Fitness Landscape Analysis of Feed-Forward Neural Networks

by

Anna Sergeevna Bosman

Submitted in partial fulfillment of the requirements for the degree
Philosophiae Doctor (Computer Science)
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria, Pretoria

February 2019

Publication data:

Anna Sergeevna Bosman. Fitness Landscape Analysis of Feed-Forward Neural Networks. Doctoral thesis, University of Pretoria, Department of Computer Science, Pretoria, South Africa, February 2019.

Electronic, hyperlinked versions of this thesis are available online, as Adobe PDF files, at:

http://cirg.cs.up.ac.za/

http://upetd.up.ac.za/UPeTD.htm

Fitness Landscape Analysis of Feed-Forward Neural Networks

by

Anna Sergeevna Bosman
E-mail: annar@cs.up.ac.za
ORCID: 0000-0003-3546-1467

Abstract

Neural network training is a highly non-convex optimisation problem with poorly understood properties. Due to the inherent high dimensionality, neural network search spaces cannot be intuitively visualised, thus other means to establish search space properties have to be employed. Fitness landscape analysis encompasses a selection of techniques designed to estimate the properties of a search landscape associated with an optimisation problem. Applied to neural network training, fitness landscape analysis can be used to establish a link between the properties of the error landscape and various neural network hyperparameters. This study applies fitness landscape analysis to investigate the influence of the search space boundaries, regularisation parameters, loss functions, activation functions, and feed-forward neural network architectures on the properties of the resulting error landscape. A novel gradient-based sampling technique is proposed, together with a novel method to quantify and visualise stationary points and the associated basins of attraction in neural network error landscapes.

Keywords: neural networks, fitness landscape analysis, classification, visualisation, loss surfaces, modality.

Supervisors : Prof. A. P. Engelbrecht
Dr. M. Helbig
Department : Department of Computer Science
Degree : Doctor of Philosophy

"Begin at the beginning," the King said, very gravely, "and go on till you come to the end: then stop."

Lewis Carroll, *Alice in Wonderland*

"Not all those who wander are lost."

J.R.R. Tolkien, *The Fellowship of the Ring*

"Would you tell me, please, which way I ought to go from here?"
"That depends a good deal on where you want to get to," said the Cat.
"I don't much care where –" said Alice.
"Then it doesn't matter which way you go," said the Cat.
"– so long as I get SOMEWHERE," Alice added as an explanation.
"Oh, you're sure to do that," said the Cat, "if you only walk long enough."

Lewis Carroll, *Alice in Wonderland*

Acknowledgements

I would like to express my deepest gratitude to the following people and organisations:

- Prof Andries Engelbrecht, my supervisor, who supported this venture into *terra incognita*, and was always generous with constructive criticism, instructive discussions, and the strictest quality control I have ever encountered.
- Dr Mardé Helbig, my co-supervisor, who provided me with immense amounts of emotional support, and was always there when I needed coffee and a person to bounce ideas off.
- Dr Katherine Malan, my colleague and role model, who bravely pioneered the field of continuous fitness landscape analysis, and inspired this study to a very large extent.
- Gregory Mark Bosman, my husband, who inevitably took this roller-coaster with me, witnessed all the ups and downs, and never doubted my ability to see it all through.
- My parents, who just could not imagine me not doing a PhD. And look, I did it!
- Anastassia Rakitianskaia, my sister, who travelled to the edge of the world with me, and is just the dearest.
- My friends, online and offline alike, who helped me keep existential fears at bay, and often provided that spark which ignites the sense of wonder.
- All members of the Computational Intelligence Research Group (CIRG), who provided a research environment that was intellectually stimulating and emotionally supportive at the same time.
- The Centre for High Performance Computing (CHPC), for the use of their infrastructure and valuable computing resources. I would probably still be running experiments if not for CHPC.

It has been a long journey. I am glad I did not have to go alone.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

Artificial neural networks (NNs) are mathematical models capable of representing an arbitrary non-linear mapping from inputs to outputs [10, 35]. Due to their non-linear information capacity, NNs have enjoyed unprecedented success in application areas such as image and speech recognition [30, 69], sequence modelling [129], and function approximation [55], amongst others. However, despite the practical success of NNs, certain theoretical properties of these models remain poorly understood. Specifically, the topological properties of the loss function surfaces associated with supervised NN training are hard to quantify and visualise due to the inherent high dimensionality of the search space [24, 68, 118]. Certain loss surface properties of the NN loss functions, such as the presence of saddle points [28, 48], plateaus, and narrow ridges [39, 57], have already been established. However, the relationship between these landscape features and the corresponding NN hyperparameters such as the number of neurons and hidden layers, or the activation functions employed, remains unclear [68]. There are on-going debates and theories regarding the presence or absence of local minima in NN error landscapes, as well as the properties of stationary points and the associated basins of attraction in the search space [28, 63, 114].

The poor understanding of the link between the NN hyperparameters and the resulting loss surface characteristics causes practitioners to make arbitrary choices for the hyperparameter values, potentially yielding sub-par performance of the final model. Failure or success of a particular combination of NN architecture and training algorithm

parameters is hard to predict. Poor understanding of the NN loss surfaces also hinders the development of new training algorithms that would take the discovered properties of the search space into consideration.

One of the main reasons for this lack of insight is the inherent high dimensionality of NN problems. High-dimensional spaces are not intuitively visualisable, thus other means of analysis have to be employed. Empirical studies of the link between the loss surface characteristics and various NN hyperparameters can be performed using fitness landscape analysis (FLA). FLA comprises of a large set of techniques designed to capture and quantify significant topological features of fitness landscapes such as ruggedness, neutrality, modality, dispersion, and searchability [81, 105]. The FLA techniques can be used to better understand the problem at hand, and to aid the process of algorithm selection and dynamic hyperparameter adaptation [56, 81]. The concept of FLA comes from the evolutionary context, and most metrics were originally defined for discrete search spaces [59, 88]. Fitness landscape properties are estimated by taking multiple samples of the search space, calculating the objective function value for every point in each sample, and analysing the relationship between the spatial and the qualitative characteristics of the sampled points. This concept can easily be translated to continuous search spaces, provided that an adequate sampling method is defined [79]. FLA of continuous fitness landscapes has recently attracted a significant amount of research [75, 81, 97, 127].

Sample analysis makes no assumptions regarding the problem at hand, and can easily be applied to black box optimisation problems such as NNs. The search space of a NN is made up of all possible real-valued weight combinations, where each weight combination corresponds to a certain measure of error. Training algorithms seek to minimise the error by searching for an optimal weight combination. Thus, the surface of a NN loss function defined in terms of the weights constitutes the NN fitness landscape, also referred to as the error landscape.

The main purpose of this study is to apply FLA to NN surfaces, and to develop a better understanding of the relationship between various NN hyperparameters and the corresponding error landscape properties. The rest of this chapter is structured as follows: Section 1.1 lists the primary objectives of this thesis. Original contributions are

summarised in Section 1.2. The structure of the thesis is outlined in Section 1.3.

1.1 Objectives

The primary objectives and sub-objectives of this thesis are summarised as follows:

1. To perform FLA of NN error landscapes.
   (a) To provide an overview of the computational intelligence techniques used in this study, namely NNs and FLA.
   (b) To investigate the sensitivity of the FLA metrics to the chosen search space boundaries.
   (c) To establish FLA as a viable method for NN error landscape analysis by performing a case study. The effect of the regularisation term on the error landscape was chosen for the case study.
   (d) To identify weaknesses in the existing FLA techniques, and to propose new FLA metrics and algorithms accordingly.
2. To establish a link between various NN hyperparameters and the corresponding error landscape properties. The following hyperparameters were considered:
   (a) Loss functions.
   (b) Activation functions.
   (c) The number and size of the hidden layers in a feed-forward NN architecture.

1.2 Contributions

The novel contributions of this work are summarised as follows:

1. A review of the existing continuous FLA metrics is performed. It is determined that no reliable modality metric for continuous high-dimensional spaces exists to date.

2. The behaviour of various existing FLA metrics on NN search spaces under different search space boundaries is investigated. All FLA metrics are shown to exhibit a sensitivity to the boundaries chosen. The weights with absolute values within the $[0.1, 1]$ interval are shown to capture information potentially useful to an optimisation algorithm. Asymmetric regions of the search space are shown to be less searchable than the symmetric regions.

3. The influence of the weight elimination term on the NN error landscape characteristics is studied using the existing FLA techniques. The weight elimination term is shown to smooth the error landscape while introducing additional minima. The backpropagation algorithm is demonstrated to efficiently search very rugged landscapes, and to struggle on step-like landscapes with rare and sudden fitness changes. Optimisation ranges for the w_0 and λ regularisation parameters are proposed.

4. The *progressive gradient walk* is proposed as a new adaptive sampling algorithm for the analysis of NN fitness landscapes. The proposed walk is more computationally efficient than a population-based adaptive walk, and has better guarantees of finding areas of high fitness. It is shown that, even though random walks provide wider search space coverage than the gradient walk, they fail to capture areas of high fitness. The gradient walk, on the other hand, is strongly biased towards the areas of high fitness, while also covering some of the poor fitness areas. Thus, the gradient walk is more representative of the search space in the context of applicability to function optimisation. In addition, the unbounded progressive gradient walk is shown to provide a truer picture of the error landscape than the bounded gradient walk.

5. A novel intuitive visualisation technique for the local minima and the associated basins of attraction is proposed, namely *loss-gradient clouds*. Loss-gradient clouds visualise the stationary points by plotting the loss values against the corresponding gradient vector magnitudes as sampled by the progressive gradient walk. Hessian matrix information is then used to identify the curvature of each sampled point to identify local minima and saddle points.

6. Two new metrics to quantify the number and extent of unique attraction basins as sampled by the gradient walks are proposed. Calculation of the statistical metrics over a number of samples provides an idea of the connectedness of the various attraction basins, as well as the likelihood of escape from the basins.

7. A visual and numerical analysis of local minima and the associated basins of attraction for two common NN loss functions, namely quadratic loss and entropic loss, is performed. Both loss functions are shown to exhibit convex local minima for the XOR problem. The majority of the classification problems considered exhibit a single main attractor around the global optimum, indicating that no local minima could be detected. The quadratic error is shown to consistently exhibit more local stationary points and associated attractors than the entropic error. The quadratic error is shown to yield superior generalisation performance in some cases.

8. The effect of five different activation functions on the resulting NN error landscapes is analysed. Rectified linear unit (ReLU) is shown to exhibit the most convexity out of all the activation functions considered, and exponential linear unit (ELU) is shown to exhibit the least flatness. The stationary points exhibited by ReLU and ELU are generally more connected than the ones exhibited by the hyperbolic tangent (TanH), indicating that ReLU and ELU yield more searchable landscapes. However, ReLU and ELU are shown to exhibit stronger sensitivity to the step size and the initialisation range than TanH. All activation functions exhibit a split into two clusters of steep and shallow gradients, associated with narrow and wide valleys. Narrow valleys are shown to be associated with saturated neurons and embedded regularised minima. The ELU activation function is shown to have superior generalisation properties compared to the other activation functions.

9. FLA of the NN loss surfaces yielded by feed-forward NN architectures with a varied number and size of the hidden layers is performed. An increase in the number of hidden layers is shown to yield a more rapid increase in flat curvature than an increase in the hidden layer size. Despite the observed flatness, an increase in the problem dimensionality is shown to yield a more searchable and more exploitable error landscape. An increase in the hidden layer size is shown to effectively reduce

the number of local minima, and to simplify the shape of the global attractor by widening the attraction basin. An increase in the number of hidden layers is shown to sharpen the global attractor, thus making it more exploitable. An increase in the number of hidden layers had no effect on the total number of stationary attractors, indicating that depth without width does not guarantee a good final solution. An increase in the number of hidden layers is shown to exaggerate the steep gradient cluster, associated with inferior generalisation performance for most problems considered. Thus, deeper architectures are shown to promote narrow valley structures.

1.3 Thesis Outline

The remainder of the thesis is organised as follows:

- **Chapter 2** provides the necessary background on NNs, and discusses the loss functions and the activation functions used in this study.
- **Chapter 3** reviews the FLA field, covers the existing continuous FLA metrics used in this study, and discusses NNs in the context of FLA. Previous studies of NN loss surfaces are reviewed in this chapter.
- **Chapter 4** investigates the sensitivity of the FLA metrics to the chosen search space boundaries.
- **Chapter 5** presents a case study that uses FLA metrics to analyse the landscape changes induced by the addition of a regularisation term to the NN loss function.
- **Chapter 6** discusses the problem of quantifying local minima and basins of attraction for NN error landscapes. A new sampling technique, as well as a novel visualisation method for the analysis of stationary points and the associated attraction basins, is proposed.
- **Chapter 7** evaluates the techniques proposed in Chapter 6 by performing a modality study of two common NN loss functions, namely quadratic loss and entropic

loss. The existing FLA techniques discussed in Chapter 3 are used to complete the FLA study of the loss functions.

- **Chapter 8** performs FLA of NN error landscapes associated with the various activation functions, namely sigmoid, TanH, ReLU, ELU, and softmax.
- **Chapter 9** analyses the effect of the NN feed-forward architecture parameters, namely the number and size of the hidden layers, on the resulting NN loss surfaces.
- **Chapter 10** summarises the findings of this thesis, and proposes some topics for future research.

The following appendices are included:

- **Appendix A** lists the benchmark problems used throughout this thesis. Each problems is briefly discussed, and the default NN architecture parameters used for each problem are provided.
- **Appendix B** presents the final classification accuracies obtained for the various benchmark problems under the various NN hyperparameter settings.
- **Appendix C** discusses the hardware and software used in the experiments conducted for this thesis, and provides a reference to the original code developed for the study.
- **Appendix D** provides a list of the acronyms used or newly defined in this thesis, as well as their associated definitions.
- **Appendix E** lists and defines the mathematical symbols used in this thesis, categorised according to the relevant chapter in which they appear.
- **Appendix F** lists the publications derived from this thesis.

Chapter 2

Artificial Neural Networks

The most advanced biological structure known to man is undoubtedly the human brain. No computer can compete with the brain's ability to learn through experience, execute complex behaviours, think, and create. Understanding the human brain is one of the most important and challenging tasks of modern cognitive science [7], as understanding the human brain can ultimately lead to understanding – and modelling – the physical nature of intelligence and consciousness itself.

The mammalian brain is a complex system of interconnected nerve cells, or neurons. Neurons communicate by sending and receiving electrochemical signals, or impulses. If the strength of the incoming impulses exceeds a certain biological threshold, a neuron will fire, i.e. send the signal further. Connections between neurons are called synapses. The synapses between actively communicating neurons strengthen over time, while unused synapses weaken. The learning process consists of strengthening and weakening the synapses such that specific incoming signal combinations trigger the desired outputs, i.e behaviours and reactions.

NNs are mathematical models inspired by the working of the mammalian brain described above, capable of representing an arbitrary non-linear mapping from inputs to outputs [10, 35]. Due to their non-linear information capacity, NNs have enjoyed unprecedented success in application areas such as image and speech recognition [30, 69], sequence modelling [129], and function approximation [55], amongst others. This chapter discusses the aspects of NNs relevant to this study. The rest of the chapter is structured

as follows: Section 2.1 describes the artificial neuron. Section 2.2 discusses the structure of feed forward NNs. Section 2.3 describes the NN training process. Section 2.4 discusses the NN loss functions used in this study. Section 2.5 discusses the relevant NN activation functions. Section 2.6 discusses regularisation of NNs. Section 2.7 describes the data preparation necessary for NN training. Finally, Section 2.8 provides a summary of the chapter.

2.1 Artificial Neuron

The history of NNs began with the invention of the artificial neuron by McCulloch and Pitts in 1943 [83], where the functionality of a biological neuron was described mathematically for the first time. Similarly to a biological neuron, an artificial neuron [49] receives a set of inputs, and produces an output based on the total strength of the incoming signal. The output is modulated using an activation function, which determines whether the neuron should fire or not. Synaptic connections are modelled as real-valued weight parameters associated with each input. Figure 2.1 illustrates the structure of a single artificial neuron.

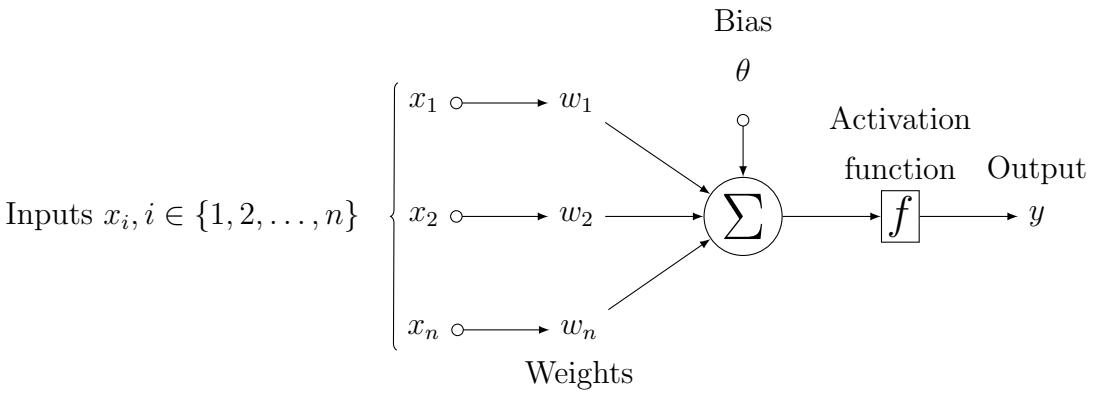


Figure 2.1: Artificial neuron

As shown in Figure 2.1, a neuron receives n inputs, where every input x_i, $i \in \{1, 2, \ldots, n\}$, is multiplied by the corresponding weight w_i. The *net* input signal, corre-

sponding to the linear combination of inputs, is offset by the bias, θ:

$$net = \sum_{i=1}^{n} x_i w_i - \theta \tag{2.1}$$

Equation (2.1) describes an n-dimensional hyperplane. Therefore, each unique set of weights w and a bias θ corresponds to a unique hyperplane that a single neuron can model. To determine the output signal, y, the activation function f is applied to the net input signal:

$$y = f(net) \tag{2.2}$$

One of the first applications of artificial neurons was to model a binary classifier, called a perceptron [112]. For this purpose, the output of a neuron had to be converted to a binary signal, i.e. $\{0, 1\}$. Such conversion can be easily obtained by comparing the weighted input signal to the threshold θ, and outputting a zero if the total signal is below the threshold, and one otherwise. This function is known as the step function, and is defined as follows:

$$f(net) = \begin{cases} 1, & \text{if } net > \theta \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

Combined with the step activation function, the hyperplane modelled by the neuron can be interpreted as a boundary between two mutually exclusive classes. The step function can be replaced by other non-linear activation functions, discussed in Section 2.5. Since a single neuron models a hyperplane, only linearly separable classes can be separated using a perceptron. This limitation was discovered in 1969 by Minsky and Papert [90], who have shown that a perceptron cannot model the exclusive-OR logical gate. However, if multiple perceptrons are combined together in layers to form a *multi-layer perceptron*, i.e. a NN, then the non-linear modelling capabilities can be established mathematically [53].

2.2 Feed Forward Neural Networks

Feed forward NNs, also referred to as multi-layer perceptrons, are acyclic collections of interconnected neurons aligned in layers. The network structure is inspired by the human brain. While each individual neuron is limited by the ability to only model

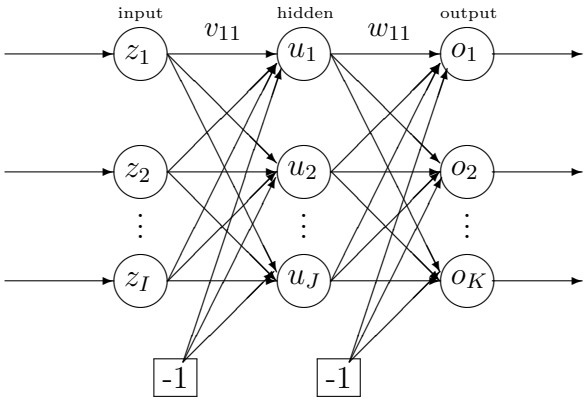


Figure 2.2: A feed forward NN with a single hidden layer.

linearly separable functions, combining multiple neurons with non-linear activations in successive layers increases the information capacity of the NN, and enables modelling non-linear functions of arbitrary complexity. An example of a feed forward NN is shown in Figure 2.2. Three layer types are shown in Figure 2.2: input, hidden, and output layers. The functionality of each layer is discussed below.

Input Layer

The input layer simulates the input stimulus that the NN receives from the environment. The purpose of this layer is to receive the inputs, and transmit them to the next layer. Thus, identity activation is used in the inputs, and the number of inputs is equal to the input dimensionality of the problem. In a fully connected NN, each input is subsequently sent to each neuron in the following layer, and each connection between the input layer and the following layer bears a weight.

Hidden Layer

The purpose of the hidden layer is to learn a non-linear transformation of the inputs that will enable the output layer to perform the desired classification or regression task. Hidden layer neurons typically employ a non-linear activation function. It has been shown that a NN with enough neurons in the hidden layer is capable of modelling any non-linear function of arbitrary complexity, thus making NNs universal approximators [53]. It was

later discovered that NNs with multiple hidden layers are capable of modelling hierarchical representations of the inputs. Thus, many-layer, or deep NNs, were shown to be more successful in real-life applications than shallow, i.e. one hidden layer NNs [71, 117]. Deep NNs, however, are largely based on the same principles as shallow NNs.

Figure 2.2 shows that all hidden units receive a signal of negative one in addition to the input signals. This signal enables each neuron in the hidden layer to learn the appropriate bias weight.

Output Layer

The number of dependent variables determines the number of neurons in the output layer. A single output is typically used for regression and binary classification problems. For multinomial classification problems, the number of outputs is usually equal to the number of classes. The activation function is chosen such that the output can be interpreted as a class distribution. For regression problems, the output units typically use linear activation, to avoid signal distortion. Bias weights are learned for each output neuron in the same way as for the hidden neurons.

NN output is calculated by a forward pass of a pattern p through the network:

$$o_{k,p} = f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj}u_j\right) = f_{o_k}\left(\sum_{j=1}^{J+1} w_{kj} f_{u_j}\left(\sum_{i=1}^{I+1} v_{ji}z_i\right)\right) \tag{2.4}$$

where o_k is the output of the k'th neuron in the output layer of size K; u_j is the output of the j'th neuron in the hidden layer of size J; z_i is the output of the i'th neurons of the input layer of size I; f_{o_k} and f_{u_j} are the activation functions of the output and hidden neurons, respectively; w_{kj} is a weight connecting the k'th output neuron and the j'th hidden neuron, and v_{ji} is a weight connecting the j'th hidden neuron and the i'th input neuron; and $w_{k,J+1}$, $v_{j,I+1}$ account for the bias weights in the output and hidden layers, respectively.

2.3 Neural Network Training

NN training mimics the learning process in the human brain, where active synapses are reinforced, and inactive ones are weakened. This study deals with supervised training

only. In the supervised training paradigm, a dataset D is available that maps inputs to the desired outputs. The dataset is divided into two mutually exclusive subsets, the training set D_T, and the generalisation set D_G. Given D_T inputs, the NN tries to predict the outputs. The NN prediction is compared to the target outputs, an error metric E is calculated, and the weights are adjusted to minimise E. Common error metrics, referred to as loss functions, are discussed in Section 2.4.

The training process is repeated until some stopping criteria is met. Once the NN has been trained, D_G is used to evaluate the ability of the trained model to correctly predict the outputs for the patterns not seen during training.

NN weights can be updated for every pattern via sequential pattern presentation. This approach is referred to as stochastic training. Alternatively, every pattern in the dataset can be evaluated first, then the cumulative error is used to perform the weight updates. This approach is referred to as batch training. Batch training typically exhibits robust performance, as minimising the cumulative error should theoretically benefit every pattern in the dataset. However, calculating the cumulative error for very large datasets may become prohibitively expensive. Stochastic training is more cost-efficient, but is prone to oscillatory behaviour, because weight updates for individual patterns are likely to contradict each other, causing the NN to repetitively learn and un-learn. A middle-ground solution called mini-batch training [131] has been proposed that combines the benefits of both stochastic and batch learning. For mini-batch training, small subsets, or mini-batches are used to calculate the error. Optimisation of the mini-batch size per problem can provide a cost-efficient, yet reliable error estimate.

Gradient-based training

One of the most prominent approaches to NN training is a gradient-based algorithm, known as the backpropagation of error, formulated for the first time by Werbos in 1974 [143]. The backpropagation algorithm adjusts the weights by calculating the gradient of E with respect to each weight in the network. Backpropagation is defined recursively, thus the gradients are calculated for the outer layers first, and then back-propagated to the earlier layers. Each weight w_{ab} between neurons a and b is adjusted

by adding a negative gradient scaled by the learning rate parameter η:

$$\begin{aligned} \Delta w_{ab} &\leftarrow \eta \left(-\frac{\partial E}{\partial w_{ab}} \right) \\ w_{ab} &\leftarrow w_{ab} + \Delta w_{ab} \end{aligned} \tag{2.5}$$

For the output layer weights, the gradient is calculated, using the chain rule, as:

$$\frac{\partial E}{\partial w_{ab}} = \frac{\partial E}{\partial o_b} \frac{\partial o_b}{\partial net_b} \frac{\partial net_b}{\partial w_{ab}} = \frac{\partial E}{\partial o_b} \frac{\partial o_b}{\partial net_b} y_a \tag{2.6}$$

where y_a is the output of the a'th hidden neuron.

For the non-output layer weights, the gradient is recursively calculated, using the chain rule, as:

$$\frac{\partial E}{\partial w_{ab}} = \frac{\partial E}{\partial y_b} \frac{\partial y_b}{\partial net_b} \frac{\partial net_b}{\partial w_{ab}} = \sum_{m \in M} \left(\frac{\partial E}{\partial y_m} \frac{\partial y_m}{\partial net_m} w_{bm} \right) \frac{\partial y_b}{\partial net_b} y_a \tag{2.7}$$

where y_a and y_b are the outputs of the a'th and b'th neuron, respectively, and M is the number of neurons in the successive layer that receives an input from neuron b.

To smooth out potential oscillations caused by stochastic and mini-batch training, a momentum term is often added to the weight update:

$$w_{ab} \leftarrow w_{ab} + \Delta w_{ab}(t) + \alpha \Delta w_{ab}(t-1) \tag{2.8}$$

where t is the iteration count, and α is the momentum coefficient that controls the influence of the past weight updates on the current weight update.

Thus, backpropagation relies heavily on the derivatives of the error function, as well as the derivatives of the activation functions employed in the hidden and output layers.

2.4 Loss Functions

The two most widely used NN error metrics are the quadratic loss function and the entropic loss function, discussed in this section.

Quadratic loss, also referred to as the sum squared error (SSE), simply calculates the sum of squared errors produced by the NN:

$$E_{sse} = \sum_{p=1}^{P} \sum_{k=1}^{K} (t_{k,p} - o_{k,p})^2 \tag{2.9}$$

where P is the number of data points, K is the number of outputs, $t_{k,p}$ is the k'th target value for data point p, and $o_{k,p}$ is the k'th output obtained for data point p. Minimisation of the SSE minimises the overall error produced by the NN.

If the outputs of the NN can be interpreted as probabilities, then the cross-entropy between two distributions can be calculated, i.e. the distribution of the desired outputs (targets), and the distribution of the actual outputs. The entropic loss, also referred to as the cross-entropy (CE) error, is formulated as follows:

$$E_{ce} = \sum_{p=1}^{P} \sum_{k=1}^{K} \left(t_{k,p} \log o_{k,p} + (t_{k,p} - 1) \log (o_{k,p} - 1) \right) \tag{2.10}$$

Minimisation of the cross-entropy leads to convergence of the two distributions, i.e. the actual output distribution resembles the target distribution more and more, thus minimising the NN error.

2.5 Activation Functions

The step function defined in Equation (2.1) is non-differentiable, which proves to be an important disadvantage, since gradient-based training algorithms such as backpropagation make use of the activation function derivatives. For this reason, a smooth approximation of the step function, known as the sigmoid function [113], was proposed:

$$f(net) = \frac{1}{1 + e^{-net}} \tag{2.11}$$

The output of the sigmoid function is in the $(0, 1)$ continuous range, which corresponds to the binary range of the step function. Figure 2.3 shows the graph of the sigmoid function, and illustrates that the sigmoid exhibits linear behaviour around the origin, and saturates (approaches asymptotes) for large positive and negative input values.

The sigmoid was designed to mimic the step function, and is thus constrained to the positive range of $(0, 1)$. As the signal is propagated from one layer to the next, consistently positive signals are likely to cause hidden neuron saturation, i.e. cause some neurons to output values close to the asymptotic ends. Neuron saturation is generally undesirable [73], since the gradient is very weak near the asymptotes, and may cause stagnation in the training algorithms [109].

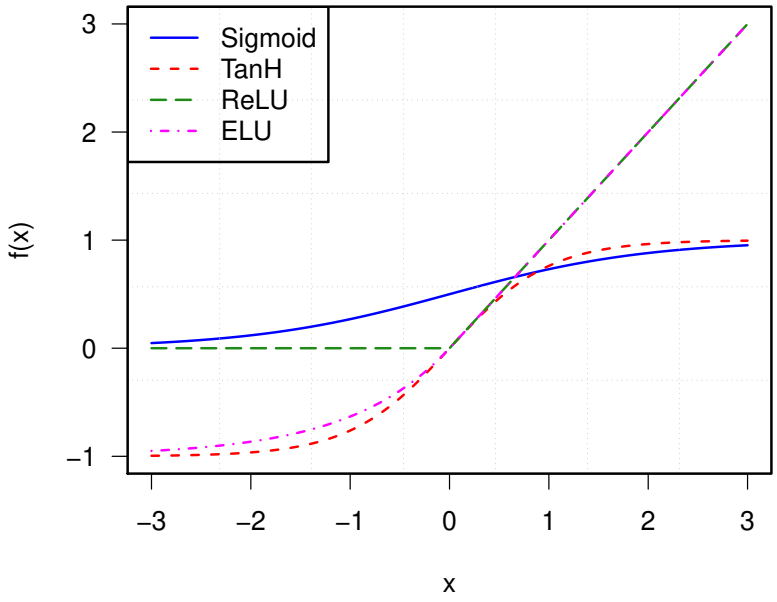


Figure 2.3: Activation functions

To alleviate saturation, the hyperbolic tangent function, further referred to as TanH, can be used in place of the sigmoid. TanH is defined as:

$$f(net) = \frac{e^{net} - e^{-net}}{e^{net} + e^{-net}} \tag{2.12}$$

The output of TanH is in the range $(-1, 1)$, and is therefore zero-centered. Figure 2.3 illustrates that TanH has a sigmoidal s-like shape, and approaches asymptotes at -1 and 1. Since the function is bounded, the neurons can still saturate, but the zero-centered range makes saturation less likely.

As discussed in Section 2.3, the backpropagation algorithm is defined recursively, and the activation function derivatives of the later layers affect the weight updates of the earlier layers. This recursive definition gives rise to the so-called vanishing gradient problem [51, 52]. If the activation function derivative values are generally smaller than one, then the error signal will become weaker and weaker during the course of backpropagation from the output layer to the input layer, due to being recursively multiplied by a small value. Figure 2.4 illustrates that both the sigmoid and the TanH activations exhibit small derivatives due to their asymptotic behaviour. The vanishing gradient problem

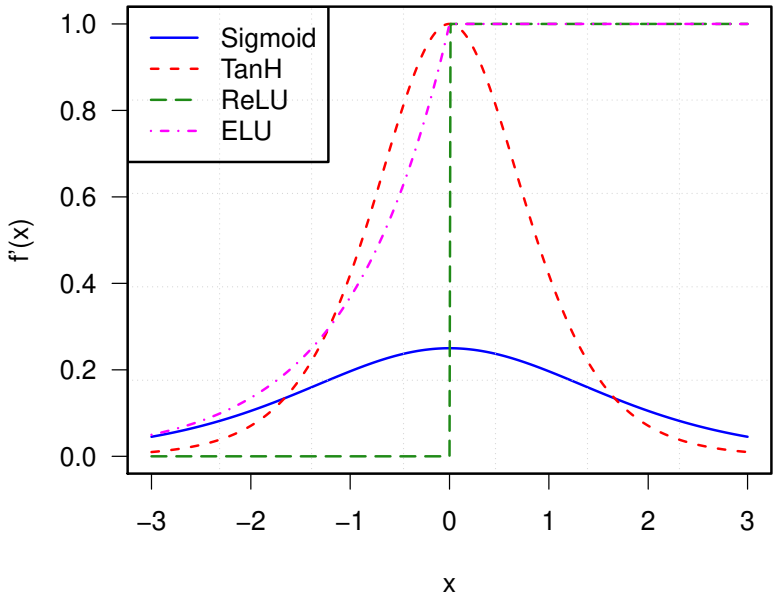


Figure 2.4: Derivatives of the activation functions

makes training deep NNs hard and largely ineffective.

To combat the vanishing gradients, the rectified linear activation function, further referred to as ReLU, was proposed as an alternative to the sigmoidal activations [45, 98]. The ReLU activation is defined as follows:

$$f(net) = \begin{cases} net, & \text{if } net > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.13}$$

Figure 2.3 illustrates that the ReLU activation is simply the identity function for all positive input signal values. As a result, the derivative of ReLU is identical to the step function, illustrated in Figure 2.4. The function is not differentiable at zero, but the derivative can be simply set to zero at the origin. Thus, the ReLU activation saturates only for negative input signals. A constant derivative of one for all positive inputs ensures that positive signals can be effectively adjusted by backpropagation. To date, ReLU is one of the most popular activation functions in deep NN learning [4, 117, 152].

ReLU has been criticised for the hard saturation (derivative of zero) imposed on all negative inputs. While hard saturation is argued to be biologically plausible [45],

the derivative of zero indicates that negative inputs will not yield any weight updates. Thus, if a particular neuron results in a saturated configuration (outputting zero for all inputs) during the course of training, the neuron will not be able to recover, and will not contribute anything to the final model.

Numerous adaptations of the ReLU activation have been proposed, most of them adding a slope with non-zero gradient for the negative inputs [98, 150]. One of the more recent successful modifications of the ReLU is the exponential linear activation [26], further referred to as ELU. The ELU activation is defined as:

$$f(net) = \begin{cases} net, & \text{if } net > 0 \\ e^{net} - 1, & \text{otherwise} \end{cases} \tag{2.14}$$

The ELU activation is illustarted in Figure 2.3, and ELU's derivative is illustrated in Figure 2.4. For the negative inputs, ELU uses an exponentially decaying curve. As a result, negative inputs have a non-zero derivative that gradually approaches the asymptote. Thus, negative inputs contribute to the final model, making the NN easier to train.

Output neuron activations should be considered as a special case. For regression problems, linear activation is often used to avoid signal distortion. For classification problems, however, one-hot binary encoding, discussed in Section 2.7, is typically used for the target variables, thus the output signals should be in the same range as the binary codes. The sigmoid activation can be used for this purpose, due to the convenient output range of $(0, 1)$. A probabilistic interpretation can also be applied to the sigmoid activations, where each output represents the probability of a pattern p belonging to a particular class. If sigmoid activations are used in the output layer, then each output neuron models the probability distribution of a class.

For multinomial classification problems, sigmoid outputs have an important disadvantage: since each neuron models a probability distribution, the NN may predict two or more mutually exclusive classes with a high probability, making the output less interpretable. As an example, for three classes, the NN may output $\{0.9, 0.9, 0.9\}$, predicting every class with 90% probability. If the classes are mutually exclusive, such behaviour is undesirable. An alternative to the sigmoid is the softmax function [17], which ties all

output neurons into a single joint probability distribution:

$$f(net) = \frac{e^{net}}{\sum_{k=1}^{K} e^{net_k}} \tag{2.15}$$

where K is the total number of output neurons. The softmax function ensures that the total activation of the output layer is equal to one.

2.6 Regularisation

The ability of a NN to correctly predict the outputs of input patterns not seen during training is known as the generalisation ability. A model that cannot generalise has no practical use, therefore maximising the generalisation potential of a NN is a major goal of NN training. A simple, yet effective way to improve the generalisation ability of a NN is to add a weight regularisation term to the loss function [92, 91, 125]. Weight regularisation is applied to minimise both NN error and NN complexity. If E_p is a penalty function that quantifies the complexity of a NN, the objective function can be modified as follows:

$$E_{nn} = E + \lambda E_p \tag{2.16}$$

where λ is a hyperparameter controlling the "strength" of regularisation. If λ is too small, the value of the penalty function will be much smaller than the error value, and the error is likely to "overshadow" the penalty, thus causing the penalty to be disregarded. On the other hand, if λ is too big, the penalty contribution to the objective function will become larger than the error term contribution, and the algorithm will focus on minimising the NN complexity instead of minimising the error. In practice, λ is chosen empirically per problem and per penalty function E_p.

The complexity of a NN can be expressed by the overall number of NN weights. Simplistic architectures with too few weights may be incapable of learning a complex problem representation. Excessive architectures with too many weights, on the other hand, may promote overfitting, i.e. poor generalisation. Thus, penalty functions are usually designed to optimise the total number of NN weights.

A well-known L2 (i.e. quadratic) penalty function proposed in the literature is weight

decay [50], given by

$$E_p = \frac{1}{2}\sum_{l=1}^{W} w_l^2 \tag{2.17}$$

where W is the total number of weights in the NN, and w_l is the l-th weight. The weight decay penalty essentially calculates the magnitude of the weight vector. The larger the magnitude, the more the NN will be penalised. Limiting the weight growth tends to improve NN generalisation [92], since the relevant weights are reinforced by the training algorithm at every iteration, while the irrelevant ones decay towards zero over time.

A disadvantage of weight decay is that no differentiation between relevant and irrelevant weights is explicitly made, thus both large and small weights are penalised with the same rigour. Weigend et al. [140] introduced an alternative L2 penalty function, which uses an extra parameter w_0 to specify the threshold that separates relevant weights from irrelevant weights:

$$E_p = \sum_{l=1}^{W} \frac{w_l^2/w_0^2}{1 + w_l^2/w_0^2} \tag{2.18}$$

This penalty function is known as weight elimination. Parameter w_0 defines a threshold that distinguishes between significantly and insignificantly large weights. Weights with $|w| >> w_0$ yield a complexity cost close to 1, and contribute towards the penalty term in Equation (2.18). Thus, weights with $|w| >> w_0$ are seen as "too large" and in need of regularisation. Weights with $|w| << w_0$ yield a complexity cost close to zero, and contribute very little to the weight elimination term. Thus, weights with $|w| << w_0$ are not penalised. A small w_0 value will result in more weights being penalised, thus only the most persistent weights will survive, yielding an architecture comprised of few larger weights. On the other hand, for a large w_0 value, small weights will not be subject to the penalty, resulting in an architecture made up of many small weights.

The preference of a few large weights or many small weights is problem-dependent, although it should be noted that large weights may cause the NN to saturate. Very large weights increase the signal strength, causing the bounded activation functions to output near-asymptotic values.

2.7 Data Preparation

For the purpose of supervised training, data has to be converted to a format usable by the NN. Inputs and outputs must be numeric, therefore nominal attributes have to be either binary coded, or binned into real-valued intervals. For multinomial classification, the outputs are typically one-hot encoded. One-hot encoding uses a bit vector of the same length as the total number of classes. For each pattern p, only one bit in the vector is set to '1', while the rest of the bits are set to '0'. The position of the '1' bit indicates the nominal value of p.

The inputs are typically scaled to a small range around zero to avoid saturation. A study by LeCun [73] indicated that the scaling of each input attribute to have a mean of zero and a variance of one benefits the gradient-based training process.

For classification problems, and for sigmoidal activation functions with range $(0, 1)$, the binary outputs are often scaled to lie in the $[0.1, 0.9]$ interval, as opposed to $[0, 1]$. Such scaling enables the sigmoid output layer to achieve an error of zero, as $[0.1, 0.9]$ lies within the asymptotic range.

2.8 Summary

This chapter discussed the structure and the training process of NNs. Loss functions, activation functions, regularisation, and data preparation relevant to this study were also described. The next chapter provides an overview of fitness landscape analysis, and discusses the existing insights into the nature of NN error landscapes.

Chapter 3

Fitness Landscape Analysis

A fitness landscape refers to the hypersurface formed by the objective function values of an optimisation problem calculated across the search space. The goal of FLA is to estimate and quantify various features of the objective function hypersurface, such as ruggedness, neutrality, and searchability, and to discover correlations between landscape features and algorithm performance [81, 105]. Through quantifying landscape features, FLA can be used to gain a better understanding of the problem nature, and thus aid the development of new algorithms. The term "fitness landscape" was coined in the evolutionary optimisation community [59, 88]. FLA techniques were inspired by genetics research [148], and therefore were originally developed for discrete binary search spaces. However, the notion of fitness landscapes was recently extended to continuous spaces [77, 97, 105]. Any optimisation problem with a well-defined objective function can be studied from the FLA perspective.

The purpose of this chapter is to review the existing FLA approaches for continuous problems, and to discuss the applicability of FLA for NN analysis. The rest of the chapter is structured as follows: Section 3.1 formally defines fitness landscapes. Section 3.2 discusses the characteristics of fitness landscapes that can have an effect on the performance of an optimisation algorithm. Section 3.3 describes sampling techniques used for FLA. Section 3.4 discusses FLA techniques developed for continuous search spaces. Section 3.5 discusses NN training in the context of FLA, and reviews the existing research on NN fitness landscapes. Finally, Section 3.6 provides a summary of the chapter.

3.1 Fitness Landscapes

A formal definition of a fitness landscape was proposed by Stadler [126], where the fitness landscape of an optimisation problem was described by three components:

1. A set of X configurations, i.e. a set of solutions.
2. A notion d of distance, neighbourhood, or accessibility on X.
3. A fitness function $g : X \to \mathbb{R}$.

X defines the solution space, i.e. the domain of g, and d defines the structure of X, i.e. the nature of transition between distinct solutions. Thus, a fitness landscape F is determined by two functions, g and d, which define the fitness, i.e. the quality of solutions, and the distances, or transitions between solutions [105]:

$$F = (X, g, d) \tag{3.1}$$

In the context of continuous optimisation, X represents a continuous range, containing all valid solutions. For constrained problems, X may be limited to a certain interval. For unconstrained problems, X is defined as all points in $\mathbb{R}$.

The neighbourhood $\mathcal{N}_\varepsilon$ of a solution $\mathbf{x} \in X$ is defined as all points $\mathbf{x}'$ for which $d(\mathbf{x}, \mathbf{x}') \leq \varepsilon$, where ε defines the neighbourhood radius. In a binary space, d can refer to the Hamming distance, i.e. the number of different bits between two bit strings. Thus, in binary space and other discrete spaces, every point $\mathbf{x}$ will have a finite number of neighbours. In continuous spaces, the Euclidean distance is typically used as d; thus $\mathbf{x}$ will have an infinite number of neighbours, using the same definition of $\mathcal{N}_\varepsilon$.

If X is all points in $\mathbb{R}$, and d is the Euclidean distance, then the fitness function g defines F for continuous optimisation problems. Without loss of generality, minimisation problems will be considered. The goal of a continuous minimisation problem is to discover a global minimum $\mathbf{x}^*$ given $g(\mathbf{x})$:

$$\mathbf{x}^* = \arg\min_X g(\mathbf{x}) \tag{3.2}$$

Studying the fitness landscapes of continuous problems is difficult, because X is an infinite set, and cannot be exhaustively enumerated. Given that continuous problems

are typically simulated using computer architectures with limited precision, the representation of the continuous space used in practice can be considered finite. However, exhaustively enumerating this finite space remains practically infeasible. Thus, sampling has to be employed to obtain X', an approximation of X. Sampling techniques for continuous problems are discussed in Section 3.3.

3.2 Characteristics of Fitness Landscapes

A number of surveys are available [77, 87, 97, 105] that list and discuss various quantifiable properties of fitness landscapes that may have an influence on the performance of an optimisation algorithm. All of these properties fall into four broad categories: modality, structure, separability, and searchability. Each category is discussed below.

3.2.1 Modality

Modality refers to the number and structure of optima in a fitness landscape. Maximisation problems are concerned with locating maxima, and minimisation problems are concerned with locating minima. The rest of this chapter assumes minimisation. The definition of a global minimum $\mathbf{x}^*$ is given in Equation (3.2). A local minimum $\tilde{\mathbf{x}}$ can be defined as a point in X that has the smallest fitness value in its neighbourhood:

$$\tilde{\mathbf{x}} \in X \iff g(\tilde{\mathbf{x}}) < g(\mathbf{x}) \; \forall \; \mathbf{x} \in \mathcal{N}_\varepsilon(\tilde{\mathbf{x}}) \tag{3.3}$$

Optimisation problems can be unimodal, i.e. contain a single optimum in the landscape, or multimodal, i.e. contain multiple optima in the landscape. An important property of a multimodal fitness landscape is the number of local optima, since local optima have the potential to trap an optimisation algorithm. In addition to the number of local optima, it is important to understand the structure of the optima, i.e. the properties of the basins of attraction. Basins of attraction are defined as the regions around optima that would lead a local search to the optima. Optima with large basins of attraction are easier to find, while small and narrow basins make local search harder [61, 127]. Thus, the size and shape of attraction basins are important fitness landscape properties.

Closely related to the local optima is the concept of stationary regions in the landscape. If the inequality in Equation (3.3) is relaxed to $g(\tilde{\mathbf{x}}) \leq g(\mathbf{x})$, then a local minimum can be extended to a local plateau, i.e. a flat region where an optimisation algorithm may also stagnate.

Distribution, frequency, and distance between local minima and other stationary regions are likely to have an effect on the performance of an optimisation algorithm. Another important property is the difficulty of transition from one optimum to another optimum, i.e. the maximum fitness value on the minimal path between two optima. This value is often referred to as a fitness barrier [105].

Finally, saddle points, i.e. stationary points in the landscape with opposite curvatures at perpendicular planes, also form a notable landscape feature. The stationary part of a saddle region, especially extended to a plateau, may attract an optimisation algorithm, and slow down the search. However, the presence of downward curvature makes saddles significantly easier to escape than the local optima.

3.2.2 Structure

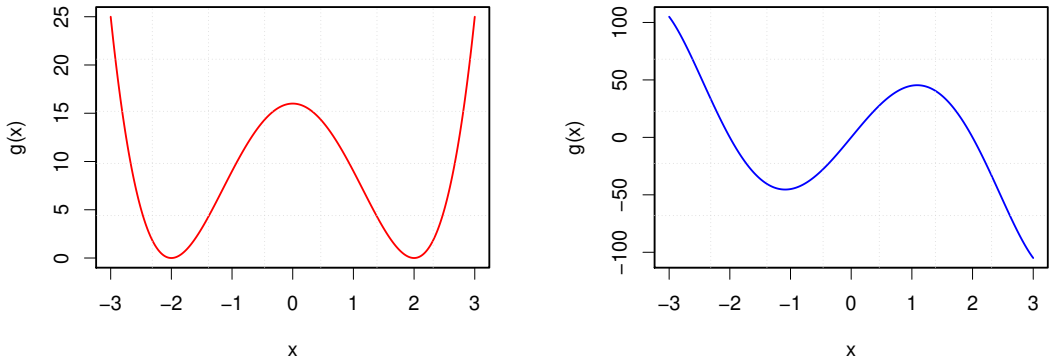
The structure of a fitness landscape is described by the amount of variability in the fitness values, and is closely related to the number of local optima. A rugged landscape is a landscape that yields a lot of variation in the fitness values when traversed, i.e. decreasing as well as increasing fitness. A smooth landscape is a landscape that yields a consistent trend when traversed, i.e. either consistently decreasing or increasing fitness, but not both. Finally, a neutral landscape is a landscape that yields very little to no variation in the fitness values when traversed, i.e. a flat landscape. A rugged landscape is more likely to have multiple local minima than a smooth landscape, and is typically harder to search [127]. A neutral landscape may also be non-trivial to optimise, because the lack of fitness variability implies the lack of information to guide the search.

An important structural property of a fitness landscape is the magnitude of fitness changes, i.e. the gradient information available to an optimisation algorithm [81]. The gradient information is particularly important to gradient-based search algorithms which use the analytical gradient directly. Gradient magnitude is also related to fitness barriers, and may indicate how difficult it will be for an optimisation algorithm to escape a local

optimum.

The presence or absence of symmetry in the landscape may also contribute to problem difficulty. A fitness landscape may be symmetric with respect to one of the axis, or a single point, eg. the origin. Symmetry implies that mutually exclusive subsets of the fitness landscape will have identical sets of fitness values arranged in the same topological structure. Figure 3.1 illustrates two types of function symmetry: even and odd. Both functions are symmetric about the origin. In the even case, two identical global minima can be found on either side of zero. In general, if a global or local minimum is found in a landscape subset with an even symmetric counterpart, another minimum of the same quality will exist in the symmetric subset. Thus, symmetry may yield a redundancy in local and global minima. The effect of symmetry on algorithm performance was studied in the context of genetic algorithms [27, 99, 144], showing that different types of symmetry can be either detrimental or conducive to algorithm performance.

Figure 3.1: Examples of symmetric functions



(a) Even symmetry (b) Odd symmetry

3.2.3 Separability

Separability is closely related to the concept of epistasis, borrowed from the field of genetics [29]. A chromosome is said to have low epistasis if the genes contribute independently to the fitness of the chromosome. High epistasis, on the other hand, means that the contribution of each gene is dependant on the values of the other genes in the chromosome.

This concept directly maps to variable inter-dependency in an optimisation problem: if each variable contributes independently to the fitness function, then each variable can be optimised separately, making the problem separable. If the contribution of a variable to the final fitness depends on other variable contributions, then per-variable optimisation becomes problematic. Problems with inter-dependent variables are referred to as non-separable. Separability is an important problem characteristic, since the degree of separability exhibited by the problem determines whether the problem can be broken down into simpler sub-problems. Studies have shown that genetic algorithms perform better on separable problems [27].

3.2.4 Searchability

Searchability [78], also referred to as evolvability in the genetic algorithm context [2, 133], refers to the ability of a search algorithm to find a better solution given the current solution, or a pool of solutions. In the genetic algorithm context, evolvability is defined as the ability of a population to produce fitter offspring [2, 133]. In the continuous optimisation problem context, searchability refers to the probability of success of a local search, and is typically quantified in terms of the presence or absence of the information that can guide the search, as well as the presence or absence of the information that can "deceive" the search. Thus, searchability is a metric of problem hardness. Whether algorithm-independent problem hardness can be estimated is not clear [96], because a problem that is difficult for a population-based algorithm may be easy for a gradient-based algorithm. Potential problem hardness predictors are the global problem structure, and the general deceptiveness of the landscape [78]. A deceptive landscape is a landscape where increasingly worse fitness is observed on a path leading to an optimum.

Searchability, or problem hardness, is a collective problem characteristic that correlates with modality, structure, and separability of the problem.

3.3 Sampling Techniques

The goal of FLA is to estimate various fitness landscape properties as discussed in Section 3.2. In continuous spaces, it is important to be able to provide adequate estimates

without a complete enumeration of every point in the search space, since complete enumeration is not available. Thus, fitness landscape properties have to be estimated by taking random or biased samples of the search space, calculating the objective function value for every point in each sample, and analysing the relationship between the spatial and the qualitative characteristics of the sampled points. Two types of sampling are used: random sampling, discussed in Section 3.3.1, and random walks, discussed in Section 3.3.2.

3.3.1 Random sampling

Random sampling entails sampling points from the search space using a probability distribution. Uniform random sampling is commonly used, although uniform samples are known to cluster around the origin as the dimensionality of the problem increases. To improve search space coverage, the Latin hypercube was suggested as a sampling technique by Mersmann et al. [87]. Regardless of the probability distribution and the chosen sampling technique, an important property of random samples is spatial independence of the sampled points, i.e. a random sample does not reflect the distance between individual sampled points.

3.3.2 Random walks

Random walks are used in multiple scientific fields, such as physics, biology, and economics [104, 122]. In the context of FLA, random walks provide an alternative to random sampling from a given probability distribution [142, 79]. As opposed to a random sample, where individual points in the sample are spatially uncorrelated, random walks generate spatially correlated samples. It is precisely the spatial correlation between the individual steps of the sample that can be exploited to obtain descriptive information such as the degree of ruggedness, neutrality, or gradients present in the search space [81].

Definition of a random walk

A random walk of length L is a sequence of points in an m-dimensional search space, obtained by starting at a certain point in the search space, $\mathbf{x}_0$, and generating the next

point, $\mathbf{x}_1$, by randomly selecting a neighbour of $\mathbf{x}_0$. In general, every $\mathbf{x}_{l+1}$ is obtained by randomly selecting a neighbour of $\mathbf{x}_l$. Thus, a random walk, X_L, is a sequence of points $(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_L)$, where every $\mathbf{x}_{l+1}$ is derived from $\mathbf{x}_l$ using a neighbourhood function, $\mathbf{x}_{l+1} \leftarrow \mathcal{N}_\varepsilon(\mathbf{x}_l)$.

Adaptive walks

The unbiased nature of random walks ensures that each point in the search space has an approximately equal probability of being selected. However, if the purpose of the analysis is to quantify the presence and extent of local minima, such a randomised approach may not prove very useful. Indeed, if a random sample does not contain any points associated with good fitness, no conclusions about the landscape features of the areas associated with good fitness can be made. Thus, fitness landscape characteristics are often derived from an *adaptive* rather than a random walk [62, 105]. Adaptive walks were originally defined for binary problems. To perform an adaptive walk, a neighbour $\mathbf{x}_o$ of $\mathbf{x}_l$ is randomly chosen. The neighbour $\mathbf{x}_o$ is accepted as the next step of the walk, $\mathbf{x}_{l+1}$, if and only if the fitness of $\mathbf{x}_o$ is better than the fitness of $\mathbf{x}_l$ [62]. In the context of genetic algorithms, a neighbour of $\mathbf{x}_l$ can be generated by applying a random mutation to $\mathbf{x}_l$, i.e. randomly flipping one or more bits of $\mathbf{x}_l$. This approach is equivalent to stochastic hill climbing in a binary space. Kauffman and Levin [62] estimated the ruggedness of the landscapes based on the average length of the adaptive walk. A shorter average length would indicate a rugged landscape, whereas a longer average length would be indicative of larger areas of consistently decreasing fitness.

Random walks in continuous search spaces

Discrete space sampling can be performed exhaustively, because each point $\mathbf{x}_l$ will at all times have a finite number of neighbours. This is not the case in continuous spaces, where every point $\mathbf{x}_l$ has an infinite number of neighbours in every dimension. Therefore, both random walks and adaptive walks can only be used in continuous spaces if neighbour selection is defined as a finite process.

The neighbourhood of a point $\mathbf{x}_l$ in a continuous m-dimensional space can be defined as all points within a certain distance from $\mathbf{x}_l$. Malan and Engelbrecht [79] proposed the

following hypercube definition of the continuous neighbourhood of $\mathbf{x}_l$:

$$\mathbf{x}_o \in \mathcal{N}_\varepsilon(\mathbf{x}_l) \iff |x_{oj} - x_{lj}| \leq \varepsilon, \forall j \in \{1, \ldots, m\} \tag{3.4}$$

where $\mathbf{x}_o$ is a neighbour of $\mathbf{x}_l$ if and only if for every dimension j the absolute difference between x_{oj} and x_{lj} does not exceed some ε.

Using Equation (3.4), a single step of a simple random walk can be defined as randomly generating an m-dimensional step vector $\Delta\mathbf{x}_l$, such that $\Delta\mathbf{x}_{lj} \in [-\varepsilon, \varepsilon]\ \forall j \in \{1, \ldots, m\}$, and adding $\Delta\mathbf{x}_l$ to $\mathbf{x}_l$ to generate $\mathbf{x}_{l+1}$:

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \Delta\mathbf{x}_l \tag{3.5}$$

A simple random walk is isotropic, i.e. not biased towards a particular direction, since the direction of each step is randomised. An anisotropic, or directionally biased variant of a random walk was proposed by Malan and Engelbrecht [79], called a progressive random walk. The progressive random walk assigns a randomly chosen direction bias to each walk in order to improve overall search space coverage. Direction bias is represented by an m-dimensional randomly generated bit mask $\mathbf{b}$. A single step of a progressive random walk can be defined as randomly generating an m-dimensional step vector $\Delta\mathbf{x}_l$, such that $\Delta x_{lj} \in [0, \varepsilon]\ \forall j \in \{1, \ldots, m\}$, and setting the sign of each Δx_{lj} according to the corresponding b_j:

$$\Delta x_{lj} = \begin{cases} -\Delta x_{lj}, & \text{if } b_j = 0. \\ \Delta x_{lj}, & \text{otherwise.} \end{cases}$$

Equation (3.5) is then used to generate $\mathbf{x}_{l+1}$. Thus, the magnitude of the step is randomised per dimension, but the overall direction of movement remains persistent. For a more detailed discussion of the algorithm, refer to [79].

The Manhattan progressive random walk is a variation of the progressive random walk, proposed in [76]. For the Manhattan walk, the step size is set to ε, and each step modifies only one randomly chosen dimension of $\mathbf{x}_l$. Manhattan progressive random walks were used in [76, 80] to provide an efficient estimate of the fitness landscape gradients.

Both the simple random walk and the progressive random walk do not take the fitness of the neighbours into account when generating the next step, $\mathbf{x}_{l+1}$. To perform

an adaptive walk in a continuous search space, an optimisation algorithm can be used to calculate $\Delta \mathbf{x}_l$. For example, particle swarm optimisation (PSO) has been proposed in the past as a sampling method [81]. Each particle in the swarm represents a candidate solution, and the next step of the walk can be defined in terms of the next step of the global best particle in the swarm. The drawback of optimisation algorithm sampling is that the success of the sampling is tightly coupled with the success of the chosen algorithm.

3.4 Fitness Landscape Analysis for Continuous Landscapes

The purpose of this section is to provide an overview of existing FLA metrics developed for continuous optimisation problems. The list is not exhaustive, and aims to outline the existing research directions, as well as to discuss the FLA metrics used in this study. The techniques are grouped into four broad categories identified in Section 3.2: modality, structure, separability, and searchability.

3.4.1 Modality

For combinatorial landscapes, application of local search to a selection of random starting points, and analysis of the best fitness solutions as discovered by the local search from different starting points, was proposed as a method to estimate the total number of optima and the size of the corresponding basins of attraction [41, 42]. The method was adapted to continuous spaces [18, 19], and involves performing local searches with gradient-based updates, and then calculating the Euclidean distance between discovered minima to determine whether the minima belong to the same basin of attraction. However, it was demonstrated by Morgan and Gallagher [95] that using the Euclidean distance in high-dimensional spaces can be very misleading, since all distances begin to look the same when the dimensionality of the problem is high enough. Thus, the applicability of the modality metrics proposed in [18, 19] to high-dimensional problems has not been established yet.

3.4.2 Structure

Structural properties estimated by FLA metrics include ruggedness, smoothness, neutrality, and gradients.

Ruggedness

In continuous spaces, ruggedness is often measured in terms of the information entropy present in a random walk sample. Existing ruggedness metrics are based on the information characteristics' analysis first introduced by Vassilev et al. [137] for discrete problem spaces. One of the successful continuous adaptations of Vassilev's approach is the first entropic measure of ruggedness (FEM), proposed by Malan and Engelbrecht [75]. To calculate FEM, a progressive random walk through the search space is taken, generating a time series of fitness values $\{g_l\}_{l=0}^{L}$. A symbol sequence, $S(\epsilon) = s_1 s_2 ... s_L$, is generated from $\{g_l\}_{l=0}^{L}$, where $s_l \in \{\bar{1}, 0, 1\}$ is given by

$$s_l = \Psi_{g_l}(l, \epsilon) = \begin{cases} \bar{1}, & \text{if } g_l - g_{l-1} < -\epsilon \\ 0, & \text{if } |g_l - g_{l-1}| \leq \epsilon \\ 1, & \text{if } g_l - g_{l-1} > \epsilon \end{cases}$$

where ϵ is the chosen sensitivity threshold. An entropic measure $H(\epsilon)$ is now defined as

$$H(\epsilon) = -\sum_{p \neq q} P_{[pq]} \log_6 P_{[pq]}$$

where $p, q \in \{\bar{1}, 0, 1\}$, and $P_{[pq]}$ is given by

$$P_{[pq]} = \frac{n_{[pq]}}{n}$$

where $n_{[pq]}$ is the number of sub-blocks pq in $S(\epsilon)$. Note that $p \neq q$, thus the total number of unique pq value combinations is 6. The value of $H(\epsilon)$ depends on the chosen value for ϵ. It was shown in [75, 81] that for a certain ϵ^*, $H(\epsilon^*)$ converges on the value of 0 for any $\{g_l\}_{l=0}^{L}$. The value of ϵ^* is defined as the smallest value of ϵ for which the landscape becomes flat. The value of FEM is calculated as follows:

$$FEM = \max_{\forall \epsilon \in [0, \epsilon^*]} \{H(\epsilon)\}$$

Two FEM measures are usually used to describe a fitness landscape: micro-ruggedness, $FEM_{0.01}$, where the maximum size of the random walk step is equal to 1% of the objective function domain, and macro-ruggedness, $FEM_{0.1}$, where the maximum size of the random walk step is equal to 10% of the objective function domain. Pseudocode for the progressive random walk, as well as the FEM calculations, can be found in [81].

The value of FEM is continuous and falls in the $[0, 1]$ range, where 0 indicates a smooth landscape (no entropy), and 1 indicates maximal ruggedness (highest entropy). Thus, FEM can be used to quantify ruggedness/smoothness of a fitness landscape. FEM was shown to be a good predictor of PSO performance in the continuous domain [80, 81].

Neutrality

Two neutrality metrics[1] for continuous spaces were proposed in [134]. The metrics are based on progressive random walks, and operate on overlapping three-point structures that make up the walks. A three-point structure is a set of three steps, $\mathcal{S} = \{\mathbf{x}_{l-1}, \mathbf{x}_l, \mathbf{x}_{l+1}\}$, with corresponding fitness values $\{g_{l-1}, g_l, g_{l+1}\}$. $\mathcal{S}$ is regarded as neutral when the following condition holds:

$$g_{r^{\max}} - g_{r^{\min}} \leq \xi \tag{3.6}$$

where $r^{\max} = \arg\max_r\{g_r\}$, $r^{\min} = \arg\min_r\{g_r\}$, $r \in \{l-1, l, l+1\}$, and ξ is a threshold value. Setting ξ to zero yields strict neutrality, and using a small positive value allows for small variation to also be quantified as neutral. In [134], ξ was set to 1×10^{-8}.

Two neutrality metrics can be extracted from the neighbourhood information provided by a progressive random walk, $\mathcal{W} = \{g_l\}_{l=0}^{L}$:

- M_1: The proportion of neutral structures in $\mathcal{W}$. This measure estimates the degree of neutrality present in the landscape. M_1 is defined as

$$M_1 = \frac{s_{\text{neutral}}}{|\mathcal{W}|} \tag{3.7}$$

[1] Aspects of this section were published as a paper in the proceedings of the IEEE Congress on Evolutionary Computation [134].

where s_{neutral} is the number of neutral 3-point structures in $\mathcal{W}$ (see Equation (3.6)), and $|\mathcal{W}|$ is the total number of 3-point structures in $\mathcal{W}$.

- M_2: The length of the largest neutral subset of $\mathcal{W}$, proportionate to $|\mathcal{W}|$. M_2 estimates the relative size of the largest neutral region, and is defined as

$$M_2 = \frac{|\omega_{\max}|}{|\mathcal{W}|} \tag{3.8}$$

where $\omega_{\max}$ is the longest sub-walk of $\mathcal{W}$ consisting of neutral 3-point objects only.

Both measures produce an output in the range $[0, 1]$, where 0 indicates that the landscape contains no neutral regions (as per the granularity of the algorithm parameters), and 1 indicates a completely flat landscape where all solutions lie on a single fitness level. The fitness values in $\mathcal{W}$ are normalised to the $[0, 1]$ interval prior to the $M1$ and $M2$ calculation to ensure that the same ξ value used in Equation (3.6) can be used across problems of varied fitness scale. Pseudocode for the generation of $M1$ and $M2$ can be found in [134].

Gradients

To quantify the fitness change magnitudes, i.e. gradients, Malan and Engelbrecht [80] proposed two gradient measures:

- The average estimated gradient, G_{avg}, and
- the standard deviation of the gradient, G_{dev}.

G_{avg} and G_{dev} are calculated based on Manhattan progressive random walk [76] samples of the search space. G_{avg} is defined as

$$G_{avg} = \frac{\sum_{l=0}^{L-1} |grad(l)|}{L}$$

where L is the number of steps in the Manhattan progressive random walk, and $grad(l)$ is defined as:

$$grad(l) = \frac{\Delta e_l}{d(\mathbf{x}_l, \mathbf{x}_{l+1})}$$

where Δe_l is the difference between the fitness values of two points $\mathbf{x}_l$ and $\mathbf{x}_{l+1}$, which define step l of the random walk, and $d(\mathbf{x}_l, \mathbf{x}_{l+1})$ is the Euclidean distance between $\mathbf{x}_l$ and $\mathbf{x}_{l+1}$. Since the Manhattan progressive random walk is suggested as the sampling strategy of choice, the Euclidean distance $d(\mathbf{x}_l, \mathbf{x}_{l+1})$ can be replaced by the fixed step size used for sampling. This simplification makes the calculation of $grad(l)$ computationally inexpensive. The absolute value of $grad(l)$ is used in the G_{avg} calculation, since Δe_l can be either positive or negative (the error can increase or decrease). The aim of G_{avg} is to quantify the magnitude of changes rather than their direction, thus the sign of $grad(l)$ is of no consequence. Positive values of $grad(l)$ are also required to ensure that the negative fitness slopes do not cancel out the positive fitness slopes.

The standard deviation of the gradient G_{dev} is defined as:

$$G_{dev} = \sqrt{\frac{\sum_{l=0}^{L-1}(G_{avg} - |g(l)|)^2}{L-1}}$$

The G_{avg} metric quantifies the mean magnitude of change in fitness values, while G_{dev} represents the corresponding standard deviation. Pseudocode for the Manhattan progressive random walk, as well as G_{avg} and G_{dev} calculations, can be found in [81].

Both the G_{avg} and the G_{dev} metrics produce a single output value based on multiple walks. Morgan and Gallagher [94] proposed analysis of the length scale distribution, where the definition of length scale corresponds to the definition of $grad(l)$. The entropy of the length scale distribution was suggested as a potentially useful metric to categorise continuous and combinatorial problems [94].

3.4.3 Separability

An FLA metric to quantify variable interactions in continuous spaces, i.e. separability, was proposed by Sun et al. [128]. The metric is called maximum entropic epistasis (MEE), and is closely related to the maximal information coefficient (MIC), a statistical measure of interaction between two variables, proposed in [110]. MEE constructs a matrix that captures the relationship between every two distinct variables in a problem. For every two decision variables v and u, MIC is calculated between v and the partial derivative of the fitness function with respect to u. The matrix of MIC values is then used to

derive three measures: the degree of direct variable interaction (DDVI), the degree of indirect variable interaction (DIVI), and the degree of variable interaction (DVI) [128]. The MEE metric was shown to give robust and accurate results on a wide selection of continuous problems.

3.4.4 Searchability

Searchability is arguably the hardest metric to estimate accurately [96], since searchability, i.e. hardness of a problem depends on the optimisation algorithm to a very large extent. However, common landscape characteristics such as the presence of the global structure have been identified as generic hardness predictors. The FLA metrics discussed in this section attempt to capture such generic landscape properties.

Fitness distance correlation

The fitness distance correlation (FDC) metric, proposed by Jones and Forrest [60], is designed to quantify global problem hardness. FDC estimates the global shape of the fitness landscape by calculating the covariance between the fitness of a solution and its distance to the nearest optimum. FDC requires knowledge of the global optima locations.

FDC_s, proposed in [78], is an adaptation of FDC for continuous landscapes with unknown optima. FDC_s is defined as:

$$\text{FDC}_s = \frac{\sum_{l=1}^{L}(g_l - \overline{g})(d_l - \overline{d})}{\sqrt{\sum_{l=1}^{L}(g_l - \overline{g})^2}\sqrt{\sum_{l=1}^{L}(d_l - \overline{d})^2}}$$

where L is the size of a uniform search space sample with associated fitness values $\mathcal{G} = \{g_1, ..., g_n\}$; $\overline{g}$ is the mean of $\mathcal{G}$, d_l is the Euclidean distance from $\mathbf{x}_l$ to the point in the sample with the highest fitness value, and $\overline{d}$ is the mean of all d_l.

FDC_s generates values in the range $[-1, 1]$. For minimisation problems, a value close to 1 indicates a highly searchable landscape, i.e. the closer the sample points are to the fittest point, the higher their fitness values are. A value close to 0 indicates a lack of information in the landscape, i.e. points both far from the fittest point and close to the fittest point may have similar fitness values. A negative FDC_s value indicates

a "deceptive" search landscape, i.e. approaching the fittest point in the sample may produce points of increasingly worse fitness.

Note that FDC_s uses random uniform samples of the search space rather than samples gathered by an optimisation algorithm or a random walk. The main advantage of random samples over samples gathered along a trajectory is that the random samples are independent of the optimisation algorithm. Thus, the information gathered from the randomised samples is less biased, and provides a more general view of the fitness landscape characteristics.

Information landscape negative searchability measure

Another measure of problem hardness was proposed by Borenstein and Poli [11]. In [11], an "information landscape" of a problem is generated by taking a random sample of the search space and performing pairwise fitness comparisons between the sampled points. To evaluate the amount and quality of information in the given landscape, the difference between the information landscape of the problem and the information landscape of an "optimal" landscape is calculated.

Malan and Engelbrecht [78] proposed an information landscape negative searchability (IL_{ns}) measure based on the Borenstein and Poli [11] approach. In [78], a random sample R is generated, and its information landscape is calculated. The spherical function is chosen as the "optimal" landscape, as it remains robustly searchable when scaled to higher dimensions. The spherical function is shifted such that its minimum coincides with the best solution found in R. The difference between the information landscape of the problem on sample R and the information landscape of the spherical function on sample R is reported as the IL_{ns} value.

IL_{ns} essentially measures the distance of the given fitness landscape from the spherical function fitness landscape of the same dimensionality. IL_{ns} is bounded to $[0, 1]$, where a value of 0 indicates maximum search information (no difference between the optimal landscape and the actual landscape), and a value of 1 indicates poor quality and quantity of information.

3.5 Loss Surfaces of Neural Networks

NN training is the process of finding the best possible combination of weights that connect the neurons between layers. Each unique combination of weights can be treated as a candidate solution that represents the mapping between the inputs and the outputs. Thus, given m weights and biases, the search space is a continuous m-dimensional space of all possible weight combinations. The complete search space of all possible NN weight vectors with corresponding error values constitutes the error landscape, or loss surface of a NN.

NNs have been studied and successfully used in numerous practical applications for decades [10, 35], yet the landscape properties of the loss functions associated with supervised NN training are still poorly understood [24]. The inherent high dimensionality of NNs prevents intuitive visualisation, making NNs a classic example of "black box" optimisation.

NN error landscapes have been studied before in an attempt to better understand the inner workings of a NN. This section summarises the main existing insights into the nature of NN error landscapes. Known NN error landscape properties are grouped into the four broad categories identified in Section 3.2: modality, structure, separability, and searchability. The provided list is not exhaustive, and is meant to outline the main directions of the existing research.

3.5.1 Modality

Many studies of local minima in NNs were carried out on the XOR (exclusive-or) problem. XOR is a simple, but a linearly non-separable problem that can be solved by a feed forward NN with at least two hidden neurons. As such, XOR is often used to analyse the basic properties of NNs. Studies of the XOR error landscape are especially interesting, because researchers have arrived at somewhat contradictory conclusions. Hamey [48] has claimed that the NN error surface associated with XOR has no local minima. A year later, Sprinkhuizen-Kuyper et al. [123, 124] have shown that stationary points are present in the XOR NN search space, but that the stationary points are in fact saddle points. A more recent study of the XOR error surface was published by Mehta et al. [86],

where techniques developed for potential energy landscapes were used to quantify local minima of the XOR problem under a varied number of hidden neurons and regularisation coefficient values. Mehta et al. [86] have shown that the XOR problem exhibits local minima, and that the number of local minima grows with the increase in the size of the hidden layer.

Further theoretical analysis performed for more complex problems than XOR has highlighted the fact that saddle points are more prevalent in high-dimensional spaces than local minima, and that the number of local minima decreases with an increase in the problem dimensionality [23, 28]. Counter examples have also been published, artificially constructing problems with difficult local minima that can potentially trap the training algorithm [130]. Current understanding of the stationary points in NN error surfaces remains incomplete, partially due to the lack of empirical evidence and intuitive visualisations.

The discovery of the prevalence of saddle points in NN error landscapes has led researchers to question the nature of the basins of attraction associated with the stationary points [114]. It has been observed that NN error landscapes are comprised of wide and narrow valleys, and that the solutions discovered at the bottom of such valleys may have different generalisation behaviour [21, 39, 65]. It has also been observed that it may be possible to find a path of non-increasing error value that connects any two valleys, thus indicating that the valleys may all be part of a single manifold, or attraction basin, with no fitness barriers between them [34].

3.5.2 Structure

Gallagher [39] used techniques such as principal component analysis to simplify error landscape representation in order to visualise NN error landscapes. It was determined that error landscapes have many flat areas with sudden cliffs or ravines. The presence of valleys that stretch into infinity is widely accepted as the most prominent structural feature of NN error landscapes [21, 39, 40, 68, 65]. Denker et al. [31] have described NN error landscapes as having a "sombrero" shape, symmetrically warped about the origin, with solutions of poor quality at zero, and multiple valleys and ridges radiating in various direction. Kordos and Duch [68] referred to similar properties as the "starfish"

structure.

An important characteristic to note here is the symmetry about the origin, studied by Chen et al. [22]. Error landscape symmetry results from two decision variable transformations that have no effect on the fitness of a NN:

1. **Hidden neuron permutation**: If the incoming and outgoing weights of two hidden neurons located in the same layer are swapped, the output of the NN will not be affected. Similarly, any permutation of the hidden neurons, without changing the order of the associated weights, will yield identical fitness values. Thus, larger hidden layers will yield increased redundancy in the error landscape due to a larger number of available permutations.

2. **Sign flip transformation**: If the signs of all the incoming weights of a single hidden neuron are flipped, and the signs of the output weights of this neuron are also flipped, the NN output will remain unchanged. The redundancy in the landscape will increase linearly with an increase in the hidden layer size.

Thus, the NN structure is known to contain ravines and valleys radiating from the origin, and is also known to be highly redundant (symmetric). Studies have been published claiming that the solutions found at the bottom of wider valleys generalise better than the solutions found at the bottom of narrow valleys [21, 65]. Other studies have provided counter examples, artificially constructing sharp minima with good generalisation properties [32, 64]. Therefore, the relationship between the NN error landscape structure and NN generalisation performance is not yet fully understood.

3.5.3 Separabilty

NNs are known to be highly non-separable due to their interconnected structure. While the neurons within a particular layer typically do not communicate, the successive layers are often fully-connected, meaning that each neuron in the preceding layer contributes to each neuron in the successive layer. Despite the obvious non-separability, techniques that rely on subdividing the NN into smaller sub-problems have achieved moderate success in the past [135]. Separability of NNs falls outside the scope of this work.

3.5.4 Searchability

NNs were shown to be highly non-convex [28], yet highly searchable. The apparent lack of high error local minima [121] in deep networks was argued to be the reason for the success of gradient-based algorithms on very high-dimensional NN problems. Chen et al. [22] provided theoretical evidence that the error landscapes contain a spherical throng of optima of similar quality, which makes NN optimisation possible even in very high-dimensional spaces. The increase in problem dimensionality makes the throng more dense [22], which increases the probability of locating a good solution. However, the generalisation performance of the available minima has not been systematically evaluated.

3.5.5 Hyperparameters that influence the neural network error landscapes

Even though some properties of the NN error landscapes have been identified, the relationship between various NN hyperparameters and the resulting error surface remains an open problem [24]. Kordos and Duch [68] provided an overview of the hyperparameters that influence the NN error landscapes, and used PCA projections to study the effect of changing the hyperparameter values on the resulting error landscapes. Observations made in [68] can be summarised as follows:

1. **NN architecture**: The complexity of the error landscape was reported to increase as more hidden layers were added to the architecture. NNs with more than one hidden layer were reported to contain multiple high-laying plateaus that could trap an optimisation algorithm more easily. Overparametrisation, i.e. excessive weights, were shown to induce flatness.

2. **Training dataset**: More complex and less separable datasets yielded more complex and "crinkled" error landscapes, with a larger number of ravines. Balanced class representation was associated with more symmetric error landscapes.

3. **Activation functions**: Only sigmoidal and non-monotone functions such as sine were considered in [68]. Activation functions were reported to have a strong influence on the error landscapes. Non-monotone functions yielded numerous local

minima, and monotone sigmoidal functions yielded step-like geometry with sudden transitions between levels.

4. **Loss functions**: SSE was compared to CE, and CE was observed to yield a more complex error surface. Weight decay regularisation was shown to impose a parabolic shape on the error landscape. Different exponents of SSE were tested, with the conclusion that higher exponents reduced weight growth and thus performed implicit regularisation.

Though the observations made in [68] are highly valuable, they are mostly based on visualisations, and thus do not provide a numerical comparison between the various hyperparameter settings. Terms used to describe error landscapes, such as "more complex" or "less complex", are somewhat vague, and do not provide a hands-on guide for algorithm design or hyperparameter tuning. Additionally, certain important aspects, such as the generalisation properties of the error landscapes, are not covered.

3.6 Summary

This chapter provided an overview of the FLA field. In general, the properties of fitness landscapes can be grouped into four main categories: modality, structure, separability, and searchability. These four categories were discussed in the FLA and the NN contexts. Although general properties of NN error landscapes have been described in the literature, the link between NN hyperparameters and the corresponding error landscape features has not been systematically investigated.

The next chapter presents a sensitivity study of the FLA metrics in relation to the chosen search space boundaries.

Chapter 4

Search Space Boundaries

An important difference between NN training and many other real-world continuous optimisation problems is that the NN search space is unbounded. Even though the weights are usually initialised in a small region around the origin [73], the weights may take on any values during the course of training. How can such a search space be sampled in a representative way? If random sampling techniques are used to estimate the FLA error landscape properties, what part of the infinite search space should be considered? This chapter attempts to answer the above questions by studying the FLA properties of a selection of NN error surfaces under different search space boundaries[1].

The rest of the chapter is structured as follows: Section 4.1 discusses the problem of choosing the search space boundaries for the purpose of applying FLA to NNs. Section 4.2 describes the experiment conducted to study the sensitivity of FLA metrics to search space boundaries. Section 4.3 presents the results of the experiment, and Section 4.4 concludes the chapter.

4.1 Unbounded Landscapes of Neural Networks

Error landscapes of NNs are in many ways similar to fitness landscapes of continuous optimisation problems, but there is an important difference: NN error landscapes are

[1]Aspects of this chapter were published as a paper in the Proceedings of the IEEE Symposium Series on Computational Intelligence [12].

unbounded, whereas optimisation problems usually have bounded decision variables. NN weights do not have any meaning by themselves, and can have any value in $\mathbb{R}$, as opposed to decision variables in optimisation problems that relate to some limited resource in the real world.

The unbounded search spaces of NNs pose a problem to FLA, since the sampling algorithms used by FLA metrics, such as the progressive random walk [79] and the Manhattan random walk [80], require knowledge of the minimum and maximum values per dimension. A range also needs to be specified to generate a random sample for the FDC_s and the IL_{ns} metrics discussed in Section 3.4.

No such range is defined for NNs. It is known, however, that NN weights are usually initialised in a small range around zero [73]. One reason behind choosing a small range is the avoidance of preliminary saturation. Indeed, if very large weights are used, the weighted sum of inputs is likely to have a large magnitude, causing the bounded activation functions to output near-asymptotic values. Saturated units make gradient descent learning slow and inefficient due to small derivative values near the asymptotes [44]. It was also shown that non-gradient descent learning can be hindered by NN saturation [109].

Therefore, it is not unreasonable to study the NN error landscapes on a small area around the origin, since that is exactly where the search for a solution begins. The search space, however, is unbounded. Thus, the properties of the error landscape on a larger scale may provide insight into the dynamics of a training algorithm and the complexity of the problem. FLA on large and small subsets of the search space will also demonstrate how FLA metrics scale, and whether the metrics converge to a similar value for different problems when the scale is increased. The purpose of this study was to perform FLA on a selection of NN problems, under various search space boundaries, and to determine the relationship between the FLA metrics and the search space boundaries imposed.

4.2 Experimentation

This section details the experiment conducted to study the sensitivity of FLA metrics to the chosen search space boundaries. Section 4.2.1 outlines the benchmark problems and

the corresponding NN architectures used. Section 4.2.2 lists the FLA metrics considered in this experiment. Section 4.2.3 describes the selection of search space boundaries tested.

4.2.1 Benchmark problems

For the purpose of this thesis, only classification problems were considered. The benchmark problems used throughout this thesis, together with the corresponding NN architectures and sources, are summarised in Appendix A. In this chapter, results obtained for the following four benchmark problems are reported:

1. Iris (4 inputs, 4 hidden neurons, 3 outputs)
2. Diabetes (8 inputs, 8 hidden neurons, 1 output)
3. Glass (9 inputs, 9 hidden neurons, 6 outputs)
4. Heart (32 inputs, 10 hidden neurons, 1 output)

The above four problems were selected for brevity, as a representative subset.

All NNs employed the identity activation function in the input layer, and the sigmoid activation function in the hidden and output layers.

For the purpose of this study, SSE, defined in Equation (2.9) in Section 2.4, was used as the NN error metric. Kordos and Duch [68] have studied the fitness landscapes associated with the SSE and CE loss functions, and concluded that the choice of the loss function does not alter the general "starfish" structure of the NN error landscapes. Therefore, the observations made in this chapter are expected to hold for the CE loss function. The fitness landscape generated by SSE was analysed under a selection of boundaries discussed in the next section.

4.2.2 Fitness landscape analysis metrics

The purpose of this study was to determine the influence of the search space boundaries on the FLA metrics. The following metrics which rely on bounded sampling were considered:

1. Estimated gradients, G_{avg} and G_{dev}. Manhattan progressive random walks of 1000 steps were used to obtain the measures, where the size of each step was equal to 1% of the search space.

2. Ruggedness on micro and macro scale, $FEM_{0.01}$ and $FEM_{0.1}$. Progressive random walks were used to obtain the measures, where the maximum size of the random walk step was equal to 1% of the search space for $FEM_{0.01}$, and 10% for $FEM_{0.1}$. Each random walk consisted of 1000 steps.

3. Fitness distance correlation, FDC_s. Uniform random samples of the search space were used to obtain the FDC_s measure, and each sample contained 1000 points.

4. Information landscape negative searchability measure, IL_{ns}. Uniform random samples of the search space were used to obtain the IL_{ns} measure, each sample contained 1000 points.

A detailed description of each metric was provided in Section 3.4. Malan and Engelbrecht [79] suggested that the ideal number of walks to perform is equal to 2^m, where m is the dimensionality of the problem. However, Malan and Engelbrecht [79] argued that obtaining 2^m samples can become infeasible for high-dimensional problems, and is unnecessary to obtain robust estimates. Therefore, this study set the number of walks/samples to be one order of magnitude greater than the dimension of the problem, $10 \times m$.

Note that this study is not the first to apply FDC in the NN context. Gallagher [38] used FDC to estimate the searchability of NN error landscapes. In [38], a "teacher" NN was randomly generated, and a "student" NN was subsequently trained to replicate the teacher. Thus, the weight configuration of the "teacher" represented the true global minimum of the search space. This study applies FDC_s to a selection of real-world datasets instead, without the explicit knowledge of the global minimum.

4.2.3 Search space boundaries

From the SSE perspective, the search space is infinite. However, from the perspective of a training algorithm, only a subspace of the infinite search space is ever traversed. Therefore, the part of the search space actually visited by a training algorithm has the

most practical significance. The question is, what part of the search space does a training algorithm typically visit, and how much does the visited sub-space vary per problem and per algorithm?

According to [73], the interval defined by $[-fanin^{-1/2}, fanin^{-1/2}]$, where $fanin$ is the number of connections leading into a neuron, is a good interval for weight initialisation, since the net input signal is dampened proportionally to the total number of inputs. For the architectures listed in Table A.1, $fanin^{-1/2}$ varies from 0.036 (MNIST) to 0.58 (XOR). The larger the architecture, the smaller the weight initialisation range will be, and vice versa, but in all cases, $fanin^{-1/2} \leq 1$.

The weights are thus typically initialised within the $[-1, 1]$ interval. Do the training algorithms ever leave this interval? The easiest way to determine this is to train a NN and observe the resulting distribution of the weights. Figure 4.1 illustrates the weight distributions after 1000 iterations of 30 runs of the stochastic backpropagation algorithm with a learning rate of 0.1 and a momentum of 0.9 on the four problems listed in Section 4.2.1. All weights were initialised in the corresponding $[-fanin^{-1/2}, fanin^{-1/2}]$ intervals, but, as Figure 4.1 shows, the final weights lay within the $[-15, 15]$ interval, a significantly wider interval than the initial interval. Thus, the initial $[-1, 1]$ interval is not representative enough for the purposes of FLA.

Backpropagation is indeed not the only training algorithm used in practice. In [107], a selection of PSO algorithms were used to train NNs, and it was shown that a non-regularised PSO produces weights in the $[-200, 200]$ interval for the Iris problem. It was also shown in [136] that PSO tends to diverge on NN training problems, producing very large weights. Perhaps the shape of the error landscape is one of the reasons behind such divergent behaviour.

Another important property of NN error landscapes is their inherent symmetry, discussed in Section 3.5.2. Various permutations of hidden neurons in a layer yield identical NN models [22]. Flipping the signs of all the incoming and outgoing weights of a single neuron will also leave the neuron's output unchanged [22]. Reduction of the search space to an asymmetric subspace can thus yield a less redundant search subspace, potentially easier to search.

Based on the insights above, a selection of intervals for random sampling for the

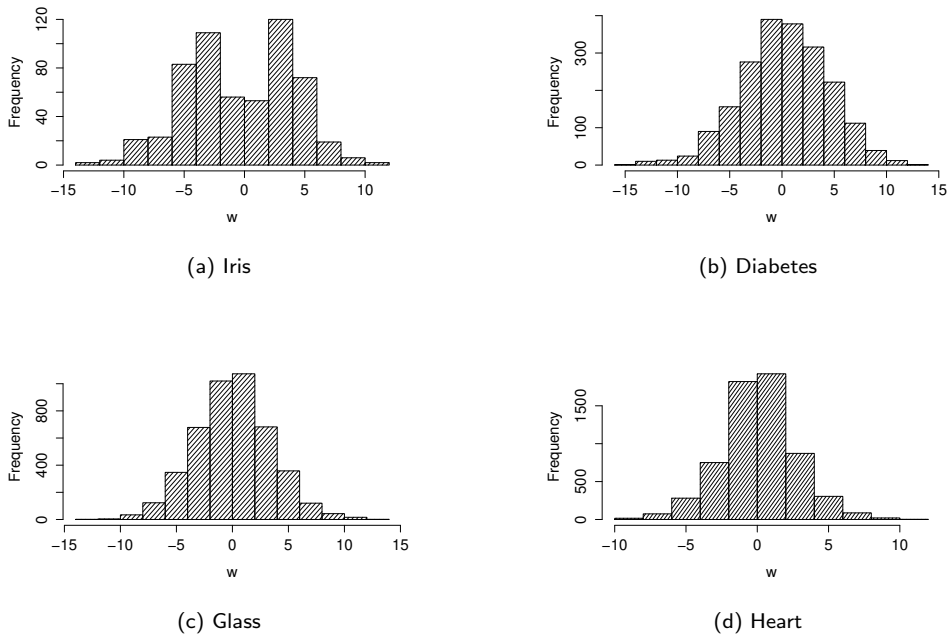


Figure 4.1: Frequency diagrams of the weight values obtained during NN training

FLA metrics, both symmetric and asymmetric about the origin, was chosen for this experiment. The intervals used were $[-N, N], N \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$, and $[0, N], N \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$.

4.3 Experimental Results

Experimental results obtained under various search space boundaries for the four problems considered are presented in this section. The six FLA metrics listed in Section 4.2.2 were used to analyse the NN error landscapes with respect to SSE. Section 4.3.1 discusses the gradient measures. Section 4.3.2 discusses the ruggedness measures. Section 4.3.3 discusses the searchability measures. All reported results are averages over 30 independent runs.

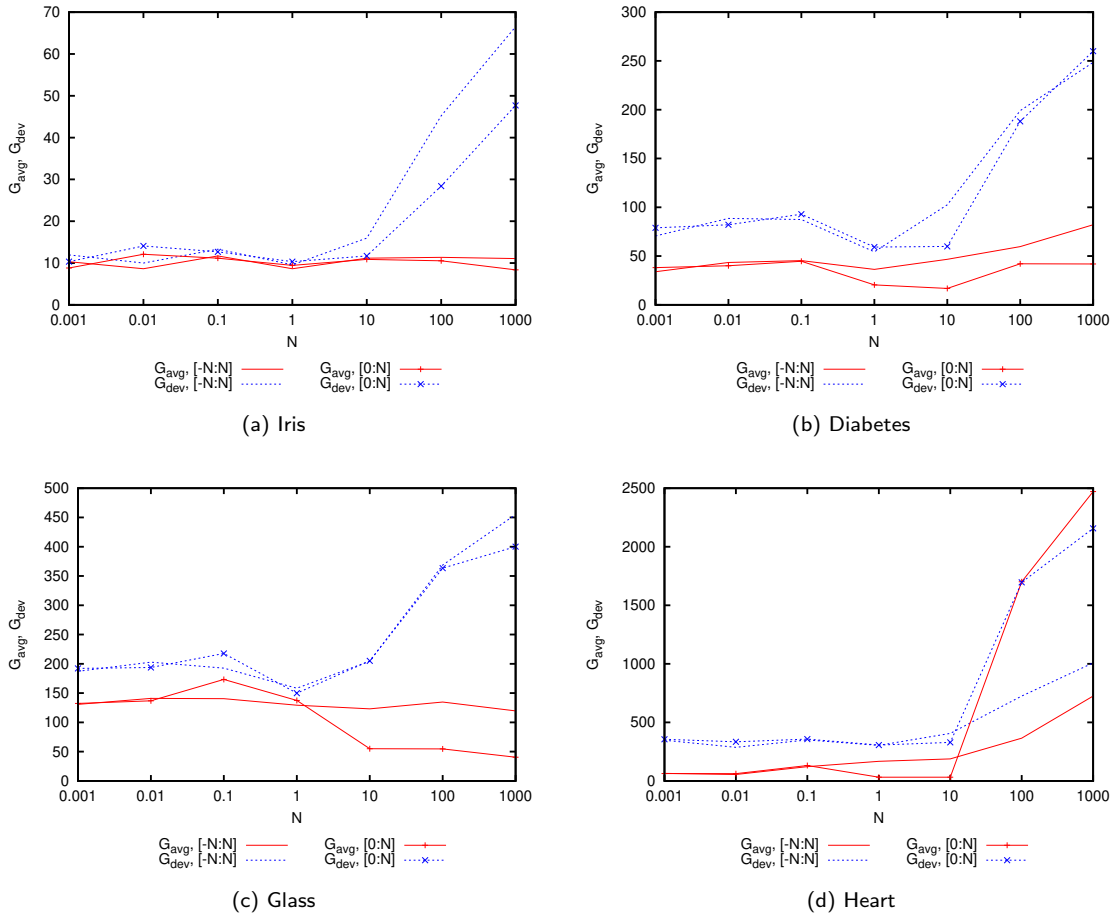


Figure 4.2: Gradient measures obtained under various search space boundaries.

4.3.1 Gradients

Figure 4.2 shows the G_{avg} and G_{dev} values obtained under different search space boundaries. The first notable feature of the gradient metrics is that even inside the smallest bounds ($[-0.001, 0.001], [0, 0.001]$), reasonably large G_{avg} and G_{dev} were obtained for all problems. This result can be explained by the presence of a staircase-like, or "layered" structure of the error landscape, with sudden jumps from one layer to the next, as previously described in the literature [23, 39, 57]. The fact that very small intervals yielded high gradients implies that the "layered" structure was not a result of neuron saturation, since saturation could not occur for such small weights.

The magnitude of gradients increased with an increase in problem dimensionality.

Thus, the jumps between layers became more and more drastic as the dimensionality grew. This observation corresponds to the previously made observations that increasing dimensionality in NNs increases the number of saddle points surrounded by high error plateaus in the error landscapes [28]. This observation is also confirmed by the growing gap between G_{avg} and G_{dev}, where $G_{dev} > G_{avg}$, associated with the dimensionality increase, as shown in Figure 4.2. Malan [81] theorised that $G_{dev} >> G_{avg}$ is indicative of landscapes with step-like, sudden fitness changes.

The gradient measures remained mostly consistent for symmetric and asymmetric bounds alike for all $N \leq 1$. This is in fact the recommended weight initialisation range. Gradient behaviour changed as the boundaries widened: G_{dev} went steadily upwards for all problems considered. The corresponding G_{avg} remained stable (Iris, Glass) or also increased (Diabetes, Heart). Wider boundaries used for the gradient metrics caused the step size of the random walks to become larger, too. For the estimation of gradients, Manhattan random walks were used: the maximum step size was fixed to 1% of the search space, and at every step, only one randomly chosen dimension was incremented or decremented by the given step. Thus, G_{avg} and G_{dev} measures provided an estimate of how rapidly the gradients changed in reaction to a change in a single random dimension. As the boundaries increased, G_{dev} increased faster than G_{avg}, indicating high variance in G_{avg}. Thus, bigger steps through the search space, even if made in only one dimension (i.e. weight), were likely to change the fitness of a solution drastically. This effect is further enhanced in higher-dimensional NNs, due to stronger gradients exhibited, as shown in Figure 4.2.

Asymmetric bounds yielded similar or lower gradients than the symmetric bounds on all problems for $N \leq 10$, which is attributed to the fact that a smaller subspace of the search space was considered. However, the results were not consistent on problems of higher dimensionality. For the Glass problem (150 weights), G_{avg} evidently decreased on large asymmetric regions. This behaviour is attributed to a higher degree of neutrality, or plateaus, due to saturation. Van Aardt et al. [134] showed that neutrality does indeed increase when the NN error landscapes are sampled using search space boundaries with $N \geq 10$. Indeed, if all weights are positive and potentially large ($\forall w \in [0, 1000]$), the likelihood of a large net input signal is higher, resulting in a higher degree of saturation.

For the Heart problem (341 weights), both G_{avg} and G_{dev} yielded much higher values on large asymmetric regions than on the corresponding symmetric regions, indicating that stronger and less consistent gradients were observed for positive-only weights. Thus, asymmetric regions yielded inconsistent results across problems, and were more data-sensitive than the symmetric regions.

4.3.2 Ruggedness

Figure 4.3 shows the FEM values obtained under different search space boundaries. For all problems considered, $FEM_{0.01}$ and $FEM_{0.1}$ were within the range $[0.2, 0.3]$ for all $N \leq 0.1$, indicating mostly non-rugged, consistent landscapes. Van Aardt et al. [134] confirmed that for $N \leq 0.1$, a high degree of neutrality was detected for NN error landscapes. However, Figure 4.3 shows that there was a sharp transition from low to high macro-ruggedness as N increased from 0.1 to 1. For all problems considered, macro-ruggedness on the symmetric $[-1, 1]$ region exceeded 0.5, indicating a change from mostly uniform to mostly rugged. Corresponding asymmetric regions did not exhibit an increase in $\text{FEM}_{0.1}$, or exhibited a less drastic increase. Thus, weights with absolute values between $[0.1, 1]$ constituted a search landscape with high variability of fitness values. However, if the search space consisted of positive weights only, the variability was greatly decreased. Therefore, asymmetric regions likely contained less information for NN training.

For all problems considered, both $FEM_{0.01}$ and $FEM_{0.1}$ increased as the symmetric search space widened, and $FEM_{0.1}$ produced larger values than $FEM_{0.01}$ at all times. Thus, the error landscapes were relatively smooth and consistent on the micro scale ($FEM_{0.01}$), but rather rugged on the macro scale ($FEM_{0.1}$). This observation is attributed to the aforementioned layered structure of the NN error landscapes: little change is observed on a given "level", but a transition from one level to the next represents a significant change in fitness.

FEM characteristics of the symmetric regions were more consistent than that of the asymmetric regions. The instability of FEM observed on the asymmetric regions indicates that the asymmetric regions did not provide a good representation of the error landscapes, and often contained little information to guide an optimisation algorithm.

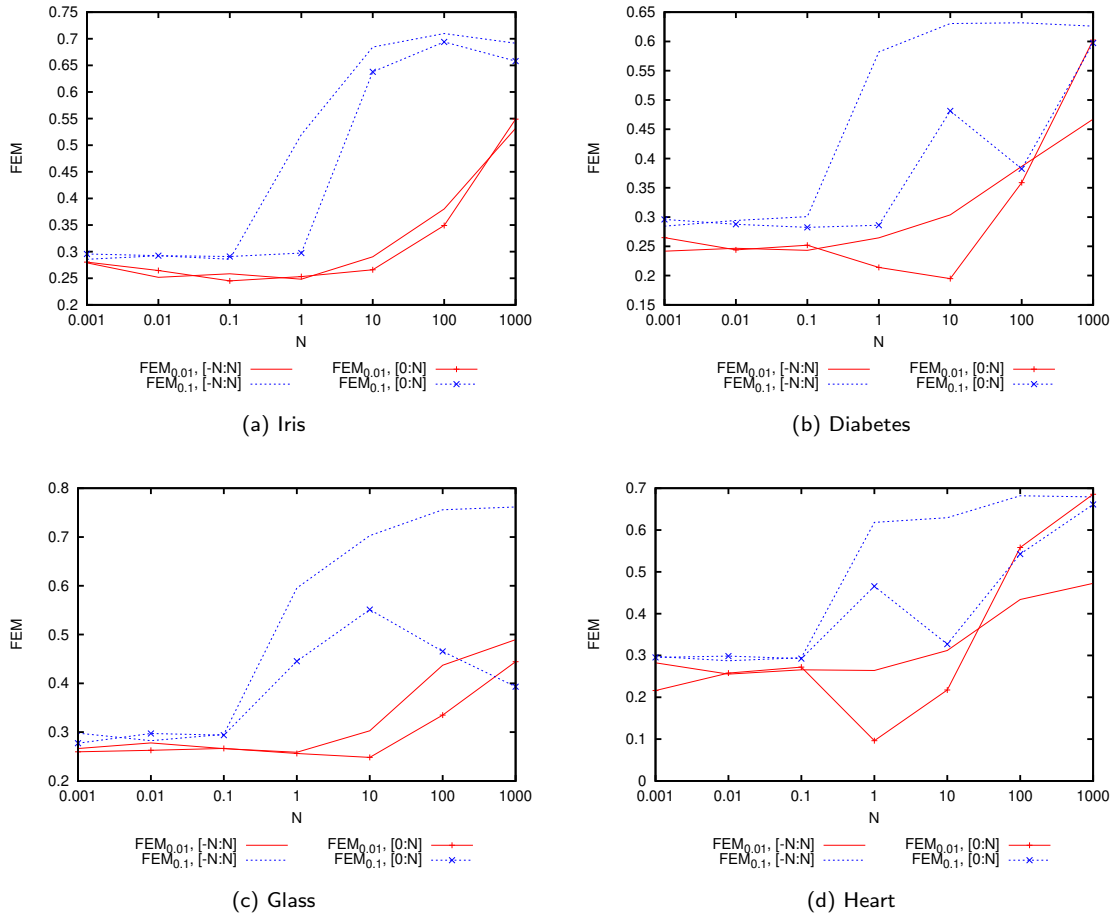


Figure 4.3: FEM measures obtained under various search space boundaries.

Shifting the search space region to the positive weights implied that some of the random walks started from the origin, while symmetric regions guaranteed that the random walks started on the outer boundaries of the selected regions. Inconsistencies in the asymmetric FEM values indicate that the view of the search space as seen from the origin is quite different from the view as seen from the boundaries of the search space. It would be interesting to design a training algorithm that starts the search at the boundaries, and gravitates towards the origin, somewhat similar to NN weight regularisation. Implementation of such a learning strategy is left for future research.

4.3.3 Searchability

Figure 4.4 shows the FDC_s and IL_{ns} measures obtained under different search space boundaries. IL_{ns} consistently increased as the boundaries increased, indicating that wider search spaces contained less and/or poorer information to guide the search. This applied to both symmetric and asymmetric bounds. Indeed, a wider search space implies larger weights, and larger weights imply a higher degree of saturation, while saturated regions of the search space are known to be hard to search. The amount of information quantified by IL_{ns} also decreased as the dimensionality of the problem increased, rightfully indicating that higher-dimensional problems were harder to search.

FDC_s, similar to IL_{ns}, identified higher-dimensional problems as less searchable, which is to be expected. FDC_s decreased as the boundaries increased, and on most problems the transition from $N = 0.1$ to $N = 1$ yielded a drastic drop in searchability. It was previously observed that the same transition yielded a drastic increase in $FEM_{0.1}$ values. Thus, higher fitness variability corresponded to poorer searchability, and the regions identified as "more searchable" by the FDC_s were simply more flat. The FDC analysis performed in [38] with the teacher-student approach also indicated that flat plateaus form an integral part of NN error landscapes.

Symmetric regions were quantified as less searchable than the asymmetric regions by FDC_s. Indeed, the asymmetric regions were less redundant, and thus contained less variability. For higher-dimensional problems (Glass, Diabetes), the FDC_s measurements of the asymmetric regions strongly correlated with the corresponding gradient measures shown in Figure 4.2. Lower gradients resulted in lower searchability, while steep gradients were associated with high searchability. Indeed, an absence of gradients would leave a training algorithm without the information necessary to guide the search.

4.4 Conclusion

This study investigated the behaviour of various FLA metrics on NN search spaces under different boundaries. All FLA metrics used in this study exhibited a sensitivity to the boundaries chosen. NNs generate complex error landscapes, and the properties of landscapes observed on a small area around the origin do not apply to the entire

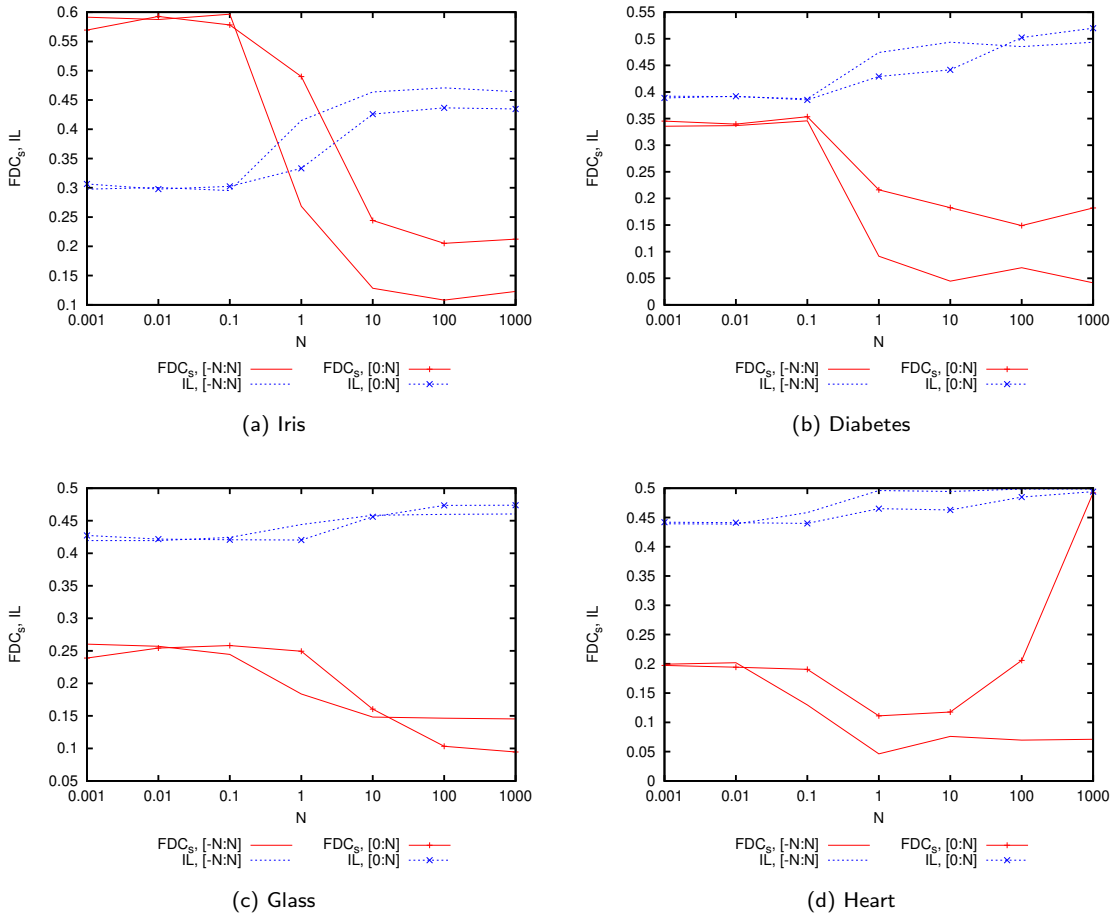


Figure 4.4: FDC_s and IL_{ns} measures obtained under various search space boundaries.

unbounded search space.

High gradient values were obtained on both small and large search subspaces, indicating that steep gradients constitute an inherent NN error landscape property. Gradient magnitudes increased with an increase in problem dimensionality. An increase in search space boundaries increased the variance of gradients, indicating that away from the origin, the step-like jumps between plateaus become more and more drastic.

According to the ruggedness metric, the NN error landscapes exhibited very little variation in the $[-0.1, 0.1]$ region, but the entropy increased drastically for weights with absolute values within $[0.1, 1]$. Thus, the $[0.1, 1]$ range contained information potentially useful to an optimisation algorithm, and can be employed for weight initialisation.

For increased search space boundaries, the micro-ruggedness increased slower than the macro-ruggedness, indicating the relevant consistency for small steps through the search space, and more drastic changes for larger steps.

Searchability metrics indicated a decrease in searchability associated with the increase of search space boundaries and dimensionality. Asymmetric regions appeared less steep, less rugged, and more searchable than the symmetric regions. This behaviour is attributed to a higher saturation degree exhibited by positive weights, as well as a lower level of optima redundancy due to symmetry about the origin [22].

The FLA measures for NN error landscapes clearly depend on the search space boundaries chosen. Based on the observations made in this study, a range of regions rather than a single region should be used to perform FLA of NNs. The two suggested regions are the region in which weights are initialised, as well as the region explored by the training algorithm of choice. More regions can be added to gain more insight into the problem.

Larger regions of the search space were classified as highly rugged, with extremely steep gradients and little information to guide a training algorithm. This explains why weight-dampening techniques such as regularisation are so effective. In general, making training algorithms gravitate towards the origin while allowing exploration may prove to be a viable search strategy.

The next chapter presents a case study that uses FLA metrics to analyse the landscape changes induced by the addition of a regularisation term to the NN loss function.

Chapter 5

Case Study: Regularisation

The ability to correctly predict the outputs of input patterns not seen during training is known as the generalisation ability of a NN. A model that cannot generalise has no practical use, therefore maximising the generalisation potential of a NN is a major goal of NN training. A simple, yet effective way to improve the generalisation ability of a NN is to add a weight regularisation term to the objective function [92, 91, 125]. Weight regularisation aims to penalise network complexity by decreasing the rate of weight growth, as well as by driving irrelevant weights to zero. Regularisation was shown to be beneficial in practical NN applications [85, 125]. Therefore, investigating the effect of regularisation on the NN training problem and the associated error landscape is important.

It is easy to understand the regularisation process intuitively: if large and irrelevant weights are penalised, the final model will be more compact. It is, however, harder to imagine the surface of the loss function after a penalty term has been added to it – will the penalty term introduce new optima, or make the function smoother? Will the chosen training algorithm find the problem easier or harder to optimise?

The relationship between the regularisation term and the resulting error landscape is far from trivial [43], especially given the fact that regularisation parameters typically have to be empirically tuned before an improvement in generalisation performance is observed. One way to investigate the relationship between the regularisation parameters and the resulting error surface is to use FLA techniques. FLA provides an easy and convenient

method to quantify and visualise the correlation between the error landscape changes and the chosen regularisation scheme. This chapter[1] applies selected FLA metrics to study the NN error landscapes under the weight elimination regularisation scheme. The obtained results provide interesting insights into the nature of regularised NN error landscapes, give some guidance for the corresponding parameter tuning, and establish the usability of FLA in the NN context.

The rest of the chapter is structured as follows: Section 5.1 discusses weight elimination in NNs. Section 5.2 describes the experimental procedure. Section 5.3 presents the empirical study of NN error landscaps under various weight elimination settings. Section 5.4 summarises the findings and concludes the chapter.

5.1 Neural Network Weight Elimination

Weight elimination, previously discussed in Section 2.6, offers a refined approach to NN regularisation that allows problem-specific parameter tuning. A recent study by Wang et al. [138] provided a theoretical analysis of boundedness and convergence of weight-elimination NNs, and confirmed good generalisation and pruning capabilities of weight elimination. However, there are two parameters that need to be tuned: the regularisation factor, λ, and the weight elimination threshold, w_0, as defined in Equations (2.16) and (2.18) in Section 2.6. This study analyses the relationship between different settings of these two parameters and the corresponding NN error landscapes. A sensible parameter optimisation range for λ and w_0 is proposed. It should be noted that weight elimination was chosen based on its relative simplicity. Even though other more complex regularisation schemes have also been proposed in the literature, an investigation of the relationship between the penalty function and the error function must begin at the most interpretable point. While weight decay, discussed in Section 2.6, is perhaps too trivial to provide interesting insights, and simply imposes a quadratic convex shape on the error function, weight elimination, with its two tunable parameters, is harder to visualise intuitively [138]. It was also shown that modern regularisation techniques benefit when

[1]The contents of this chapter have been published as an article in the *Neural Processing Letters* journal [13].

combined with simpler quadratic penalty functions [125]. Thus, the results of this study can be extended and applied to numerous recently proposed regularisation schemes.

5.2 Experimental Procedure

The aim of the experiments was to apply FLA metrics to regularised NN error landscapes, and to observe the influence of regularisation parameters on both error landscape characteristics and training algorithm performance. Thus, insight into the regularised NN error landscapes can be gained, and the expressiveness of FLA metrics in the NN context can be evaluated.

The rest of the section is structured as follows: Section 5.2.1 outlines the benchmark problems used in this study, Section 5.2.2 lists the FLA metrics used, Section 5.2.3 describes the chosen search space boundaries, Section 5.2.4 lists the regularisation parameter settings investigated in this study, Section 5.2.5 describes the NN training algorithm used, and Section 5.2.6 lists the NN training algorithm parameters.

5.2.1 Benchmark problems

For the purpose of this study, the following three classification benchmark problems were considered:

1. Iris (4 inputs, 4 hidden neurons, 3 outputs)
2. Diabetes (8 inputs, 6 hidden neurons, 1 output)
3. Glass (9 inputs, 9 hidden neurons, 6 outputs)

The characteristics of the problems are discussed in Appendix A. The three problems exhibited consistent FLA trends, and therefore were deemed sufficient for the experiment. All NNs employed the identity activation function in the input layer, and the sigmoid activation function in the hidden and output layers. For the purpose of this study, SSE, defined in Equation (2.9) in Section 2.4, was used as the NN error metric.

5.2.2 Fitness landscape analysis metrics

The purpose of this study was to determine the effectiveness of FLA to characterise NN error landscapes under various regularisation parameter settings. The following metrics, discussed in Section 3.4, were considered:

1. Estimated gradients, G_{avg} and G_{dev}. Manhattan progressive random walks of 1000 steps were used to obtain the measures, where the size of each step was equal to 1% of the search space.

2. Ruggedness on the micro and macro scale, $FEM_{0.01}$ and $FEM_{0.1}$. Progressive random walks were used to obtain the measures, where the maximum size of the random walk step was equal to 1% of the sampling range for $FEM_{0.01}$, and 10% for $FEM_{0.1}$. Each random walk consisted of 1000 steps.

3. Fitness distance correlation, FDC_s. The information landscape negative searchability measure, IL_{ns}, was not considered, since the results in Chapter 4 indicated that IL_{ns} and FDC_s provide similar searchability estimates. Uniform random samples of the search space were used to obtain the FDC_s measure, where each sample contained 1000 points.

The number of walks/samples per experiment was set to $10 \times m$, where m is the dimensionality of the problem. Averages of 30 independent experiments are reported.

5.2.3 Search space boundaries

NN weights are defined to be any numbers in $\mathbb{R}$, thus sensible boundaries had to be chosen for the sampling to take place, as discussed in Chapter 4. For this study, three search space boundary settings were considered: $[-0.5, 0.5], [-1, 1]$, and $[-5, 5]$. These regions correspond to the typical weight initialisation area [73], as well as the areas where a search algorithm may find an acceptable solution [12]. The three different boundary settings yielded similar FLA results, therefore only $[-1, 1]$ results are reported in this chapter.

5.2.4 Regularisation parameters

The success of weight elimination is heavily dependent on the regularisation parameters λ and w_0 (see Equations (2.16) and (2.18)), which are usually chosen empirically [140]. FLA offers an intuitive way to visualise the effects that the regularisation parameters have on the resulting error landscape. To study these effects, different combinations of λ and w_0 had to be considered. Previous studies have shown that w_0 of order unity is usually a good choice [141], and that small values of λ tend to give better results [107], because λ significantly larger than 1 causes the error to be dominated by the penalty function. This study considered all combinations of λ and w_0 values listed in Table 5.1 for each problem.

Table 5.1: Values of λ and w_0 considered in this study.

	Discrete value range
λ	$\{1\times10^{-6}, 5\times10^{-6}, 1\times10^{-5}, 5\times10^{-5}, 1\times10^{-4}, 5\times10^{-4},$ $1\times10^{-3}, 5\times10^{-3}, 1\times10^{-2}, 5\times10^{-2}, 0.1, 0.5, 1\}$
w_0	$\{1\times10^{-6}, 5\times10^{-6}, 1\times10^{-5}, 5\times10^{-5}, 1\times10^{-4}, 5\times10^{-4},$ $1\times10^{-3}, 5\times10^{-3}, 1\times10^{-2}, 5\times10^{-2}, 0.1, 0.5, 1, 2, 5\}$

5.2.5 Neural network training

Characterisation of error landscapes is not very useful unless insight into the nature of the problem is provided that can aid the training process. In addition to studying the error landscapes of regularised NNs, this study investigates the relationship between NN training algorithm performance and the FLA characteristics of the problem.

Backpropagation (BP), discussed in Section 2.3, was used in this study. BP uses gradient descent to iteratively adjust NN weights and biases in the direction of the negative gradient of the objective function. Weight regularisation is applied to a NN by incorporating the desired penalty term in the gradient calculations.

5.2.6 Training algorithm parameters

To investigate the relationship between NN training algorithm performance and the FLA characteristics of the problem, the corresponding training algorithm parameters had to be optimised to ensure that the algorithm performed adequately. An iterative approach to algorithm parameter optimisation was used. Algorithm parameters were optimised one at a time. For each parameter, the algorithm was tested under a selected range of possible values for this parameter, while the other parameters remained fixed. In order to keep the optimisation process statistically sound, 30 independent runs were conducted for every value in the chosen discrete range. The parameter value yielding the lowest average generalisation error for the current parameter being optimised was subsequently chosen, and optimisation proceeded to the next parameter. For optimisation of the remaining parameters, all the parameters already optimised were fixed to their best values. This approach to parameter optimisation is sensitive to the order in which the parameters are tuned, and varying the order of parameter tuning is likely to have an effect on the resulting algorithm performance. It should be noted that the focus of the study is on investigating the relationship between the error landscape characteristics and regularisation parameters, rather than algorithm performance. Thus, adequate performance rather than optimal performance is sufficient for the purpose of this study.

For stochastic BP, the learning rate, η, and momentum, α (discussed in Section 2.3) had to be optimised. Values considered during the optimisation process are listed in Table 5.2. Final parameter values used in the experiments are listed in Table 5.3. Table 5.3 indicates that the problems of higher dimensionality required a larger learning rate (Glass, Diabetes), and sometimes a smaller momentum (Diabetes) than the problem of lower dimensionality (Iris). This observation implies that higher dimensional problems exhibited better overall searchability, allowing for larger step sizes (learning rate), and requiring less gradient fluctuation smoothing (momentum).

Each reported result is an average over 30 independent simulations that ran for 1000 iterations. Datasets were divided into a training set and a generalisation set; 80% of the patterns were randomly chosen to form the training set, and the remaining 20% were used for testing. Test data used to calculate the generalisation errors was not used for parameter optimisation.

Table 5.2: BP parameter values considered in the optimisation process.

	Discrete value range
Learning rate η	$\{0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5\}$
Momentum α	$\{0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5\}$

Table 5.3: Optimised BP parameter values.

	Iris	Glass	Diabetes
Learning rate η	0.1	0.4	0.3
Momentum α	0.9	0.9	0.8

5.3 Experimental Results

The purpose of the experiments is twofold: first, to investigate the influence of the regularisation term on the NN error landscapes under different regularisation parameter settings (in Section 5.3.1); secondly, to observe the training algorithm's response to the error landscape changes induced by weight elimination (in Section 5.3.2).

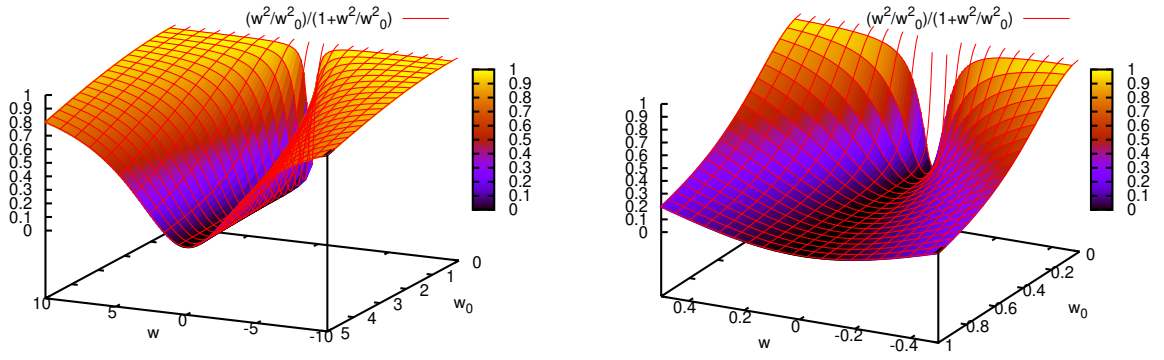
5.3.1 Characterising regularised neural network error landscapes

This section presents an analysis of the relationship between the regularisation term and the NN error landscapes. The effect of λ and w_0 on the average gradients observed in the landscapes is investigated first, followed by a study of ruggedness and the searchability of NN error landscapes under various λ and w_0 values.

Gradients

To understand the impact of the penalty term, consider the weight elimination penalty for a single weight over various values of w_0, illustrated in Figure 5.1. As can be seen from Figure 5.1, the weight elimination term has a clear minimum in one dimension: a weight of zero yields no penalty. The value of w_0 controls the "sharpness" of the minimum. It can be hypothesised that an increase in the value of λ increases the contribution of the penalty term to the objective function, thus "simplifying" the objective function by

adding a global attractor in the form of a global minimum imposed by the penalty term.



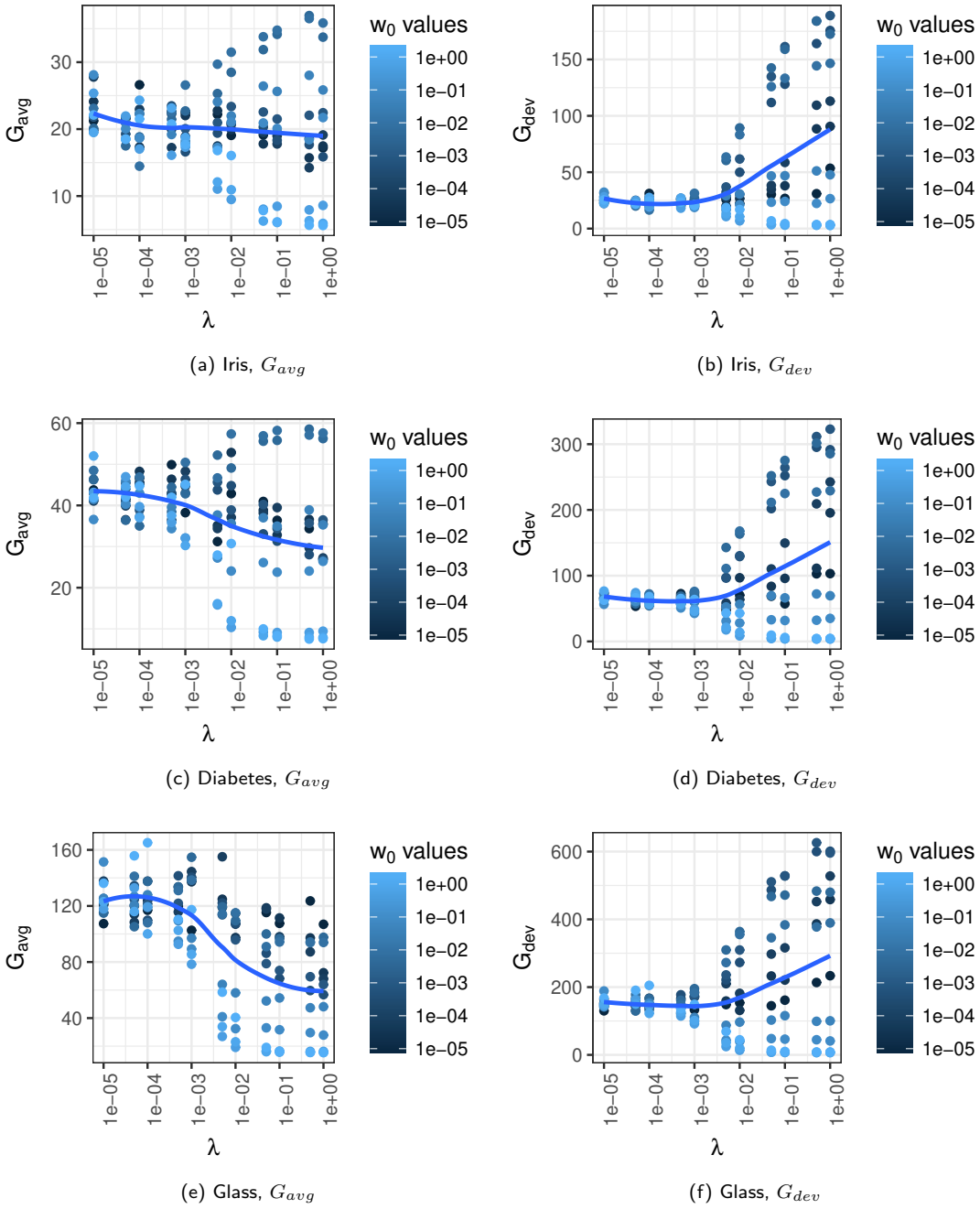
(a) $w \in [-10, 10]$ (b) $w \in [-0.5, 0.5]$

Figure 5.1: Weight elimination function for a single weight, w, and weight elimination threshold, w_0

Figure 5.2 shows the average values of G_{avg} and G_{dev} associated with different combinations of λ and w_0. For interpretability, every scatter plot in Figure 5.2 includes a locally estimated scatterplot smoothing (LOESS) curve [25], representing local polynomial regression.

Across all problems considered, an overall downward trend in G_{avg} was associated with an increase in λ. Indeed, the surface of the penalty function only has one minimum and is otherwise smooth. Therefore, increasing the contribution of the penalty term to the loss function is expected to smooth the error landscape. However, the effect of the penalty strongly depended on the chosen value of w_0: as shown in Figure 5.2, larger values of w_0 indeed yielded smaller G_{avg} gradients for all problems considered. According to Figure 5.1, larger w_0 implies that the imposed minimum is less sharp, thus smaller gradients are to be expected.

In Figure 5.2, the problems are presented in ascending order of dimensionality. Figure 5.2 shows that the average magnitudes of the gradients increased with an increase in problem dimensionality. The same phenomenon was observed in Chapter 4, confirming that higher gradients associated with a higher dimensionality form an inherent property of NN error landscapes. The downward trend in G_{avg} associated with the



(a) Iris, G_{avg} (b) Iris, G_{dev}

(c) Diabetes, G_{avg} (d) Diabetes, G_{dev}

(e) Glass, G_{avg} (f) Glass, G_{dev}

Figure 5.2: G_{avg} and G_{dev} results obtained for different combinations of λ and w_0 on the $[-1, 1]$ interval

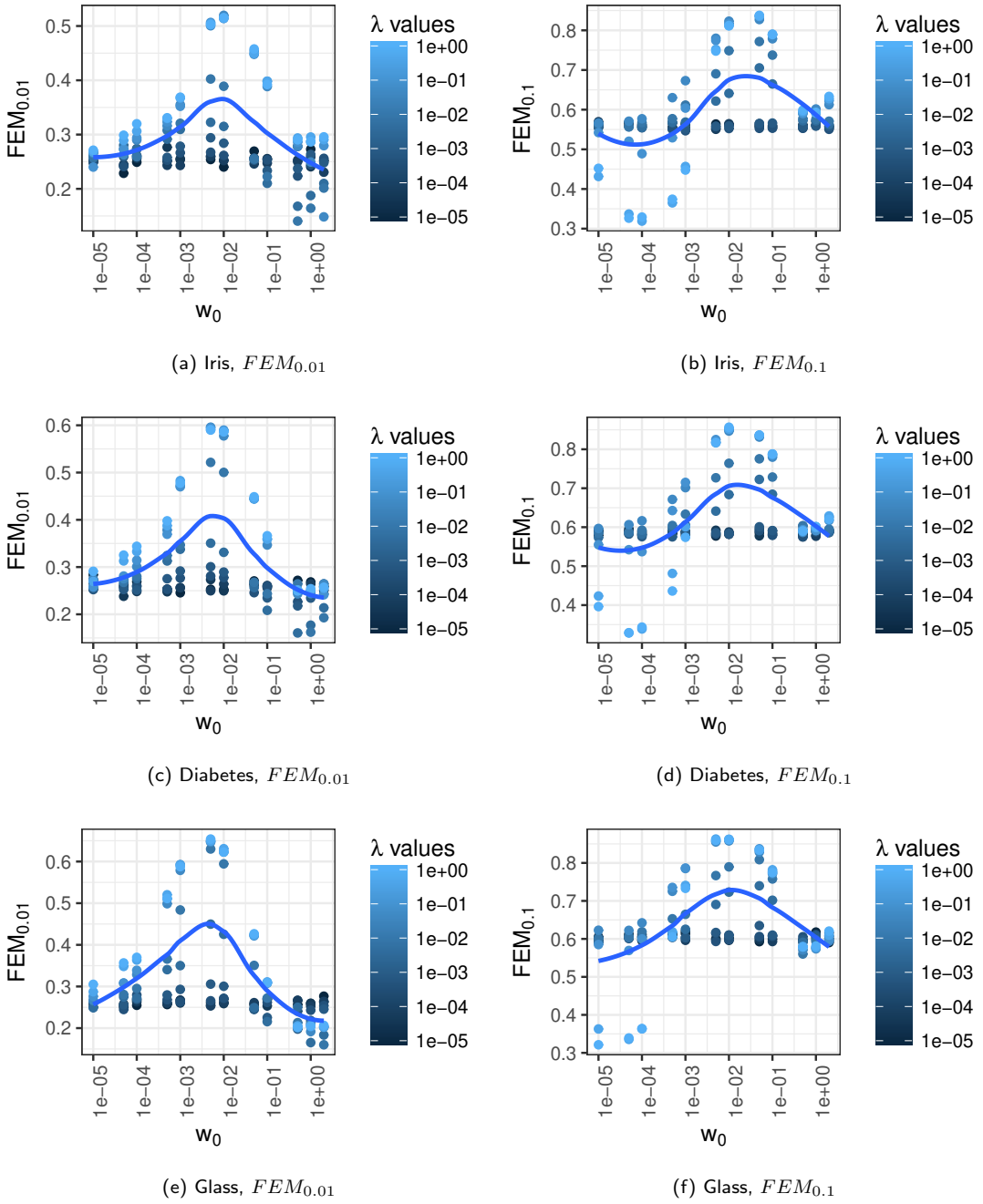
penalty term contribution became more definite with an increase in dimensionality. A high-dimensional fully-connected NN architecture is more likely to have redundant free parameters than a low-dimensional architecture, thus the effect of the penalty term becomes more pronounced in high dimensions.

Figure 5.2 also shows that an increase in λ was associated with an overall upward trend in G_{dev}. In other words, a stronger contribution of the penalty term to the loss function yielded smaller gradients of higher variability. As Figure 5.1 illustrates, weight elimination introduced sharp minima, surrounded by a plateau-like surface. On the plateau, the gradients will be small. Around the minima, however, the fitness value will change rapidly. Thus, a high G_{dev} resulted due to the contrast between the plateaus and the sudden minima. Indeed, a large difference between G_{avg} and G_{dev} is indicative of a step-like landscape with sudden jumps, according to [81]. The hypothesis is further confirmed by observing that the larger values of w_0, as illustrated in Figure 5.2, were not associated with an increase in G_{dev}: higher values of w_0 decreased the sharpness of the introduced minima.

Thus, the introduction of the weight elimination term decreased the overall gradients of the error surface, but added sharp, narrow optima that may not be very easy to find.

First Entropic Measure of Ruggedness

The first entropic measure of ruggedness, FEM, quantifies the change in fitness values based on entropy. Figure 5.3 illustrates how the micro- and macro-ruggedness of the regularised error landscape changed in relation to different values of λ and w_0. Variation in ruggedness for different values of w_0 was only observed for larger values of λ. If λ is too small, the contribution of the penalty term may become negligible. As the value of w_0 increased, so did the ruggedness: small w_0 yielded sharp narrow optima that altered a small part of the search space; increasing w_0 widened the "diameter" of the penalty-induced optima, thus influencing a larger part of the error landscape. The ruggedness began to drop again as w_0 became larger than 0.01: the induced optima gradually "flattened" and became lost among other error landscape fluctuations. When w_0 became larger than the chosen error landscape boundaries, no sampled weights were deemed large enough to be penalised, thus the contribution of the penalty term vanished



(a) Iris, $FEM_{0.01}$

(b) Iris, $FEM_{0.1}$

(c) Diabetes, $FEM_{0.01}$

(d) Diabetes, $FEM_{0.1}$

(e) Glass, $FEM_{0.01}$

(f) Glass, $FEM_{0.1}$

Figure 5.3: $FEM_{0.01}$ and $FEM_{0.1}$ results obtained for different combinations of λ and w_0 on the $[-1, 1]$ interval

altogether.

Macro-ruggedness results, also shown in Figure 5.3, illustrate the same trends as micro-ruggedness, but in a more pronounced manner. Average values of $FEM_{0.1}$ exceeded the corresponding $FEM_{0.01}$ values, indicating that larger step sizes experienced more variation in the landscape. Once again, values of w_0 close to 0.01 induced the most ruggedness across all problems considered.

Entropy is used to estimate the level of ruggedness in fitness landscapes. From the information theory perspective, the amount of entropy can be interpreted as the amount of "information", or variability. Clearly, specific values of w_0 and λ maximised the amount of variability present in the error landscape. The question that remains to be answered is whether this "information" was indeed useful to the training algorithms, and whether the penalty term made the error landscape easier to search.

Fitness Distance Correlation

FDC_s results for different values of λ and w_0 are shown in Figure 5.4. Once again, the effect of the penalty term on the error landscape only became noticeable for larger values of λ. It is evident from Figure 5.4 that FDC_s decreased at first as the value of w_0 increased from 1×10^{-5} to 1×10^{-2}. It was shown in Figure 5.3 that increasing the value of w_0 resulted in increased ruggedness. Ruggedness implies that the fitness value fluctuates instead of persistently going up or down as the landscape is traversed by an algorithm. Increased fluctuations were thus labelled as "less searchable".

After the highest peak of ruggedness was reached at $w_0 \approx 0.01$, and the ruggedness began to decline with an increase in w_0, the landscape was progressively perceived as more and more searchable (FDC_s increased). Highest values of w_0, combined with the highest values of λ, yielded the higest searchability on all problems considered. Thus, only large weights were penalised, and the applied penalty was strong. Therefore, the FDC_s results support the hypothesis that the application of a penalty simplifies the landscape. High values of w_0 combined with high values of λ have also been shown to yield error landscapes with low and consistent gradients, shown in Figure 5.2, which once again confirms the landscape simplification via regularisation.

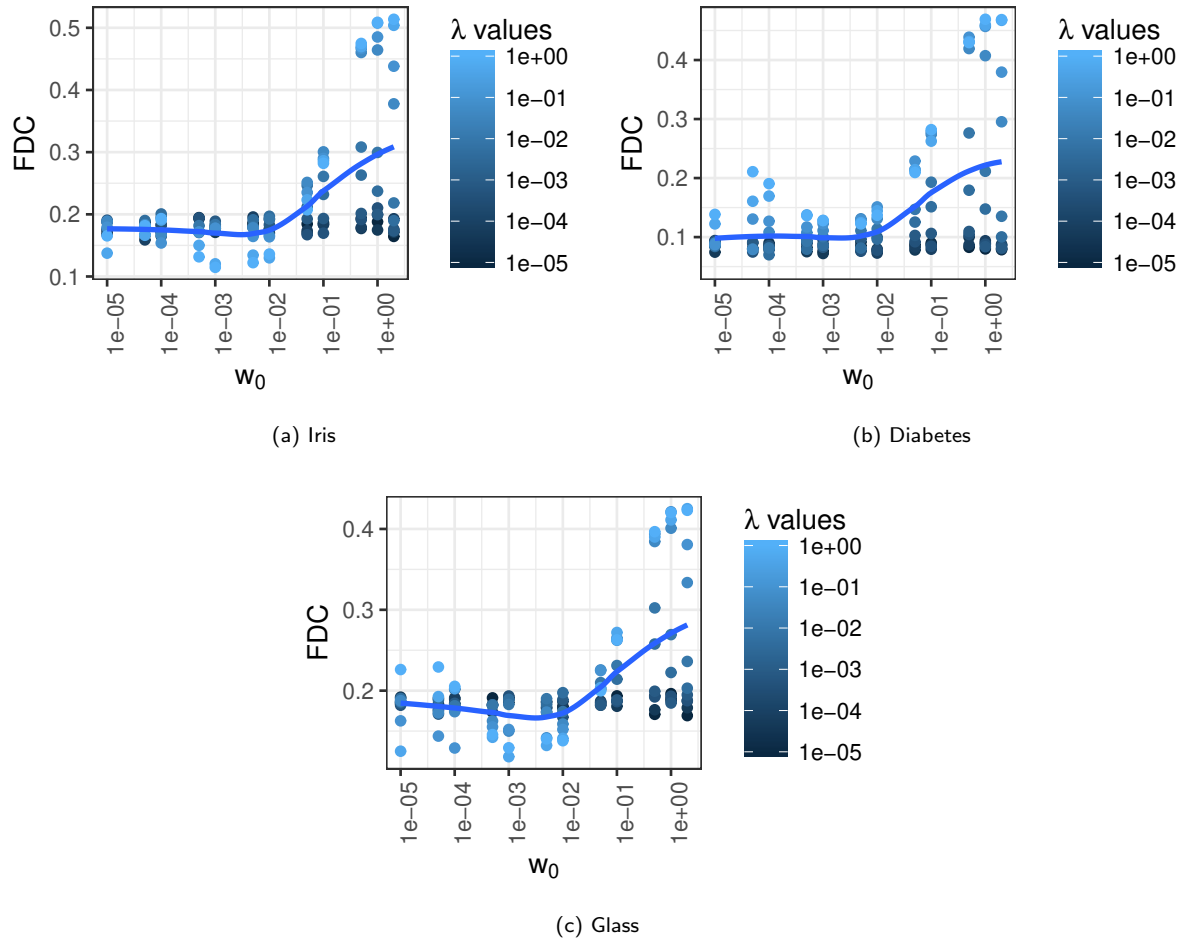


Figure 5.4: FDC_S results obtained for different combinations of λ and w_0 on the $[-1, 1]$ interval

Summary

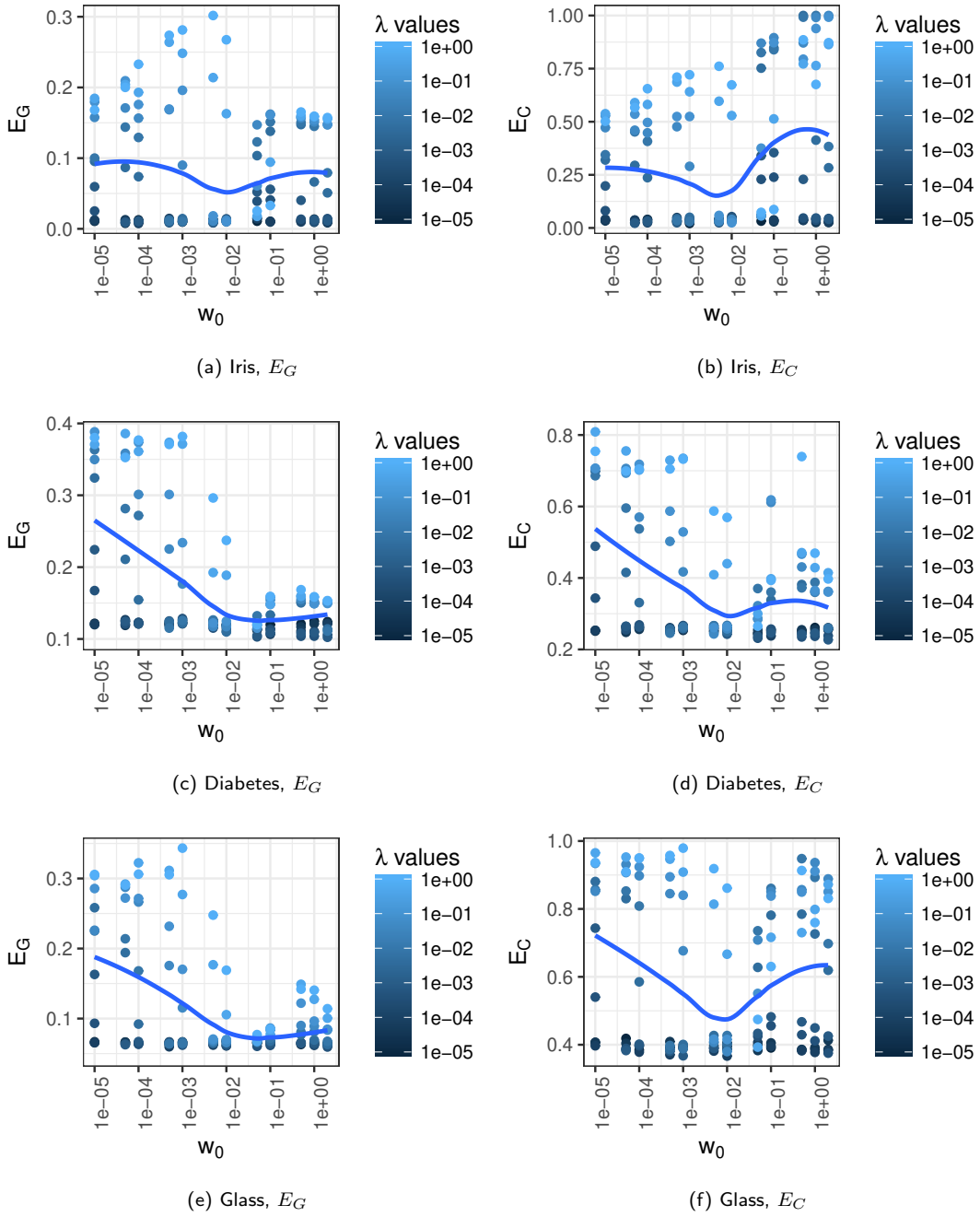
Regularised NN error landscapes were considered in terms of approximate gradients, ruggedness, and searchability. It was observed that the addition of a penalty term had a visible impact on the resulting error landscape, and that properties of the error surface such as gradients and ruggedness can be controlled by tuning the regularisation parameters. The next section puts these observations in the context of NN training.

5.3.2 Fitness landscape analysis and neural network training

Now that it has been established how the penalty term changes NN error landscapes, it is important to understand whether the induced changes make the landscape easier or harder to search for the NN training algorithms. This study considered the classical BP algorithm for NN training, as outlined in Section 5.2.5. No search space boundaries were enforced during training, since NN weights are defined to be any real numbers in $\mathbb{R}$. The goal of the study was to execute an instance of an algorithm on a problem, and to observe any difference in algorithm performance induced by the various combinations of λ and w_0 values. All NN weights were randomly initialised in the $[-0.5, 0.5]$ interval. Algorithm performance was evaluated in terms of the mean squared training error, E_T, the mean squared generalisation error, E_G, and the mean classification error, E_C. Both E_G and E_C were calculated on the test set, which constituted a randomly selected 20% of the data set not used during training or parameter optimisation. If at least one value in the output vector differed from the corresponding target value by more than 0.5, the pattern was labelled as incorrectly classified.

Figure 5.5 summarises the average E_G and E_C values obtained for different values of λ and w_0. Across all problems and both error metrics, high values of λ were associated with inferior performance. An increase in λ implies that the contribution of the penalty term to the objective function becomes stronger. Indeed, if the training algorithm focuses on eliminating the weights rather than minimising the error, the training will produce a minimal architecture that is utterly useless.

The situation looked quite different when observed from the perspective of w_0. On all problems, the smallest values of w_0 yielded poor training and generalisation performance. It has been observed in Section 5.3.1 that low values of w_0 corresponded to error landscapes of low ruggedness and drastic gradient changes due to the nature of the penalty term. BP, being a gradient-descent method, struggled to find a way through the plateaus of the resulting staircase-like error surface. An increase in w_0, that corresponded to an increase in ruggedness, and a decrease in gradient variation, yielded a predictable improvement in BP performance. As w_0 increased further, penalising fewer and fewer weights, the error began to grow again, which is especially evident from the E_C values.



(a) Iris, E_G (b) Iris, E_C

(c) Diabetes, E_G (d) Diabetes, E_C

(e) Glass, E_G (f) Glass, E_C

Figure 5.5: Backpropagation results obtained for different combinations of λ and w_0

What parameter selection guidance can be induced from the above observations? First of all, w_0 in the range $[0.001, 0.01]$ generated the landscapes with most variability,

i.e. information, which resulted in the lowest average E_T, E_G, and E_C values. The authors suggest that values less than or equal to 0.01 are considered for w_0 for low-dimensional problems, and values greater or equal to 0.01 are considered for higher-dimensional problems. Values around 0.01 are likely to produce a sensible result, thus 0.01 can be used as a starting point in the optimisation process.

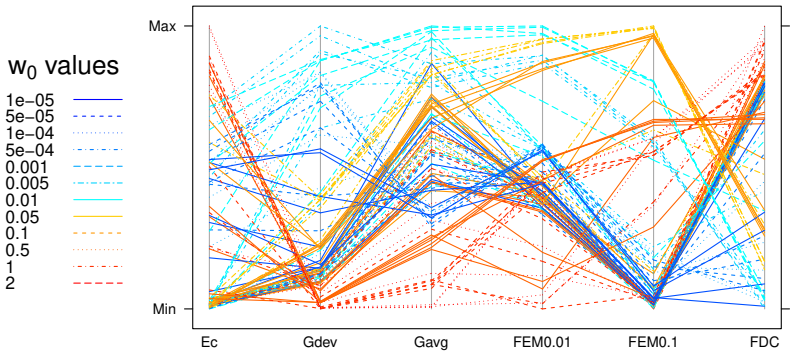
Even though excessively large λ may hinder training by overshadowing the error by the penalty, a value of $\lambda < 0.001$ is not likely to influence the error landscape significantly. Therefore, λ in the range $[0.001, 0.1]$ is suggested to be considered in the parameter optimisation process.

It should also be noted that very low values of λ yielded low error values in some scenarios, especially on lower-dimensional problems. Thus, regularised models should be compared to non-regularised models as a part of the optimisation process, to ensure that regularisation does indeed improve the generalisation performance.

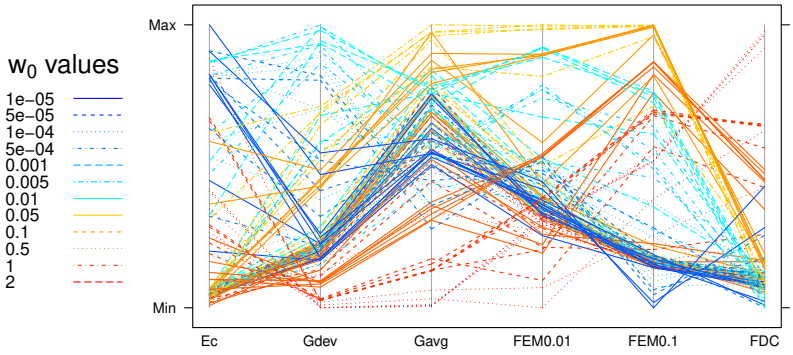
To further visualise the relationship between the obtained FLA measures and the corresponding algorithm performance, parallel coordinate plots are presented in Figure 5.6. Parallel coordinate plots were first proposed by Wegman [139] as a technique to visualise the relationships between various dimensions in high-dimensional spaces. In Figure 5.6, each FLA metric is represented as a parallel coordinate axis, and E_C is used as a metric representing BP performance. Each line represents a combination of averages over 30 simulations of each metric, for a given combination of λ and w_0 values.

Even though BP's performance differed per problem (Figures 5.6a, 5.6b, and 5.6c), some general trends can be observed. For all problems considered, G_{avg} higher than G_{dev} was associated with lower E_C. Consistent but prominent gradients imply a more searchable and cohesive landscape, with enough gradient information to guide BP. Small G_{avg} and G_{dev}, indicative of a fairly flat landscape, yielded poor BP performance, which is to be expected. Mid-range G_{avg} with $G_{dev} >> G_{avg}$ also yielded poor performance, indicating that BP does not perform well on step-like error landscapes with abrupt fitness changes.

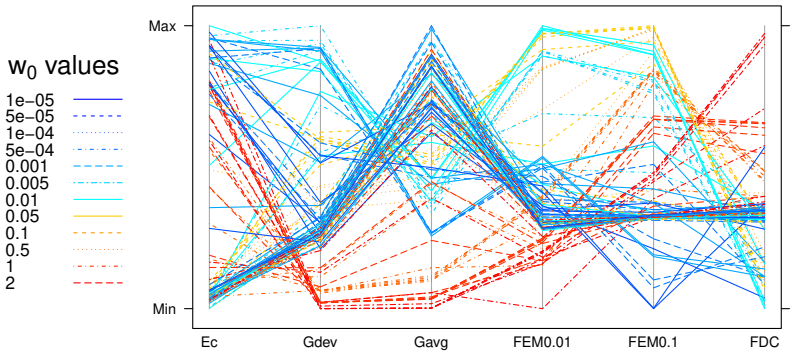
Figure 5.6 shows that high ruggedness did not hinder BP. Good BP performance on highly rugged surfaces indicates that BP may be much more resilient to local minima than previously suspected. These results correlate well with recent theoretical findings



(a) Iris



(b) Diabetes



(c) Glass

Figure 5.6: Parallel coordinate plots for various FLA metrics obtained on the $[-1, 1]$ interval.

showing that the NN error landscapes contain more saddle points than local minima [28], and that the number of local minima reduces exponentially as the dimensionality of the problem increases [23, 28]. The problems considered in this study were not very high-dimensional, yet resilience to potential local minima has already been observed. The modality of NNs is investigated further in Chapters 7, 8, and 9.

Low micro-ruggedness, $FEM_{0.01}$, combined with higher macro-ruggedness, $FEM_{0.1}$, resulted in poor BP performance. Low $FEM_{0.01}$ and high $FEM_{0.1}$ also corresponded to low G_{avg}. All of these properties combined describe landscapes with wide plateaus, with sudden changes observable only on the macro-level. Such landscapes are not very searchable from the gradient descent perspective.

Interestingly, the searchability measure FDC_s provided the least useful and the most misleading information: the highest FDC_s values corresponded to the flattest landscapes with low G_{avg} and G_{dev}. BP struggled to perform well on such landscapes for the lack of gradient information. Low values of FDC_s, on the other hand, corresponded to better BP performance. Perhaps NN error surfaces are too non-trivial to be considered from the "global shape" perspective that FDC_s offers.

5.4 Conclusion

This chapter investigated the applicability of FLA metrics to regularised NN error landscapes. The influence of the weight elimination term on the NN error landscape characteristics was studied. It was observed that the addition of a weight elimination term to the loss function alters the error landscape, and does not necessarily make the error landscape easier to search. Five continuous FLA metrics were used to study the properties of the regularised error surfaces: gradient measures G_{avg} and G_{dev}, ruggedness measures $FEM_{0.01}$ and $FEM_{0.1}$, and the searchability measure FDC_s. Different combinations of regularisation parameters λ and w_0 were used, and the BP training algorithm was considered in the FLA context. FLA was shown to be a useful tool for visualising the properties of NN error landscapes.

The weight elimination term was shown to smooth the error landscape while introducing additional minima. Tuning of the w_0 parameter allows to tune the sharpness of

the introduced minima. Sharper minima result in more drastic, highly varied gradients. Values chosen from the $[0.001, 0.1]$ range for the w_0 parameter maximised the variability, or ruggedness of the landscapes, and yielded the lowest average NN errors. It was shown that the BP algorithm is capable of efficiently searching very rugged landscapes. On the other hand, step-like landscapes with rare and sudden fitness changes rendered BP inefficient.

Very small λ values render regularisation insignificant, while excessively large values of λ overshadow the error by the penalty. Values chosen from the $[0.001, 0.1]$ range resulted in visible error landscape transformations and did not hinder training, provided that the w_0 value was sensible. The necessity to optimise λ can be eliminated by employing a multi-objective algorithm to optimise both the loss function and the weight elimination term separately, and to find a suitable trade-off solution thereof. This is a topic of future research.

The searchability metric, FDC_s, evaluated rugged landscapes as less searchable, even though BP actually benefited from the variability in the landscape. Perhaps NN error surfaces are too complex for the crude "global shape" estimation that FDC_s provides. Out of the five metrics considered, FDC_s produced the least valuable results.

This study only considered weight elimination. It will be interesting to compare weight elimination error landscapes to other regularised error landscapes. FLA can potentially be used to optimise the penalty parameters involved, because FLA metrics provide a handy visualisation tool for the corresponding error landscapes.

Thus, existing FLA metrics were shown to be a usable visualisation tool in the NN context. The next chapter discusses the problem of quantifying local minima and basins of attraction on NN error landscapes. A new sampling technique, as well as a novel visualisation method for the analysis of stationary points and the associated attraction basins, are proposed.

Chapter 6

Modality Quantification

A selection of existing FLA metrics were successfully used in Chapters 4 and 5 to estimate the structural attributes of NN error landscapes such as ruggedness and gradients under various boundary and regularisation settings. However, these techniques do not provide a means to quantify the modality of the NN error landscapes. As previously discussed in Section 3.5.1, the modality of NN error landscapes is poorly understood. Recent studies [21, 39, 65] suggest that NN error landscapes are comprised of wide and narrow valleys, and that the solutions discovered at the bottom of such valleys may have different generalisation behaviour. Studies have also been published stating that there is no bad local minima in high-dimensional NNs, largely due to the nature of high-dimensional spaces [28, 63, 115, 121]. However, counter examples have also been presented, where local minima on NN error landscapes were either constructed artificially [130], or discovered empirically [86]. Thus, a better understanding of local minima and the associated basins of attraction has to be developed.

Existing FLA modality metrics for continuous spaces, previously discussed in Section 3.4.1, rely on iteratively performing multiple instances of a local search, and comparing the Euclidean distances between the best fitness points discovered by the local search instances. An analysis of the distance between the discovered points is then used to estimate the number of local minima. The lengths of the local search trajectories are proposed as a measure to quantify the width of the attraction basins. There are two problems with this approach: Firstly, using the Euclidean distance in high-dimensional

spaces was shown to be misleading [95]. Secondly, using a local search to find minima is limited by the ability of the local search to locate the minima. Stagnation of a local search is not a definite indication that a local minimum has been discovered.

This chapter proposes a new sampling algorithm biased towards the discovery of high fitness points. Based on this sampling algorithm, a new technique to visualise and quantify local minima and other stationary points, together with the associated basins of attraction, is proposed. Section 6.1 discusses the shortfalls of the existing sampling algorithms, and proposes a new adaptive sampling algorithm for NN error landscapes. Section 6.2 proposes loss-gradient clouds, a new visualisation technique for stationary points, and two statistical metrics to quantify the number and width of attraction basins. Section 6.3 concludes the chapter.

6.1 Adaptive Sampling for Neural Network Fitness Landscape Analysis[1]

To quantify the properties of local minima and attraction basins, spatially connected samples of the search space are required that would capture the spatial relationships between individual points in the sample. As discussed in Section 3.3.2, random and progressive random walks can provide such spatially-correlated samples. However, both the simple random walk and the progressive random walk do not take the fitness of the neighbours into account when generating the $l+1$ step of the walk, $\mathbf{x}_{l+1} \in \{\mathbf{x}_l\}_{l=1}^{L}$. Smith et al. [119] have shown that when the distribution of fitness values across the search space is highly skewed towards poor fitness, random sampling may produce an inadequate sample that does not capture enough points of high fitness. Indeed, if the random sample fails to capture any points of good fitness, it will not be possible for the FLA metrics to correctly quantify certain landscape properties such as modality. Thus, an adaptive walk for continuous search spaces is necessary.

Adaptive walks in discrete spaces, discussed in Section 3.3.2, rely on random muta-

[1] Aspects of this section were published as a paper in the Proceedings of the Genetic and Evolutionary Computation Conference Companion [14].

tions of a candidate solution. The same approach can be employed in continuous spaces, thus emulating stochastic hill climbing. Mutations can be performed by adding random noise in one or more dimensions. If the mutated position has a higher fitness than the current position, the mutated position will be added to the walk. However, if the search space is high-dimensional and skewed towards poor fitness areas, such random mutations are likely to not produce neighbours of higher fitness. Thus, stochastic hill climbing in continuous search spaces will be computationally expensive, and may produce very short walks that neither adequately cover the search space, nor find areas of high fitness.

Particle swarm optimisation (PSO) has also been proposed in the past as a sampling method. Each particle in the swarm represents a candidate solution, and the next step of the walk can be defined in terms of the next step of the global best particle in the swarm [81]. There are two problems with this approach: Firstly, PSO sampling is algorithm-specific, and the trajectory will be highly sensitive to algorithm parameters. Secondly, PSO has been shown to exhibit divergent behaviour on NN training [136], which yields this algorithm a suboptimal choice for NN error landscape sampling.

A number of attempts have been made to study the NN error landscapes from the perspective of the gradient descent trajectory [39, 40, 68]. Gradient descent uses the numerical gradient of the error function, thus the fitness is likely to increase per step, provided there is an incline. However, analysis of the gradient descent trajectory is algorithm-specific. Steep gradients combined with the learning rate parameter may induce large steps through the search space, while weak gradients may produce small steps. Thus, the step sizes are bound to be inconsistent, providing an unrealistic view of the search space. Additionally, the lack of stochasticity makes gradient descent unlikely to investigate the areas of poor and average fitness.

This study proposes a randomised gradient sampling technique based on the progressive random walk [79]. The proposed algorithm uses the error function gradient to choose the general direction for each step. The magnitude of the step is randomised within a closed interval per dimension, thus introducing stochasticity. Therefore, the proposed sampling algorithm is biased towards the search space areas that contain high fitness solutions. The added stochasticity makes the sampling general enough to not be algorithm-specific, and allows for the coverage of poor fitness as well as good fitness

areas.

The rest of this section is structured as follows: Section 6.1.1 formally defines the progressive gradient walk for NN error landscapes. Section 6.1.2 presents the empirical study of the proposed gradient walk compared to two random walks commonly used in the FLA literature.

6.1.1 Progressive gradient walk

Gradient information, when available, is clearly the most direct way of performing hill-climbing in a continuous search space. In addition to being a reliable source of direction, the gradient is also more efficient to compute than choosing the best individual in a population. Population-based approaches such as PSO require each individual to be evaluated separately, whereas the gradient is computed once per step. Computational efficiency is an important concern for NNs, since NN search spaces are inherently high-dimensional.

To alleviate gradient descent specificity of the proposed adaptive walk, and to study the error landscape as a whole rather than as an algorithm trajectory, the following approach is proposed:

1. Gradient vector $\mathbf{g}_l$ is calculated for a point $\mathbf{x}_l$.

2. A binary direction mask $\mathbf{b}_l$ is extracted from $\mathbf{g}_l$ as follows:

$$b_{lj} = \begin{cases} 0, & \text{if } g_{lj} < 0 \\ 1, & \text{otherwise} \end{cases}$$

   where $j \in \{1, \ldots, m\}$ for the m-dimensional vector $\mathbf{g}_l$.

3. The progressive random walk algorithm, previously discussed in Section 3.3.2, is used to generate the next step $\mathbf{x}_{l+1}$.

The progressive random walk algorithm requires two parameters to be set: the maximum dimension-wise step size, s, and the boundaries of the search space. The progressive gradient walk requires the same two parameters. The behaviour of the progressive gradient walk with no search space boundaries is also investigated in this study.

6.1.2 Empirical analysis of the gradient walk

The aim of this study is to illustrate that random sampling fails to capture high fitness solutions, and that the proposed progressive gradient walk generates more representative samples than the random walks. The rest of this section details the experiments conducted to illustrate these points.

Dataset

The XOR problem was chosen for the purpose of this study as the simplest classification problem requiring a non-linear solution. The entire dataset is provided in Table 6.1. Despite being a seemingly trivial problem, the XOR is not linearly separable, and generates a complex error landscape that is still not fully understood [48, 86, 123].

Table 6.1: The XOR problem dataset

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Neural network architecture

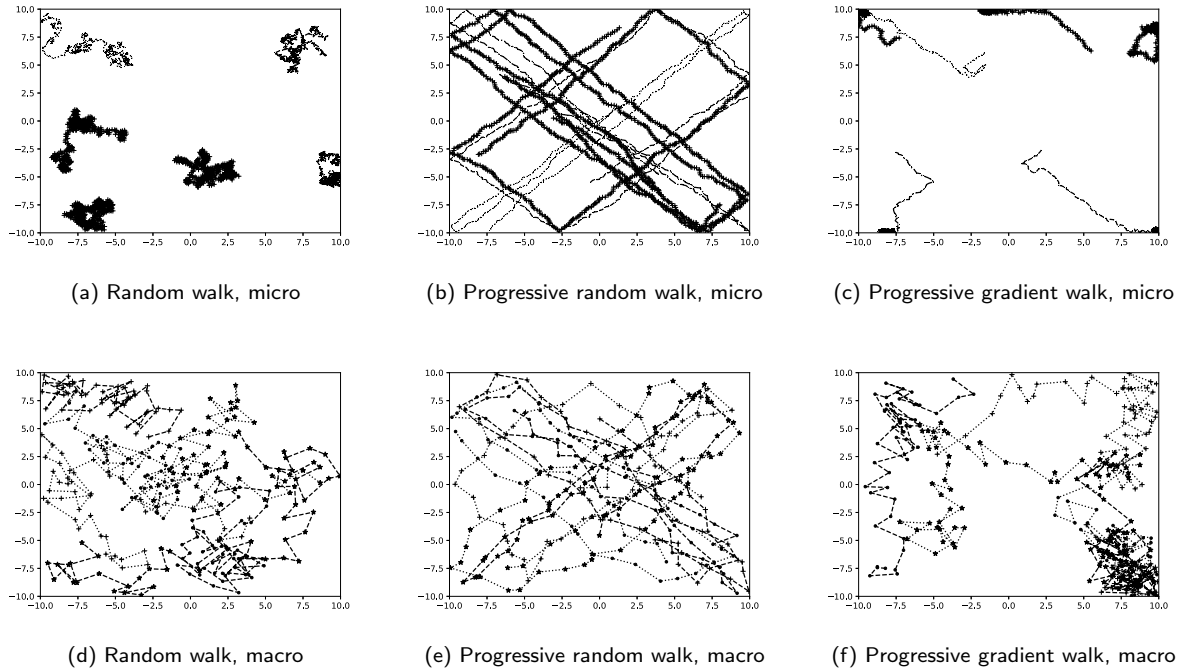
Given the classic XOR problem, a corresponding fully-connected feed-forward NN architecture was chosen. The NN comprised of two input neurons, two hidden neurons, and one output neuron [123]. Bias weights were associated with the hidden and the output neurons. The total number of weights was equal to 9. The sigmoid activation function was employed in the hidden and the output units. The SSE loss, defined in Equation (2.9) in Section 2.4, was used as an error metric to calculate the gradients and the fitness of any given point in the search space.

Experiments

Random sampling is typically performed within some predefined bounds. For the purpose of this study, search space bounds were set to $[-10, 10]$. This range was chosen as the range likely to contain high fitness solutions [12]. Since the granularity of the walk, i.e. the average step size, has a bearing on the resulting FLA metrics [75], two granularity settings were used throughout the experiments: micro, where the maximum step size was set to 1% of the search space, and macro, where the maximum step size was set to 10% of the search space.

To illustrate the basic movement dynamics of the various walks, a sample of points obtained by a random walk, a progressive random walk, and a progressive gradient walk under micro and macro settings are shown in Figure 6.1. Each walk was performed in nine dimensions corresponding to the NN weights. The 2D projections of the first six dimensions of two independent walks are plotted in pairs along the axes, resulting in a total of 6 projections, 3 projections per walk. Each axis corresponds to a weight of the NN, and each 2D projection illustrates the movement dynamics of the walk in a 2-dimensional subset. Micro walks performed 500 steps, and macro walks performed 50 steps. It is evident from Figure 6.1 that the progressive gradient walk was biased compared to the random walks, but did perform a reasonable amount of exploration. Smaller step sizes led to more consistent trajectories (see Figure 6.1c), indicating that the surface was locally smooth. Previous theoretical studies indicated that NN error landscapes are comprised of plateaus and narrow ravines [39, 68]. The consistent direction of movement observed for the progressive gradient walk is attributed to these consistent structures.

To estimate the search space coverage of the three walks, 10 000 points were generated per macro walk, and 100 000 points were generated per micro walk. A total of 100 walks were performed under both micro and macro settings. Micro walks performed 1000 steps each, and macro walks performed 100 steps each. Values across all dimensions were plotted in histograms of 100 equally sized bins each. The resulting histograms are shown in Figure 6.2. Mean and standard deviation values of the samples are also displayed above each histogram. While the random and the progressive random walks covered the search space near-uniformly (mean close to zero), the gradient walk leaned strongly towards the borders of the search space, especially in the micro case. Even

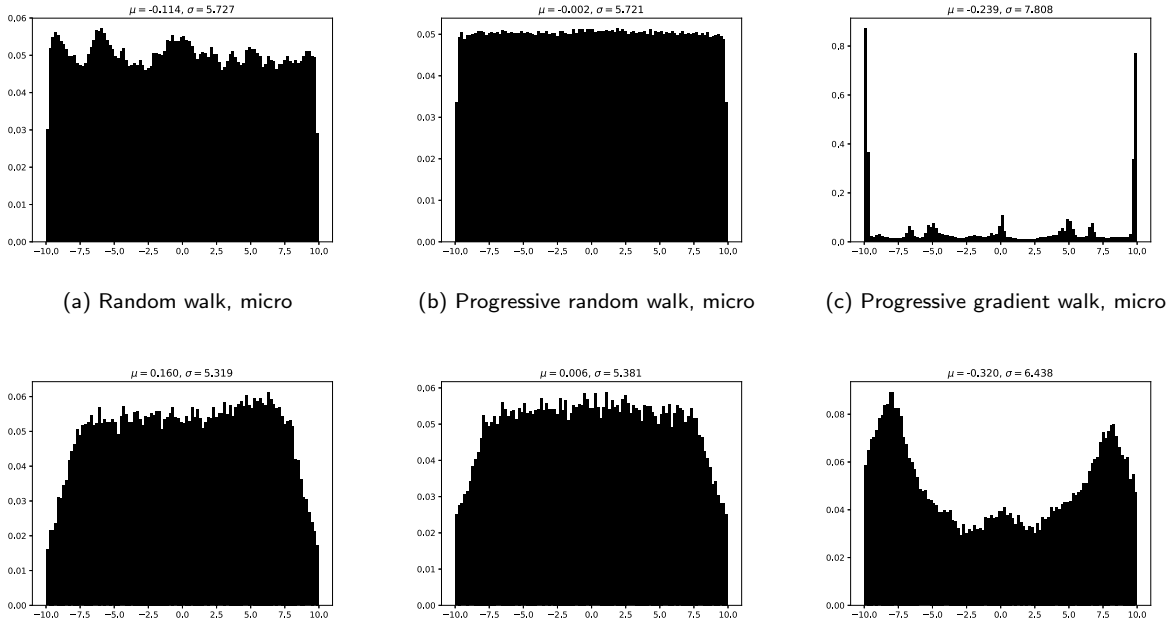


(a) Random walk, micro (b) Progressive random walk, micro (c) Progressive gradient walk, micro

(d) Random walk, macro (e) Progressive random walk, macro (f) Progressive gradient walk, macro

Figure 6.1: Plots of the positions of paired dimensions of sample walks. Micro walks were performed over 500 steps. Macro walks were performed over 50 steps.

though the walks were not allowed to leave the search space, the gradient direction often pointed outwards, thus causing gradient walks to cluster around the boundaries. This observation is in line with previous studies, proposing that NN error landscapes have a "starfish" or "sombrero" structure, with ravines of lower error leading outwards [31, 68]. A question thus needs to be answered: should NN error landscapes be studied within predefined boundaries? Previous studies have argued that the boundaries are necessary, since random sampling can not be performed in unbounded space [12, 79]. However, if progressive gradient sampling is used instead of random sampling, the gradient information should lead the walks to "interesting" areas of the landscape, rather than causing meaningless wandering. Unbounded progressive gradient sampling was performed for the purpose of this study, and the resulting walk samples are given in Figures 6.3a and 6.3d. In both micro and macro settings, gradient walks tended to move away from the origin, once again aligning with the "starfish" structure. Search space coverage histograms for the unbounded gradient walks are shown in Figures 6.3b and 6.3e. Unbounded walks ex-



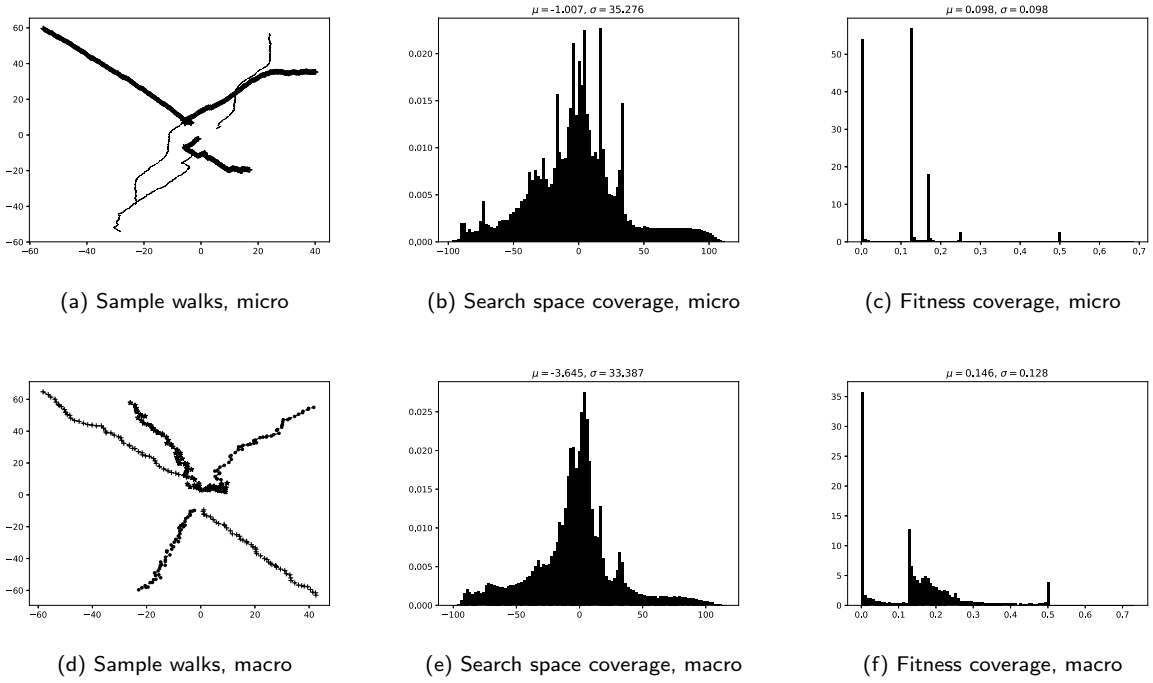
(a) Random walk, micro (b) Progressive random walk, micro (c) Progressive gradient walk, micro

(d) Random walk, macro (e) Progressive random walk, macro (f) Progressive gradient walk, macro

Figure 6.2: Frequency diagrams illustrating search space coverage by the various walks.

hibited less clustering than bounded walks, and covered the search space better. Spikes associated with particular ranges are attributed to the presence of local minima or saddle points that could have trapped the gradient walk.

Progressive gradient walks are suggested as a method of search space sampling that is more likely to find areas of good fitness than the random walks. To estimate the fitness coverage, the fitness frequency distribution of the sample points obtained by each of the walks under both the micro and macro settings were plotted in histograms of 100 equally-sized bins. The resulting histograms are shown in Figure 6.4. The means and standard deviations of each distribution are shown above the histograms. It is evident from Figure 6.4 that both the random and the progressive random walk failed to discover areas of good fitness. For both random sampling techniques, the average SSE was around 0.45, with a very sharp peak on the average value, and a heavy tail on the left, corresponding to areas of above average fitness. The lowest error sampled by the random walks hovered around 0.2. Thus, the random walks sampled mostly average (random guess) fitness areas, and the areas of optimal fitness (near zero) were almost



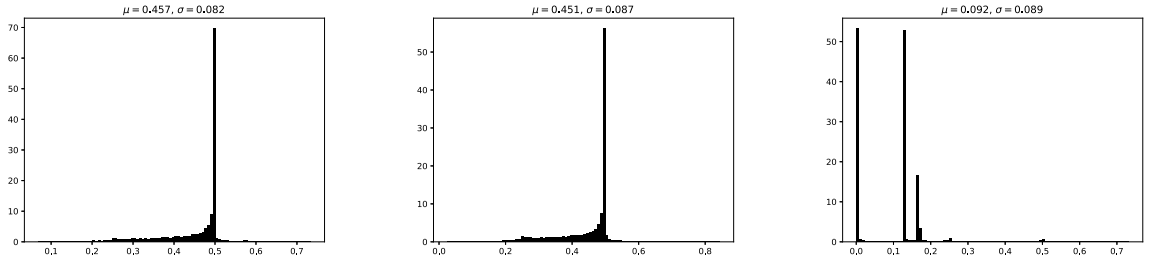
(a) Sample walks, micro (b) Search space coverage, micro (c) Fitness coverage, micro

(d) Sample walks, macro (e) Search space coverage, macro (f) Fitness coverage, macro

Figure 6.3: Unbounded progressive gradient walk.

not sampled at all. Low fitness areas are the least interesting areas from an optimisation algorithm perspective, and an optimisation algorithm is expected to spend the least amount of time in these areas. Thus, more emphasis should be placed on studying the areas of high fitness.

The progressive gradient walk, on the other hand, has successfully captured error values around zero (optimal fitness). Figures 6.4c and 6.4f indicate that the mean error of the gradient walks was below the lowest error of the random walks. Interestingly, both micro and macro gradient walks exhibited peaks around specific error values. This can be an indication of the presence of local minima or saddle points at those fitness values. The macro progressive gradient walk exhibited a good spread of fitness values between 0.0 and 0.5, indicating that macro samples captured information relevant to a potential training algorithm. Fitness frequency histograms were also plotted for the unbounded gradient walks, in Figures 6.3c and 6.3f. Unbounded gradient walks captured a similar distribution of fitness values as bounded gradient walks, exhibiting similar peaks. The peaks can be indicative of the modality of the error landscape.

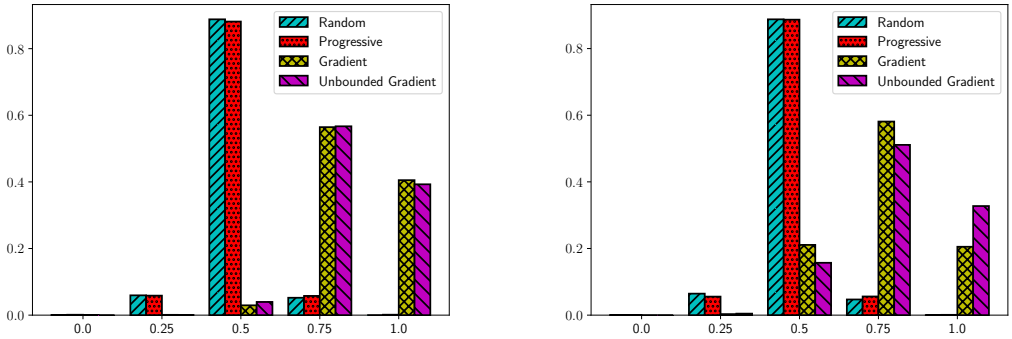


(a) Random walk, micro (b) Progressive random walk, micro (c) Progressive gradient walk, micro

(d) Random walk, macro (e) Progressive random walk, macro (f) Progressive gradient walk, macro

Figure 6.4: Frequency diagrams of the fitness (SSE) associated with the samples obtained by the various walks.

The spread of fitness values was also calculated in terms of classification accuracy. Since the XOR problem has only four data points, the set of possible classification accuracy values is discrete, and is comprised of the following values: $\{0.0, 0.25, 0.5, 0.75, 1\}$, where 0.0 indicates incorrect output for all patterns, and 1.0 indicates correct output for all patterns. A frequency histogram for the various walks under micro and macro setting is shown in Figure 6.5. It is evident from Figure 6.5 that the random walks failed to capture areas of 100% accuracy, and sampled mostly average, or random guess accuracy points instead. The gradient walks, on the other hand, sampled mostly above average accuracy and 100% accuracy. The macro setting yielded a better coverage of average accuracy by the gradient walks. Once again, the gradient walks captured areas of the landscape that are of more interest to a potential optimisation algorithm.



(a) Micro setting

(b) Macro setting

Figure 6.5: Frequency diagrams of the classification accuracy associated with the samples obtained by the various walks.

Fitness landscape metrics

Since the progressive gradient walk captures a different distribution of fitness values compared to the random walks, the FLA metrics are expected to yield different results when calculated over the gradient walks. To test this hypothesis, three FLA metrics were used, originally proposed as metrics calculated over the random walks. The metrics are:

1. The first entropic measure of ruggedness, FEM, discussed in Section 3.4.2. FEM quantifies the change in fitness values based on the entropy of a random walk. The value of FEM ranges between 0 and 1, where 0 indicates a perfectly smooth landscape, and 1 indicates maximal ruggedness.

2. The neutrality measures, M_1 and M_2, discussed in Section 3.4.2. M_1 measures the proportion of neutral 3-point structures in a walk, and M_2 measures the relative length of the largest sequence of neutral steps in the walk. Both M_1 and M_2 range between 0 and 1, where 0 indicates a landscape with no neutral regions, and 1 indicates a completely flat landscape.

The resulting FEM, M_1, and M_2 values calculated over the various walks are summarised in Table 6.2. Table 6.2 shows average values obtained over 30 independent runs, together

Table 6.2: Some FLA metrics obtained over various walks

	Random	Progressive	Gradient	Unbounded
FEM (micro)	0.41178 (0.02238)	0.30623 (0.00989)	0.20239 (0.00263)	0.20234 0.00263
FEM (macro)	0.47964 (0.01450)	0.46006 (0.00611)	0.65177 (0.00707)	0.56348 (0.01880)
M_1 (micro)	0.01897 (0.01192)	0.02956 (0.00502)	0.01003 (0.01287)	0.36281 (0.03622)
M_1 (macro)	0.00310 (0.00194)	0.00238 (0.00084)	0.00967 (0.00964)	0.12001 (0.02479)
M_2 (micro)	0.00635 (0.00398)	0.01095 (0.00162)	0.00003 (0.00003)	0.00142 (0.00026)
M_2 (macro)	0.00252 (0.00149)	0.00188 (0.00064)	0.00000 (0.00002)	0.01999 (0.01042)

with the corresponding standard deviations shown in parenthesis. Each run comprised of 100 walks. Each micro walk performed 1000 steps, and each macro walk performed 100 steps.

Table 6.2 shows that both bounded and unbounded progressive gradient walks exhibited a higher disparity between the micro and macro FEM values than the random walks. Indeed, the random sampling algorithms covered more or less the same areas of average fitness, while the gradient walks focused on areas of higher fitness. The maximal size of the step had an influence on the resulting FEM, indicating that the error landscape was smooth when observed locally, and exhibited ruggedness when observed at a larger scale. Perhaps FEM values can be used to suggest an appropriate step size scaling for NN optimisation algorithms.

Table 6.2 shows that the values of $M1$ were more or less the same for all the bounded walks, and only the unbounded gradient walk captured areas of increased neutrality. Indeed, if the gradient walk is allowed to leave the bounded search space, it is likely to fall into one of the "ravines" that the NN landscapes are known to exhibit. Thus, the

unbounded gradient walk captured a property of NN landscapes that the other walks did not.

The values of $M2$ in Table 6.2 indicate that none of the walks have experienced a long stretch of unchanging fitness values. Thus, the progressive gradient walk did not simply converge on a single point in the search space and sample it indefinitely.

Thus, the different walks do indeed capture different properties of NN error landscapes, and analysis of the gradient walks rather than the random walks may highlight more interesting and important features of the landscapes than that captured by the unbiased random walks.

6.2 Visualisation and Quantification of Neural Network Attraction Basins

One of the simplest FLA approaches to estimate the presence of local minima is to take a uniform random sample of the search space, and then to calculate the proportion of local minima within the sample [3]. To identify minima, stationary points need to be identified first. Since the loss functions are differentiable, the gradient can be calculated for each point in the sample. Points with a gradient of zero are stationary points. Stationary points can be further categorised into local minima, local maxima, and saddle points by calculating the eigenvalues of the corresponding Hessian matrix. A positive-definite Hessian is indicative of a local minimum [36].

However, as demonstrated in Section 6.1, random samples capture very few points of high fitness even for such a simple problem as XOR, and thus are unlikely to discover local or global minima. Additionally, random samples do not capture the neighbourhood relationship between individual sample points, which is crucial to the analysis of the basins of attraction. Besides simply identifying the presence or absence of local minima, the possibility of escaping the minima, as well as the structure of the minima, should also be quantified.

Progressive gradient walks, proposed in Section 6.1, constitute an algorithm-agnostic and spatially correlated sampling method biased towards areas of good fitness. Progressive gradient walks can therefore be used to identify local minima, and to analyse the

associated basins of attraction.

This section describes two novel FLA techniques proposed in this study for the purpose of visualisation and quantification of the stationary points and the associated basins of attraction exhibited by NN loss surfaces. Section 6.2.1 introduces loss-gradient clouds, which offer a 2-dimensional visualisation of the stationary points discovered by the gradient walks. Section 6.2.2 proposes two metrics to quantify the properties of the basins of attraction encountered by the gradient walks.

6.2.1 Loss-gradient clouds

A simple way to visualise stationary points discovered by a search space sample is to plot the error, or loss values, against the corresponding gradient in a 2-dimensional scatterplot, referred to as the loss-gradient cloud, or l-g cloud. All points of zero gradient are stationary points. Stationary points of non-zero loss can be either local minima, local maxima, or saddle points. To determine if a particular stationary point is a local minimum, local maximum, or a saddle point, local curvature information can be derived from the eigenvalues of the corresponding Hessian matrix. If the eigenvalues of the Hessian are positive, the point is a maximum. If the eigenvalues are negative, the point is a minimum. If the eigenvalues are positive as well as negative, the point is a saddle. If any of the eigenvalues are zero, i.e. if the Hessian is indefinite, the test is considered inconclusive.

The main benefit of l-g clouds is the 2-dimensional representation of the high-dimensional search space. Studying the discovered stationary points in 2-dimensional space allows the identification of the total number of attractors corresponding to different loss values. The gradient behaviour of the attractors is also visualised by the l-g clouds, and can provide useful insights. Since the distance between sampled points is not represented in the l-g clouds, the actual number of distinct local minima and other attractors cannot be estimated using this technique.

6.2.2 Quantifying basins of attraction

The progressive gradient walk samples the search space by taking randomised steps in the general direction of the steepest gradient descent. If a step taken in the direction of the negative gradient is too large, the step may miss the area of low error, and result in an area of higher error. Thus, the progressive gradient walk will not necessarily produce a sequence of points with strictly non-increasing error values. In fact, any gradient-based sample or algorithm trajectory is likely to exhibit oscillatory behaviour if the gradient in some dimensions is significantly steeper than in others [113].

Even though the gradient step sequence will not necessarily be strictly decreasing in error, the sample is nonetheless expected to travel in the general direction of a global minimum. The areas of the landscape where a gradient-based walk oscillates or otherwise fails to reduce the error for a number of steps are the stationary areas of the search space that may hinder the optimisation process. Quantification of the number and extent of such areas will provide an indication of the "difficulty" of the search space, as well as an empirical estimate of the landscape modality. Thus, an important error landscape property to estimate is the number of times that the sampling algorithm will become "stuck" along the way.

To smooth out potential oscillations of a sample, an exponential moving average of the sample can be calculated. An exponentially weighted moving average (EWMA) [147] is a smoothing filter commonly used for time series prediction. EWMA calculates the moving average for each step in the time series by taking all previous steps into account, and assigning exponentially decaying weights to the previous steps, such that the weight for each older step in the series decreases exponentially, never reaching zero. Given a sequence $T = \{T_l\}_{l=1}^{L}$ of length L, the EWMA-smoothed sequence T' is given by:

$$T'_l = \begin{cases} T_l & \text{if } l = 1 \\ \beta \cdot T_l + (1 - \beta) \cdot T'_{l-1} & \text{if } l > 1 \end{cases} \tag{6.1}$$

The decay coefficient, $\beta \in [0, 1]$, determines the degree of smoothing, where larger values of β facilitate faster decay and weaker smoothing, and smaller values of β facilitate slower decay and thus stronger smoothing.

To identify the sections of the sample where the behaviour is stagnant, the standard

deviation of the smoothed sample is calculated first. Then, a sliding window approach is used to generate a sequence of the moving standard deviations of the sample. If the standard deviation of the values in the current window is less than the standard deviation of the entire sample for a number of steps, then these steps can be said to form a stagnant sequence. The average number of stagnant regions encountered per sample, n_{stag}, and the average length of the stagnant regions, l_{stag}, can be used to quantify the number and size of the basins of attraction present in the search space.

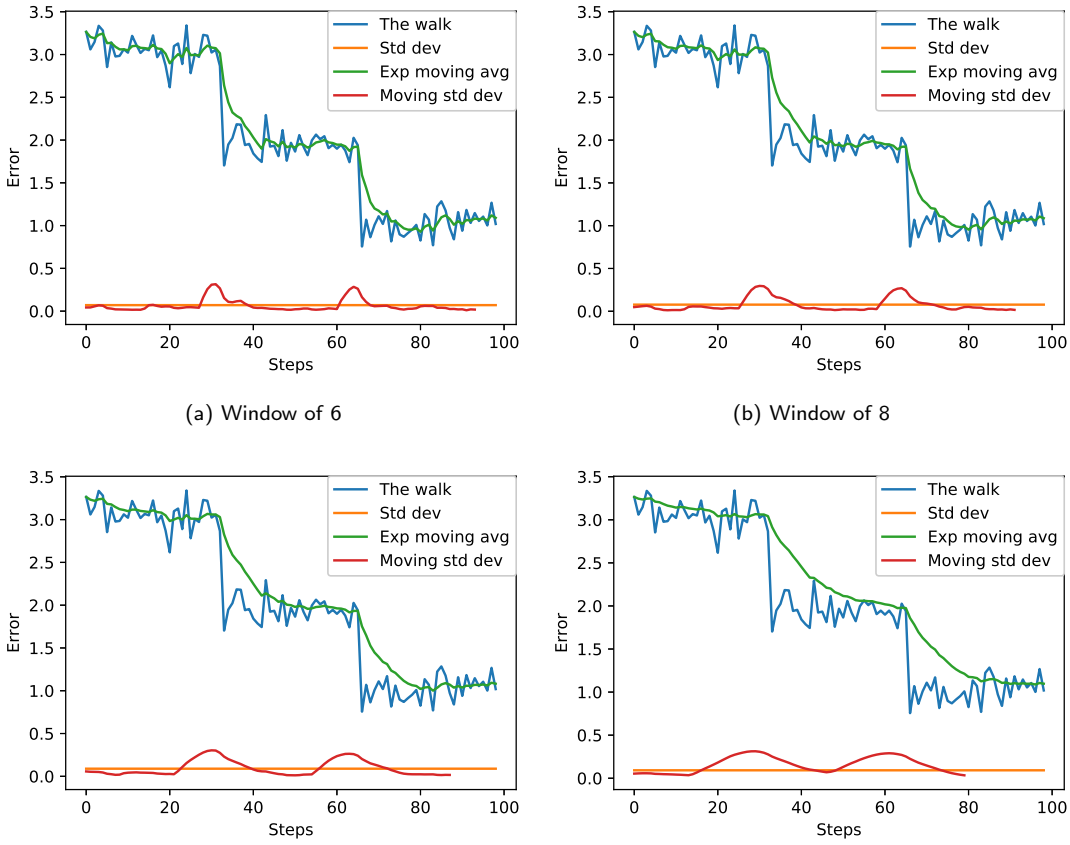
The proposed approach is illustrated in Figure 6.6. The simulated walk oscillates around three different error values. The moving standard deviation line dips below the all-sample standard deviation threshold three times, which corresponds to the three simulated stagnant areas.

Figure 6.6 illustrates that the window size has a significant effect on the attraction basin estimates: too little smoothing (Figure 6.6a) may cause fluctuations to be perceived as stationary regions. Excessive smoothing, on the other hand (Figure 6.6d), may fail to detect all stationary regions. Therefore, the window size has to be optimised per sample. If the sequence contains oscillations, then too little smoothing will cause multiple "spikes" in the walk to be regarded as areas of stagnation. These short bursts of "stagnation" will yield a small average basin length, l_{stag}. If the sequence is smoothed excessively, the sample will start to resemble a wave more and more, perceiving flat areas as areas with an incline, which will once again cause the l_{stag} to decrease. Thus, too little as well as too much smoothing will shrink the value of l_{stag}. Therefore, the window size w can be optimised by maximising l_{stag}. Table 6.3 lists l_{stag} values obtained on the simulated walk shown in Figure 6.6 under various values of w. Table 6.3 shows that l_{stag} reaches its maximum for $w = 8$, and decreases for smaller, as well as larger, values of w.

Table 6.3: Effect of window size w on l_{stag}

w	6	8	10	12	14	16	18	20
l_{stag}	18.75	**22.0**	20.67	18.0	16.33	14.33	14.0	12.0

The window size w can therefore be automatically optimised by calculating l_{stag} over a range of w values, and picking the value of w that yields the highest l_{stag} value. In this study, w is optimised by successively applying $w \in \{6, 8, \ldots, 18, 20\}$. Given a window of



(a) Window of 6

(b) Window of 8

(c) Window of 12

(d) Window of 20

Figure 6.6: Illustration of the proposed technique to estimate the number and extent of the basins of attraction. Figures 6.6a to 6.6d show the effect of window size on the sample smoothing.

size w, the EWMA value of β is calculated as $\beta = 2/(w+1)$. The w value yielding the largest l_{stag} is subsequently used for the final l_{stag} and n_{stag} estimates.

Thus, two estimates to quantify the basins of attraction are proposed:

1. The average number of times that stagnation was observed, n_{stag}.
2. The average length of the stagnant sequence, l_{stag}.

Pseudocode given in Algorithms 1 and 2 summarises the proposed method to obtain both metrics.

Algorithm 1 Basins of attraction estimates

Initialise n_{stag}, average number of basins, to 0;
Initialise l_{stag}, average basin size, to 0;
Initialise n_w to the number of walks to perform;
Initialise *walk*, the sample, to $\emptyset$;
for $\forall i \in \{1, ..., n_w\}$ **do**
  walk $\leftarrow$ sample the input problem using a progressive gradient walk [14];
  Normalise the sample fitness range in *walk* to $[0, 1]$;
  Initialise $n_{stag,i}$ and $l_{stag,i}$ to 0 for walk i;
  for $\forall j \in \{6, 8, ..., 18, 20\}$ **do**
    walk $\leftarrow$ calculate the EWMA of *walk* using Equantion (6.1), $\beta = 2/(j+1)$
    $\varsigma \leftarrow$ calculate the standard deviation of *walk*
    $\sigma \leftarrow$ calculate the sequence of moving standard deviations of *walk*, $w = j$
    Obtain a list of stagnant regions, *list*, using Algorithm 2 with inputs σ, ς.
    if average length of regions in *list* $> l_{stag,i}$ **then**
      $l_{stag,i} \leftarrow$ average(*list*)
      $n_{stag,i} \leftarrow$ number of regions in l
    end if
  end for
  $n_{stag} \leftarrow n_{stag} + n_{stag,i}$
  $l_{stag} \leftarrow l_{stag} + l_{stag,i}$
end for
return n_{stag}/n_w, l_{stag}/n_w

It is important to note that n_{stag} and l_{stag} are approximations, and may produce misleading results in some scenarios. Specifically, if the observed sequence is chaotic, i.e. does not exhibit convergence or stagnant areas, the estimates provided by n_{stag} and l_{stag} are likely to be overly optimistic. In order to maximise l_{stag}, the algorithm will apply excessive smoothing to the chaotic sequence, potentially interpreting chaotic fluctuations as multiple stagnation regions. In general, because the algorithm is designed to maximise the stagnation length of the estimate, erroneous results are expected for sequences that

Algorithm 2 Basins of attraction identification

```
Inputs: σ, ς;
Initialise l_stag, average basin size, to 0;
Initialise stuck to false
Initialise len, length of a stagnant region, to 0;
Initialise list, the list of stagnant regions, to ∅;
for each step s_i in σ do
  if stuck then
    if s_i < ς then
      len ← len + 1
    else
      stuck ←false
      list ← add len to list
      len ← 0
    end if
  else
    if s_i < ς then
      len ← len + 1
      stuck ←true
    end if
  end if
end for
if len > 0 then
  list ← add len to list
end if
return list
```

do not exhibit any form of stagnation.

6.3 Conclusions

This chapter proposed a progressive gradient walk as an adaptive sampling mechanism for the analysis of NN fitness landscapes. The gradient walk is more computationally efficient than a population-based adaptive walk, and has better guarantees of finding areas of high fitness. The gradient information is used to calculate the direction of the next step, but the magnitude of the step is randomised per dimension within the given bounds, thus adding stochasticity and preventing convergence.

Both bounded and unbounded progressive gradient walks were compared to the random and progressive random walks in terms of search space coverage and fitness coverage. It was shown that, even though random walks provide wider search space coverage, they fail to capture areas of high fitness. The gradient walk, on the other hand, is strongly biased towards the areas of high fitness, while also covering some of the poor fitness areas. Thus, the gradient walk is more representative of the search space in the context of applicability to function optimisation. In addition, the unbounded progressive gradient walk was shown to provide a truer picture of the error landscape than the bounded gradient walk.

A selection of FLA metrics were calculated over the random and the gradient walks. While the obtained FLA metrics did not disagree with one another, the FLA metrics obtained from the gradient walks captured the specific known characteristics of NN error landscapes with better precision.

This chapter also proposed an intuitive visualisation of the local minima and the associated basins of attraction, namely the loss-gradient clouds. By plotting the loss values against the corresponding gradient vector magnitudes as sampled by the progressive gradient walk, stationary points could be easily identified. To classify the identified stationary points as minima, maxima, or saddles, Hessian matrix information can be used to identify the curvature of each sampled point. Additionally, this chapter proposed two simple metrics to quantify the number and extent of attraction basins as sampled by the gradient walks. Calculation of statistical metrics over a number of walks provides an idea of the connectedness of the various basins, as well as the likelihood of escaping from the basins.

The proposed modality quantification approaches are evaluated in the next chapter

by performing a visual and numerical analysis of local minima and the associated basins of attraction for two common NN loss functions, namely quadratic loss and entropic loss.

Chapter 7

Loss Functions

Quantification of the stationary points and the associated basins of attraction of NN loss surfaces is an important step towards a better understanding of NN loss surfaces at large. Loss-gradient clouds were proposed in Chapter 6 for the purpose of visual analysis of stationary points discovered by progressive gradient walks, together with two additional metrics for numeric quantification of the attraction basins. This chapter[1] evaluates the proposed metrics by performing FLA of two common NN error functions: quadratic loss and entropic loss.

The rest of the chapter is structured as follows: Section 7.1 briefly reviews the previously published literature on local minima, stationary points, and attraction basins associated with the two loss functions. Section 7.2 details the experimental procedure. Section 7.3 presents a visual and numerical analysis of stationary points and basins of attraction of the quadratic and the entropic error landscapes. Section 7.4 presents FLA measures of gradients, ruggedness, and neutrality associated with the two losses. Section 7.5 concludes the chapter.

[1]Research presented in this chapter has been submitted for publication as an article to the *Neural Networks* journal [15].

7.1 Quadratic and Entropic Loss Surfaces

The number of local minima, as well as the properties of local minima, were theoretically shown to depend on the chosen error metric [120], among other parameters. The two most widely used error metrics are the quadratic loss function (SSE) and the entropic loss function (CE), discussed earlier in Section 2.4. Solla et al. [120] analysed quadratic loss and entropic loss theoretically, and came to the conclusion that quadratic loss exhibits a higher density of local minima. Solla et al. [120] further showed that entropic loss must generate a "steeper" landscape with stronger gradients, which may be the reason for the observed faster convergence of gradient descent on CE compared to SSE. In addition to faster convergence, entropic loss was shown to exhibit better statistical properties, such as more precise estimation of the true posterior probability on average [67]. Superior probabilistic properties of entropic loss has lead to entropic loss becoming more popular than quadratic loss in the deep learning community [44, 46].

From a theoretical standpoint, however, the global minima of both SSE and CE will correspond to the true posterior probability derived from the given dataset [16]. Thus, if a global minimum is found on either of the error landscapes, the quality of either minimum will be equally good. A study by Golik et al. [46] has shown that, although squared loss may cause the training algorithm to converge to a poor minimum, this behaviour is only exhibited if the algorithm was initialised poorly. Golik et al. [46] demonstrated the benefit of applying gradient descent to the error landscape generated by entropic loss at first, and then "switching" to quadratic loss to further refine the solution discovered on the entropic loss surface. Such a training scheme may be successful due to the fact that entropic loss is known to turn flat around the global minima [5].

As previously discussed in Section 3.5.5, Kordos and Duch [68] compared the loss surfaces of SSE and CE using PCA visualisations, and observed that CE yielded a more complex error surface. The same study indicated that NN loss surfaces exhibited the "starfish" structure regardless of the error metric chosen.

To better understand the landscape properties of the two loss functions, and to evaluate the expressiveness of the techniques proposed in Chapter 6, the proposed FLA techniques were applied to investigate the quadratic and entropic loss surfaces for a selection of benchmark problems of varied dimensionality.

7.2 Experimental Procedure

The aim of the study was to visually and numerically investigate the local minima and basins of attraction exhibited by the quadratic and entropic loss functions. This section discusses the experimental set-up of the study, and is structured as follows: Section 7.2.1 describes the NN hyperparameters employed in the experiments, Section 7.2.2 lists the benchmark problems used, and Section 7.2.3 outlines the sampling algorithm parameters, and the data recorded for each sampled point.

7.2.1 Neural network hyperparameters

All experiments employed feed-forward NNs with a single hidden layer. The sigmoid activation function, given by Equation (2.11) in Section 2.5, was used in the experiments. While the choice of activation function has an effect on the resulting error landscape, the aim of this study was to investigate the difference between quadratic and entropic loss.

7.2.2 Benchmark problems

For the purpose of this study, seven classification problems described in Appendix A were used, namely XOR, Iris, Diabetes, Glass, Cancer, Heart, and MNIST.

7.2.3 Sampling parameters

For the purpose of sampling the areas of low error, the progressive gradient walk, discussed in Section 6.1.1, was used as the sampling mechanism. To allow for adequate coverage of the search space, the number of independent walks was set to be one order of magnitude higher than the dimensionality of the problem, i.e. for a problem of m dimensions, $10 \times m$ independent progressive gradient walks were performed. The walks were not restricted by search space bounds, however, two different initialisation ranges were considered, namely $[-1, 1]$ and $[-10, 10]$. The smaller range is typically used for NN weight initialisation. The larger range is likely to contain high fitness solutions [12]. Since the granularity of the walk, i.e. the average step size, has a bearing on the resulting

FLA metrics [75], two granularity settings were used throughout the experiments: micro, where the maximum step size was set to 1% of the initialisation range, and macro, where the maximum step size was set to 10% of the initialisation range. Micro walks performed 1000 steps each, and macro walks performed 100 steps each.

For all problems except the XOR problem, the dataset was split into 80% training and 20% test subsets. The training set was used to calculate the direction of the gradient, as well as the error of the current point on the walk. The test set was used to evaluate the generalisation ability of each point in the walk. To calculate the training and the generalisation errors, the entire train/test subsets were used for all problems except MNIST. For MNIST, random batches of 100 patterns were sampled from the respective training and test sets.

In order to identify stationary points discovered by the gradient walks, the magnitude of the gradient vector was recorded for each step together with the loss value. Additionally, the eigenvalues of the Hessian matrix were calculated for each step, and used to classify each step as convex, concave, saddle, or singular.

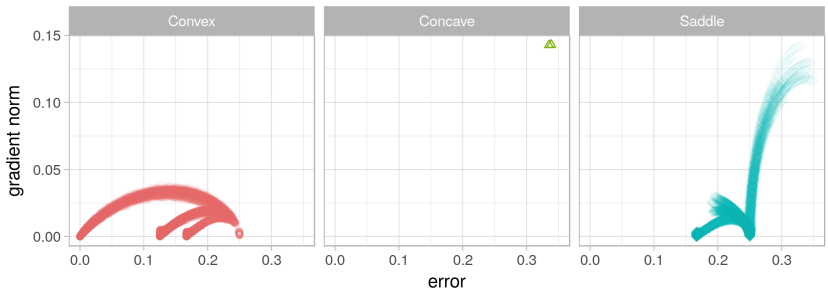
7.3 Empirical Study of Modality

This section presents the analysis of apparent local minima and the corresponding basins of attraction as captured by the progressive gradient walks. The results obtained for each problem are discussed separately.

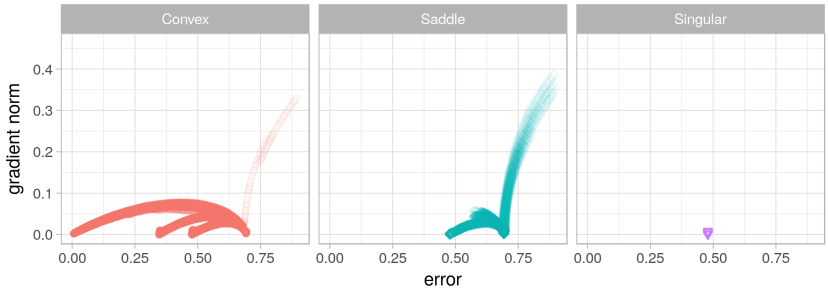
7.3.1 XOR

Figures 7.1 and 7.2 show the l-g clouds (proposed in Section 6.2.1), obtained for the XOR problem, separated into panes according to the curvature.

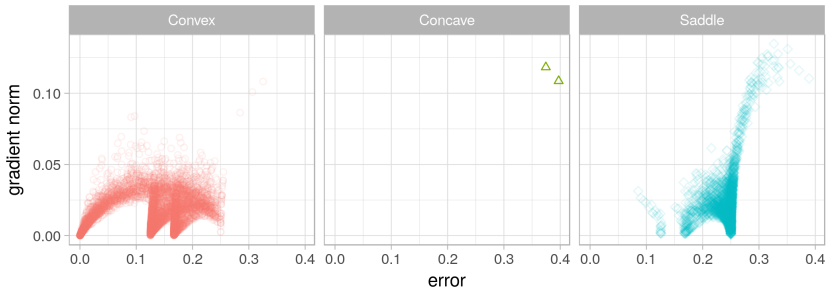
The first observation that can immediately be made from Figure 7.1 is that both SSE and CE yielded exactly four stationary points on the walks initialised in the $[-1, 1]$ range. Furthermore, these four points were classified as convex according to the Hessian eigenvalues, indicating that the points can be classified as local minima rather than saddle points. A transition from saddle curvature to convex curvature was observed for both SSE and CE. Points further away from the global minimum attractor were classified



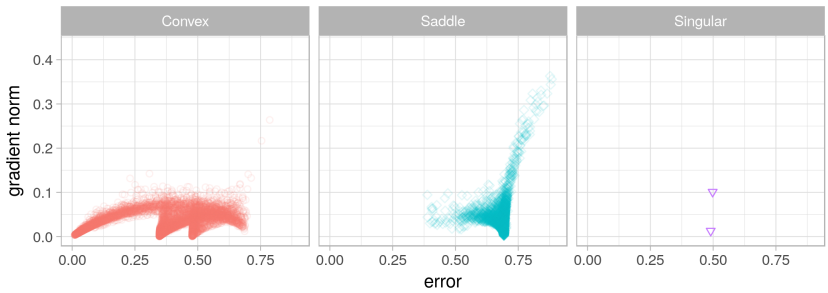
(a) SSE, micro, $[-1, 1]$



(b) CE, micro, $[-1, 1]$



(c) SSE, macro, $[-1, 1]$



(d) CE, macro, $[-1, 1]$

Figure 7.1: L-g clouds for the gradient walks initialised in the $[-1, 1]$ range for the XOR problem.

as exhibiting saddle curvature. Two stationary points furthest away from the global minimum attractor were sometimes classified as saddles, indicating that both saddles and local minima of equal loss value were discovered. Under the macro setting (larger steps), a few singular points have been sampled in the same apparent basin, indicating that the area was flat (no curvature) in some dimensions.

However, the global minimum points discovered by the gradient walks initialised in the $[-1, 1]$ range appeared perfectly convex. The area surrounding the global attractor, as well as the two adjacent local minima, also exhibited convexity. Thus, the XOR problem definitely exhibits convex local minima.

Another interesting observation can be made by observing the trajectories connecting the apparent local minima: It is evident from Figure 7.1 that most high loss, high gradient points first descended to the local minimum furthest away from the global minimum, and from thereon proceeded to one of the three better minima. The three convex minima, however, were not connected by trajectories. In other words, once the gradient walk descended into one of the basins, escape from the basin became unlikely, given the limited step size. To further support this claim, the n_{stag} and l_{stag} measures (discussed in Section 6.2.2) were calculated for the various XOR gradient walks, and are reported in Table 7.1. According to Table 7.1, the average number of basins visited by the $[-1, 1]$ micro-step walks was 1.88889 for SSE, and 2.04444 for CE. Thus, the walks visited two or fewer basins. The n_{stag} values are even smaller for macro-step walks initialised in the same range, i.e. 1.33333 for SSE, and 1.35556 for CE. Figures 7.1c and 7.1d illustrate that larger step sizes allowed some of the walk trajectories to skip the poor loss area, while the smaller steps consistently became stuck, and proceeded directly to one of the better minima. Small n_{stag} values indicate that transition between adjacent minima was still unlikely for the given step size.

CE and SSE thus exhibited very similar properties when sampled with $[-1, 1]$ gradient walks. The same number of local minima was observed, and the basins of attraction exhibited similar behaviour in terms of basin-to-basin transitions. According to Figure 7.1, CE exhibited stronger gradients. This corresponds to the theoretical predictions made in [120]. A comparison of Figures 7.1c and 7.1d shows that SSE exhibited more non-convex behaviour around the apparent local minima, which indicates that SSE would

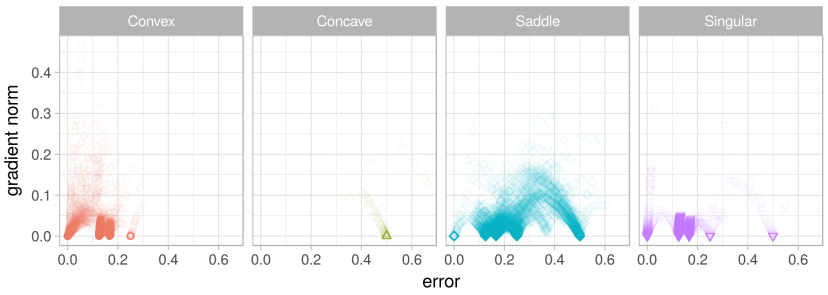
Table 7.1: Basin of attraction estimates calculated for the XOR problem. Standard deviation shown in parenthesis.

	SSE		CE	
	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$, micro	1.88889	367.04444	2.04444	313.32130
	(0.31427)	(134.84453)	(0.44500)	(148.75671)
$[-1, 1]$, macro	1.33333	37.14815	1.35556	30.35000
	(0.49441)	(16.68220)	(0.50136)	(13.32863)
$[-10, 10]$, micro	1.63333	684.77778	1.16667	870.87222
	(0.72188)	(263.72374)	(0.37268)	(180.17149)
$[-10, 10]$, macro	1.10000	57.98889	1.03333	74.79444
	(0.39581)	(24.91864)	(0.23333)	(20.49253)

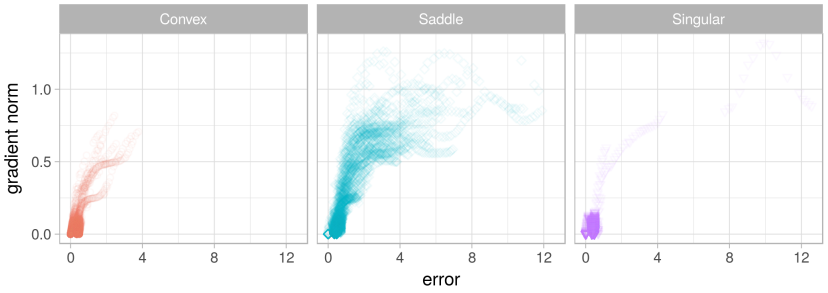
be harder to search for an optimisation algorithm than CE.

Figure 7.2 shows the l-g clouds obtained for gradient walks initialised in the $[-10, 10]$ range. Figures 7.2a and 7.2c indicate that initialisation in a wider range caused the gradient walks to discover more stationary points on the SSE loss surface: Instead of four points of zero gradient, six can be seen in the figures. Out of these six, only four exhibited convexity. Even the points that exhibited convexity were surrounded by points with saddle curvature or no curvature. Such overlap between convex and non-convex structure indicates that the surface around the minima was not smooth. Overlap of convexity and non-convexity can also indicate that multiple minima of the same loss value exist that exhibit different landscape curvature properties.

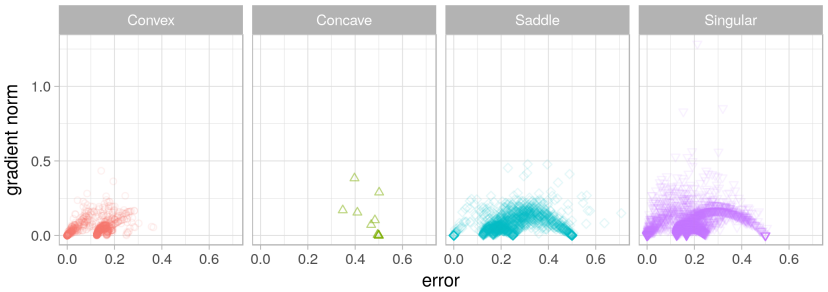
Figures 7.2b and 7.2d show that the loss surface of CE exhibited noticeably different properties when probed in a larger range. While non-convex curvature remained prevalent, CE, as opposed to SSE, did not exhibit additional stationary points. Instead, points of high loss exhibited high fitness, leading the gradient walks towards the same basins as discovered with the $[-1, 1]$ walks. Figure 7.3 displays only those points of the gradient walks that yielded a CE loss value less than 1. Four stationary points can be observed, only three of which exhibited convexity. Thus, CE exhibited fewer local minima than



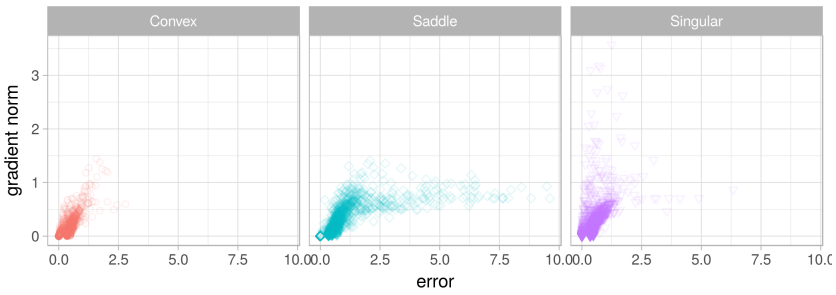
(a) SSE, micro, $[-10, 10]$



(b) CE, micro, $[-10, 10]$

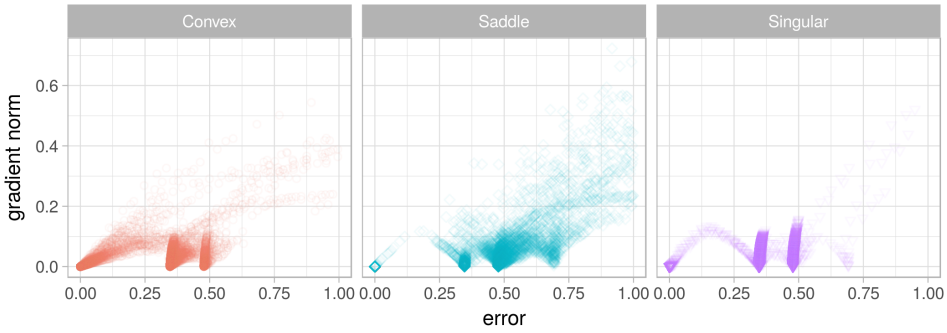


(c) SSE, macro, $[-10, 10]$

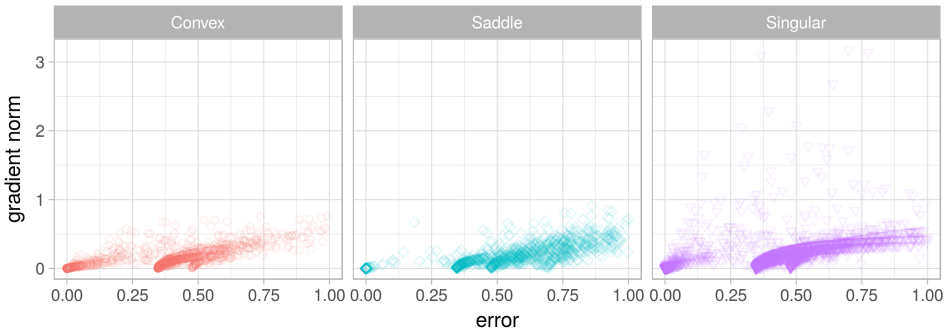


(d) CE, macro, $[-10, 10]$

Figure 7.2: L-g clouds for the gradient walks initialised in the $[-10, 10]$ range for the XOR problem.



(a) CE, micro, $[-10, 10]$, error < 1



(b) CE, macro, $[-10, 10]$, error < 1

Figure 7.3: L-g clouds for the CE values less than 1, sampled by the gradient walks initialised in the $[-10, 10]$ range for the XOR problem.

SSE. This observation corresponds with the theoretical predictions made in [120].

Once again, the convex minima observed in Figures 7.2 and 7.3 were disconnected from one another. No convex trajectory has been captured that visited all the stationary points present. In fact, according to Figure 7.3a, the only transition between the global optimum and the adjacent local optima corresponded to the indefinite Hessians. Thus, to make a transition from one convex minimum to another one, the algorithm had to traverse a flat area with little to no convexity. With reference to Table 7.1, the n_{stag} values were smaller for the $[-10, 10]$ initialisation range, and the l_{stag} values were larger than those yielded by the $[-1, 1]$ walks. Thus, the walks were more likely to stagnate once, and to remain in the stagnated state for the entire walk.

A comparison of Figures 7.2a and 7.2b shows that CE demonstrated a smoother, more consistent relationship between the gradient and the loss values than SSE. Together with

evidently fewer stationary points, this property makes CE an easier loss surface to search.

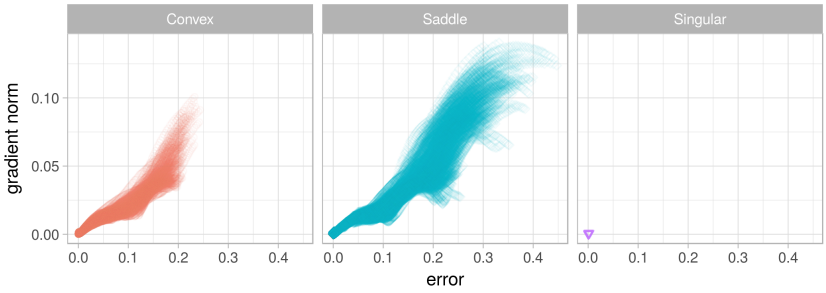
Figures 7.2c and 7.2d indicate that gradient walks with a macro step size, initialised in a larger area, still managed to find the global optima for both SSE and CE, but on fewer occasions than the micro walks. A large portion of the points yielded indefinite Hessians, indicating flatness. This is to be expected, as the loss surfaces of NNs with sigmoidal activation functions are known to exhibit increasing hidden neuron saturation with an increased distance from the origin [134].

7.3.2 Iris

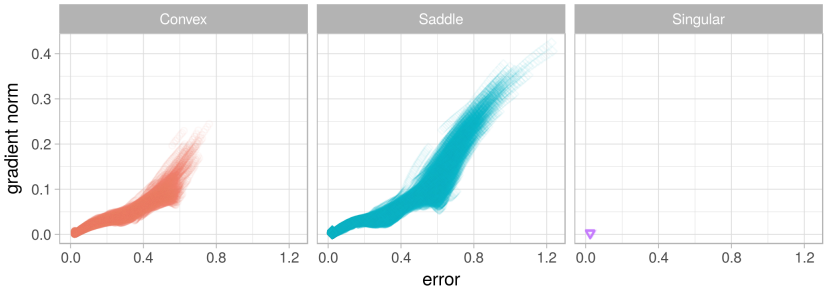
The Iris dataset, despite being rather compact and trivial, is large enough to be split into the training and testing subsets. The training subset can then be used to sample the loss surface, and the testing set can be used to evaluate the discovered minima and stationary points for their ability to generalise. For the rest of the chapter, the training set loss values are referred to as E_t, and the test set loss values are referred to as E_g.

Figures 7.4 and 7.5 show the l-g clouds obtained for the Iris problem. According to Figure 7.4, only one attractor with zero gradient has been discovered on both the SSE and CE loss surfaces by gradient walks initialised in the $[-1, 1]$ range. Two more attractors of non-zero gradient can also be observed, however, these attractors do not constitute local minima. Transition from non-convex space to convex space was still present, but was less distinct than for XOR. Points around the global minimum attractor exhibited convex as well as saddle behaviour, and saddle behaviour was prevalent. Both the SSE and CE surfaces exhibited flatness (indicated by the singular Hessians) around the global optima. This corresponds to theoretical claims that the loss surface around the global minima is flat [5]. However, the flatness was not prevalent.

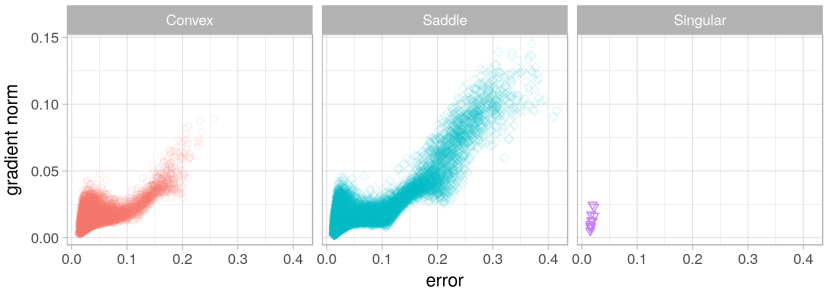
A comparison of the micro and macro steps in Figure 7.4 indicates that the macro steps discovered the same landscape characteristics as the micro steps. In the macro setting, a wider range of gradient values around the global minimum was discovered. This can be explained by the fact that NN loss surfaces are known to contain ravines and valleys [39], and optima is typically found at the bottom of such structures. The macro step size may have caused the gradient walks to oscillate and to sample points on the sides of the valley where the global minima was discovered.



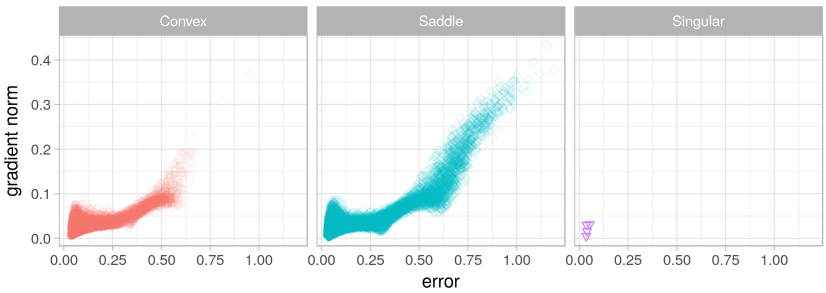
(a) SSE, micro, $[-1, 1]$



(b) CE, micro, $[-1, 1]$

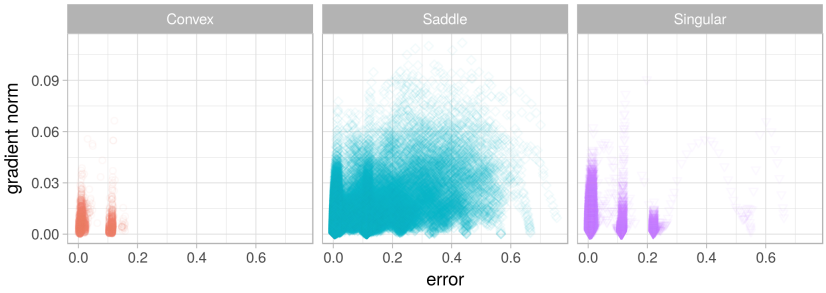


(c) SSE, macro, $[-1, 1]$

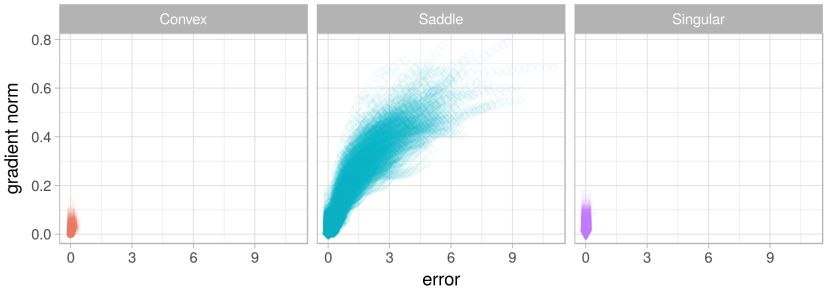


(d) CE, macro, $[-1, 1]$

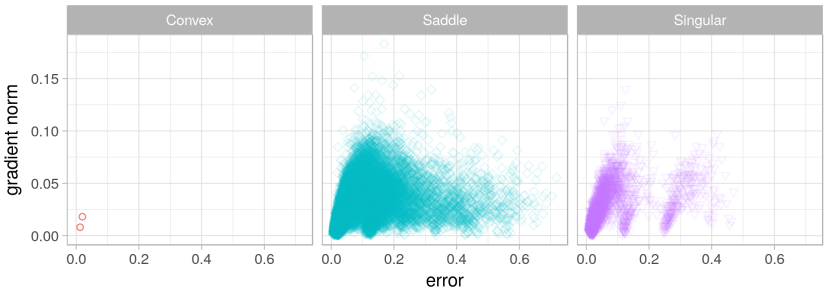
Figure 7.4: L-g clouds for the gradient walks initialised in the $[-1, 1]$ range for the Iris problem.



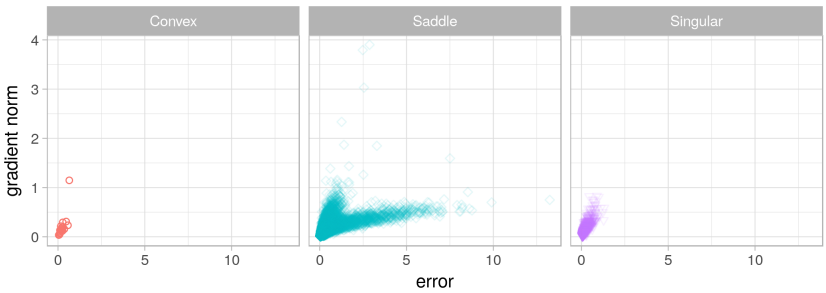
(a) SSE, micro, $[-10, 10]$



(b) CE, micro, $[-10, 10]$



(c) SSE, macro, $[-10, 10]$



(d) CE, macro, $[-10, 10]$

Figure 7.5: L-g clouds for the gradient walks initialised in the $[-10, 10]$ range for the Iris problem.

To further analyse the landscape properties sampled by the gradient walks, the n_{stag} and l_{stag} values were calculated for the E_t values sampled by the gradient walks, as well as for the corresponding E_g values. Table 7.2 lists the E_t and E_g values obtained. According to Table 7.2, both SSE and CE yielded an average n_{stag} very close or equal to 1 for all gradient walks initialised in the $[-1, 1]$ range. Thus, a single basin of attraction was discovered by each individual walk. This correlates well with the results shown in Figure 7.4. For the macro setting in the $[-1, 1]$ range, both SSE and CE produced an n_{stag} average of 1, with a standard deviation of zero. This observation indicates that the macro steps in the $[-1, 1]$ range were sufficient to prevent stagnation in suboptimal areas, yet convergence in an attraction basin still took place. The generalisation error exhibited similar behaviour, as shown in Table 7.2. The presence of a single global attractor makes the loss surface associated with the Iris problem trivial to search using a gradient-based method.

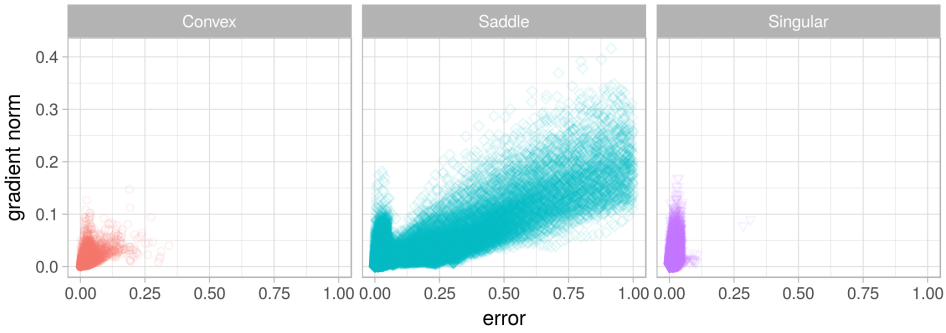
Figure 7.5 shows the l-g clouds obtained for the gradient walks initialised in the $[-10, 10]$ interval. According to Figures 7.5a and 7.5c, multiple stationary points were discovered on the SSE loss surface. Two of the discovered stationary points, including the global minimum, have exhibited convexity. Thus, there is at least one local minimum on the SSE loss surface associated with the Iris problem. Additionally, the discovered stationary points were disjoint in the convex and singular (flat) space. The saddle space was more connected; however, the n_{stag} values presented in Table 7.2 indicate that the gradient walks did not generally become stuck more than twice. Thus, multiple stationary points discovered were not trivial to escape from.

CE, on the other hand, exhibited only one attractor at the global minimum, as illustrated in Figure 7.5. Figure 7.6 shows only those points of the gradient walks that yielded a CE loss value less than 1. Even though all points belong to the same global attraction basin, two distinct clusters can be observed in Figure 7.6a: points that lie in the low error region, and exhibit higher gradients, and points that lie in the higher error region, and exhibit lower gradients. The same tendency can be observed in Figure 7.5d. These observations indicate that the gradient walks have explored wide (higher error, lower gradient) as well as narrow (higher gradient, lower error) valleys, which the NN error landscapes are known to exhibit [21].

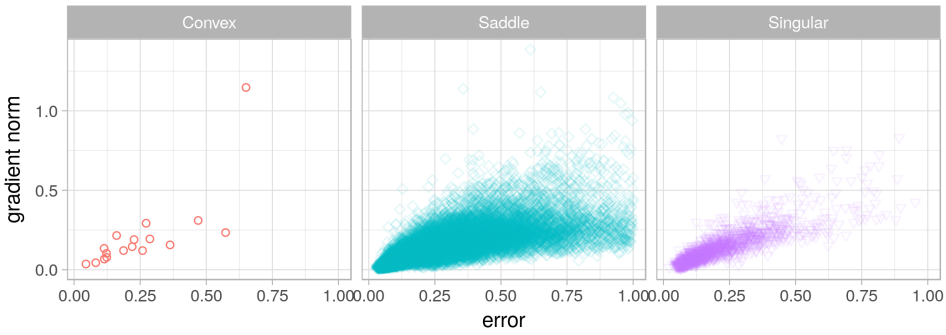
Table 7.2: Basin of attraction estimates calculated for the Iris problem on the E_t and E_g walks. Standard deviation shown in parenthesis.

	SSE		CE	
E_t	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$, micro	1.00857 (0.11922)	848.82048 (52.82897)	1.00571 (0.07538)	820.20429 (54.28522)
$[-1, 1]$, macro	1.00000 (0.00000)	76.46571 (4.03382)	1.00000 (0.00000)	73.40857 (4.72216)
$[-10, 10]$, micro	1.28571 (0.48234)	796.07000 (212.33922)	1.00000 (0.00000)	953.26286 (10.64382)
$[-10, 10]$, macro	1.02571 (0.15828)	73.77000 (13.50309)	1.00571 (0.07538)	84.55857 (6.27217)
E_g	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$, micro	1.10571 (0.38206)	820.07167 (143.30209)	1.02286 (0.18375)	818.33905 (77.64190)
$[-1, 1]$, macro	1.00000 (0.00000)	74.69429 (4.52494)	1.00286 (0.05338)	67.71143 (7.26987)
$[-10, 10]$, micro	1.36000 (0.57231)	770.39048 (229.39260)	1.12286 (0.75160)	917.17541 (138.19499)
$[-10, 10]$, macro	1.03143 (0.17447)	75.41714 (13.86152)	1.01429 (0.11867)	83.85857 (8.78615)

The CE loss surface once again exhibited fewer local minima than SSE. However, the quality of the minima should also be evaluated in terms of the generalisation capabilities, before any final conclusions can be drawn. Figure 7.7 shows the l-g clouds colourised according to the corresponding E_g values. It is evident from Figure 7.7 that CE yielded poor generalisation performance in the area of the global minimum: All E_g values reported were an order of magnitude larger than the corresponding E_t values. This observation is to be expected: Achieving 100% accuracy on the training set can



(a) CE, micro, $[-10, 10]$, $E_t < 1$



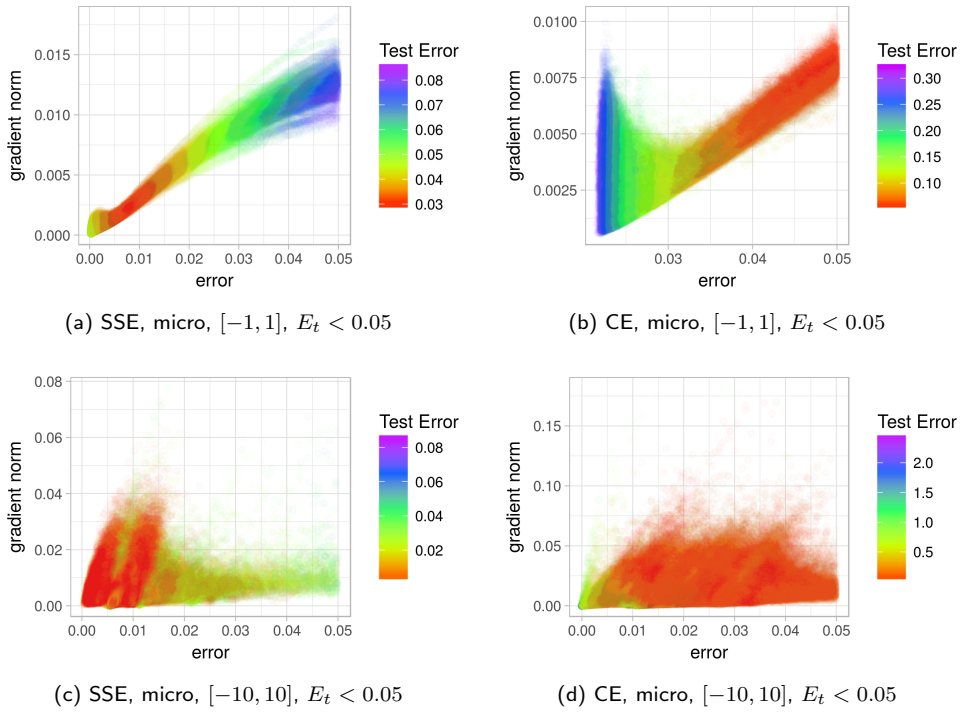
(b) CE, macro, $[-10, 10]$, $E_t < 1$

Figure 7.6: L-g clouds for the CE loss values less than 1 sampled by the gradient walks initialised in the $[-10, 10]$ range for the Iris problem.

easily lead to overfitting. SSE also exhibited overfitting at the global minimum, but not as strongly as CE. CE exhibited stronger gradients around the global optimum, which can cause gradient-based methods to overfit more easily on CE than on SSE.

Appendix B lists all classification errors obtained by the gradient walks on the various problems. Table B.1 indicates that, for the Iris problem, SSE has indeed yielded better generalisation in most scenarios.

Thus, CE exhibited better global structure than SSE on the Iris problem, and was more searchable from the gradient descent perspective. However, stronger gradients around the global optimum indicate that CE exhibited sharper minima, causing stronger overfitting on the CE loss surface.



(a) SSE, micro, $[-1, 1]$, $E_t < 0.05$

(b) CE, micro, $[-1, 1]$, $E_t < 0.05$

(c) SSE, micro, $[-10, 10]$, $E_t < 0.05$

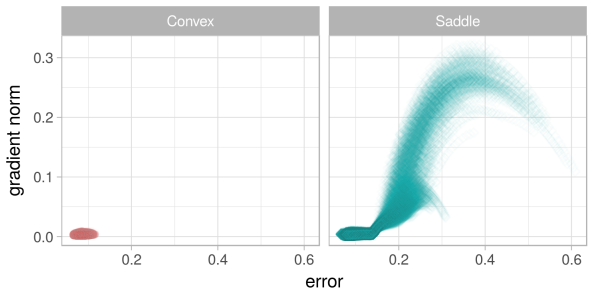
(d) CE, micro, $[-10, 10]$, $E_t < 0.05$

Figure 7.7: L-g clouds colourised according to the corresponding E_g values for the Iris problem.

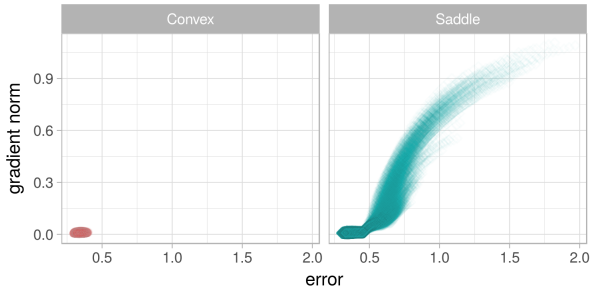
7.3.3 Diabetes

Figures 7.8 and 7.9 show the l-g clouds obtained for the Diabetes problem. According to Figure 7.8, both SSE and CE exhibited a single attractor of near-zero gradient, and that attractor constituted a wide area of low gradients around the loss of zero. Both SSE and CE exhibited convexity around zero loss, especially when sampled with micro steps. The majority of the sampled points, however, were once again classified as a saddle according to their Hessians. This corresponds well with the observations made by Dauphin et al. [28], where the prevalence of saddle points in non-convex optimisation was studied.

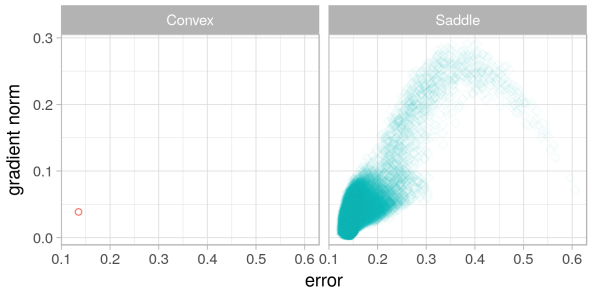
An arch-like curve can be observed in Figures 7.8a and 7.8c, indicating that higher errors were associated with weaker gradients on the SSE loss surface. A transition to the area of higher fitness was associated with a gradient signal that became stronger for



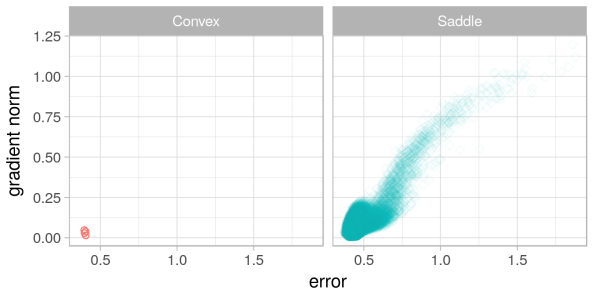
(a) SSE, micro, $[-1, 1]$



(b) CE, micro, $[-1, 1]$

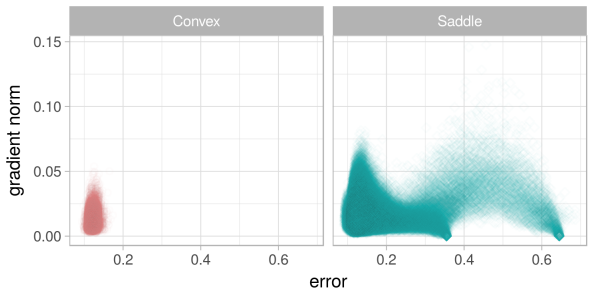


(c) SSE, macro, $[-1, 1]$

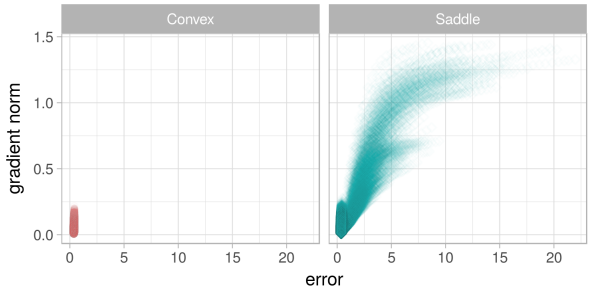


(d) CE, macro, $[-1, 1]$

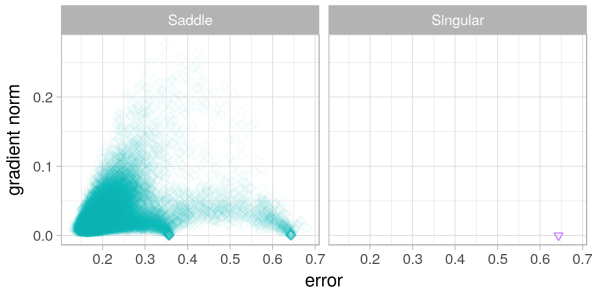
Figure 7.8: L-g clouds for the gradient walks initialised in the $[-1, 1]$ range for the Diabetes problem.



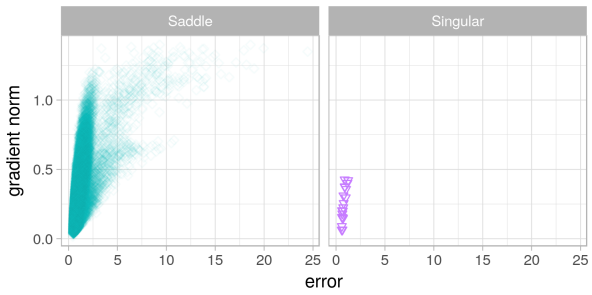
(a) SSE, micro, $[-10, 10]$



(b) CE, micro, $[-10, 10]$



(c) SSE, macro, $[-10, 10]$



(d) CE, macro, $[-10, 10]$

Figure 7.9: L-g clouds for the gradient walks initialised in the $[-10, 10]$ range for the Diabetes problem.

some time, and then began to weaken again as a global optimum was approached. The CE l-g clouds in Figure 7.8 indicate that the CE loss surface did not have the tendency to exhibit weaker gradients for higher errors, which makes CE favourable from the gradient descent perspective. This corresponds well with the theoretical properties of both loss functions, which indicate that SSE is expected to exhibit weaker gradients for higher errors, as opposed to CE [46].

Figure 7.9 shows the l-g clouds obtained for the points sampled by the gradient walks initialised in the $[-10, 10]$ interval. SSE loss once again exhibited multiple near-zero gradient attractors (three), and CE loss exhibited only one attractor. The majority of the points sampled by larger steps in a larger area had a saddle curvature. The convex attractors sampled by the $[-10, 10]$ walks exhibited more variation in gradient than the corresponding attractors discovered by the $[-1, 1]$ walks. This observation can be attributed to the valley structure of the optima: Larger steps induced oscillations around the walls of the valley. Macro $[-10, 10]$ walks did not discover any convexity, and yielded a few points of singular (flat) curvature. This behaviour is likely due to the hidden neuron saturation: Unconstrained macro gradient walks initialised in a larger range are more likely to explore search space areas that are further away from the origin. Larger weights increase the magnitude of net input signals that the hidden neurons receive, and exceedingly large net input signals cause hidden neuron saturation.

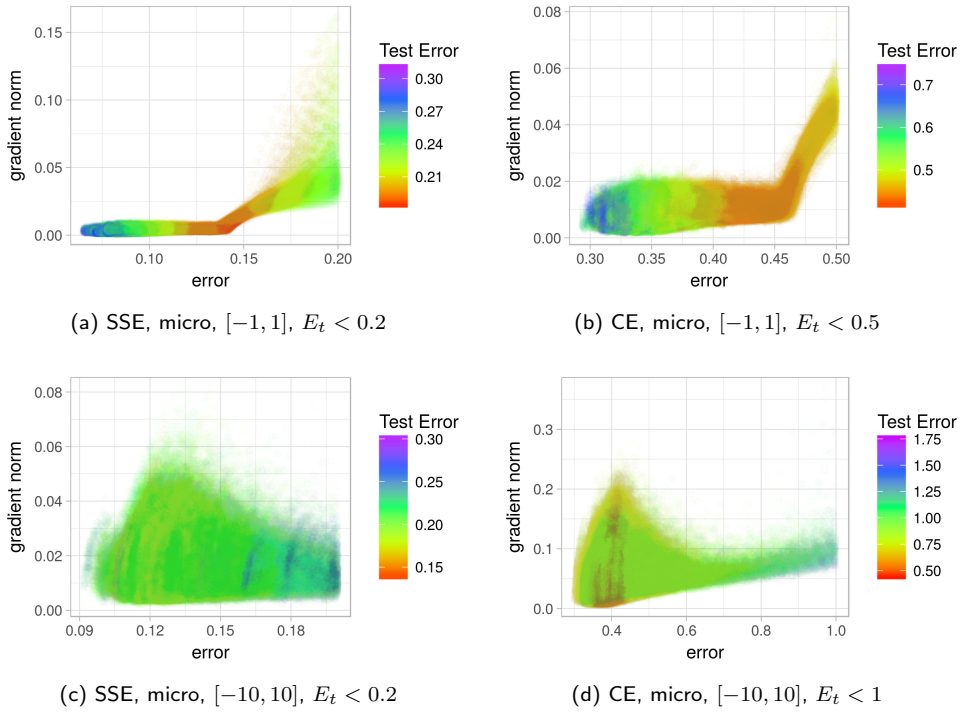
The n_{stag} and l_{stag} values reported in Table 7.3 indicate that most walks discovered a single attractor only, which correlates well with Figures 7.8 and 7.9, and also indicates that the two suboptimal attractors discovered on the SSE loss surface were not easy to escape from. Table 7.3 also shows that the generalisation performance of the points discovered on the SSE loss surface was somewhat volatile when sampled using micro walks. Micro walks took smaller steps, and thus were more likely to exploit a particular attractor, causing overfitting.

Figure 7.10 shows a close-up depiction of the convex attractors, colourised according to their generalisation performance. Both SSE and CE exhibited deteriorating generalisation performance as the walks sampled points closer to the zero loss, which is to be expected. For micro $[-1, 1]$ walks, both SSE and CE exhibited a sudden drop in gradient magnitudes, and the points of low gradient with the highest error exhibited the

Table 7.3: Basin of attraction estimates calculated for the Diabetes problem on the E_t and E_g walks. Standard deviation shown in parenthesis.

	SSE		CE	
E_t	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$, micro	1.00123 (0.03511)	938.66728 (22.51936)	1.00000 (0.00000)	935.27160 (12.49791)
$[-1, 1]$, macro	1.00000 (0.00000)	85.19012 (1.54389)	1.00000 (0.00000)	84.84691 (1.97922)
$[-10, 10]$, micro	1.09259 (0.37525)	905.05504 (138.96827)	1.00000 (0.00000)	962.31235 (5.28613)
$[-10, 10]$, macro	1.03580 (0.18580)	77.06975 (13.75685)	1.02716 (0.18393)	78.23086 (16.45928)
E_g	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$, micro	1.51852 (1.21727)	794.78363 (270.89365)	1.04938 (0.31822)	925.83735 (100.01013)
$[-1, 1]$, macro	1.00494 (0.07010)	85.80988 (4.91134)	1.00123 (0.03511)	85.16543 (2.61851)
$[-10, 10]$, micro	2.76420 (3.96060)	703.52152 (343.41982)	1.00617 (0.07832)	958.96852 (39.60395)
$[-10, 10]$, macro	1.08148 (0.52189)	53.88477 (33.59152)	1.08272 (0.35041)	70.88848 (24.84558)

best generalisation performance. As previously noted by Choromanska et al. [23], finding the global minimum may be unnecessary, as the global minimum is likely to overfit the problem. Figures 7.10c and 7.10d indicate for the $[-10, 10]$ walks that points around the global minimum have exhibited various degrees of generalisation performance, with a significant overlap between good and poor generalisation. This indicates that the discovered minima had the same training error values, but different test error values. The Diabetes problem is known to contain noisy data, and noise is a common cause of over-



(a) SSE, micro, $[-1, 1]$, $E_t < 0.2$

(b) CE, micro, $[-1, 1]$, $E_t < 0.5$

(c) SSE, micro, $[-10, 10]$, $E_t < 0.2$

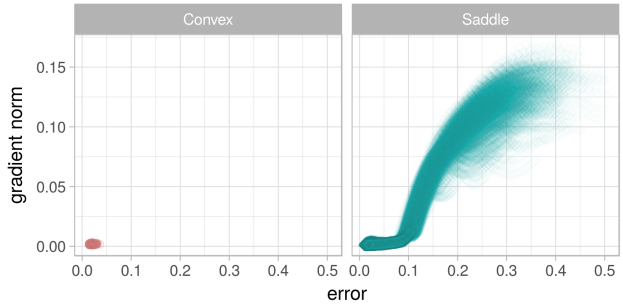
(d) CE, micro, $[-10, 10]$, $E_t < 1$

Figure 7.10: L-g clouds colourised according to the corresponding E_g values for the Diabetes problem.

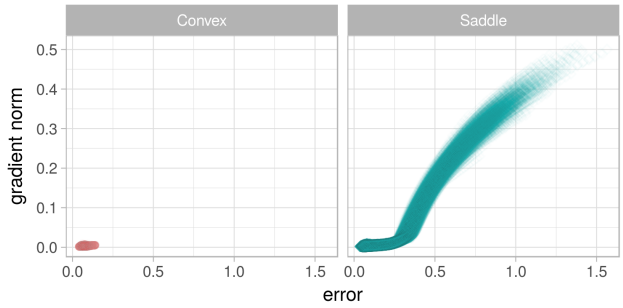
fitting. Table B.2 lists the classification errors obtained for the Diabetes problem, and shows that CE loss yielded better generalisation when sampled with the $[-1, 1]$ walks, and SSE generalised better when larger step sizes were used.

7.3.4 Glass

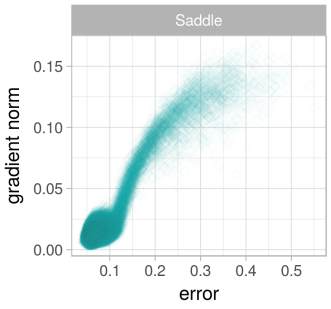
Figures 7.11 and 7.12 show the l-g clouds obtained for the Glass problem. According to Figure 7.11, convexity was found around the global minimum only, and only by the micro walks initialised in the $[-1, 1]$ range. Macro walks in the same range discovered exclusively saddle curvature points. This observation once again confirms that the search space for both SSE and CE is dominated by saddle curvature points. Convexity could only be discovered by the smallest steps tested, indicating that the convex area was sharp, and could easily be "overstepped" by a larger step size.



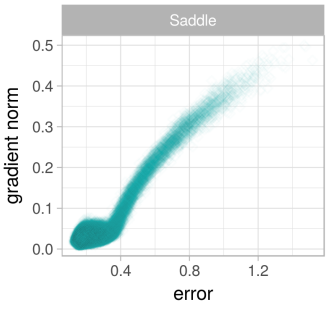
(a) SSE, micro, $[-1, 1]$



(b) CE, micro, $[-1, 1]$



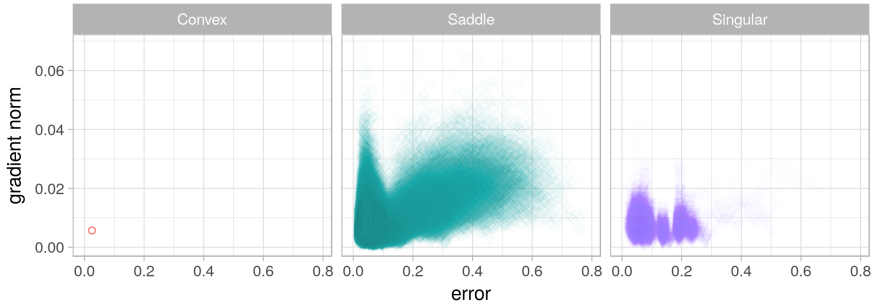
(c) SSE, macro, $[-1, 1]$



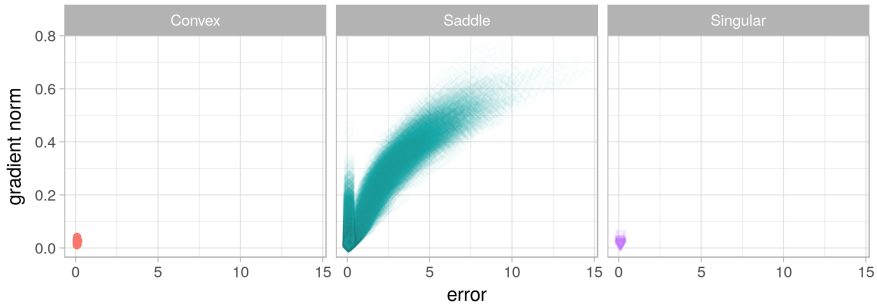
(d) CE, macro, $[-1, 1]$

Figure 7.11: L-g clouds for the gradient walks initialised in the $[-1, 1]$ range for the Glass problem.

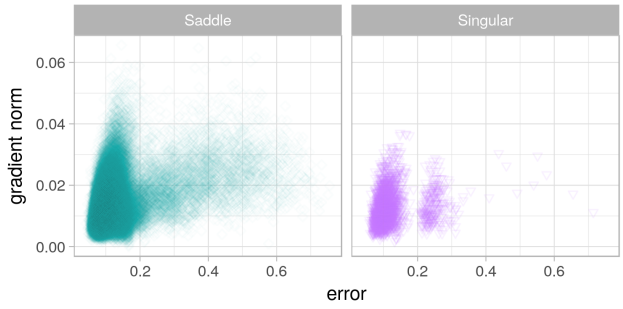
From Figure 7.11, the attractor dynamics exhibited by CE and SSE were quite similar: both losses yielded a general near-linear decline in gradient associated with a decline in error. Once the error became low enough, the gradients flattened, and a further decrease in error towards zero was performed with near-zero gradients. Both CE and SSE exhibited a single major attractor around the global minimum, indicating that all



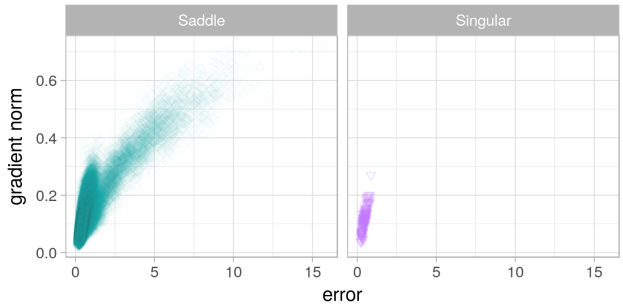
(a) SSE, micro, $[-10, 10]$



(b) CE, micro, $[-10, 10]$



(c) SSE, macro, $[-10, 10]$



(d) CE, macro, $[-10, 10]$

Figure 7.12: L-g clouds for the gradient walks initialised in the $[-10, 10]$ range for the Glass problem.

near-stationary points discovered by the walks had a similar error value. The macro steps discovered higher gradients around zero error than the micro steps, but the separation into flat and non-flat areas was still evident. This behaviour is likely to be caused by the gradient walks descending to the bottom of a valley first, and then travelling down the bottom of the valley towards a global minimum.

Table 7.4 reports the n_{stag} and l_{stag} values obtained by the various walks on the Glass problem. All walks consistently discovered only one attractor. The l_{stag} values indicate that the attractor was found within the first 10% to 20% of the steps, and from thereon the walks proceeded to explore the discovered attractor. Thus, all walks quickly descended into a valley, and then travelled at the bottom of the valley for the majority of the steps. It was clearly quite easy to find a valley, and the error values at the bottom of all discovered valleys were rather similar. No inter-valley transition was observed.

The corresponding n_{stag} and l_{stag} values obtained for E_g indicate that E_g also yielded a single attractor per walk. Standard deviations of n_{stag} and l_{stag} are higher for E_g than for E_t, indicating that a steady decrease in E_t was not always associated with a steady decrease in E_g.

To further study the generalisation behaviour of the two loss functions, Figure 7.13 shows the l-g clouds of the attractors discovered for CE and SSE, colourised according to the corresponding E_g values. According to Figures 7.13a and 7.13b, both SSE and CE exhibited a decrease in E_g associated with an initial decrease in E_t, but for both loss functions E_g increased as E_t approached zero. Thus, once an algorithm has descended into a valley, further exploitation of the valley becomes unnecessary, as far as the generalisation performance is concerned.

Figure 7.12 shows the l-g clouds obtained by the micro and macro walks initialised in the $[-10, 10]$ range. According to Figure 7.12, a larger initialisation range yielded indefinite Hessians, indicating that points of little to no curvature were discovered. A larger initialisation range is more likely to yield exploration of areas further away from the origin. Since the NNs in this study employed the sigmoid activation, the observed flatness is attributed to the saturation of the activation signals. Multiple flat attractors were observed for SSE, while CE exhibited a single major attractor. While this single attractor was at the global minimum, the sampled points clustered around two "paths":

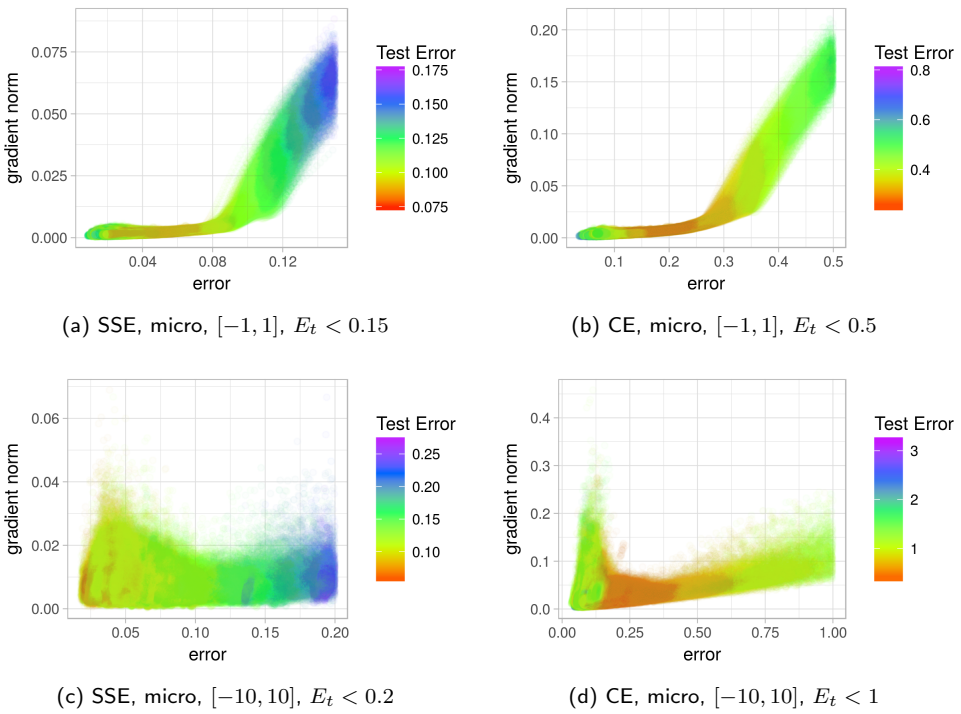
Table 7.4: Basin of attraction estimates calculated for the Glass problem. Standard deviation shown in parenthesis.

	SSE		CE	
E_t	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.00000	947.12067	1.00000	939.70667
micro	(0.00000)	(7.81525)	(0.00000)	(7.50773)
$[-1, 1]$,	1.00000	86.13867	1.00000	85.04333
macro	(0.00000)	(0.77595)	(0.00000)	(1.03672)
$[-10, 10]$,	1.04133	927.42956	1.00000	961.23867
micro	(0.20238)	(96.74308)	(0.00000)	(5.18617)
$[-10, 10]$,	1.00400	85.29156	1.00133	87.17956
macro	(0.08155)	(4.99437)	(0.05162)	(2.28182)
E_g	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.00000	951.66867	1.00800	941.96300
micro	(0.00000)	(8.18235)	(0.10925)	(42.59343)
$[-1, 1]$,	1.00000	86.67000	1.00000	86.01267
macro	(0.00000)	(0.57715)	(0.00000)	(0.71683)
$[-10, 10]$,	1.11400	902.05250	1.02733	950.75153
micro	(0.44084)	(148.80139)	(0.26568)	(73.87091)
$[-10, 10]$,	1.00533	85.01433	1.00400	86.62400
macro	(0.07283)	(6.10341)	(0.06312)	(4.83090)

lower errors associated with higher gradients, and higher errors associated with lower gradients. This indicates the presence of two structures: narrow as well as wide valleys.

It has been previously observed that wide valleys are likely to yield better generalisation performance [149, 21]. There was also a counter-argument presented, where a sharp minimum with good generalisation properties was artificially created [32]. To study the generalisation performance of the sampled points, the l-g clouds obtained for the $[-10, 10]$ micro walks, colourised according to the E_g values, are presented in Figures 7.13c and 7.13d. Figure 7.13d confirms that points of large gradient and low error

generalised poorly for CE, while points of higher error and lower gradient generalised better. Thus, points of low error exhibited overfitting for CE loss on the Glass problem, and the wide valleys have exhibited better generalisation properties. Interestingly, the same did not hold for SSE loss: according to Figure 7.13c, the smallest E_g was observed for the points of the lowest E_t. Thus, SSE loss was less prone to overfitting when sampled at the given resolution. Therefore, despite exhibiting more low gradient attractors, SSE exhibits better generalisation properties in some scenarios. The classification error values reported in Table B.3 indicate that SSE and CE have in fact performed very similarly, and have both generalised poorly. The Glass dataset is rather small, and small datasets lead to overfitting.



(a) SSE, micro, $[-1, 1]$, $E_t < 0.15$

(b) CE, micro, $[-1, 1]$, $E_t < 0.5$

(c) SSE, micro, $[-10, 10]$, $E_t < 0.2$

(d) CE, micro, $[-10, 10]$, $E_t < 1$

Figure 7.13: L-g clouds colourised according to the corresponding E_g values for the Glass problem.

7.3.5 Cancer

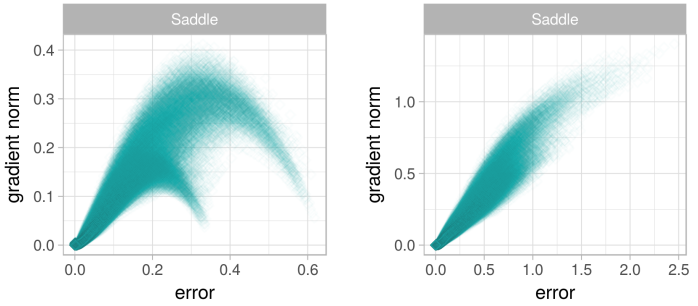
Figures 7.14 and 7.15 show the l-g clouds obtained for the Cancer problem. According to Figure 7.14, all points sampled by micro and macro walks initialised in the $[-1, 1]$ range exhibited saddle curvature. Total dimensionality of the cancer problem is 321, which is noticeably higher than that of the previous problems considered. Saddle curvature is expected to become more and more prevalent as the dimensionality increases [28].

According to Figure 7.14, both SSE and CE had one major attractor at the global minimum. In addition to the global minimum, SSE exhibited two more attractors of low, but non-zero gradient. Trajectories can be observed leading to the global minimum from either of the two high error attractors. However, there is no trajectory connecting the attractors to one another. The n_{stag} and l_{stag} values reported in Table 7.5 confirm that all walks discovered a single attractor only, thus no transition between the attractors took place.

CE, as shown in Figure 7.14, exhibited almost linear correlation between the gradient and the error. Such simple correlation implies that the CE loss surface is likely to be more searchable than the SSE loss surface from the perspective of a gradient-based optimisation algorithm. The Cancer problem is known to be an easy classification problem, which must have contributed to the simplicity of the observed attractor.

Figure 7.15 shows the l-g clouds for the micro and macro walks initialised in the $[-10, 10]$ range. The larger initialisation range once again exposed points with indefinite Hessians for both SSE and CE, i.e. points with little to no curvature. For CE, the points of no curvature aligned with the global minimum. For SSE, the global minimum, as well as the other two attractors, exhibited flatness. The majority of the points exhibited saddle curvature. The two zero-gradient attractors, away from the global minimum, were observed for the SSE loss surface. The CE loss surface did not exhibit multiple attractors. However, multiple points of high gradient close to the global minimum were sampled. This once again indicates that CE is more prone to sharp minima (narrow valleys) than SSE.

The n_{stag} and l_{stag} values yielded by E_g of the sampled points (Table 7.5) are inconsistent with the corresponding n_{stag} and l_{stag} values obtained for E_t. To further study this inconsistency, Figure 7.16 presents the l-g clouds colourised according to the E_g

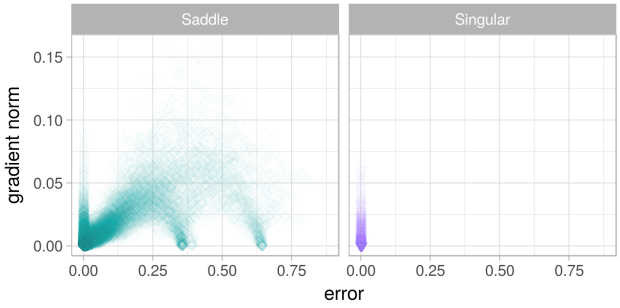


(a) SSE, micro, $[-1, 1]$ (b) CE, micro, $[-1, 1]$

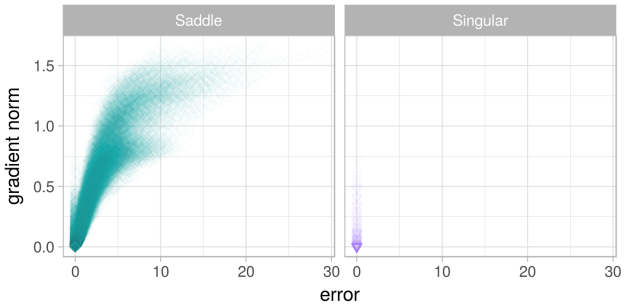
(c) SSE, macro, $[-1, 1]$ (d) CE, macro, $[-1, 1]$

Figure 7.14: L-g clouds for the gradient walks initialised in the $[-1, 1]$ range on the Cancer problem.

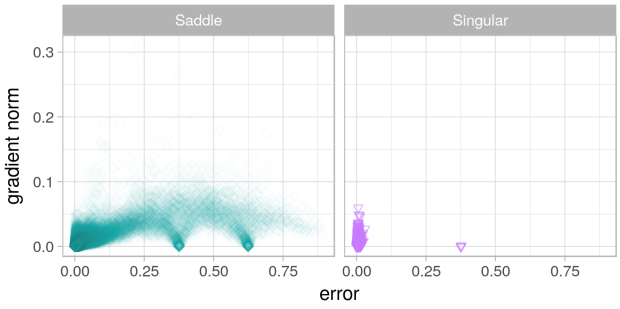
values for the points around the global minimum. Due to high disparity in the E_g values obtained for CE, the CE l-g clouds were colourised on logarithmic scale. Similar to the previous problems considered, the generalisation performance at the global optimum was poor for both SSE and CE. However, it is evident from Figures 7.16c and 7.16d that low error, high gradient points around the global minimum generalised well for SSE, and poorly for CE. SSE in general produced weaker gradients than CE, indicating that SSE was less prone to sharp minima. Figure 7.16 also shows that SSE exhibited points of zero gradient for non-zero error, while CE did not. However, the observed local minima, as well as the global minima of SSE, can yield better generalisation performance than the global minima exhibited by CE.



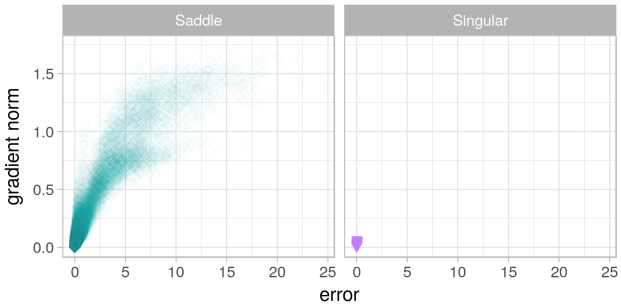
(a) SSE, micro, $[-10, 10]$



(b) CE, micro, $[-10, 10]$



(c) SSE, macro, $[-10, 10]$



(d) CE, macro, $[-10, 10]$

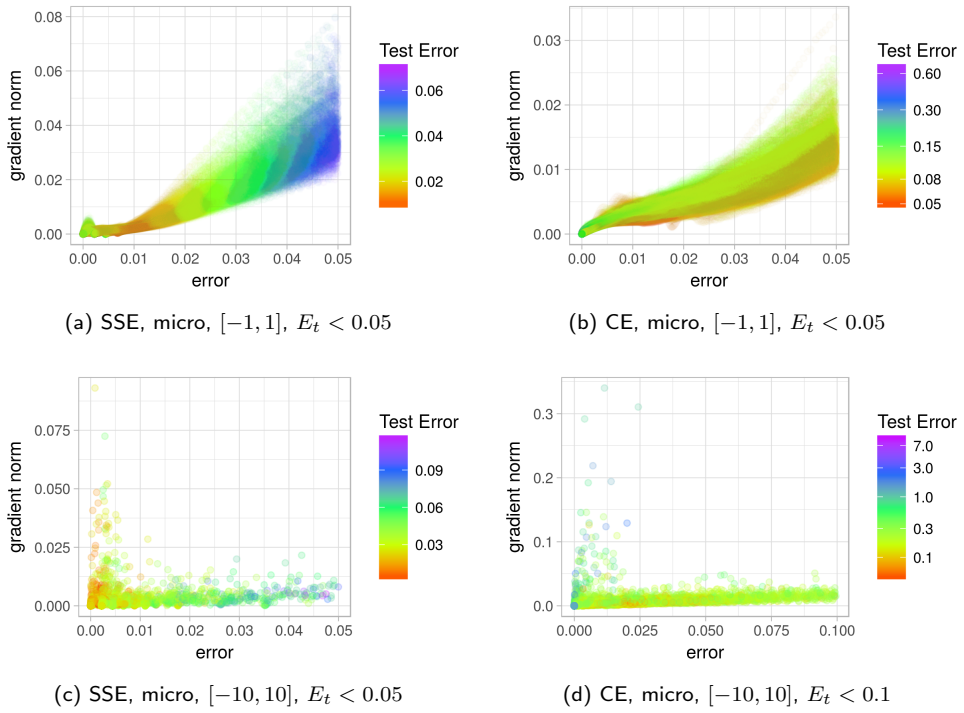
Figure 7.15: L-g clouds for the gradient walks initialised in the $[-10, 10]$ range on the Cancer problem.

Table 7.5: Basin of attraction estimates calculated for the Cancer problem. Standard deviation shown in parenthesis.

	SSE		CE	
E_t	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.00000	962.09844	1.00000	953.89307
micro	(0.00000)	(5.39294)	(0.00000)	(5.37201)
$[-1, 1]$,	1.00000	87.77788	1.00000	87.18816
macro	(0.00000)	(0.44464)	(0.00000)	(0.51044)
$[-10, 10]$,	1.00000	972.77778	1.00000	975.43836
micro	(0.00000)	(7.45025)	(0.00000)	(3.01133)
$[-10, 10]$,	1.00000	87.44517	1.00000	87.80498
macro	(0.00000)	(0.89423)	(0.00000)	(1.12240)
E_g	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.00000	959.38629	1.00725	953.80766
micro	(0.00000)	(5.62718)	(0.09002)	(41.40420)
$[-1, 1]$,	1.00000	87.81838	1.00000	87.25514
macro	(0.00000)	(0.41360)	(0.00000)	(0.54000)
$[-10, 10]$,	1.11111	932.44444	4.13699	541.93665
micro	(0.45812)	(154.23691)	(4.50666)	(384.89318)
$[-10, 10]$,	1.00125	87.16246	1.00903	86.55711
macro	(0.03528)	(2.95538)	(0.09786)	(7.86466)

7.3.6 Heart

Figures 7.17 and 7.18 show the l-g clouds obtained for the Heart problem. Similar to the Cancer problem, all points sampled by the $[-1, 1]$ walks were classified as saddle points. The total dimensionality of the heart problem is 341, which is similar to the dimensionality of the Cancer problem. Figure 7.17 illustrates that both SSE and CE had a single flat attractor in the general area of the global minimum. In addition to this attractor, SSE exhibited two more attractors of much higher error. However, the



(a) SSE, micro, $[-1, 1]$, $E_t < 0.05$

(b) CE, micro, $[-1, 1]$, $E_t < 0.05$

(c) SSE, micro, $[-10, 10]$, $E_t < 0.05$

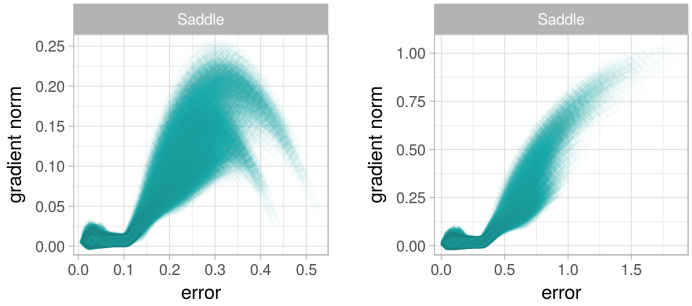
(d) CE, micro, $[-10, 10]$, $E_t < 0.1$

Figure 7.16: L-g clouds colourised according to the corresponding E_g values for the Cancer problem.

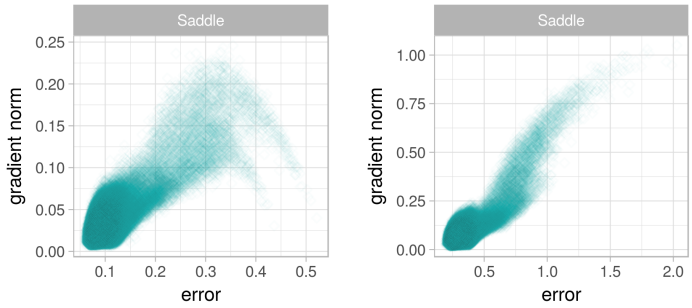
$[-1, 1]$ walks did not sample any zero-gradient (stationary) points around the high error attractors.

Larger steps and a larger initialisation range, however, allowed gradient walks to discover the stationary points of high error on the SSE loss surface, as illustrated in Figure 7.18. The CE loss surface sampled by the same walks did not reveal any additional attractors, but was again visibly split into two clusters leading towards the global minimum attractor: points of high gradient and low error, and points of lower gradient and higher error. This is once again indicative of narrow and wide valleys, which appears to be a common characteristic of the CE loss surface.

The n_{stag} and l_{stag} values reported in Table 7.6 confirm that the walks generally did not make transitions between the discovered attractors. The n_{stag} and l_{stag} values calculated over the E_g values were again less stable than the corresponding E_t values, indicating that exploiting an E_t attractor does not necessarily coincide with exploiting



(a) SSE, micro, $[-1, 1]$ (b) CE, micro, $[-1, 1]$



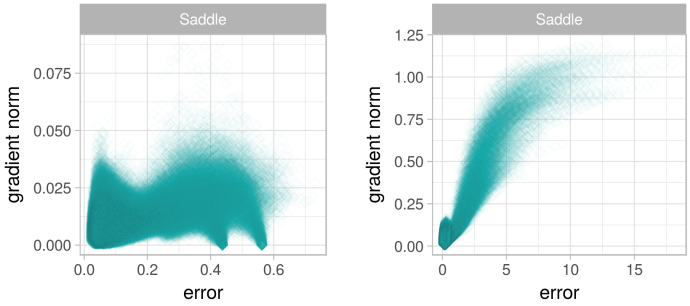
(c) SSE, macro, $[-1, 1]$ (d) CE, macro, $[-1, 1]$

Figure 7.17: L-g clouds for the gradient walks initialised in the $[-1, 1]$ range on the Heart problem.

a corresponding E_g attractor. Figure 7.19 illustrates the generalisation behaviour of the flat attractor discovered on both the SSE and CE loss surfaces by the micro $[-1, 1]$ walks: the smallest E_g values were observed on the rightmost side of the attractor, closest to the points of higher error and higher gradient. Exploitation around the global minimum yielded superior E_t values, but inferior E_g values. This once again illustrates that discovering the global optimum may be unnecessary. The success of techniques such as early stopping [93] comes precisely from preventing the algorithm from exploiting a global minimum unnecessarily.

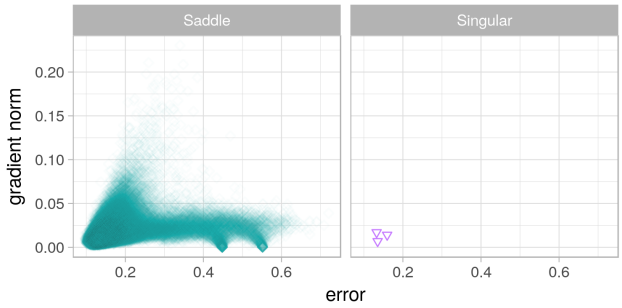
7.3.7 MNIST

Figures 7.20 and 7.21 show the l-g clouds for the MNIST problem. Due to the prohibitively expensive memory requirements, the Hessian matrices were not computed for

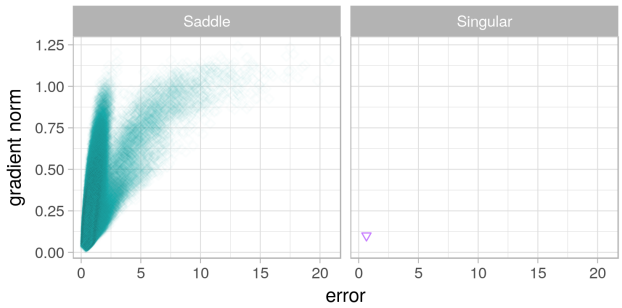


(a) SSE, micro, $[-10, 10]$

(b) CE, micro, $[-10, 10]$



(c) SSE, macro, $[-10, 10]$



(d) CE, macro, $[-10, 10]$

Figure 7.18: L-g clouds for the gradient walks initialised in the $[-10, 10]$ range on the Heart problem.

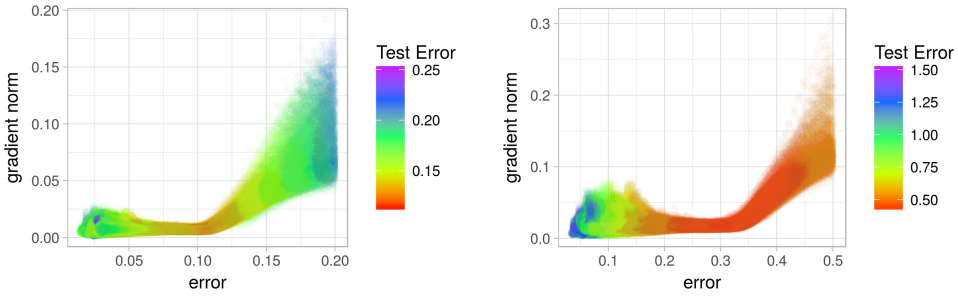
the MNIST dataset. Thus, the curvature of the loss functions for the MNIST dataset is not reported in this study. The reader is referred to the previous studies of the MNIST Hessians [114] for a discussion of curvature characteristics, where it was shown that the gradient descent algorithm discovered points of saddle and singular curvature only.

Table 7.6: Basin of attraction estimates calculated for the Heart problem. Standard deviation shown in parenthesis.

	SSE		CE	
E_t	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.00000	952.36276	1.00000	947.81432
micro	(0.00000)	(6.49124)	(0.00000)	(7.04222)
$[-1, 1]$,	1.00000	86.70645	1.00000	86.40587
macro	(0.00000)	(0.66921)	(0.00000)	(0.97088)
$[-10, 10]$,	1.02493	937.01486	1.00000	966.72036
micro	(0.15962)	(76.63092)	(0.00000)	(3.70200)
$[-10, 10]$,	1.00176	84.85293	1.00411	84.43886
macro	(0.04191)	(3.68841)	(0.06394)	(7.00978)
E_g	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.00733	957.87269	2.14920	710.77930
micro	(0.10669)	(40.88520)	(2.07157)	(342.24282)
$[-1, 1]$,	1.00000	87.70880	1.00088	87.54971
macro	(0.00000)	(0.60395)	(0.02965)	(1.97717)
$[-10, 10]$,	1.54927	821.66135	1.00298	965.68084
micro	(1.78543)	(251.33524)	(0.05453)	(27.96970)
$[-10, 10]$,	1.00440	84.55381	1.04956	79.23624
macro	(0.06618)	(5.61240)	(0.24735)	(17.04847)

Figure 7.20 shows that both SSE and CE exhibited one major attractor around the global minimum. Additionally, SSE exhibited two more attractors of non-zero gradient. Thus, the error landscape of CE was more searchable than the error landscape of SSE. The n_{stag} and l_{stag} results reported in Table 7.7 indicate that most walks have discovered a single attractor only, which corresponds to the results in Figures 7.20 and 7.21.

A cluster of values of high gradient and low error can be observed for both SSE and CE, indicating that both exhibited sharp minima. SSE, however, exhibited lower gradients overall. Figure 7.20 illustrates that the generalisation performance improved



(a) SSE, micro, $[-1, 1]$, $E_t < 0.2$ (b) CE, micro, $[-1, 1]$, $E_t < 0.5$

Figure 7.19: L-g clouds colourised according to the corresponding E_g values for the Heart problem.

as the error approached the global minimum. Figure 7.22 shows the generalisation performance of the points sampled around the global minimum. SSE once again exhibited better generalisation performance around the global minimum than CE, confirming the earlier made hypothesis that SSE is less prone to overfitting due to weaker gradients. The classification error results reported in Table B.6, however, indicate that, although SSE yielded a smaller disparity between the E_t and E_g values, both loss functions performed similarly in terms of final classification.

Figure 7.21 indicates that SSE exhibited a much weaker correlation between the gradient and the error when sampled by gradient walks initialised in the $[-10, 10]$ interval. For CE, the positive correlation was still clearly manifested. Thus, CE exhibited a more searchable landscape when sampled by the $[-10, 10]$ walks. Both loss functions clustered along the error and the gradient axis.

The landscape properties exhibited by the MNIST problem were thus very similar to the landscape properties exhibited by the problems of lower dimensionality. The CE loss surface was more searchable for all problems considered, and exhibited fewer non-global attractors. SSE, however, exhibited somewhat better generalisation capabilities under some of the considered scenarios. Perhaps the two loss functions should be combined to construct an error landscape that is both searchable and robust to overfitting.

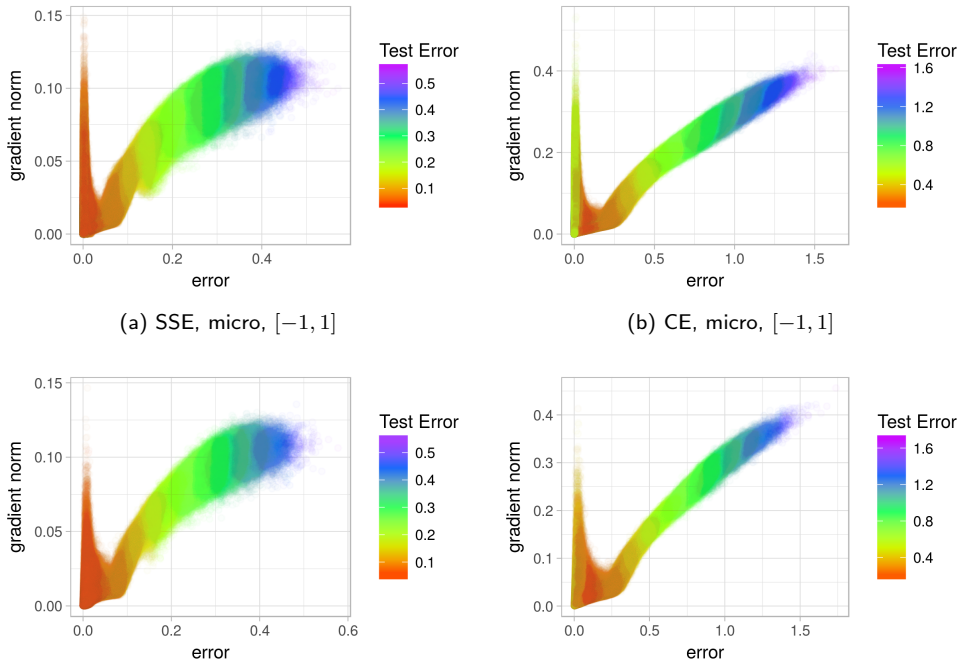
Table 7.7: Basin of attraction estimates calculated for the MNIST problem. Standard deviation shown in parenthesis.

	SSE		CE	
E_t	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.00003	948.96269	1.00020	943.65445
micro	(0.00557)	(7.73257)	(0.01418)	(10.31210)
$[-1, 1]$,	1.00000	89.59761	1.00000	88.94583
macro	(0.00000)	(0.62686)	(0.00000)	(0.79698)
$[-10, 10]$,	1.00338	944.23606	1.00020	955.48716
micro	(0.06420)	(29.25325)	(0.01418)	(9.14026)
$[-10, 10]$,	1.00004	90.19884	1.00001	90.24536
macro	(0.00614)	(1.02171)	(0.00354)	(1.09101)
E_g	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.00028	944.25561	2.84430	570.85913
micro	(0.01762)	(10.01749)	(2.71734)	(334.02547)
$[-1, 1]$,	1.00000	90.04197	1.01201	85.19988
macro	(0.00000)	(0.63536)	(0.11234)	(7.54223)
$[-10, 10]$,	1.00408	943.53542	1.26670	878.55561
micro	(0.11775)	(29.40333)	(1.10976)	(191.85100)
$[-10, 10]$,	1.00005	90.31260	1.00881	88.80587
macro	(0.00709)	(1.00828)	(0.10216)	(6.14361)

7.4 Ruggedness, Gradients, Neutrality

The ruggedness, gradient, and neutrality metrics, as described in Section 3.4, were calculated for the benchmarks considered to gain more insight into the fitness landscape properties of the two loss functions. Magnitudes of the numerical gradients were used instead of gradient estimates. Figures 7.23 and 7.24 show the FEM and gradient metrics obtained.

It is evident from Figures 7.23 and 7.24 that ruggedness increased with an increase



(a) SSE, micro, $[-1, 1]$

(b) CE, micro, $[-1, 1]$

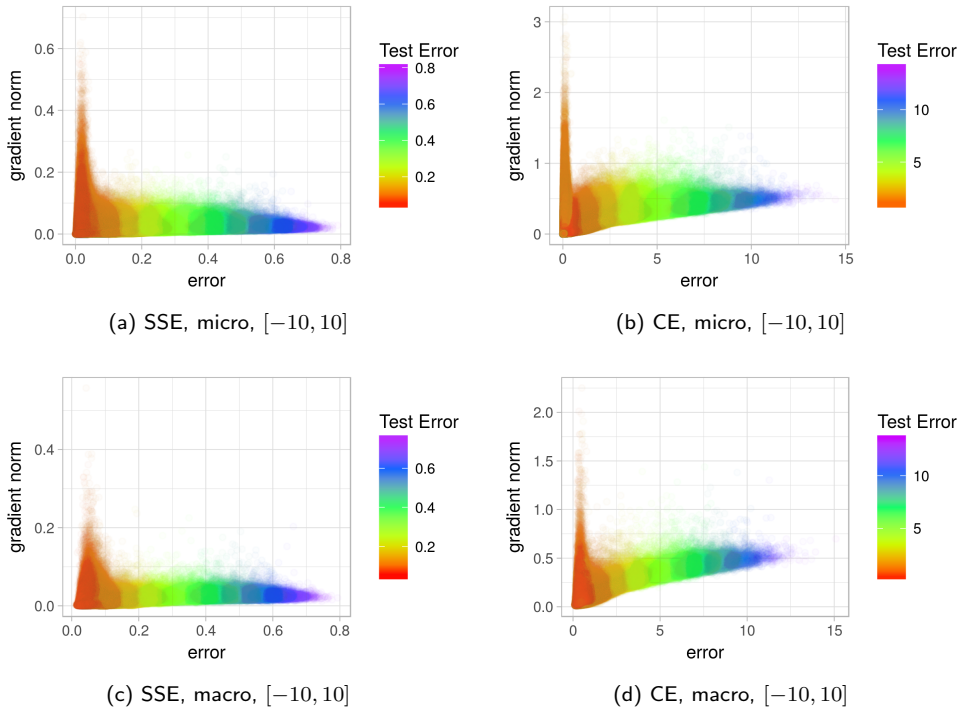
(c) SSE, macro, $[-1, 1]$

(d) CE, macro, $[-1, 1]$

Figure 7.20: L-g clouds for the gradient walks initialised in the $[-1, 1]$ range on the MNIST problem.

in step size and initialization boundaries for all problems considered. This result is interesting, because the loss surfaces are expected to become more and more flat as the search moves further away from the origin, due to hidden unit saturation. However, the reported metrics were calculated over gradient-guided walks, thus the ruggedness observed corresponded to the gradient-based trajectories. Therefore, increased ruggedness can be explained by the gradient trajectory bouncing off the walls of a ravine, with local/global minima at the bottom. Larger step sizes were more likely to overstep the width of a ravine, and thus caused stronger oscillations.

A decrease in FEM associated with the transition from the $[-1, 1]$ macro setting to the $[-10, 10]$ micro setting can be observed for multiple problems in Figures 7.23 and 7.24. Both settings used the same maximum step size, since 10% of the $[-1, 1]$ range is equal to the 1% of the $[-10, 10]$ range. Therefore, the difference in FEM values resulted from the different initialisation intervals. It has been observed in the past



(a) SSE, micro, $[-10, 10]$

(b) CE, micro, $[-10, 10]$

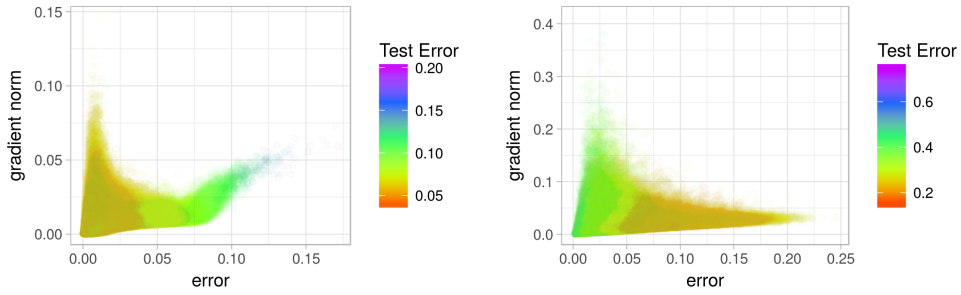
(c) SSE, macro, $[-10, 10]$

(d) CE, macro, $[-10, 10]$

Figure 7.21: L-g clouds for the gradient walks initialised in the $[-10, 10]$ range on the MNIST problem.

that SSE is likely to not find a good solution if initialised poorly [46]. Smaller FEM values indicate that a larger initialisation range caused the gradient walks to exploit different structures when initialised in the $[-10, 10]$ range. Fewer oscillations for the same step size indicate that shallower and wider valleys were discovered for the $[-10, 10]$ micro setting. The classification values reported in Appendix B indicate that $[-10, 10]$ micro walks sometimes discovered solutions with better generalisation properties than the ones discovered by the $[-1, 1]$ macro walks. This corresponds with the previously made hypothesis that wide valleys generalise better than narrow valleys [21].

While the two loss functions exhibited similar behaviour in terms of FEM, the same can not be said about the gradients. Figures 7.23 and 7.24 indicate that CE yielded higher G_{avg} and G_{dev} values for all problems considered. This observation is in line with the theoretical predictions made in [120]. Stronger gradients are responsible for faster convergence of gradient-based methods on CE compared to SSE. Figures 7.23

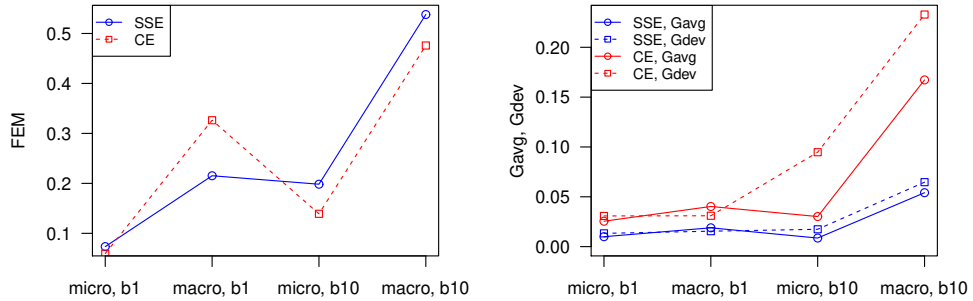


(a) SSE, macro, $[-10, 10]$, $E_t < 0.2$ (b) CE, macro, $[-10, 10]$, $E_t < 0.25$

Figure 7.22: L-g clouds colourised according to the corresponding E_g values for the MNIST problem.

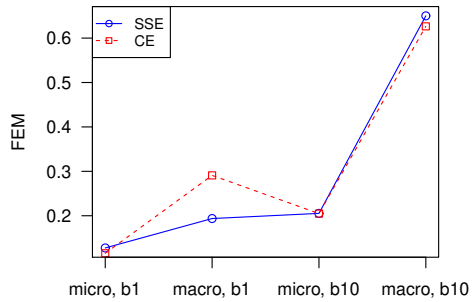
and 7.24 also show that the G_{avg} and G_{dev} values increased for CE with an increase in the initialisation boundaries, which was not the case for SSE. Thus, from the gradient perspective, the CE loss surface exhibited a stronger global shape, and contained more gradient information to guide a search. The G_{dev} values were similar or smaller than the corresponding G_{avg} values for most problems considered, with the exception of XOR ($[-10, 10]$ setting) and Cancer (all settings). For these two problems, G_{dev} exceeded G_{avg}, indicating sudden step-like changes in the landscape. The XOR problem contains four data patterns only, and is thus expected to be ill-defined. Cancer is known to be a trivial classification problem, solvable with over 95% accuracy by NNs with a single hidden layer [9]. The simplicity of the dataset clearly contributed to the simplicity of the landscape. Cancer also exhibited the smallest FEM values compared to the other datasets, indicating that the loss surface associated with this problem was smooth.

Finally, the neutrality metrics $M1$ and $M2$ calculated for all problems considered are shown in Figure 7.25. Surprisingly, none of the problems, except the Cancer problem, exhibited much neutrality. This observation indicates that the loss surfaces were generally highly searchable, and the gradient walks did not have to traverse flat areas. Lack of neutrality is also indicative of oscillations exhibited by the walks, attributed to the fact that minima is located at the bottom of ravines ans valleys, and the ravines can have steep walls. The Cancer problem is the only exception to this rule. The flatness observed on the Cancer dataset is again attributed to the triviality of the dataset. If the

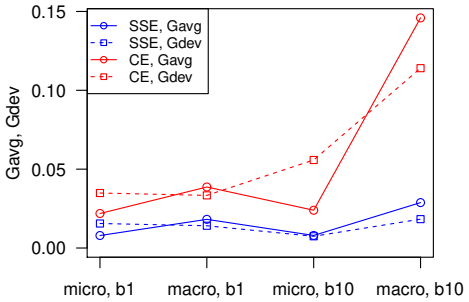


(a) XOR, FEM

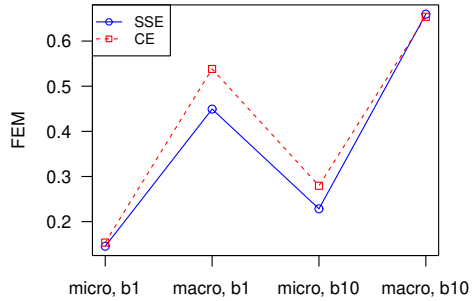
(b) XOR, gradients



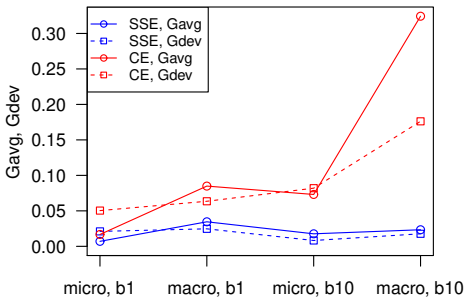
(c) Iris, FEM



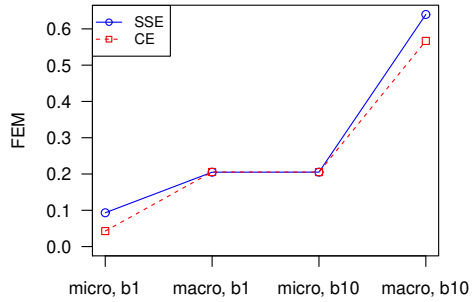
(d) Iris, gradients



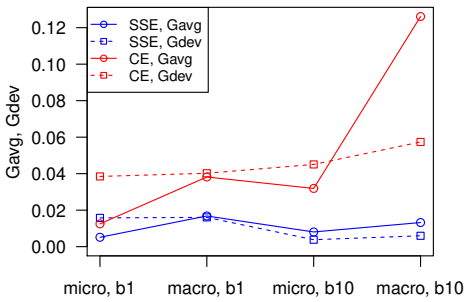
(e) Diabetes, FEM



(f) Diabetes, gradients

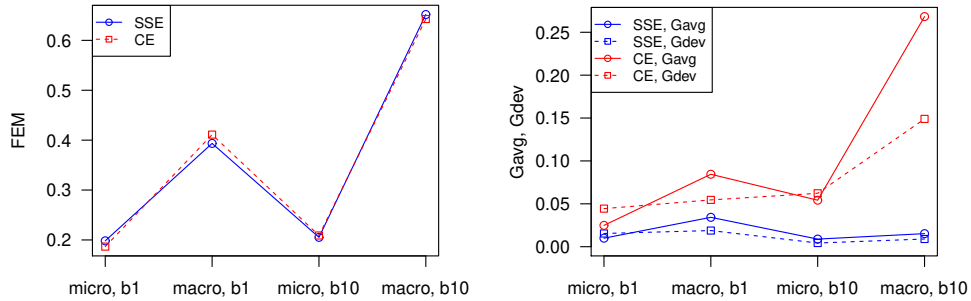


(g) Glass, FEM



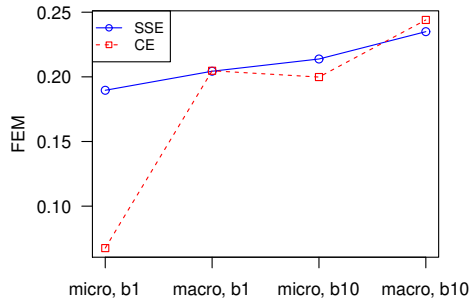
(h) Glass, gradients

Figure 7.23: FLA metrics for the XOR, Iris, Diabetes, and Glass problems.

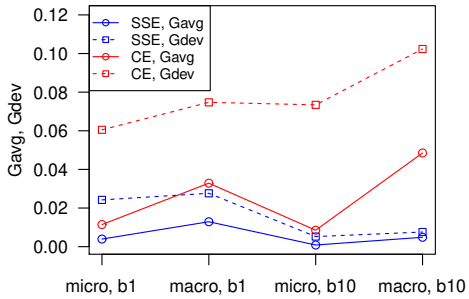


(a) Heart, FEM

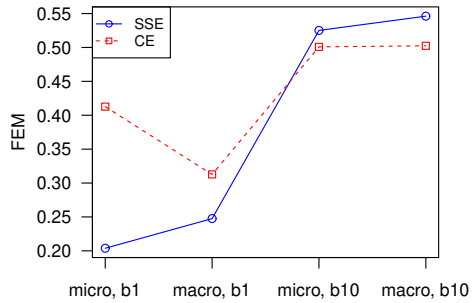
(b) Heart, gradients



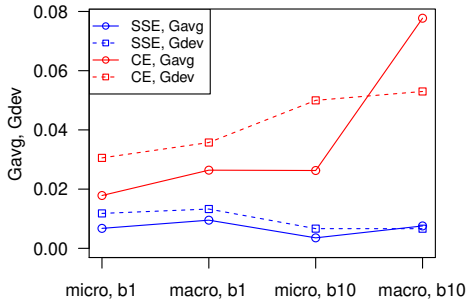
(c) Cancer, FEM



(d) Cancer, gradients



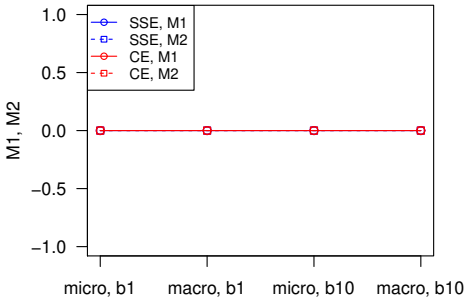
(e) MNIST, FEM



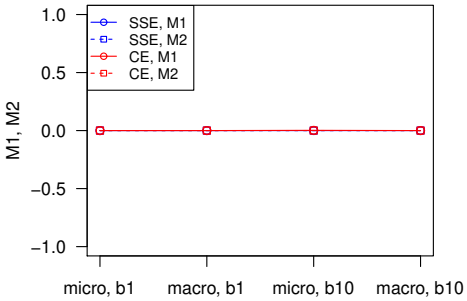
(f) MNIST, gradients

Figure 7.24: FLA metrics for the Heart, Cancer, and MNIST problems.

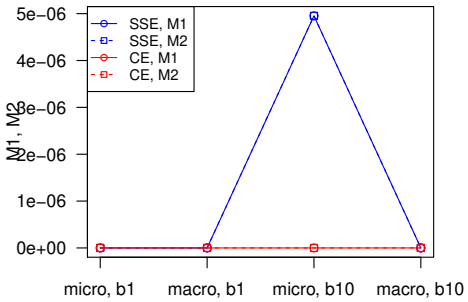
valleys were wide enough, the gradient walks would have exhibited fewer oscillations. Perhaps valley width is associated with the separability of a problem.



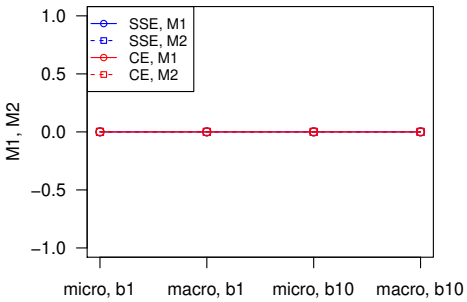
(a) XOR, neutrality



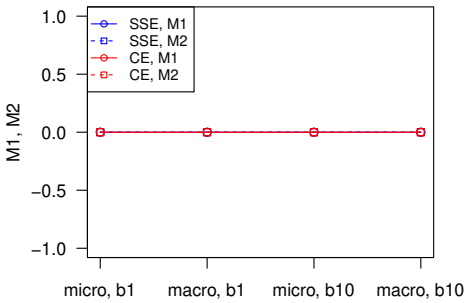
(b) Iris, neutrality



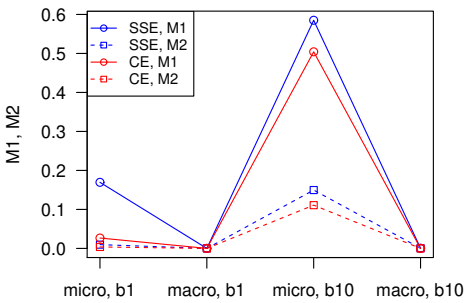
(c) Diabetes, neutrality



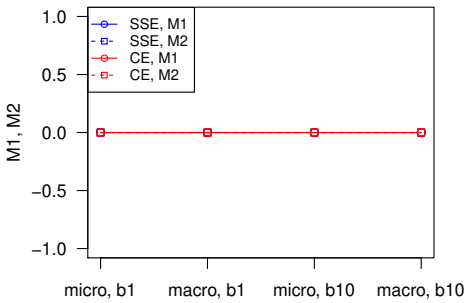
(d) Glass, neutrality



(e) Heart, neutrality



(f) Cancer, neutrality



(g) MNIST, neutrality

Figure 7.25: Neutrality metrics for the various problems considered.

7.5 Conclusions

This study presented a visual and numerical analysis of local minima and the associated basins of attraction for two common NN loss functions, i.e. quadratic loss and entropic loss. The study was performed by analysing the samples obtained by a number of progressive gradient walks proportionate to the dimensionality of the problems. The gradient walks were not restricted to any specific search space bounds, but were initialised in two distinct intervals, i.e. $[-1, 1]$ and $[-10, 10]$. Additionally, two granularity settings were considered for the gradient walks, namely micro and macro.

Both loss functions exhibited convex local minima for the XOR problem. The amount of observed convexity decreased with an increase in problem dimensionality. Saddle curvature was the most prevalent curvature observed, and some higher-dimensional problems considered exhibited only saddle curvature for all sampled points.

SSE consistently exhibited more local stationary points and associated attractors than CE. Analysis of the individual walks further revealed that transition between different attractors was unlikely, and that the paths connecting different attractors exhibited singular Hessian matrices, indicative of flatness. Thus, CE exhibited a more consistent and searchable structure across the selection of problems considered in this study.

With an increase in problem dimensionality, the number of zero or low gradient attractors decreased. The majority of the problems exhibited a single main attractor around the global optimum. For CE, the gradient was for the most part positively correlated to the error value, indicating that the CE loss surface is highly searchable from the perspective of gradient-based methods. This study did not attempt to quantify the number of optima, but the obtained results clearly indicated that the majority of the optima exhibited similar loss values.

The results confirmed previously made observations of the presence of valley-shaped optima in NN error landscapes. For the majority of the problems, descending into a valley was easily accomplished by the walks. Travelling down the bottom of the valley towards the global minimum yielded a decrease in generalisation performance for both SSE and CE. CE exhibited stronger gradients than SSE in all experiments conducted, and the stronger gradients promoted overfitting in CE. For some of the problems, SSE exhibited better generalisation performance. It can be speculated that the CE loss

surface is more prone to sharp minima (narrow valleys) than SSE; thus, CE is more easily overfitted. The experiments revealed the tendency for the points sampled on CE to fall into two major clusters: points of low error and high gradients, and points of higher error and low gradients. These are hypothesised to represent narrow and wide valleys, respectively. The results of this study confirmed that superior generalisation performance was exhibited by the points in the wide valleys.

FLA estimates of ruggedness, gradients, and neutrality were also calculated for the two losses. The ruggedness results suggested that the discovered valleys became wider further away from the origin. The gradient results confirmed that CE exhibited stronger gradients than SSE, as well as a more searchable global shape, with steeper gradients further away from the origin. Neutrality was observed for the Cancer problem only, indicating that the width of the discovered valleys may be associated with the separability of the dataset.

An analysis of the progressive gradient samples thus illustrated a number of current theories regarding the shape of NN error surfaces, and highlighted the differences between SSE and CE loss surfaces, confirming that FLA, and specifically the l-g clouds proposed in this study, provide a viable method for visualisation and analysis of NN error landscapes.

The observation that the SSE landscape may have superior generalisation properties suggests that a hybrid of SSE and CE may produce a landscape that combines the searchability of CE with the robustness of SSE. Additionally, the presence of a single attractor in the majority of the problems considered suggests that an exploitative rather than an exploratory approach should be taken for the purpose of NN training. This observation has strong implications for population-based training algorithms, which so far failed to be effectively applied to high-dimensional NN training problems. A population-based approach designed with exploitation rather than exploration in mind may perform competitively, especially if gradient information is used as one of the guides for the population. This hypothesis is further supported by a recent study of PSO in high-dimensional spaces [102], were the efficacy of exploitation over exploration in high-dimensional spaces was observed. Investigation of exploitative population-based techniques applied to NNs is an interesting topic for future research.

Another interesting observation is the impressive ability of a randomised algorithm to find the global optima, when guided by nothing besides the direction of the gradient. As Appendix B indicates, the average classification error calculated at the last step of the gradient walks approached 100% accuracy on most problems under at least one of the granularity settings. Perhaps gradient-guided stochastic training algorithms should be considered for deeper, more complex problems.

The next chapter presents an FLA study of NN error landscapes exhibited by various activation functions.

Chapter 8

Activation Functions

Non-linear activation functions enable multilayer NNs to model arbitrary complex mappings between inputs and outputs [53], and thus play a crucially important role in NN design, training, and performance. Kordos and Duch [68] have shown that the activation function used in the hidden layers of a NN has a significant effect on the resulting loss surface, but their study was limited to monotone (sigmoidal) and non-monotone bounded functions. Kordos and Duch [68] concluded that non-monotone activation functions yield more complex loss surfaces than the monotone activation functions, and that non-smooth monotone activation functions introduce more plateaus than smooth monotone activation functions.

Bounded activation functions such as sigmoid and TanH, discussed in Section 2.5, are prone to saturation, which was shown to be detrimental to NN performance for shallow [109] and deep [51, 52] architectures alike. Modern activation functions such as ReLU and ELU, also discussed in Section 2.5, are less prone to saturation, and thus became the primary choice in the deep learning community [4, 117, 152]. However, the effect of the various activation functions on the resulting NN loss surfaces has not been established yet. Recent studies have shown that, on a bounded set in the weight space, the ReLU activation function yields strong piece-wise convexity around the isolated local minima on NN loss surfaces [89, 111]. However, these theoretical studies rely on numerous limiting assumptions. To the best of the author's knowledge, no dedicated studies of the loss surfaces associated with the ELU activation or the softmax activation exist to date.

This chapter aims to investigate NN loss surfaces under various activation functions using the FLA techniques. The rest of the chapter is structured as follows: Section 8.1 discusses the experimental procedure. Section 8.2 presents a visual and numerical analysis of stationary points and basins of attraction associated with various activation functions. Section 8.3 presents FLA measures of gradients, ruggedness, and neutrality associated with the various activation functions. Section 8.4 concludes the chapter.

8.1 Experimental Procedure

The aim of the study was to visually and numerically investigate the local minima and basins of attraction exhibited by a diverse set of activation functions. This section discusses the experimental set-up of the study, and is structured as follows: Section 8.1.1 describes the NN hyperparameters and activation functions considered in this study, Section 8.1.2 lists the benchmark problems used, and Section 8.1.3 outlines the sampling algorithm parameters.

8.1.1 Activation functions

All experiments employed feed-forward NNs with a single hidden layer. The entropic loss function (CE), discussed in Section 2.4, was used in the experiments. The entropic loss was chosen based on the findings presented in Chapter 7, where CE was shown to produce more searchable landscapes with fewer local minima. The following three activation functions, previously discussed in Section 2.5, were considered for the hidden neurons:

1. Hyperbolic tangent, TanH.
2. Rectified linear units, ReLU.
3. Exponential linear units, ELU.

The three activation functions were chosen based on their widespread success in modern NN architectures [116, 117, 150]. For the output layer neurons, the sigmoid activation

function was used for binary classification problems. For multinomial classification problems, both the sigmoid and softmax activation functions were considered in the output layer.

8.1.2 Benchmark problems

For the purpose of this study, the seven classification problems described in Appendix A were used, of which four were examples of binary classification (XOR, Diabetes, Cancer, Heart), and three were examples of multinomial classification (Iris, Glass, MNIST).

8.1.3 Sampling parameters

The same sampling parameters as discussed in Section 7.2.3 were used for the experiments. Progressive gradient walks (refer to Section 6.1.1) were used for the purpose of sampling. The total number of walks was set to be $10 \times m$, where m is the dimensionality of the search space. The walks were unbounded, but two distinct initialisation ranges were considered, namely $[-1, 1]$ and $[-10, 10]$. Two granularity settings were used throughout the experiments: micro, where the maximum step size was set to 1% of the initialisation range, and macro, where the maximum step size was set to 10% of the initialisation range. Micro walks performed 1000 steps each, and macro walks performed 100 steps each. All datasets except XOR were split into 80% training and 20% test subsets. To calculate the training (E_t) and the generalisation (E_g) errors, the entire train/test subsets were used for all the problems except MNIST. For MNIST, random batches of 100 patterns were sampled from the respective training and test sets.

8.2 Empirical Study of Modality

This section presents an analysis of apparent local minima and the corresponding basins of attraction as captured by the progressive gradient walks. L-g clouds, proposed in Chapter 6, are employed for the purpose of this study. Sections 8.2.1 to 8.2.7 present the analysis of the three different hidden neuron activation functions for each problem.

Section 8.2.8 presents the analysis of the softmax activation for the multinomial classification problems.

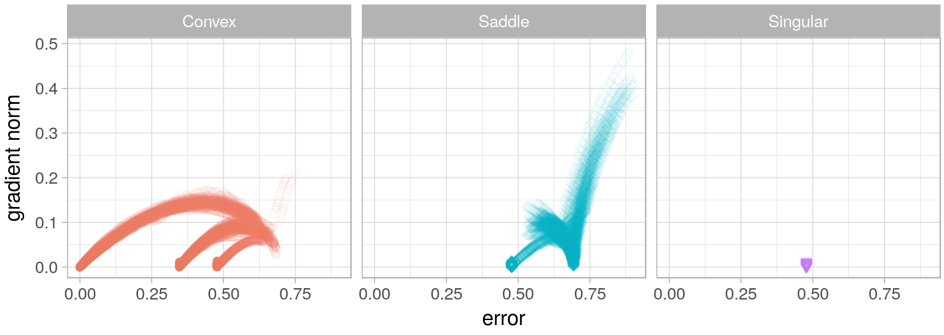
8.2.1 XOR

Figures 8.1, 8.2, 8.3, and 8.4 show the l-g clouds obtained for the XOR problem, separated into panes according to the curvature.

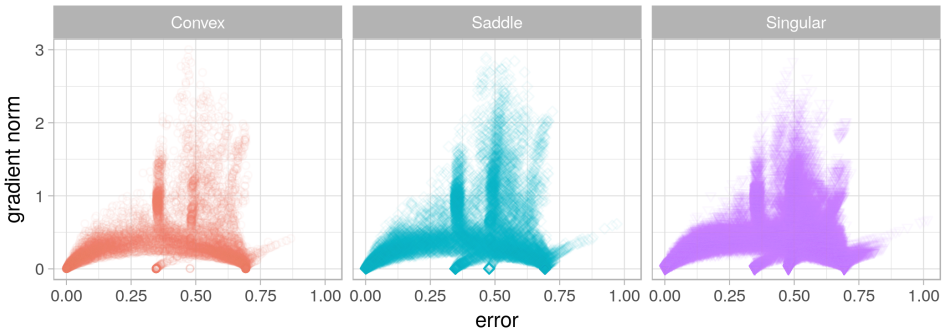
Figure 8.1 indicates that the three activation functions considered have all yielded local minima similar to the minima discovered on the sigmoid error surface (refer to Fig. 7.1 in Chapter 7). However, the loss surface characteristics around the local minima varied for the three activation functions. TanH yielded stronger gradients than the sigmoid, which is to be expected, but otherwise exhibited a loss surface with a clear transition between saddle and convex curvatures. The gradient walks generally descended to a stationary point in the saddle space, and then made the transition to one of the convex minima from there. The three convex minima discovered were disconnected, i.e. the walks generally did not make transitions between the convex minima. This observation is confirmed by the n_{stag} and l_{stag} values reported in Table 8.1.

Table 8.1: Basin of attraction estimates calculated for the XOR problem for the different hidden neuron activation functions. Standard deviation is shown in parenthesis.

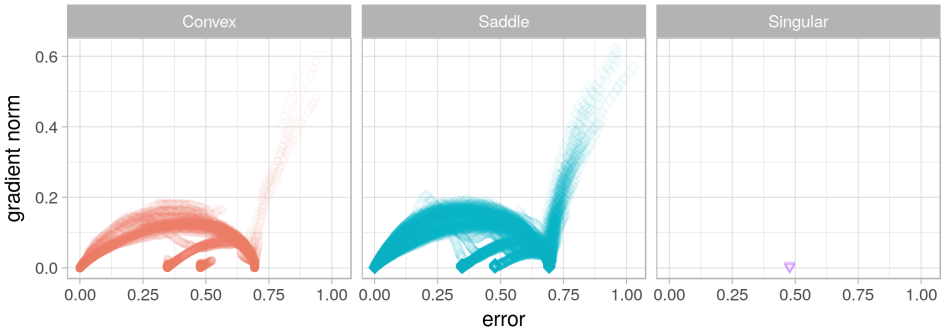
	TanH		ReLU		ELU	
	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$, micro	1.3667 (0.4819)	623.1000 (188.8277)	5.3778 (4.7972)	257.4255 (258.8352)	1.7000 (0.5467)	492.9157 (210.9447)
$[-1, 1]$, macro	1.0333 (0.1795)	60.1833 (15.7269)	1.1778 (0.5692)	26.2259 (17.7395)	1.1778 (0.3823)	47.8111 (18.3239)
$[-10, 10]$, micro	1.1222 (0.3599)	905.0130 (158.2318)	1.7556 (2.1925)	818.8688 (276.1865)	1.0889 (0.3542)	919.2556 (135.3812)
$[-10, 10]$, macro	1.0222 (0.3326)	71.3333 (26.2463)	1.1889 (0.5753)	46.0889 (32.4271)	1.0889 (0.3542)	69.0389 (22.8096)



(a) TanH, micro, $[-1, 1]$



(b) ReLU, micro, $[-1, 1]$



(c) ELU, micro, $[-1, 1]$

Figure 8.1: L-g clouds for the micro gradient walks initialised in the $[-1, 1]$ range for the XOR problem.

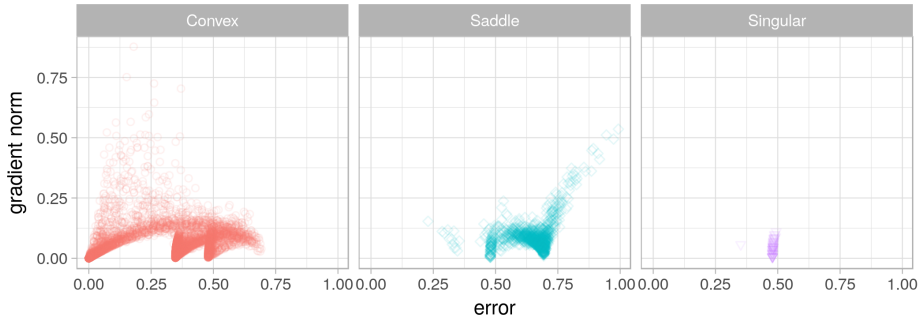
The loss surface characteristics associated with ReLU were vastly different from the sigmoidal loss surfaces. Firstly, as can be seen in Figure 8.1b, there was no clear separation between convex, saddle, and indefinite curvatures in the different areas sampled for

ReLU. Secondly, even though two convex local minima were discovered, these minima were apparently less attractive to the gradient walks than the global minimum. This observation indicates that a gradient-based algorithm would be less likely to become trapped in a local minimum exhibited by the ReLU activation function. The reason for this is the presence of saddle and indefinite (flat) points around the local minima, providing pathways through which a gradient-based algorithm can escape. Additionally, vertical clusters can be observed around the local minima, indicating that diverse gradient information was available in the neighbourhood of the local minima. The n_{stag} and l_{stag} values reported in Table 8.1 confirm that the gradient walks generally exhibited multiple transitions between the stagnant areas. The predominance of indefinite curvature is explained by the fact that the ReLU function outputs zero for all negative inputs, which inevitably causes flatness.

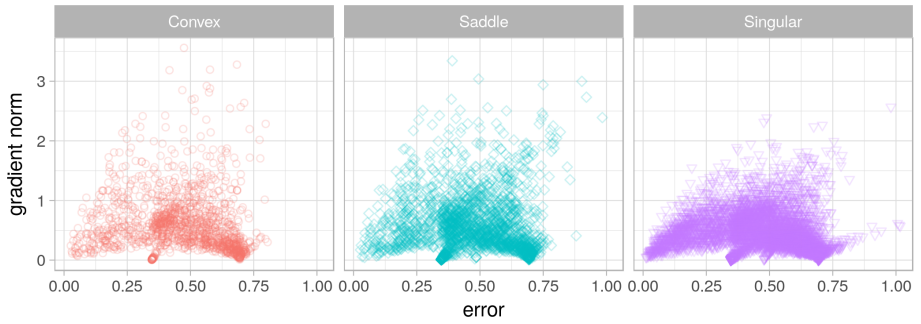
The ELU activation yielded a loss surface similar to the sigmoidal functions, but with no clear separation between the convex and saddle curvatures. Four convex minima were discovered, but the suboptimal minima were again less atrractive to the gradient walks than the global minimum. The local minima were less connected than for ReLU, but more connected than for TanH. Better connectivity is illustrated by the n_{stag} and l_{stag} values in Table 8.1. The band connecting the global minimum to the stationary attractors of high error was wider for ELU than for TanH, indicating that the surface was smoother, and the connecting valleys were likely wider.

Figure 8.2 shows the l-g clouds as sampled by the $[-1, 1]$ macro walks. Larger step sizes did not prevent the gradient walks from discovering the same convex minima for the TanH activation. Clear transition from the saddle to convex curvature can still be observed in Figure 8.2a. Stronger gradients were sampled around the global minimum attractor, indicating that larger steps bounced off the walls of the attraction basin. Larger steps also allowed gradient walks to proceed directly to one of the minima, sometimes avoiding prior convergence to a saddle stationary point, confirmed by the n_{stag} values in Table 8.1.

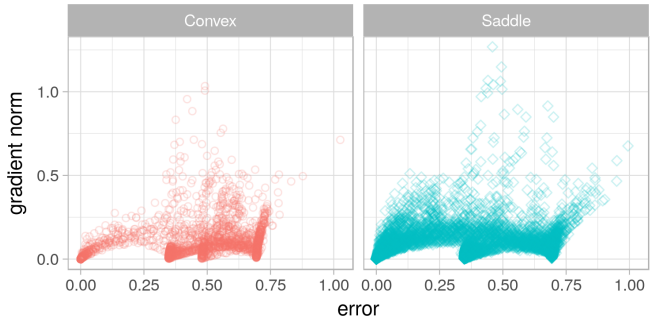
For the ReLU activation function, larger step sizes yielded very noisy behaviour, with less evident structure, as shown in Figure 8.2b. This indicates that the loss surface associated with the ReLU activation function was rugged and inconsistent when observed



(a) TanH, macro, $[-1, 1]$



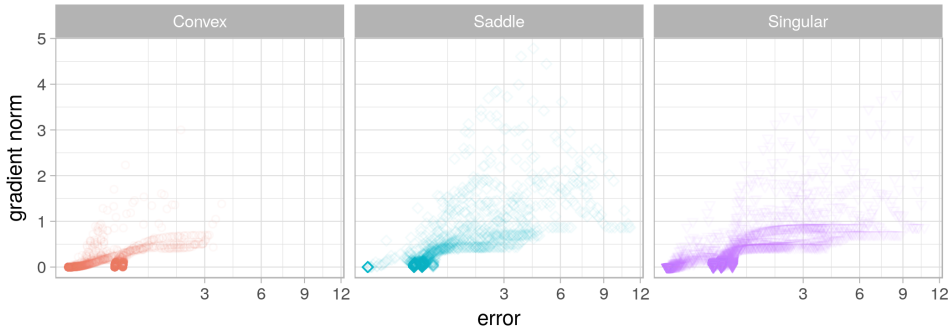
(b) ReLU, macro, $[-1, 1]$



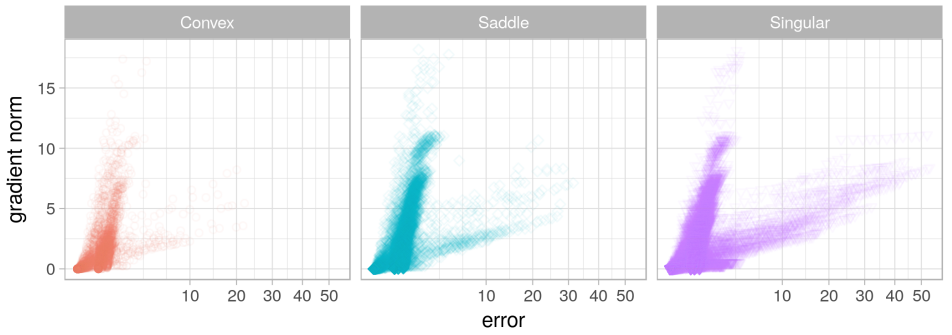
(c) ELU, macro, $[-1, 1]$

Figure 8.2: L-g clouds for the macro gradient walks initialised in the $[-1, 1]$ range for the XOR problem.

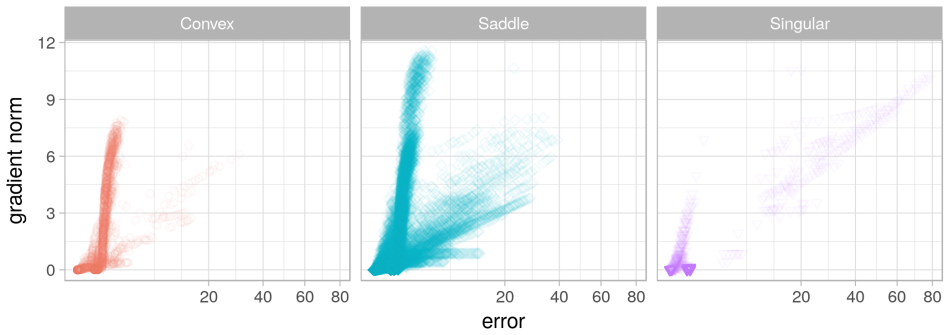
at a larger scale. Fewer walks have discovered convex global minima. Most points exhibited flatness. According to the l_{stag} values in Table 8.1, the gradient walks typically became stuck for less than a quarter of the total steps taken. Thus, although ReLU offers a higher chance of escaping local minima, searchability of the overall landscape



(a) TanH, micro, $[-10, 10]$



(b) ReLU, micro, $[-10, 10]$

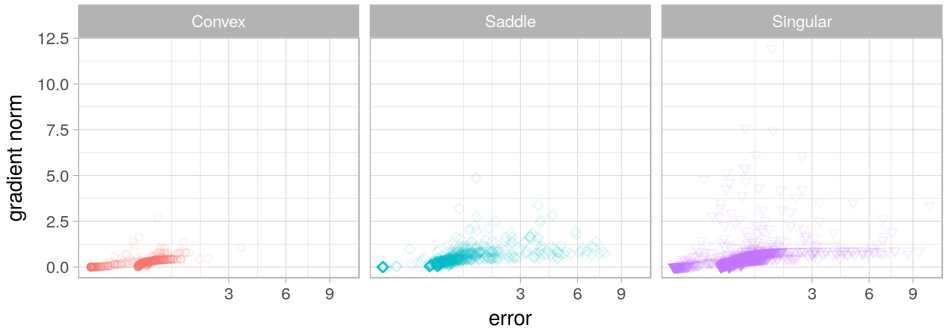


(c) ELU, micro, $[-10, 10]$

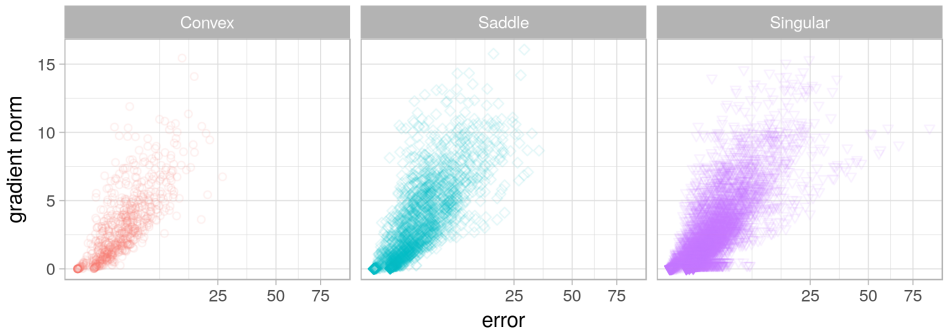
Figure 8.3: L-g clouds for the micro gradient walks initialised in the $[-10, 10]$ range for the XOR problem.

suffers, and is more dependent on the chosen step size.

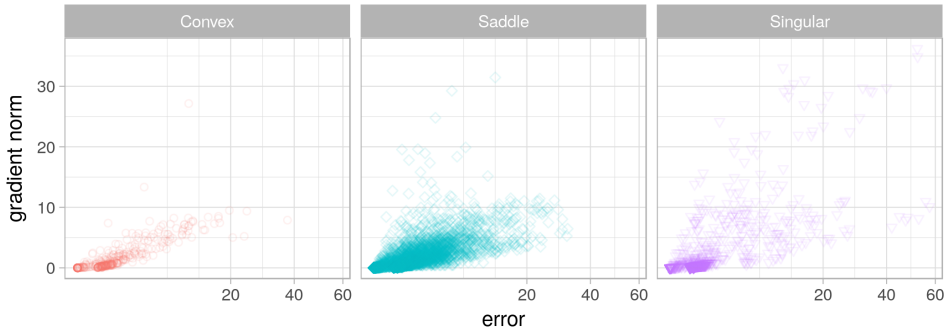
The ELU activation function sampled with larger step sizes, as shown in Figure 8.2c, exhibited more evident structure than ReLU, and stronger gradients than TanH. Com-



(a) TanH, macro, $[-10, 10]$



(b) ReLU, macro, $[-10, 10]$



(c) ELU, macro, $[-10, 10]$

Figure 8.4: L-g clouds for the macro gradient walks initialised in the $[-10, 10]$ range for the XOR problem.

pared to TanH and ReLU, ELU did not exhibit any flatness, i.e. indefinite Hessians, which indicate that the loss surface associated with ELU was more searchable. The l_{stag} values in Table 8.1 indicate that the gradient walks typically became stuck for half of the

total steps taken, thus ELU did not prevent exploitation, while preventing premature convergence to a local minimum.

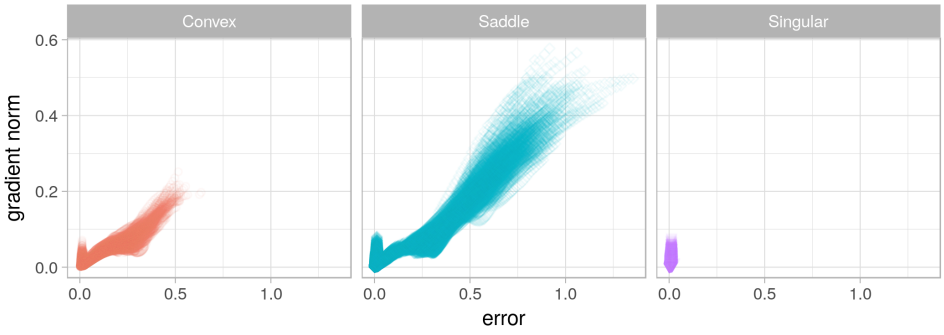
Figures 8.3 and 8.4 show the l-g clouds obtained for the $[-10, 10]$ gradient walks. The x-axis is shown in square-root scale for readability. The same four stationary attractors were sampled for TanH, with two convex local minima. Table 8.1 indicates that the walks were very unlikely to make transitions between minima. ReLU and ELU both yielded much higher errors than TanH, although convergence to the global minimum still took place. ReLU sampled the most flat curvature points, and ELU sampled the most saddle points. Both ReLU and ELU also exhibited two clusters: points of low error and high gradient, and points of low gradient and high error. These are attributed to the narrow and wide valleys present in the landscape, which could not be observed at the smaller scale.

Under the macro setting, shown in Figure 8.4, all three activation functions still yielded the discovery of local minima. Out of the three activation functions, ELU exhibited the strongest gradients, and the least number of indefinite curvature points. Thus, ELU yielded the most searchable loss surface for the XOR problem.

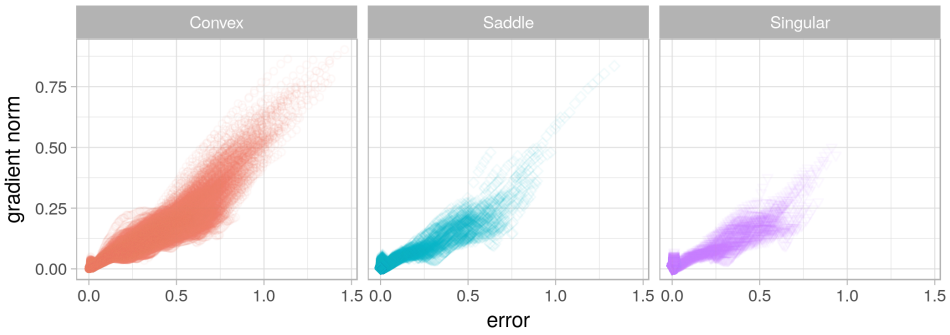
8.2.2 Iris

Figure 8.5 shows the l-g clouds obtained for the Iris problem sampled with the $[-1, 1]$ micro walks. The l-g clouds for the three activation functions exhibited a similar shape, but different curvature properties. Only one major attractor around the global minimum was discovered. Table 8.2 lists the n_{stag} and l_{stag} values generated by the activation functions for the Iris problem under various settings. The gradient walks did not generally become stuck more than once under all settings considered, confirming that global minima formed the only strong attractor.

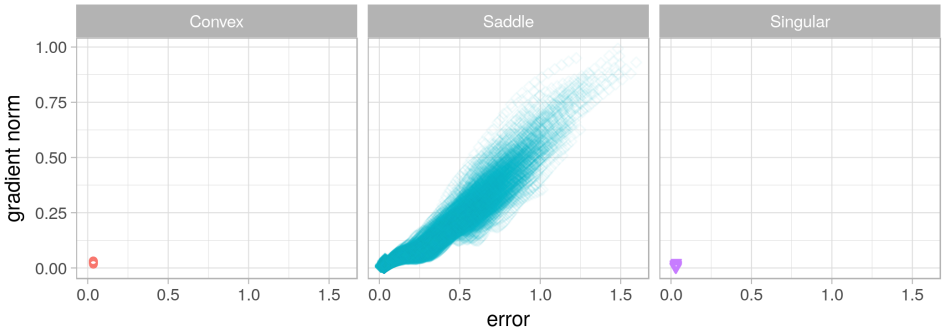
Figure 8.5a shows that TanH was dominated by the saddle curvature. The sampled points were split into two clusters around the global minimum, namely points of higher error and lower gradient, and points of higher gradient and low error. The points of high gradient and low error overlapped with the points of indefinite curvature, indicating that this cluster exhibited flatness. Lack of curvature in some dimensions means that those particular dimensions have not contributed to the final fitness. In the case of



(a) TanH, micro, $[-1, 1]$



(b) ReLU, micro, $[-1, 1]$



(c) ELU, micro, $[-1, 1]$

Figure 8.5: L-g clouds for the micro gradient walks initialised in the $[-1, 1]$ range for the Iris problem.

NNs, each dimension corresponds to a weight. A non-contributing weight indicates that the signal generated by that weight is zeroed somewhere in the architecture, and any change in the weight value would not have an effect on the NN output. For example,

Table 8.2: Basin of attraction estimates calculated for the Iris problem on the E_t and E_g walks. Standard deviation shown in parenthesis.

	TanH		ReLU		ELU	
E_t	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.0000	886.0829	1.0086	879.1043	1.0000	897.9343
micro	0.0000	22.8566	0.0922	50.2012	0.0000	23.4255
$[-1, 1]$,	1.0000	79.9514	1.0000	80.1371	1.0000	81.7000
macro	0.0000	2.6404	0.0000	3.6728	0.0000	2.8921
$[-10, 10]$,	1.0029	954.3300	1.0000	960.9057	1.0000	960.9057
micro	0.0534	27.1584	0.0000	7.9617	0.0000	7.0884
$[-10, 10]$,	1.0057	84.4300	1.0371	80.3833	1.0229	83.9881
macro	0.0754	8.1484	0.2172	14.7487	0.1675	11.3875
E_g	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.0457	848.6614	1.0371	867.5129	1.0057	912.1214
micro	0.2089	97.6926	0.1891	90.6481	0.0754	41.7026
$[-1, 1]$,	1.0000	79.2629	1.0000	79.9714	1.0000	83.0914
macro	0.0000	2.7887	0.0000	3.7438	0.0000	3.0280
$[-10, 10]$,	1.1286	935.8892	1.1543	942.3144	1.0314	951.1780
micro	1.1628	115.1048	1.2735	119.1299	0.3491	71.0568
$[-10, 10]$,	1.0171	83.3657	1.0914	71.2819	1.0343	83.3848
macro	0.1298	10.0162	0.3166	23.4619	0.2242	11.8406

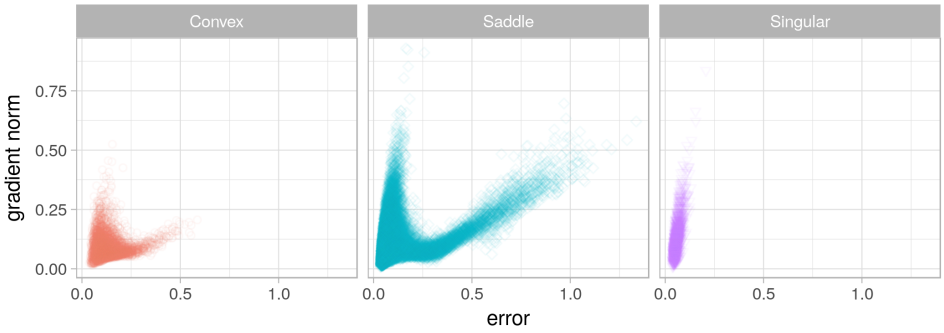
if a neuron is saturated, i.e. the neuron always outputs a value close to the asymptote, then the contribution of a single weight may become negligible. Therefore, indefinite curvature is likely associated with saturated neurons. However, neuron saturation is not the only explanation for non-contributing weights: if certain weights are unnecessary for the minimal solution to the problem at hand, then techniques such as regularisation can be employed to reduce the unnecessary weights to zero. Therefore, solutions with non-contributing weights can also be associated with regularised models. It has been

observed before that optima for smaller NN architectures are embedded in the weight space of larger NN architectures [86]. Indeed, if the unnecessary weights in an over-parametrised NN architecture are set to zero, then the large architecture will map to a smaller architecture. If there is a minimum associated with the smaller architecture, the same minimum will be available to the larger architecture, provided the irrelevant weights are set to zero. Minima associated with smaller architectures that can be found for a larger architecture are further referred to as *embedded minima*.

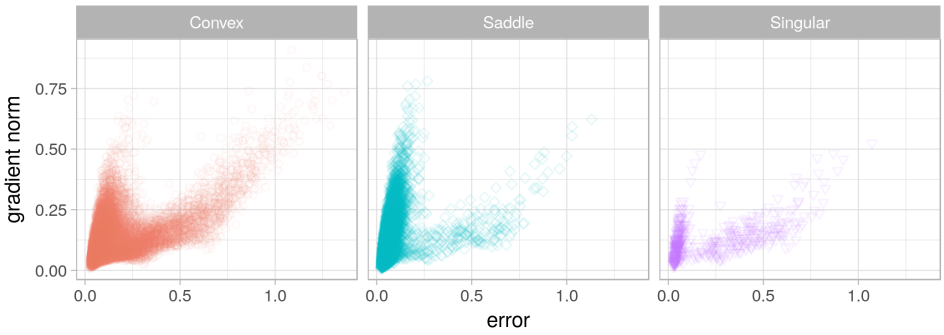
Thus, flat areas around the global minimum are attributed to the non-contributing weights, which are indicative of saturation. Some of the solutions with saturated neurons correspond to the minima that require fewer weights than available in the architecture. This hypothesis has the implication that the steep gradient minima associated with indefinite curvature can have poor performance, in case of unwanted saturation, or good performance, in case of implicit regularisation.

For the ReLU activation function, the prevalence of convex curvature is evident in Figure 8.5b. Again, ReLU exhibited more flatness than the other two activation functions considered. Such behaviour is attributed to the hard saturation of ReLU, which can easily yield non-contributing weights. The ELU activation function, as shown in Figure 8.5c, was dominated by the saddle curvature. The least amount of flatness was discovered for ELU, again making ELU a potentially good choice for algorithms that rely on the gradient information.

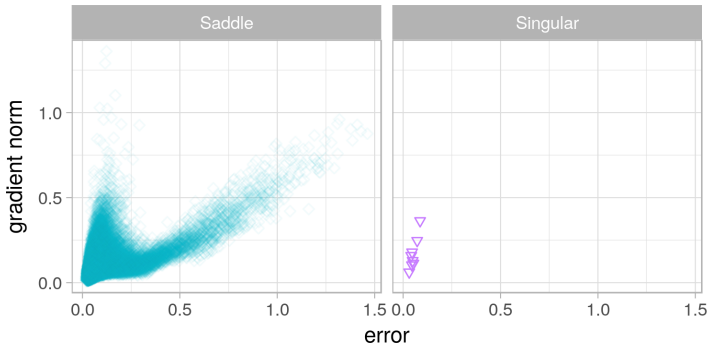
Figure 8.6 shows the l-g clouds obtained for the $[-1, 1]$ macro walks. Larger step sizes revealed similar curvature tendencies for the three activation functions, with TanH exhibiting convexity at the global minimum, but being dominated by saddle points otherwise, ReLU exhibiting strong convexity, and ELU exhibiting no convexity and almost no flatness. All three activation functions yielded a split into high error, low gradient, and high gradient, low error clusters. The high gradient, low error clusters were associated with indefinite Hessians for the three activation functions. This behaviour is again attributed to the embedded minima that require fewer weights, as well as hidden unit saturation. Larger steps are likely to arrive at larger weights, thus increasing the chances of saturation. The classification errors reported in Table B.7 in Appendix B indicate that the solution quality for the macro walks deteriorated as compared to the micro walks.



(a) TanH, macro, $[-1, 1]$



(b) ReLU, macro, $[-1, 1]$



(c) ELU, macro, $[-1, 1]$

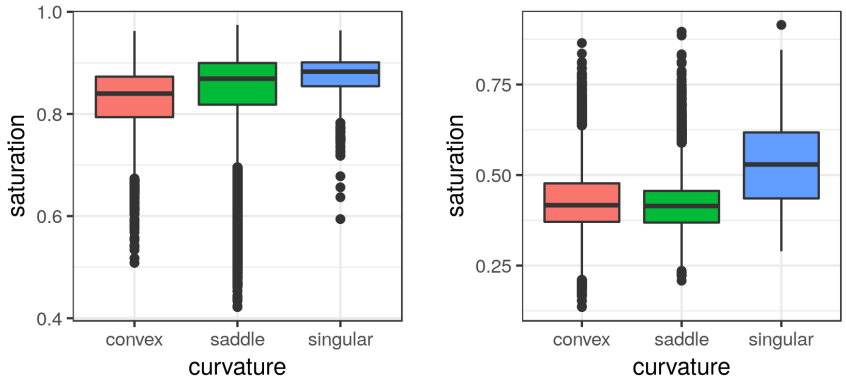
Figure 8.6: L-g clouds for the macro gradient walks initialised in the $[-1, 1]$ range for the Iris problem.

To test the saturation hypothesis, the degree of saturation was measured for TanH and ReLU. For TanH, the ς_h saturation measure was used, proposed in [108] for bounded activation functions. The value of ς_h is in the $[0, 1]$ continuous range, where 0 corresponds

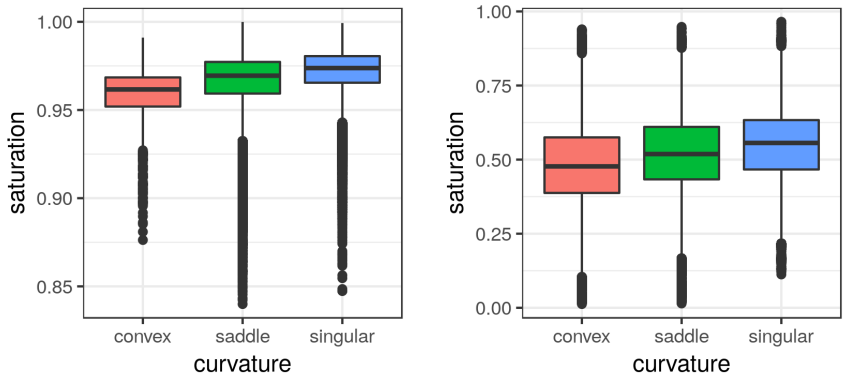
to a normal distribution of hidden neuron activations, 0.5 corresponds to a uniform distribution of hidden neuron activations, and 1 corresponds to a saturated distribution of hidden neuron activations, where most activations lie on the asymptotic ends. For ReLU, the proportion of zero activations for all hidden neurons was used as an estimate of saturation. Figure 8.7 shows the box plots [84, 132] generated for TanH and ReLU for a few selected scenarios. Box plots represent the distribution of a continuous variable, visualising the median (the line in the middle of the box), the 25^{th} and 75^{th} percentiles (hinges, or upper and lower bounds of the box), and the 1.5×inter-quartile range from the hinges (the whiskers, or lines extending from the box). Points that lie outside of the whiskers are plotted separately as outliers. Figure 8.7 shows that singular curvature (indefinite Hessians) was indeed associated with higher saturation under various scenarios, especially under the $[-1, 1]$ macro setting, which yielded a steep gradient cluster of indefinite points.

Figures 8.8 and 8.9 show the l-g clouds for the $[-10, 10]$ walks for the micro and macro settings, respectively. The x-axis is shown in square-root scaling for readability. According to Figure 8.8, TanH exhibited two convex attractors around the global minimum, but was mostly dominated by the saddle curvature. Under the macro setting, shown in Figure 8.9, TanH exhibited very few convex points. Large step sizes prevented the walks from converging to a convex basin, indicating that the width of the convex basin must have been smaller than 2 (maximum step size calculated as 10% of the $[-10, 10]$ initialisation range). ReLU exhibited the most convexity out of the three activation functions, and the most flatness. Thus, ReLU yielded a wider convex attraction basin than TanH, and was also more likely to saturate. ELU exhibited little to no convexity, and flatness only along the steeper cluster, associated with embedded minima and/or saturation. The split into two clusters was still evident for all activation functions. In general, with the increase in step size, the steep cluster became heavier than the shallow cluster. Thus, the steep cluster is confirmed to be associated with large weights, which cause saturation.

Table 8.2 shows that the n_{stag} and l_{stag} values calculated for E_g yielded higher standard deviations than for E_t, especially under the $[-10, 10]$ micro setting. To further study the generalisation behaviour of the three activation functions, Figure 8.10 shows



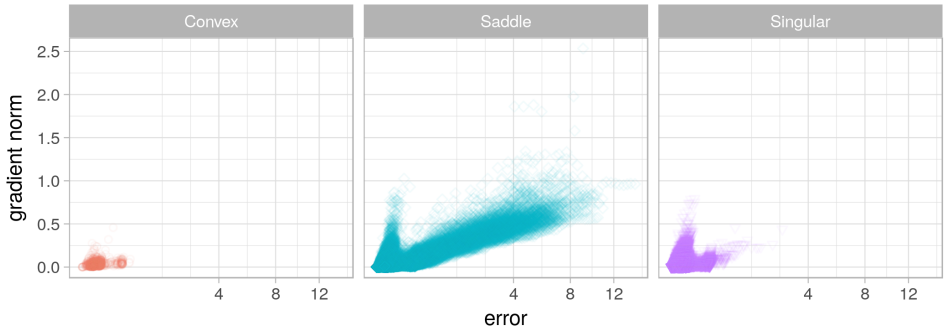
(a) TanH, macro, $[-1, 1]$ (b) ReLU, macro, $[-1, 1]$



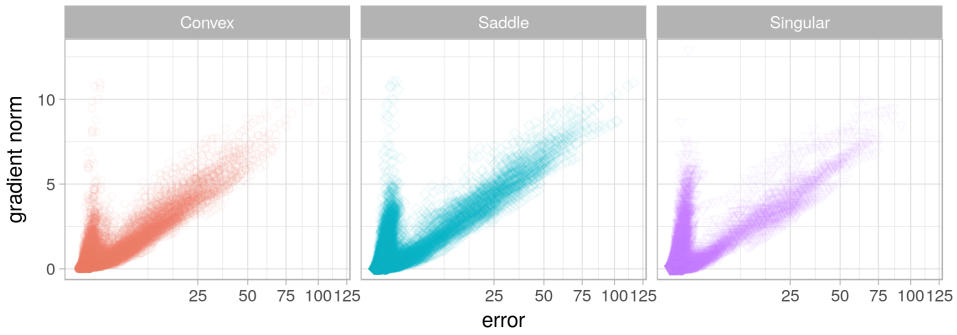
(c) TanH, micro, $[-10, 10]$ (d) ReLU, micro, $[-10, 10]$

Figure 8.7: Box plots illustrating the degree of saturation associated with the different curvatures for the Iris problem.

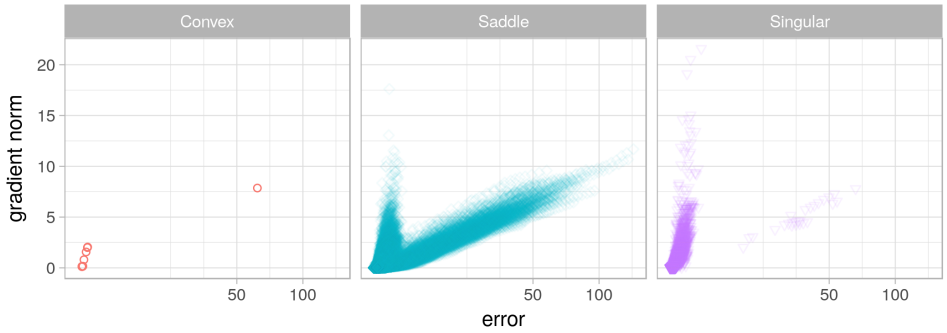
the l-g clouds for the micro walks colourised according to their E_g values. Figure 8.10 illustrates for the $[-1, 1]$ setting that ELU exhibited the best generalisation behaviour, not deteriorating in E_g as E_t approached zero. The classification errors reported in Table B.7 confirm this observation. For the $[-10, 10]$ setting, the low error, high gradient cluster generalised better than the high error, low gradient cluster for both ReLU and ELU. If the cluster of steep gradients corresponds to the embedded minima comprised of fewer contributing weights, then the steeper gradient solutions can be considered regularised, which explains better generalisation performance.



(a) TanH, micro, $[-10, 10]$



(b) ReLU, micro, $[-10, 10]$

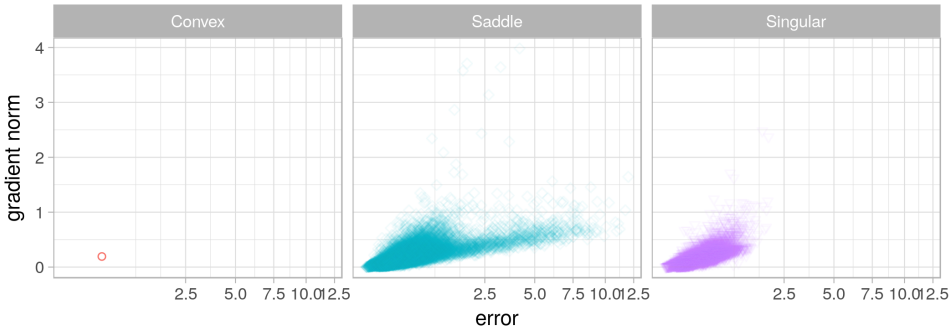


(c) ELU, micro, $[-10, 10]$

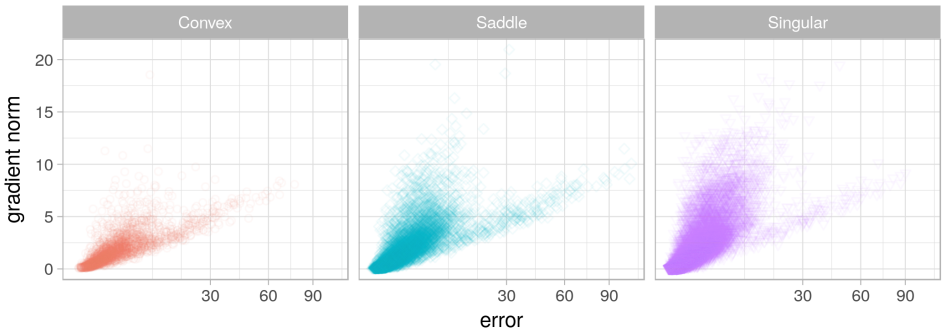
Figure 8.8: L-g clouds for the micro gradient walks initialised in the $[-10, 10]$ range for the Iris problem.

8.2.3 Diabetes

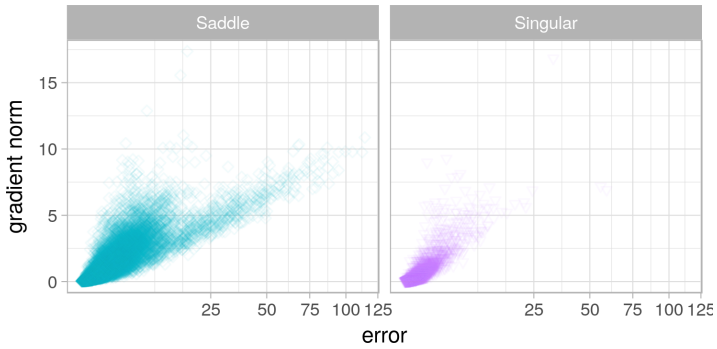
Figure 8.11 shows the l-g clouds obtained for the Diabetes problem for the $[-1, 1]$ micro setting. The three activation functions yielded l-g clouds of a similar shape: gradient



(a) TanH, macro, $[-10, 10]$



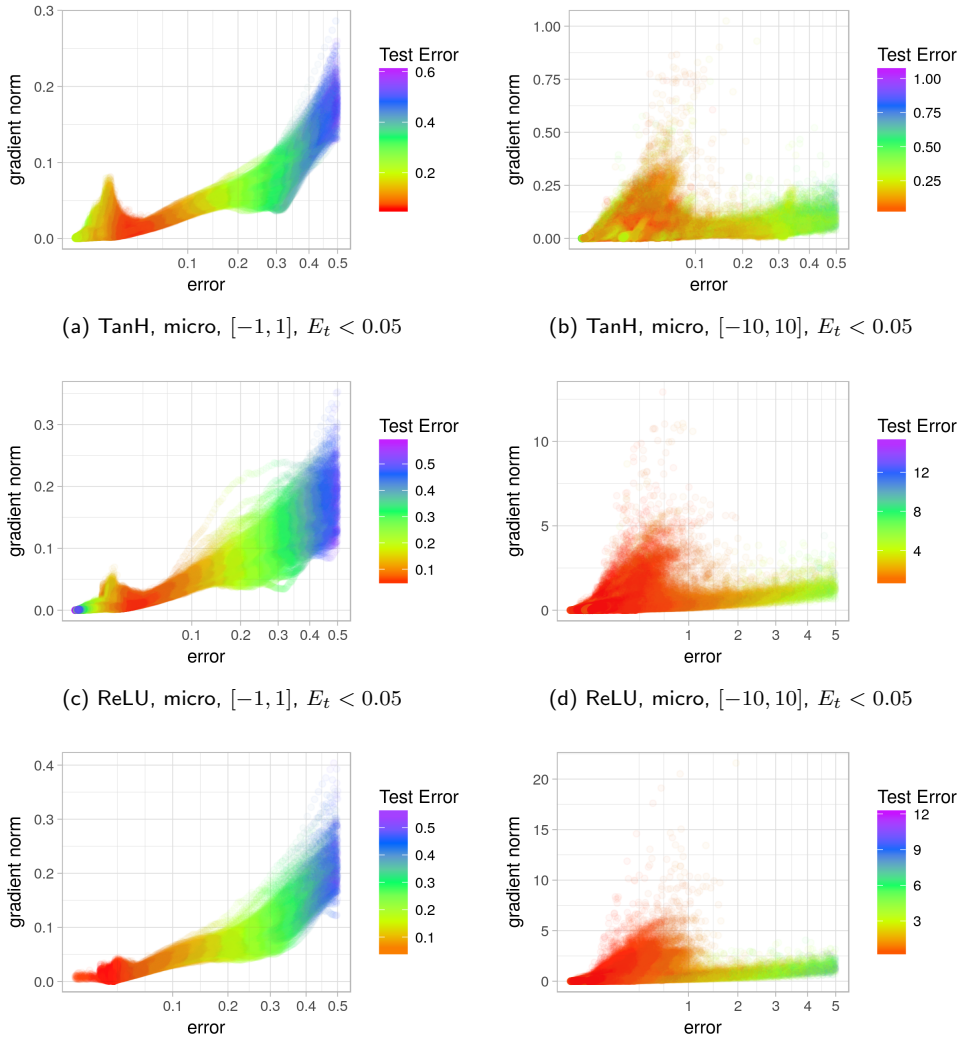
(b) ReLU, macro, $[-10, 10]$



(c) ELU, macro, $[-10, 10]$

Figure 8.9: L-g clouds for the macro gradient walks initialised in the $[-10, 10]$ range for the Iris problem.

walks descended to the general area around global minimum attractor, and then exploited that area. No disconnected local minima were found. The n_{stag} and l_{stag} values in Table 8.3 confirm that the walks generally did not become stuck more than once.



(a) TanH, micro, $[-1, 1]$, $E_t < 0.05$

(b) TanH, micro, $[-10, 10]$, $E_t < 0.05$

(c) ReLU, micro, $[-1, 1]$, $E_t < 0.05$

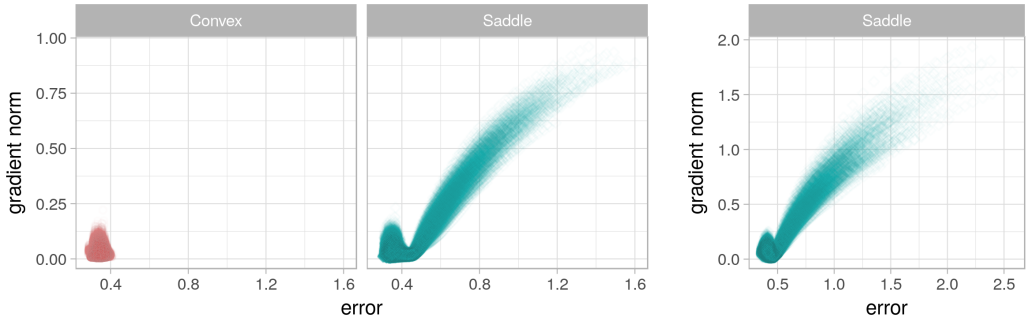
(d) ReLU, micro, $[-10, 10]$, $E_t < 0.05$

(e) ELU, micro, $[-1, 1]$, $E_t < 0.05$

(f) ELU, micro, $[-10, 10]$, $E_t < 0.05$

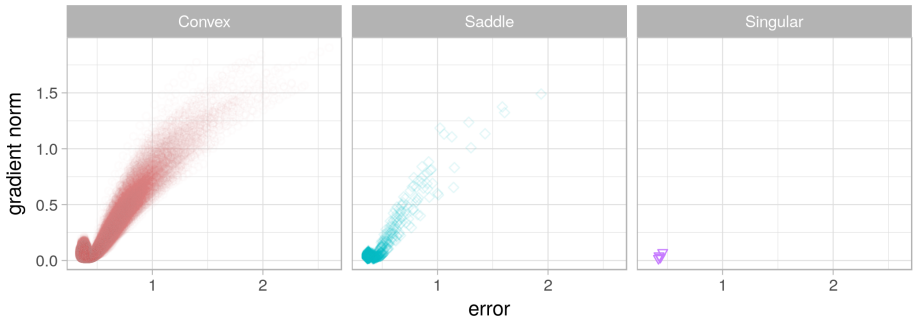
Figure 8.10: L-g clouds colourised according to the corresponding E_g values for the Iris problem.

Thus, once again, the various activation functions did not introduce new minima. However, the curvature and width of the global minimum was affected by the activation function choice. TanH exhibited convexity around the global minimum, but was otherwise dominated by the saddle curvature. ReLU was dominated by the convex curvature, with some saddle and indefinite curvature discovered around the global minimum. ReLU



(a) TanH, micro, $[-1, 1]$

(b) ELU, micro, $[-1, 1]$



(c) ReLU, micro, $[-1, 1]$

Figure 8.11: L-g clouds for the micro gradient walks initialised in the $[-1, 1]$ range for the Diabetes problem.

also exhibited steeper gradients and a wider range of error values than TanH, indicating that ReLU yielded a steeper and more convex attraction basin. ELU exhibited saddle curvature only, with the strongest gradients. Despite exhibiting the least convexity out of the three activation functions, ELU yielded a highly searchable attraction basin with no flat areas.

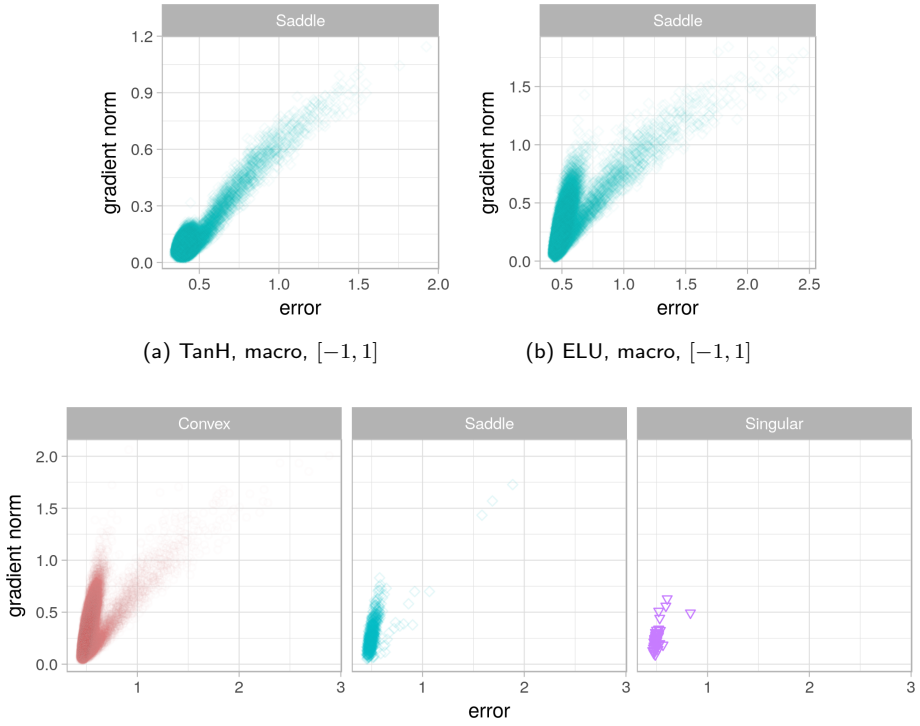
For the $[-1, 1]$ macro setting, illustrated in Figure 8.12, ReLU was the only activation function that exhibited convexity. Both TanH and ELU exhibited only saddle curvature. The split into two clusters of steep and low gradients was again observed for the ELU and ReLU activation functions. Indefinite Hessians were obtained for the steep gradient cluster using the ReLU activation. Figure 8.13a shows that indefinite curvature was associated with the largest degree of saturation for the $[-1, 1]$ macro setting. Thus,

Table 8.3: Basin of attraction estimates calculated for the Diabetes problem on the E_t and E_g walks. Standard deviation shown in parenthesis.

	TanH		ReLU		ELU	
E_t	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.0000	955.0790	1.0000	957.6765	1.0000	964.2988
micro	(0.0000)	(10.3712)	(0.0000)	(7.0533)	(0.0000)	(5.1385)
$[-1, 1]$,	1.0012	86.9463	1.0086	84.5179	1.0062	85.9068
macro	(0.0351)	(3.4553)	(0.0926)	(7.8686)	(0.0928)	(6.9681)
$[-10, 10]$,	1.0000	957.5432	1.0000	961.3630	1.0000	961.7667
micro	0.0000)	(6.0317)	(0.0000)	(5.0449)	(0.0000)	(5.1574)
$[-10, 10]$,	1.0062	84.9864	1.0728	76.2551	1.0679	77.8084
macro	(0.0783)	(5.9030)	(0.3234)	(20.1278)	(0.2705)	(18.3192)
E_g	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.1296	921.4922	1.1111	941.5796	1.0136	961.5564
micro	(0.7399)	(145.2101)	(1.0399)	(109.9153)	(0.2999)	(40.5750)
$[-1, 1]$,	1.2346	64.1361	1.0852	75.1210	1.0432	82.5895
macro	(0.5061)	(28.5435)	(0.3427)	(20.8204)	(0.2208)	(14.3887)
$[-1, 1]$,	1.0074	943.7086	1.0000	959.9605	1.0000	961.1333
micro	(0.0857)	(44.1027)	(0.0000)	(5.7900)	(0.0000)	(6.1131)
$[-1, 1]$,	1.0420	79.0010	1.1198	67.3222	1.0815	75.5739
macro	(0.2066)	(15.3752)	(0.4088)	(26.3781)	(0.3116)	(20.3869)

indefinite curvature associated with the narrow valleys once again indicates that the narrow valleys correlate with neuron saturation.

Figure 8.14 illustrates the l-g clouds obtained for the $[-10, 10]$ micro setting. When observed at a larger scale, TanH exhibited a much smaller error range than ReLU and ELU, indicating that ReLU and ELU generated steeper landscapes for the same initialisation range. ReLU again exhibited the most convexity out of the three activation functions, and ELU exhibited the least convexity, and the most saddle curvature. The split into steep and low gradient clusters is evident for ReLU and ELU, less so for TanH.



(a) TanH, macro, $[-1, 1]$

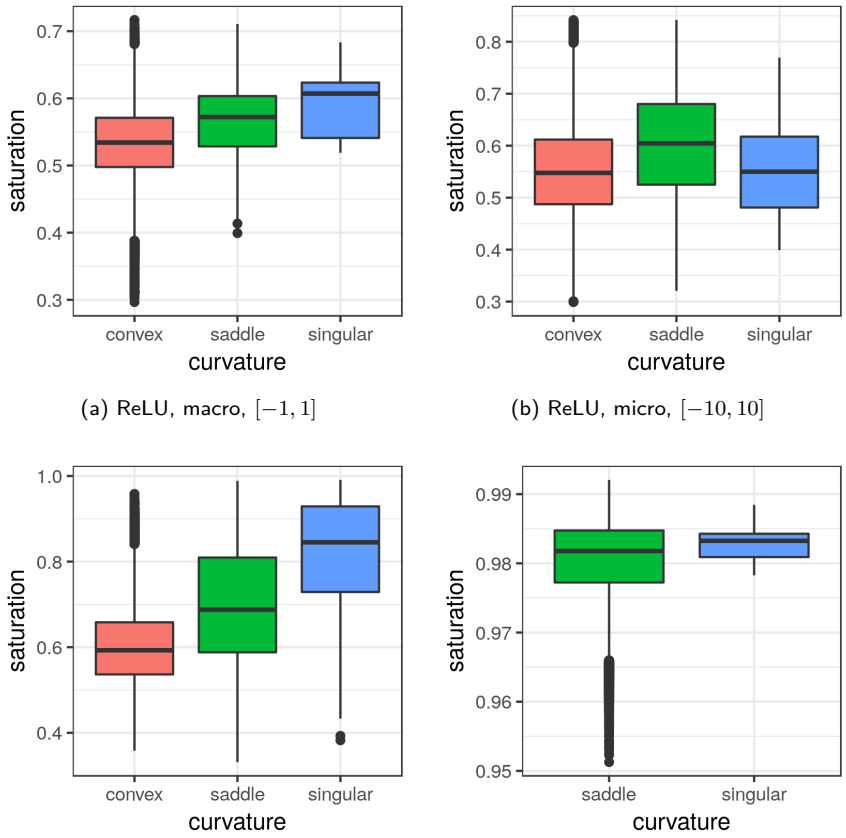
(b) ELU, macro, $[-1, 1]$

(c) ReLU, macro, $[-1, 1]$

Figure 8.12: L-g clouds for the macro gradient walks initialised in the $[-1, 1]$ range for the Diabetes problem.

This observation indicates that ReLU and ELU shared certain properties not present in TanH. The clear split into two clusters is again explained by the presence of deep and shallow valleys. Due to steeper gradients, the different valleys are more strongly manifested in ReLU and ELU than in TanH. Interestingly, the indefinite points exhibited by ReLU were present in both the steep and the shallow cluster for the $[-10, 10]$ micro setting. Figure 8.13 confirms that for this setting, indefinite points did not correspond to higher saturation.

Similar behaviour can be observed for the macro $[-10, 10]$ setting, illustrated in Figure 8.15. For large steps in the large initialisation range, ReLU still exhibited convexity, as well as flatness. For the larger steps in the $[-10, 10]$ range, TanH also exhibited the split into two clusters. For both activation functions, indefinite curvature corresponded



(a) ReLU, macro, $[-1, 1]$

(b) ReLU, micro, $[-10, 10]$

(c) ReLU, macro, $[-10, 10]$

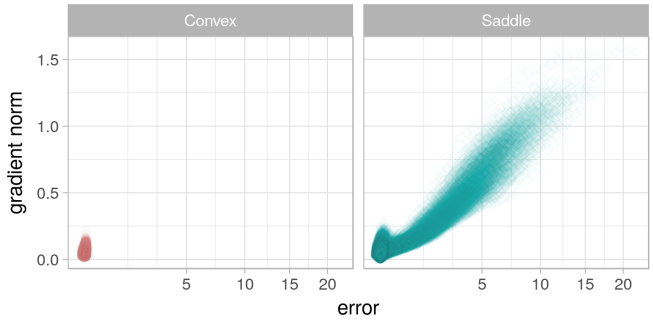
(d) TanH, macro, $[-10, 10]$

Figure 8.13: Box plots illustrating the degree of saturation associated with the different curvatures for the Diabetes problem.

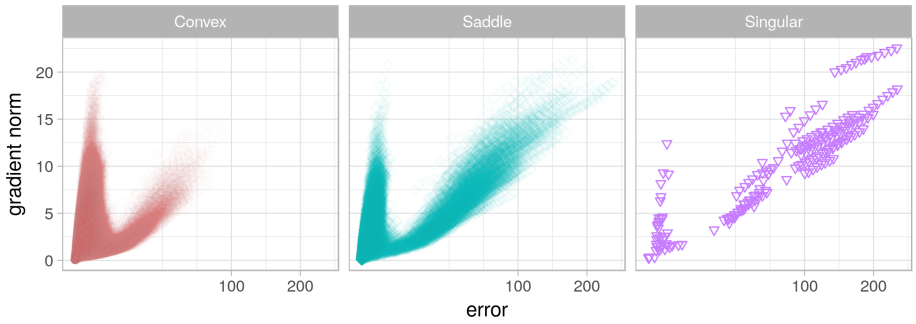
to the steeper cluster. Figure 8.13 again shows that the indefinite points in the steep cluster yielded higher degrees of saturation for both TanH and ReLU. Thus, the steep gradient cluster solutions definitely correlated with neuron saturation.

The classification results in Table B.8 in Appendix B show that TanH yielded lower errors than ReLU and ELU under the $[-10, 10]$ setting, indicating that the TanH landscape was in fact more searchable using larger steps than the ReLU and ELU landscapes.

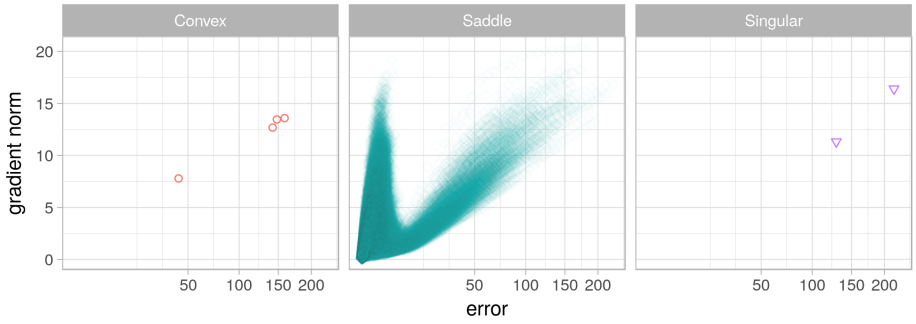
To evaluate the generalisation performance of the activation functions, l-g clouds colourised according to the E_g values are shown in Figure 8.16. Table B.8 also summarises the average classification errors obtained at the last step of the gradient walks.



(a) TanH, micro, $[-10, 10]$



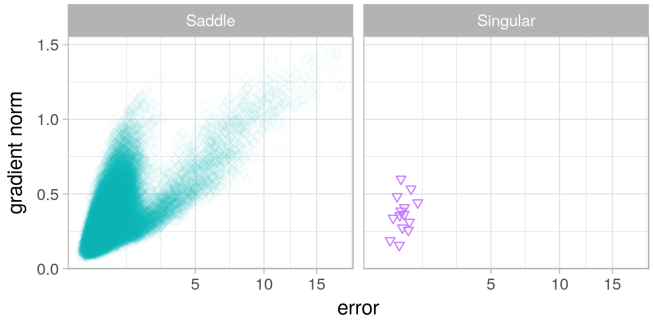
(b) ReLU, micro, $[-10, 10]$



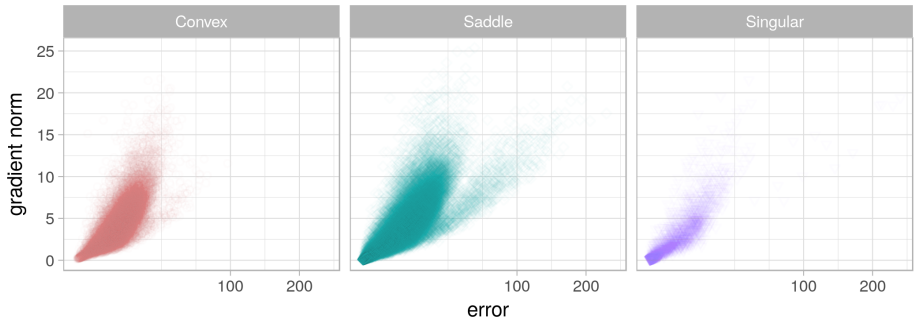
(c) ELU, micro, $[-10, 10]$

Figure 8.14: L-g clouds for the micro gradient walks initialised in the $[-10, 10]$ range for the Diabetes problem.

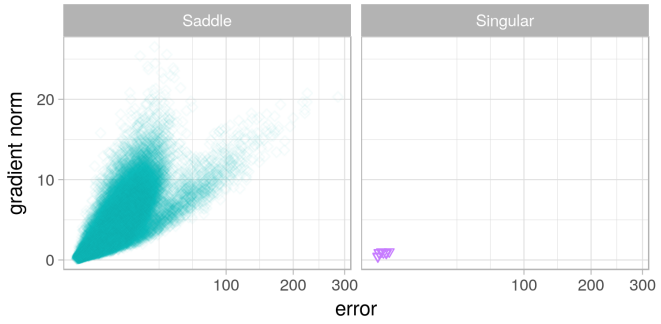
Figure 8.16 shows for the $[-1, 1]$ setting that ELU generalised the best out of the three activation functions. The results in Table B.8 confirm this observation. For ELU, exploitation of the global optimum had the least negative effect on the final generalisation performance, which indicates that ELU was resilient to overfitting. For the $[-10, 10]$



(a) TanH, macro, $[-10, 10]$



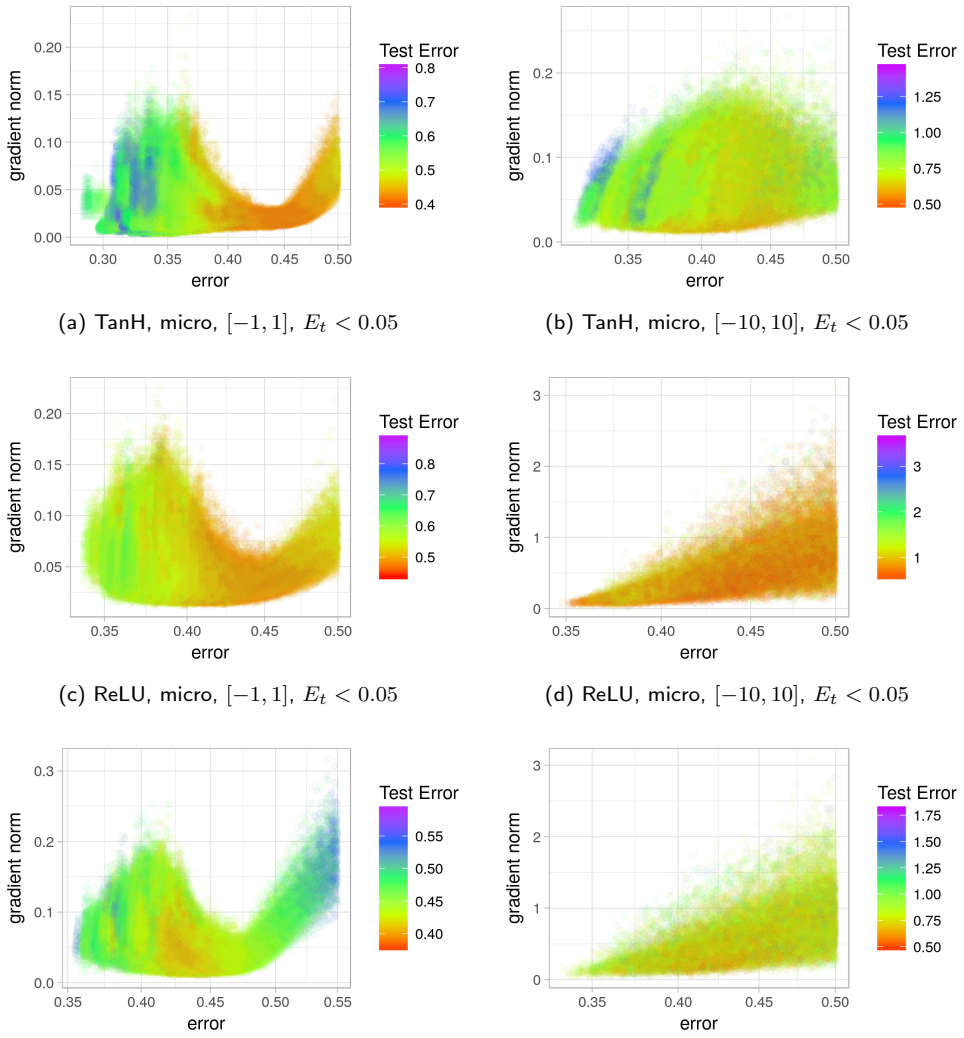
(b) ReLU, macro, $[-10, 10]$



(c) ELU, macro, $[-10, 10]$

Figure 8.15: L-g clouds for the macro gradient walks initialised in the $[-10, 10]$ range for the Diabetes problem.

setting, both ReLU and ELU yielded error landscapes that were harder to exploit than the TanH error landscape, causing poor final error. Table B.8 confirms that TanH consistently yielded better training and test errors for the $[-10, 10]$ walks. Thus, while ReLU and ELU were more robust to overfitting, they were also more sensitive to the chosen



(a) TanH, micro, $[-1, 1]$, $E_t < 0.05$

(b) TanH, micro, $[-10, 10]$, $E_t < 0.05$

(c) ReLU, micro, $[-1, 1]$, $E_t < 0.05$

(d) ReLU, micro, $[-10, 10]$, $E_t < 0.05$

(e) ELU, micro, $[-1, 1]$, $E_t < 0.05$

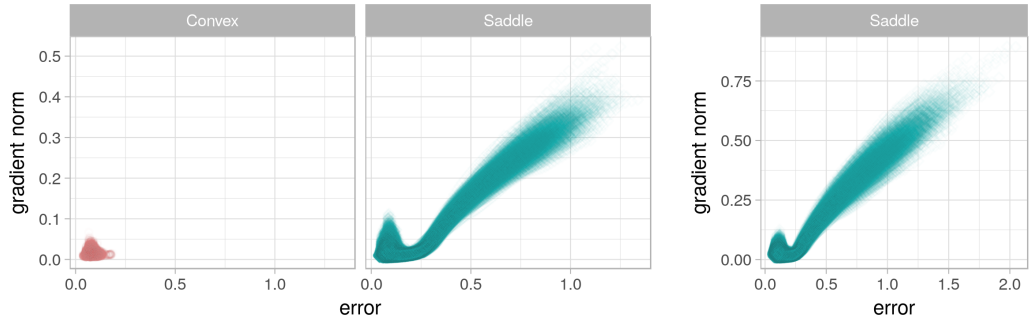
(f) ELU, micro, $[-10, 10]$, $E_t < 0.05$

Figure 8.16: L-g clouds colourised according to the corresponding E_g values for the Diabetes problem.

step size and initialisation range.

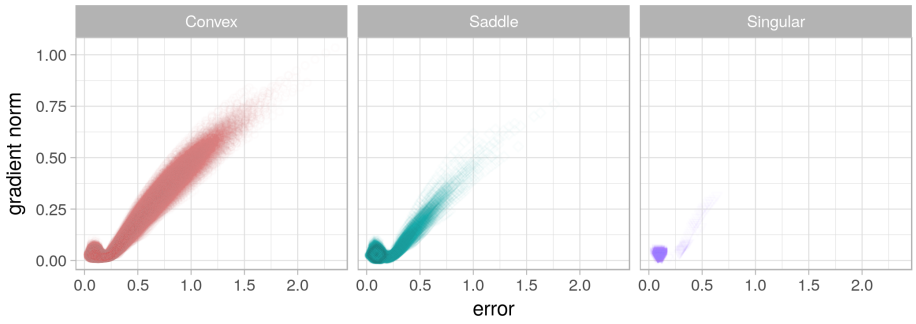
8.2.4 Glass

Figures 8.17, 8.18, 8.19, and 8.20 show the l-g clouds obtained for the Glass problem under various granularity settings. Similar to the Diabetes problem, all activation func-



(a) TanH, micro, $[-1, 1]$

(b) ELU, micro, $[-1, 1]$



(c) ReLU, micro, $[-1, 1]$

Figure 8.17: L-g clouds for the micro gradient walks initialised in the $[-1, 1]$ range for the Glass problem.

tions yielded convergence to the general area of the global minimum, and then exploited that area. The results in Table 8.4 confirm for all activation functions that the walks became stuck exactly once.

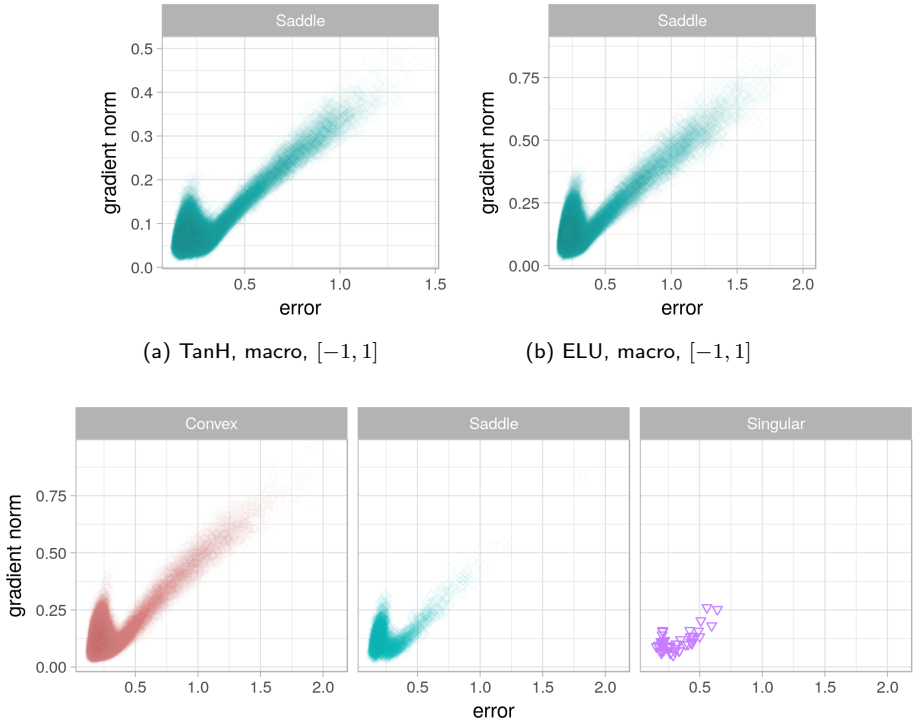
TanH exhibited convexity around the global minimum for the micro $[-1, 1]$ walks, and saddle curvature otherwise. An increase in the step size and initialisation range yielded TanH to discover fewer convex points and more points of indefinite curvature, as illustrated in Figures 8.18, 8.19, and 8.20. ELU exhibited no convexity at all, and an increase in the step size and range increased the number of indefinite points discovered, even though saddle curvature was still prevalent. ReLU once again exhibited convexity for all scenarios considered, and the number of convex points gradually reduced with an increase in the step size and initialisation range. For the $[-10, 10]$ setting, shown in Fig-

Table 8.4: Basin of attraction estimates calculated for the Glass problem on the E_t and E_g walks. Standard deviation shown in parenthesis.

	TanH		ReLU		ELU	
E_t	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.0000	938.2247	1.0000	945.6920	1.0000	949.5280
micro	0.0000	5.8471	0.0000	6.9634	0.0000	4.7040
$[-1, 1]$,	1.0000	85.7233	1.0000	86.4553	1.0000	86.8427
macro	0.0000	0.5878	0.0000	0.6282	0.0000	0.4582
$[-10, 10]$,	1.0000	956.3840	1.0000	961.2947	1.0000	960.5373
micro	0.0000	4.9462	0.0000	3.5346	0.0000	3.6248
$[-10, 10]$,	1.0000	86.8540	1.0000	87.5760	1.0000	87.5267
macro	0.0000	0.7398	0.0000	0.9523	0.0000	0.7080
E_g	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.0067	936.7922	1.1880	905.3526	1.0880	922.7396
micro	0.0892	39.4541	0.9943	162.1228	0.5332	123.1337
$[-1, 1]$,	1.0020	86.1760	1.5913	50.9346	1.0000	87.1727
macro	0.0447	2.2831	0.8924	29.2926	0.0000	1.2983
$[-10, 10]$,	1.0000	952.9680	1.0000	960.2427	1.0000	960.6493
micro	0.0000	7.3825	0.0000	4.1790	0.0000	4.3135
$[-10, 10]$,	1.0000	86.8433	1.0013	87.1951	1.0027	86.9362
macro	0.0000	1.0348	0.0516	2.8150	0.0632	3.1805

ures 8.19 and 8.20, ReLU exhibited the most indefinite curvature compared to the other two activation functions. As previously discussed, indefinite curvature is associated with non-contributing weights, and is therefore attributed to the hard saturation exhibited by ReLU.

The split into steep and low gradient clusters manifested across all settings, and became more evident as the step size and initialisation range increased. Therefore, larger step sizes and a wider initialisation range exaggerated the split into narrow and wide



(a) TanH, macro, $[-1, 1]$

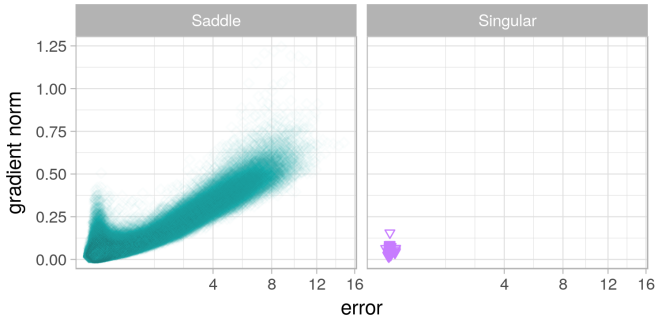
(b) ELU, macro, $[-1, 1]$

(c) ReLU, macro, $[-1, 1]$

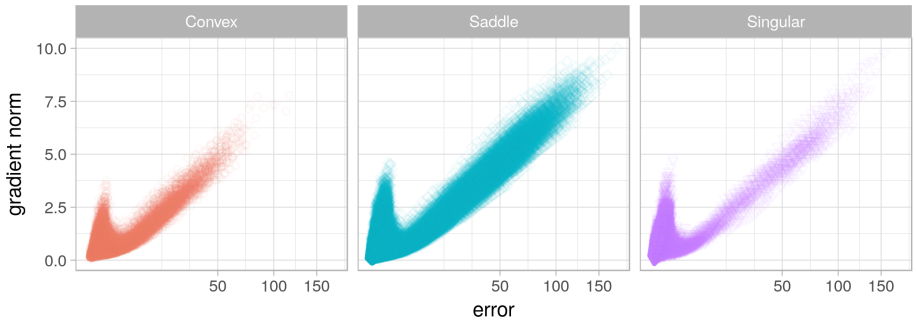
Figure 8.18: L-g clouds for the macro gradient walks initialised in the $[-1, 1]$ range for the Glass problem.

valleys. The two types of valleys clearly form a part of the global landscape structure. The higher density of indefinite points associated with the narrow valleys (steep gradients) once again correlated with a high degree of saturation, as shown in Figure 8.21. Thus, the narrow valleys were associated with saturated neurons.

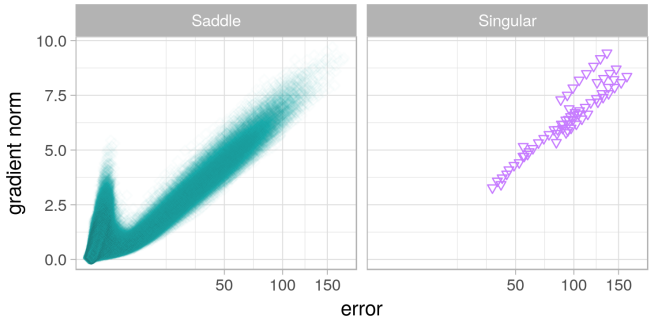
The classification results in Table B.9 (Appendix B) show that the ELU activation function yielded the best generalisation performance in three out of the four settings considered, confirming the hypothesis that ELU yields loss surfaces that are resilient to overfitting. Figure 8.22 shows the l-g clouds colourised according to the E_g values. Exploitation of the global minimum with small steps was evidently detrimental for all activation functions considered. For the larger steps, ReLU and ELU yielded high errors, and struggled to exploit. Points of varied E_g values were sampled around the global



(a) TanH, micro, $[-10, 10]$



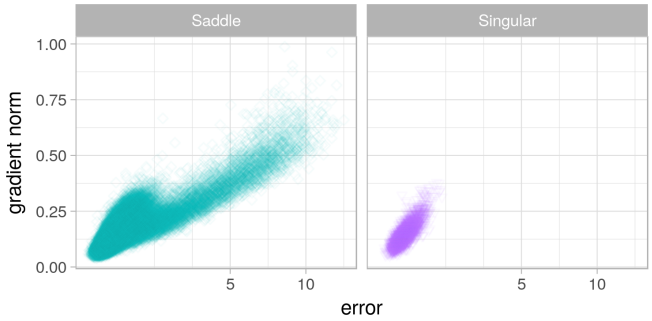
(b) ReLU, micro, $[-10, 10]$



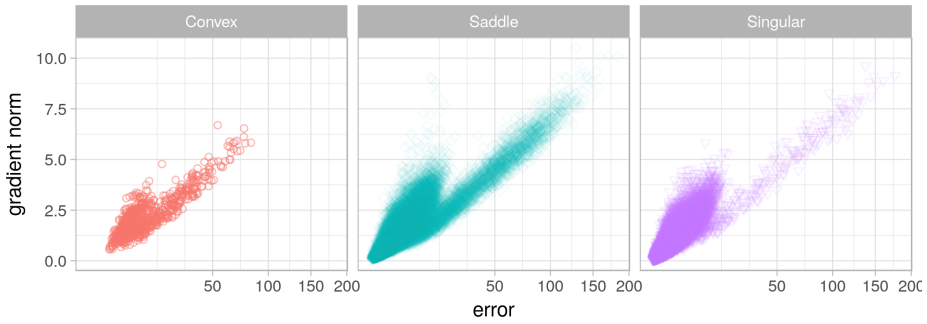
(c) ELU, micro, $[-10, 10]$

Figure 8.19: L-g clouds for the micro gradient walks initialised in the $[-10, 10]$ range for the Glass problem.

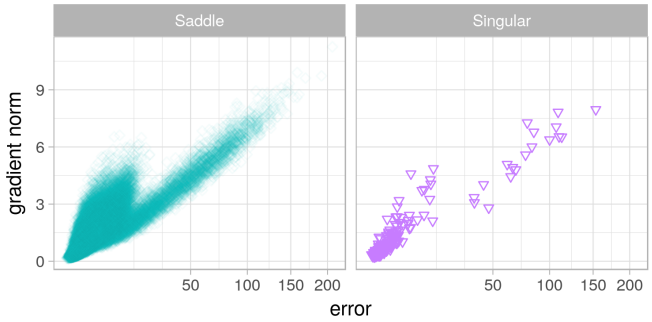
minimum attractor along the steeper gradient clusters, indicating that two points of very similar E_t values can have different E_g values. Indeed, non-contributing weights can either be a result of saturation due to over-training, or of implicit regularisation due to the discovery of embedded minima.



(a) TanH, macro, $[-10, 10]$



(b) ReLU, macro, $[-10, 10]$

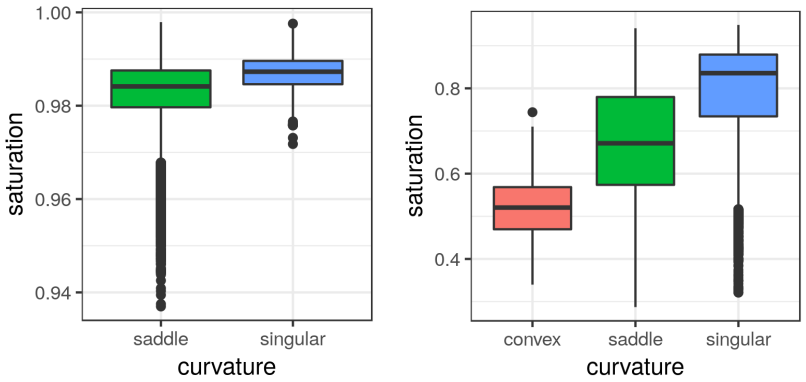


(c) ELU, macro, $[-10, 10]$

Figure 8.20: L-g clouds for the macro gradient walks initialised in the $[-10, 10]$ range for the Glass problem.

8.2.5 Cancer

Figures 8.23, 8.24, 8.25, and 8.26 show the l-g clouds obtained for the Cancer problem. All activation functions yielded convergence to a single simple attractor at the global



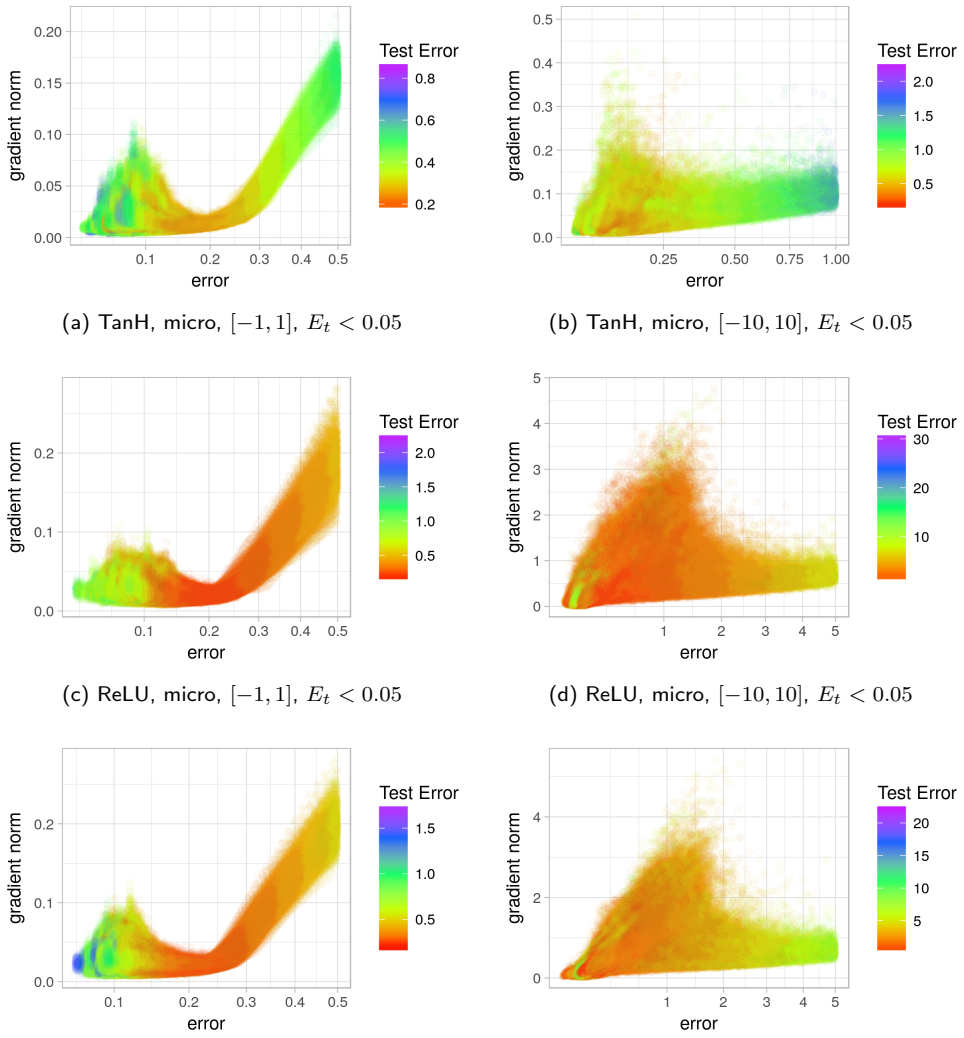
(a) TanH, macro, $[-10, 10]$ (b) ReLU, macro, $[-10, 10]$

Figure 8.21: Box plots illustrating the degree of saturation associated with the different curvatures for the Glass problem.

minimum. The results in Table 8.5 confirm for all activation functions that the walks became stuck exactly once. The l_{stag} values in Table 8.5 show that the walks converged quickly and exploited the attractor for most of the steps. The Cancer problem is known to be a trivial classification task, solvable with over 95% accuracy by NNs with a single hidden layer [9]. The simplicity of the dataset clearly contributed to the simplicity of the landscape.

Figures 8.23 and 8.24 show the l-g clouds for the $[-1, 1]$ walks. ReLU was once again the only activation function that exhibited convexity, and ELU was the only activation function that did not exhibit any indefinite, or flat curvature. Under the macro setting, the l-g clouds for TanH and ELU had a similar shape, as shown in Figure 8.24, although ELU yielded much stronger gradients. ReLU exhibited a split into two clusters of steep and shallow gradients around the global optimum attractor, with convex points associated with the shallow gradients. The results in Table B.10 (Appendix B) show that all three activation functions yielded an accuracy score between 98% and 100% under all the settings considered, confirming that the problem was very easy indeed. When sampled with the $[-1, 1]$ micro walk, all the activation functions resulted in 100% classification accuracy on the training set.

Figures 8.25 and 8.26 show the l-g clouds obtained for the $[-10, 10]$ walks. A larger



(a) TanH, micro, $[-1, 1]$, $E_t < 0.05$

(b) TanH, micro, $[-10, 10]$, $E_t < 0.05$

(c) ReLU, micro, $[-1, 1]$, $E_t < 0.05$

(d) ReLU, micro, $[-10, 10]$, $E_t < 0.05$

(e) ELU, micro, $[-1, 1]$, $E_t < 0.05$

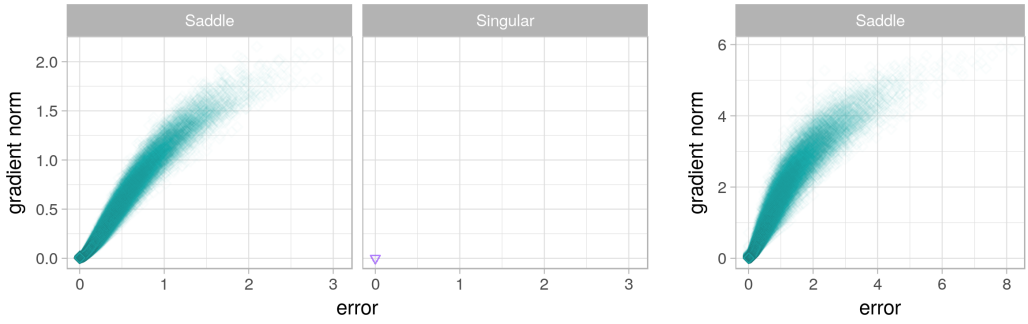
(f) ELU, micro, $[-10, 10]$, $E_t < 0.05$

Figure 8.22: L-g clouds colourised according to the corresponding E_g values for the Glass problem.

initialisation range resulted in all the activation functions producing points of indefinite curvature. As before, points of indefinite curvature exhibited a correlation with the degree of saturation, illustrated in Figure 8.27. According to Figure 8.27, TanH was much more prone to saturation than ReLU. This behaviour is to be expected, since ReLU is an unbounded function, and only saturates for negative values, while TanH is

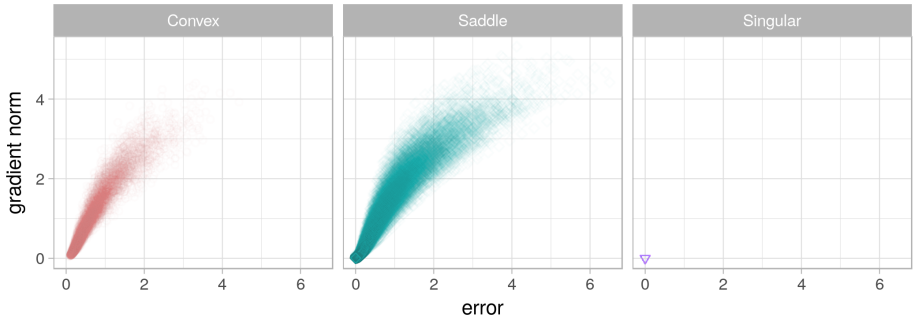
bounded. The split into steep and shallow gradient clusters around the origin manifested for all three activation functions, and the steep gradient clusters were associated with a higher density of indefinite, i.e. flat, points. Figure 8.26 shows that the split into clusters became especially evident for TanH when sampled with the $[-10, 10]$ macro walks. TanH is a bounded activation function, prone to saturation when large weights are considered, as illustrated in Figure 8.27. ReLU and ELU also exhibited the two-cluster split. Thus, the presence of narrow and wide valleys was again confirmed to be a part of the global landscape structure.

Figure 8.28 shows the l-g clouds colourised according to the E_g values. For the $[-10, 10]$ walks, points of high gradient and low error for TanH and ELU often yielded good generalisation. This observation is attributed to the discovery of embedded minima,



(a) TanH, micro, $[-1, 1]$

(b) ELU, micro, $[-1, 1]$



(c) ReLU, micro, $[-1, 1]$

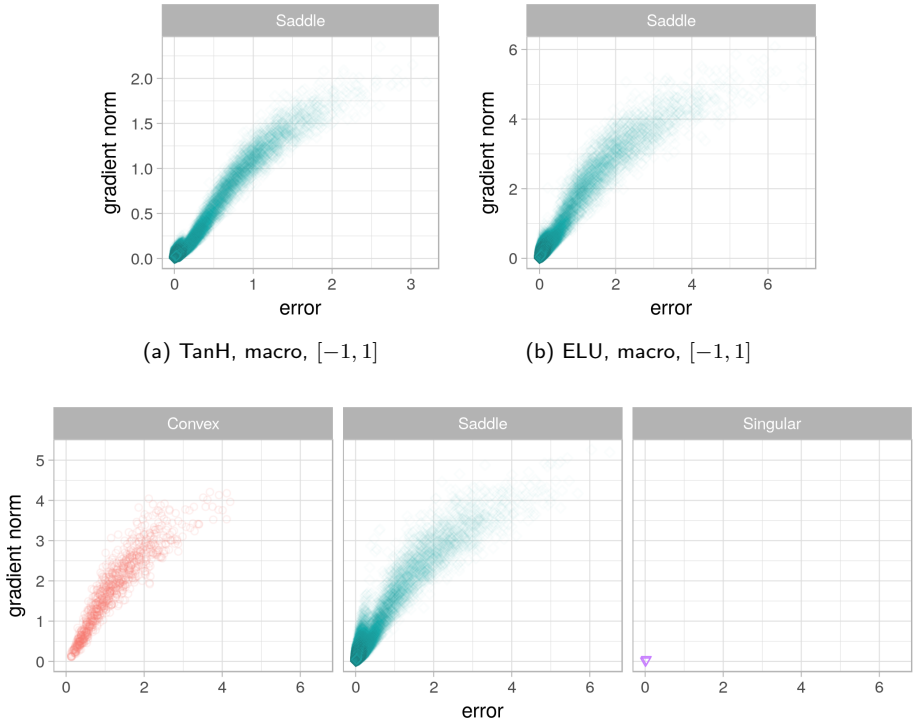
Figure 8.23: L-g clouds for the micro gradient walks initialised in the $[-1, 1]$ range for the Cancer problem.

Table 8.5: Basin of attraction estimates calculated for the Cancer problem on the E_t and E_g walks. Standard deviation shown in parenthesis.

	TanH		ReLU		ELU	
E_t	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.0000	969.2556	1.0000	974.0112	1.0000	975.3938
micro	(0.0000)	(3.4917)	(0.0000)	(3.2969)	(0.0000)	(2.6956)
$[-1, 1]$,	1.0000	87.9396	1.0003	87.7855	1.0012	87.7195
macro	(0.0000)	(0.2912)	(0.0176)	(2.0024)	(0.0353)	(2.6253)
$[-10, 10]$,	1.0000	975.3991	1.0000	977.3551	1.0000	977.5683
micro	(0.0000)	(3.4372)	(0.0000)	(2.4727)	(0.0000)	(2.5137)
$[-10, 10]$,	1.0003	87.5972	1.0022	86.6762	1.0019	87.0460
macro	(0.0176)	(1.7322)	(0.0549)	(5.3511)	(0.0466)	(4.4699)
E_g	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.2088	920.8136	1.5218	874.1151	1.8218	861.6208
micro	(1.0886)	(171.6435)	(1.7495)	(249.7582)	(2.5779)	(277.8866)
$[-1, 1]$,	1.0044	87.1337	1.0112	86.8358	1.0617	82.0722
macro	(0.0748)	(4.6555)	(0.1111)	(7.0472)	(0.2733)	(16.6101)
$[-10, 10]$,	3.3802	632.1400	1.2591	937.0848	1.0939	960.5525
micro	(3.9069)	(385.1825)	(1.4125)	(170.9971)	(0.8247)	(108.2192)
$[-10, 10]$,	1.0106	86.2966	1.0695	79.7677	1.0503	80.9173
macro	(0.1192)	(7.7003)	(0.3030)	(18.9452)	(0.2495)	(17.5740)

i.e. implicit regularisation. For ReLU, however, the points of steeper gradients generally exhibited poorer generalisation performance than the points of shallow gradients. For all scenarios and all activation functions, generalisation deteriorated at the global minimum. Table B.10 indicates that all activation functions exhibited similar generalisation performance, and that the generalisation of ELU has indeed improved with increased step size and initialisation range. Due to the simplicity of the dataset, the resulting landscape was highly searchable, enabling good performance even with larger steps.

Previous studies have suggested that better generalisation performance is typically



(a) TanH, macro, $[-1, 1]$

(b) ELU, macro, $[-1, 1]$

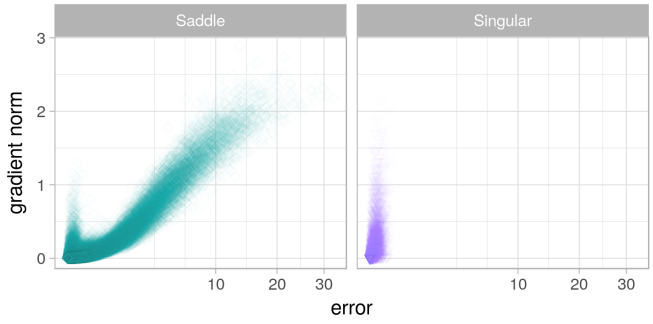
(c) ReLU, macro, $[-1, 1]$

Figure 8.24: L-g clouds for the macro gradient walks initialised in the $[-1, 1]$ range for the Cancer problem.

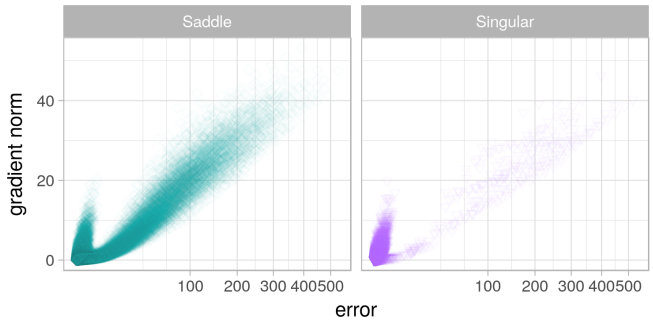
observed in the wide valleys [21]. The results in this section suggest that the above statement may not be universally applicable to all activation functions, and that for the ELU activation, better generalisation performance can sometimes be obtained in the more narrow valleys. Indeed, if narrow valleys correspond to the embedded, or regularised minima, then finding the minima of the necessary steepness can lead to implicit regularisation, and therefore better generalisation performance.

8.2.6 Heart

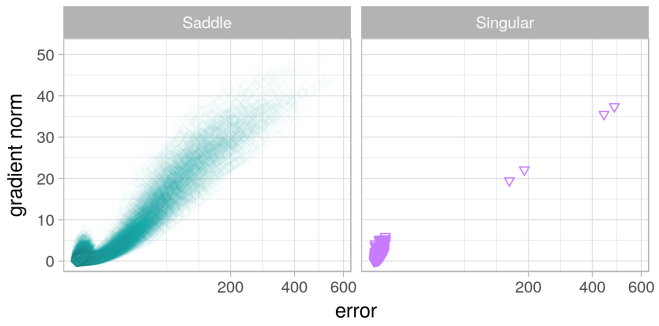
Figures 8.29, 8.30, and 8.31 show the l-g clouds obtained for the Heart problem. Convergence to the general area of global minimum with subsequent exploitation of that area was observed for all activation functions under all scenarios. The results in Table 8.6



(a) TanH, micro, $[-10, 10]$



(b) ReLU, micro, $[-10, 10]$

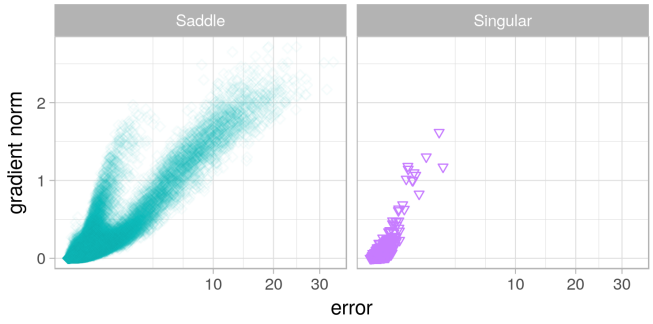


(c) ELU, micro, $[-10, 10]$

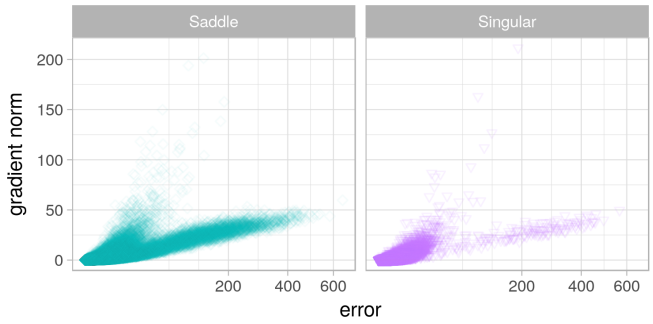
Figure 8.25: L-g clouds for the micro gradient walks initialised in the $[-10, 10]$ range for the Cancer problem.

confirm that the walks have converged once for all activation functions considered, and remained in the discovered basin for most of the steps.

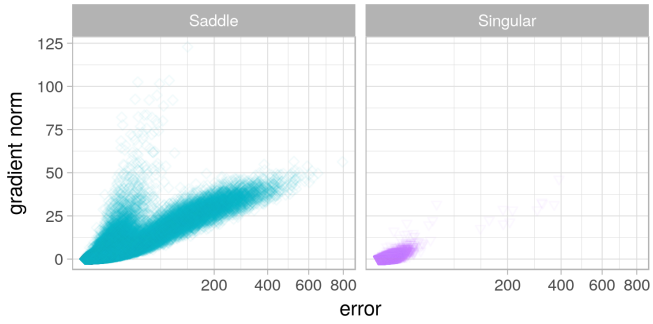
Figure 8.29 shows for the $[-1, 1]$ walks that all activation functions were dominated by the saddle curvature. Only ReLU yielded a few indefinite points around the global



(a) TanH, macro, $[-10, 10]$



(b) ReLU, macro, $[-10, 10]$

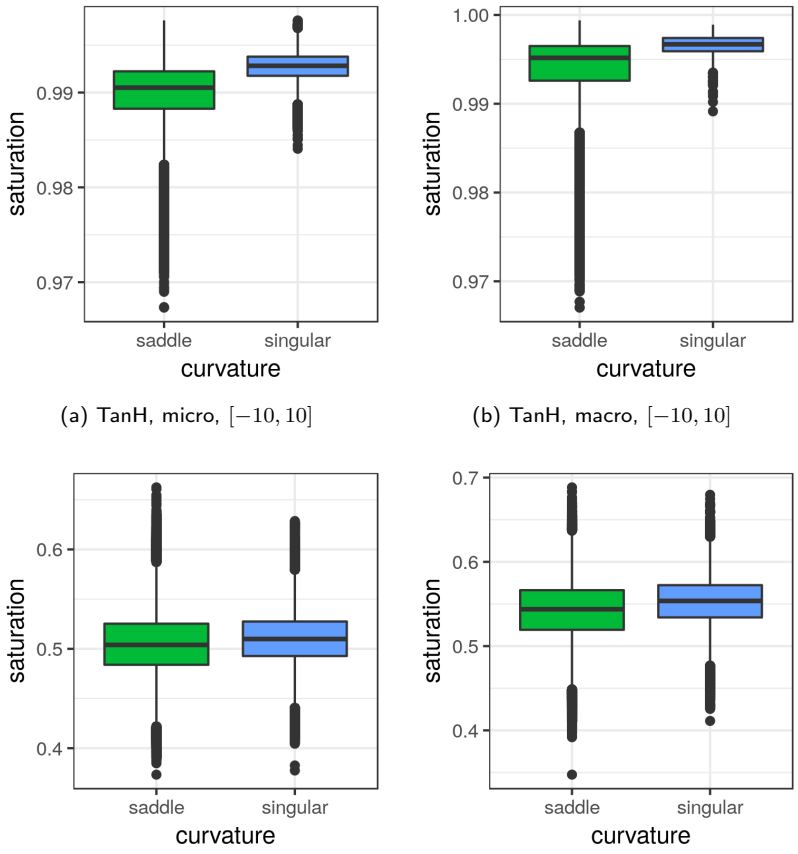


(c) ELU, macro, $[-10, 10]$

Figure 8.26: L-g clouds for the macro gradient walks initialised in the $[-10, 10]$ range for the Cancer problem.

minimum attractor. The prevalence of the saddle curvature is attributed to the relatively high dimensionality of the problem [28].

The split into two clusters under the $[-1, 1]$ setting was evident for all activation functions, especially for ReLU and ELU, as shown in Figure 8.29. The steep gradient

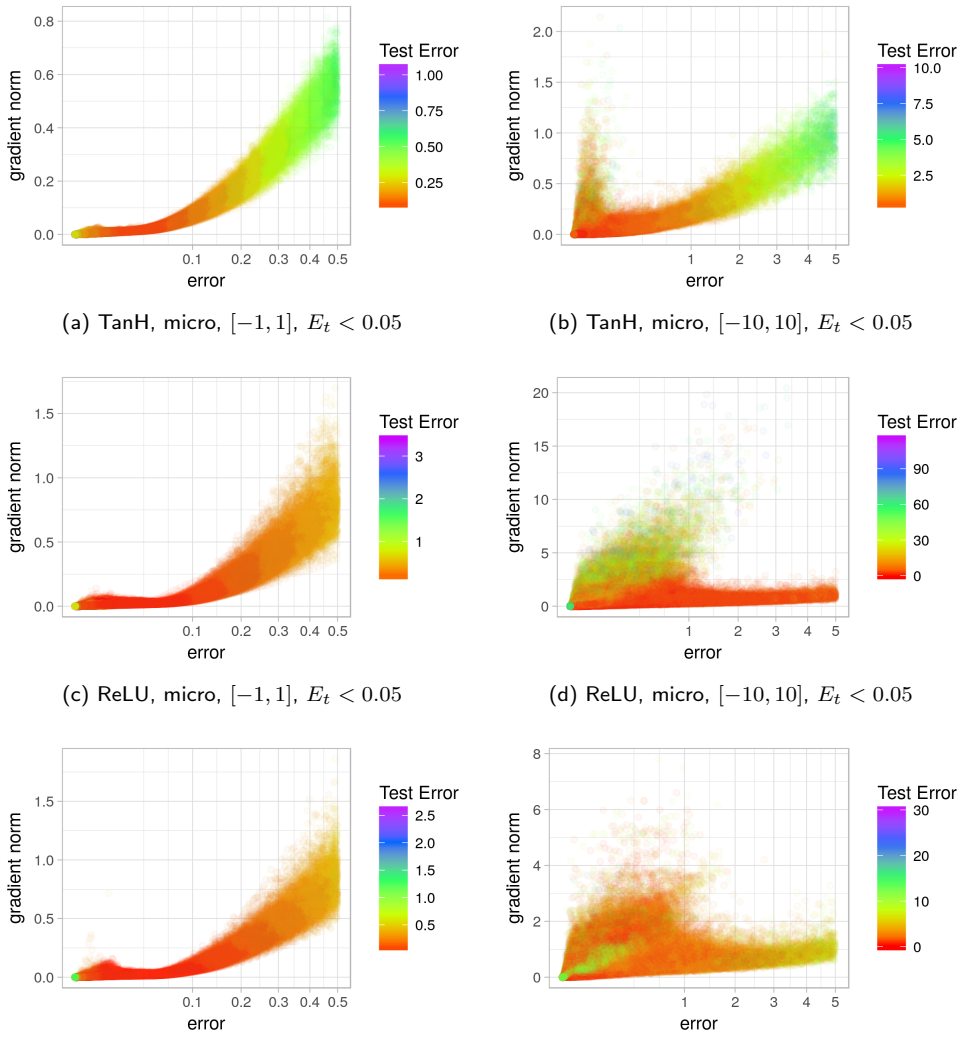


(a) TanH, micro, $[-10, 10]$ (b) TanH, macro, $[-10, 10]$

(c) ReLU, micro, $[-10, 10]$ (d) ReLU, macro, $[-10, 10]$

Figure 8.27: Box plots illustrating the degree of saturation associated with the different curvatures for the Cancer problem.

cluster became progressively more dense for ReLU and ELU as the step and initialisation range increased. Thus, for ReLU and ELU, finer granularity favoured wider basins, and coarser granularity favoured steeper, sharper basins. For TanH, the steeper basins did not manifest strongly for the macro $[-1, 1]$ and micro $[-10, 10]$ scenarios, but were clearly observable for the remaining two scenarios (micro $[-1, 1]$ and macro $[-10, 10]$). Figure 8.32 again illustrates that the indefinite points corresponded to points of higher saturation for both TanH and ReLU. Further, Figure 8.33 shows the l-g clouds for TanH and ReLU, colourised according to the estimated degree of saturation. For both activation functions, the steeper gradient cluster was clearly associated with the saturated



(a) TanH, micro, $[-1, 1]$, $E_t < 0.05$

(b) TanH, micro, $[-10, 10]$, $E_t < 0.05$

(c) ReLU, micro, $[-1, 1]$, $E_t < 0.05$

(d) ReLU, micro, $[-10, 10]$, $E_t < 0.05$

(e) ELU, micro, $[-1, 1]$, $E_t < 0.05$

(f) ELU, micro, $[-10, 10]$, $E_t < 0.05$

Figure 8.28: L-g clouds colourised according to the corresponding E_g values for the Cancer problem.

neurons.

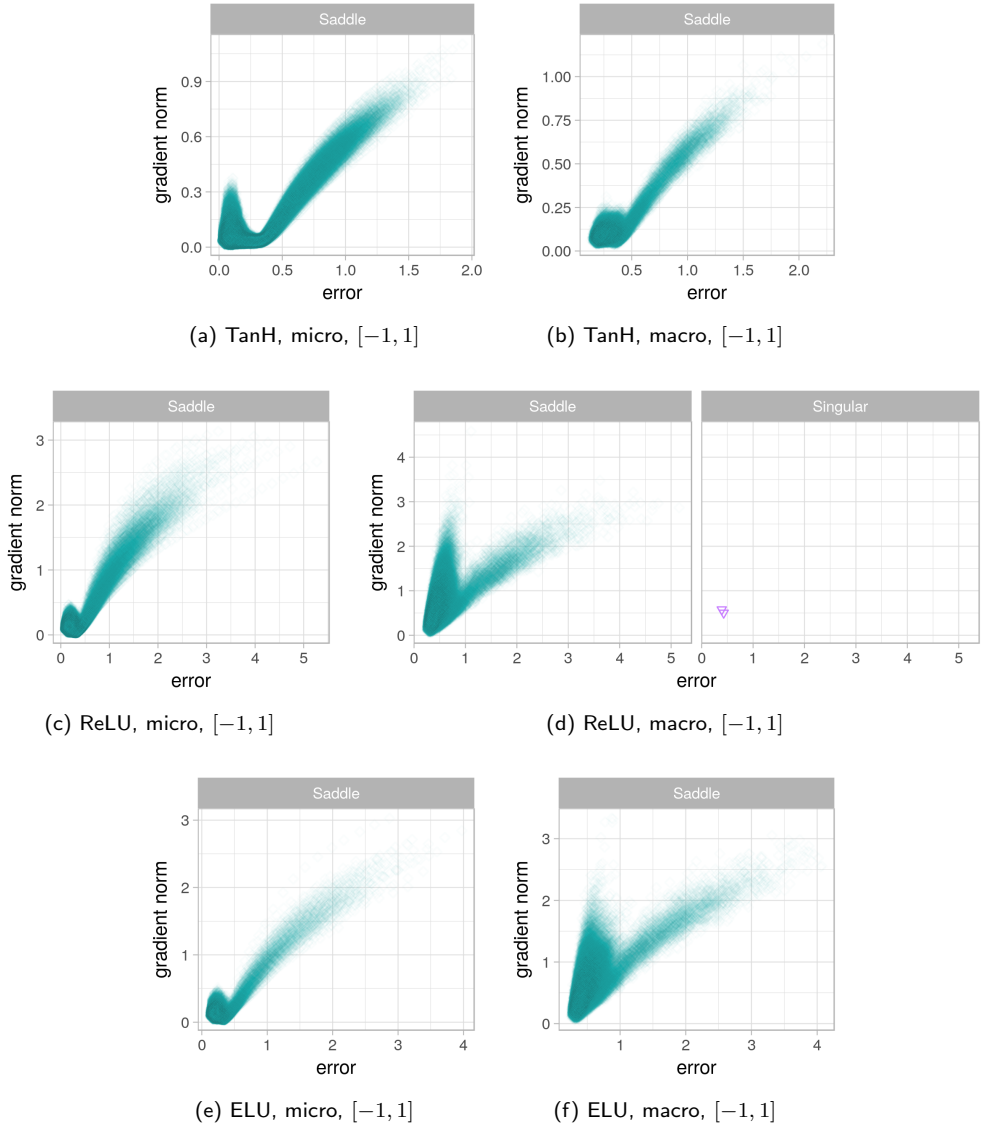
According to the classification results presented in Table B.11, TanH achieved the best generalisation performance under the scenarios in which the two-cluster split was more evident. This observation again indicates that steep valleys can contain solutions with good generalisation properties.

Table 8.6: Basin of attraction estimates calculated for the Heart problem on the E_t and E_g walks. Standard deviation shown in parenthesis.

	TanH		ReLU		ELU	
E_t	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	1.0000	954.2774	1.0000	967.9572	1.0000	969.4890
micro)	(0.0000)	(9.2356)	(0.0000)	(2.7861)	(0.0000)	(2.6436
$[-1, 1]$,	1.0000	86.9760	1.0032	85.8949	1.0050	86.0182
macro)	(0.0000)	(0.9320)	(0.0567)	(5.6751)	(0.0704)	(6.6016
$[-10, 10]$,	1.0000	960.0452	1.0000	967.3132	1.0000	967.6488
micro)	(0.0000)	(6.3670)	(0.0000)	(3.2789)	(0.0000)	(3.0297
$[-10, 10]$,	1.0000	86.6836	1.0170	83.5893	1.0103	84.2049
macro	0.0000)	(1.2558)	(0.1423)	(10.2744)	(0.1144)	(8.4039
E_g	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$,	2.4809	695.1029	1.0259	962.6718	1.0122	968.0473
micro	2.7620)	(354.5025)	(0.3175)	(65.7118)	(0.3034)	(38.4338
$[-1, 1]$,	1.3249	66.4177	1.0610	80.8311	1.0361	83.9103
macro	0.6306)	(27.4676)	(0.2736)	(16.1697)	(0.2170)	(11.9992
$[-10, 10]$,	1.0021	939.5545	1.0000	968.4547	1.0003	966.7501
micro	(0.0453)	(27.1684)	(0.0000)	(4.0867)	(0.0172)	(9.9001
$[-10, 10]$,	1.0009	86.0808	1.0463	80.6070	1.0531	79.9146
macro	(0.0296)	(3.4324)	(0.2352)	(15.1674)	(0.2525)	(15.9264

According to Table B.11, both ReLU and ELU generalised the best under the $[-1, 1]$ setting, indicating once again that these activation functions yield loss surfaces that are sensitive to the step size and initialisation range. Out of the three activation functions, ELU generalised the best.

Figure 8.34 shows the l-g clouds colourised according to the E_g values. For the $[-1, 1]$ walks, all activation functions yielded deteriorating E_g values as E_t approached zero. However, the band of good solutions was noticeably wider for both ReLU and ELU



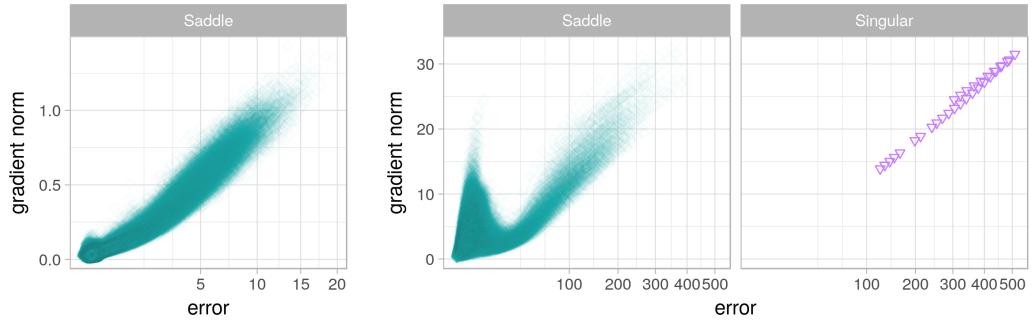
(a) TanH, micro, $[-1, 1]$ (b) TanH, macro, $[-1, 1]$

(c) ReLU, micro, $[-1, 1]$ (d) ReLU, macro, $[-1, 1]$

(e) ELU, micro, $[-1, 1]$ (f) ELU, macro, $[-1, 1]$

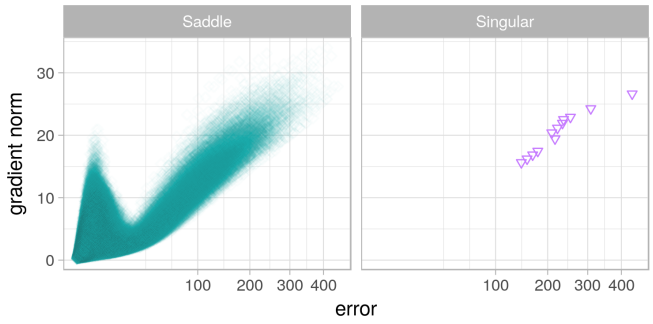
Figure 8.29: L-g clouds for the gradient walks initialised in the $[-1, 1]$ range for the Heart problem.

as compared to TanH, indicating that it was easier to find a good quality solution on the ReLU and ELU loss surfaces. For the $[-10, 10]$ initialisation range, the points of the steepest gradient produced the best generalisation performance for ReLU and ELU, confirming the hypothesis that steeper gradient cluster contains regularised minima.



(a) TanH, micro, $[-10, 10]$

(b) ReLU, micro, $[-10, 10]$



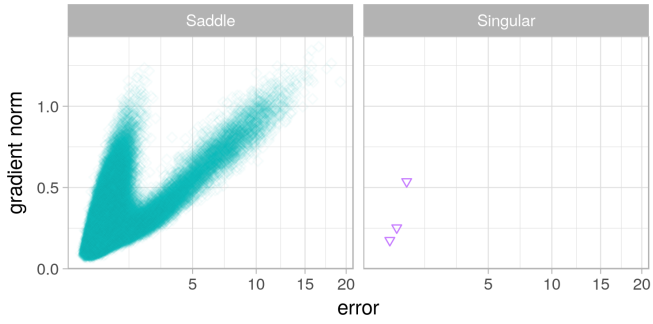
(c) ELU, micro, $[-10, 10]$

Figure 8.30: L-g clouds for the gradient walks initialised in the $[-10, 10]$ range for the Heart problem.

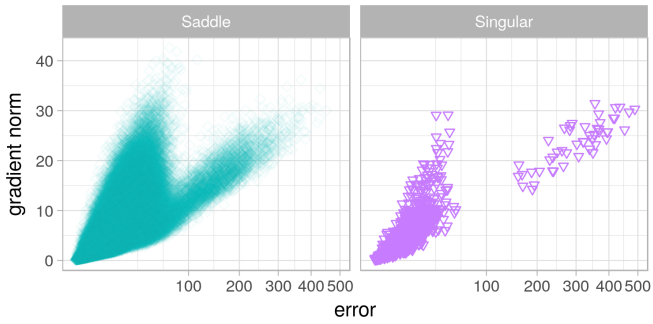
8.2.7 MNIST

Due to the high dimensionality of MNIST, Hessians were not calculated for this experiment. The reader is referred to [114] for an empirical analysis of the Hessians associated with the various NN architectures trained on the MNIST dataset.

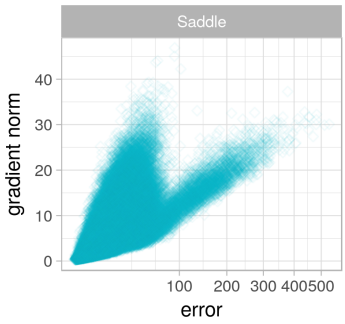
Figures 8.35 and 8.36 show the l-g clouds colourised according to the E_g values, obtained using the $[-1, 1]$ and $[-10, 10]$ initialisation ranges. Convergence to a single global attractor is evident for all activation functions. The results in Table 8.7 show that the walks generally did not become stuck more than twice. The $n_{stag} > 1$ values indicate that the MNIST error landscape was less trivial than the other problems considered, and the micro walks initialised in the $[-1, 1]$ range used half of the allocated steps to discover the global attraction basin. The classification results in Table B.12 in Appendix B show



(a) TanH, macro, $[-10, 10]$



(b) ReLU, macro, $[-10, 10]$



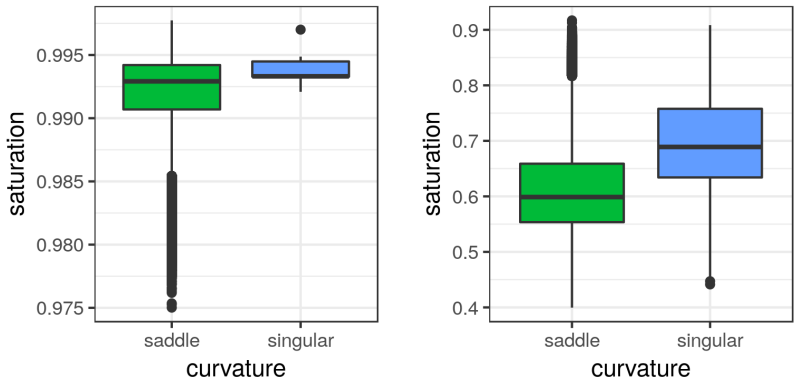
(c) ELU, macro, $[-10, 10]$

Figure 8.31: L-g clouds for the gradient walks initialised in the $[-10, 10]$ range for the Heart problem.

that a training error above 90% was achieved by each activation function under at least one of the scenarios, but the final generalisation performance generally did not exceed 60%. Out of the three activation functions considered, ELU once again yielded the most consistent generalisation performance.

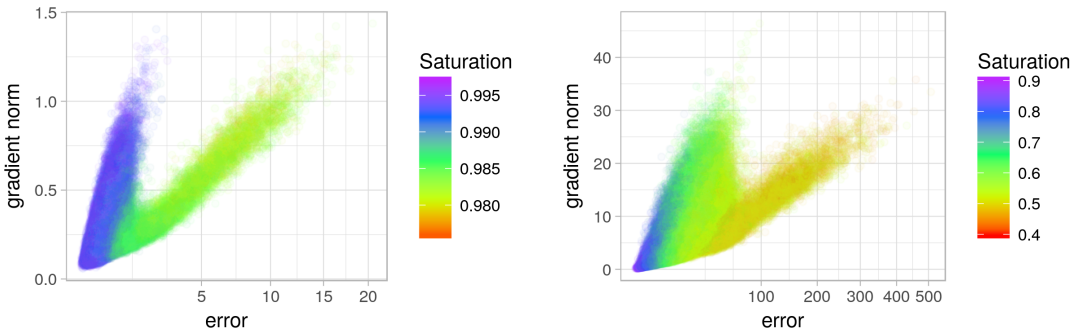
Figure 8.35 shows for the $[-1, 1]$ interval that ReLU and ELU yielded much higher gradients than TanH. The two-cluster split was evident for all activation functions. For ReLU and ELU, the steep gradient cluster generally yielded poorer generalisation performance than the shallow gradient cluster. This behaviour is especially evident for ReLU under the macro setting, as shown in Figure 8.35d. The poor generalisation performance of high gradient points is attributed to neuron saturation.

Figure 8.36 shows for the $[-10, 10]$ range that TanH yielded an l-g cloud without a well-formed structure. Low gradients and a noisy l-g cloud indicate that the TanH error



(a) TanH, macro, $[-10, 10]$ (b) ReLU, macro, $[-10, 10]$

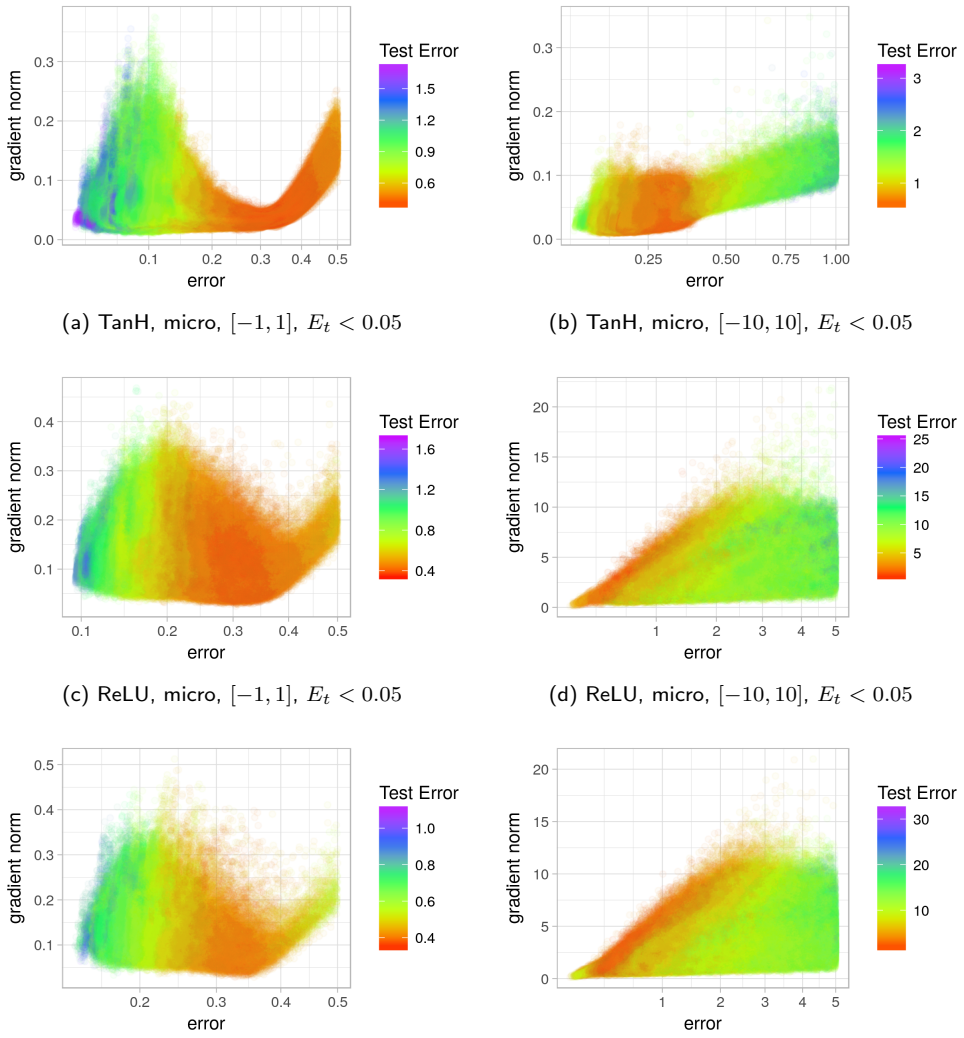
Figure 8.32: Box plots illustrating the degree of saturation associated with the different curvatures for the Heart problem.



(a) TanH, macro, $[-10, 10]$ (b) ReLU, macro, $[-10, 10]$

Figure 8.33: L-g clouds for the macro gradient walks initialised in the $[-10, 10]$ range for the Heart problem, colourised according to the estimated saturation.

landscape was not very searchable under the $[-10, 10]$ granularity setting. The results in Table B.12 confirm that both the training and the generalisation accuracy were below 50% when sampled in this range. ReLU and ELU also deteriorated with the increased step size and initialisation range, but not as drastically as TanH. The relative resilience of ReLU and ELU to the step size is attributed to the high gradients generated by these unbounded activation functions. ReLU and ELU evidently yielded a more searchable landscape for the high-dimensional MNIST problem, which correlates with the current



(a) TanH, micro, $[-1, 1]$, $E_t < 0.05$

(b) TanH, micro, $[-10, 10]$, $E_t < 0.05$

(c) ReLU, micro, $[-1, 1]$, $E_t < 0.05$

(d) ReLU, micro, $[-10, 10]$, $E_t < 0.05$

(e) ELU, micro, $[-1, 1]$, $E_t < 0.05$

(f) ELU, micro, $[-10, 10]$, $E_t < 0.05$

Figure 8.34: L-g clouds colourised according to the corresponding E_g values for the Heart problem.

deep NN learning insights [44, 117].

The split into two clusters became more pronounced for ReLU and ELU under the $[-10, 10]$ granularity setting, once again confirming the presence of narrow and wide valleys in the landscape. For ELU, points of good E_g values were found in the steep cluster, likely due to the embedded minima discovered. ReLU yielded higher E_t and

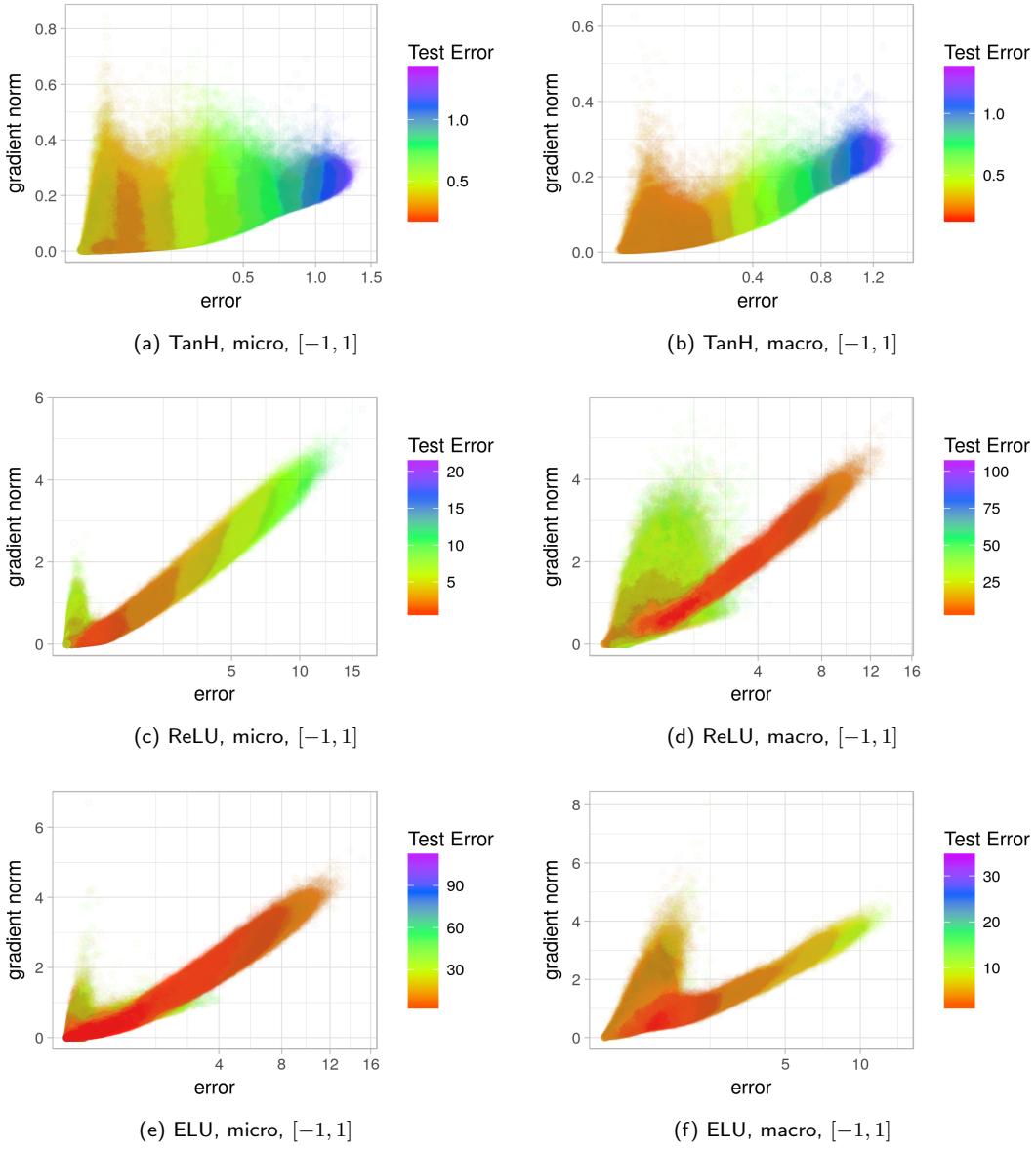
E_g values than ELU, confirming that the ELU error landscape was more resilient to overfitting.

8.2.8 Softmax

As discussed in Section 2.5, the softmax function is often used in the output NN layer for multinomial classification problems as an alternative to the sigmoid function. The softmax function ties the outputs into a single probability distribution, which makes the

Table 8.7: Basin of attraction estimates calculated for the MNIST problem on the E_t and E_g walks. Standard deviation shown in parenthesis.

	TanH		ReLU		ELU	
E_t	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$, micro	1.9275 (0.3346)	518.7926 (135.0496)	1.9556 (0.7151)	524.0519 (141.0278)	1.9206 (0.2952)	522.2484 (131.3190)
$[-1, 1]$, macro	1.0000 (0.0000)	81.2046 (1.4440)	1.0000 (0.0000)	86.7389 (1.2807)	1.0000 (0.0000)	86.7957 (0.5139)
$[-10, 10]$, micro	1.0000 (0.0000)	930.5987 (5.9444)	1.6019 (1.2018)	779.6422 (283.3337)	1.0000 (0.0000)	968.7611 (2.0536)
$[-10, 10]$, macro	1.0000 (0.0000)	84.7876 (0.7833)	1.0001 (0.0079)	86.2568 (2.1953)	1.0000 (0.0000)	87.1465 (0.5617)
E_g	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}
$[-1, 1]$, micro	2.5455 (1.0137)	406.9110 (149.8214)	8.9498 (4.1133)	124.9902 (89.5238)	8.9515 (4.3751)	123.3717 (116.0007)
$[-1, 1]$, macro	1.0000 (0.0000)	79.1788 (2.0895)	1.1930 (0.4445)	22.9150 (9.1853)	1.3948 (0.5980)	47.1115 (22.2410)
$[-10, 10]$, micro	1.0000 (0.0000)	918.5345 (7.9246)	6.5518 (1.6881)	84.1472 (28.7148)	5.6886 (5.7706)	453.8400 (381.7540)
$[-10, 10]$, macro	1.0000 (0.0000)	84.3475 (0.9705)	1.0889 (0.2990)	45.2955 (12.9575)	1.4408 (0.5790)	45.0293 (21.0153)



(a) TanH, micro, $[-1, 1]$

(b) TanH, macro, $[-1, 1]$

(c) ReLU, micro, $[-1, 1]$

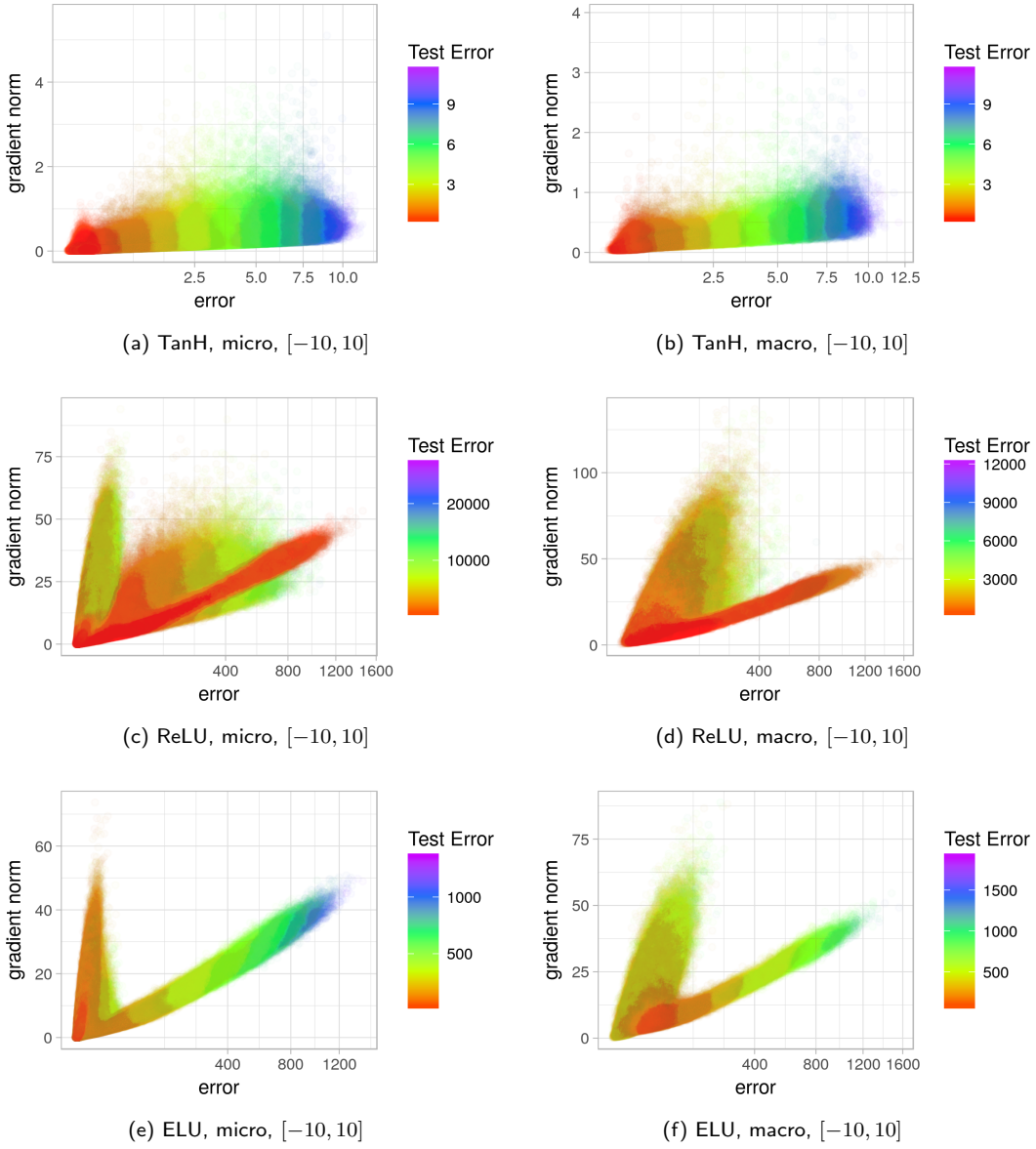
(d) ReLU, macro, $[-1, 1]$

(e) ELU, micro, $[-1, 1]$

(f) ELU, macro, $[-1, 1]$

Figure 8.35: L-g clouds for the gradient walks initialised in the $[-1, 1]$ range for the MNIST problem.

classification results more interpretable [17, 69]. To study the influence of the softmax function on the loss surfaces, the multinomial benchmark problems (Iris, Glass, and MNIST) were considered with the softmax output neurons.



(a) TanH, micro, $[-10, 10]$

(b) TanH, macro, $[-10, 10]$

(c) ReLU, micro, $[-10, 10]$

(d) ReLU, macro, $[-10, 10]$

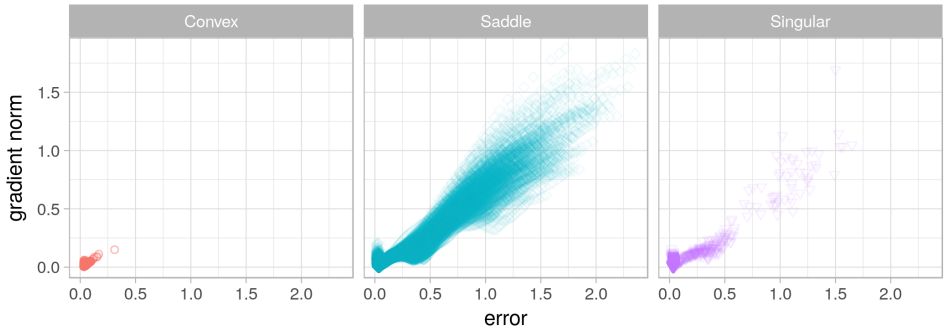
(e) ELU, micro, $[-10, 10]$

(f) ELU, macro, $[-10, 10]$

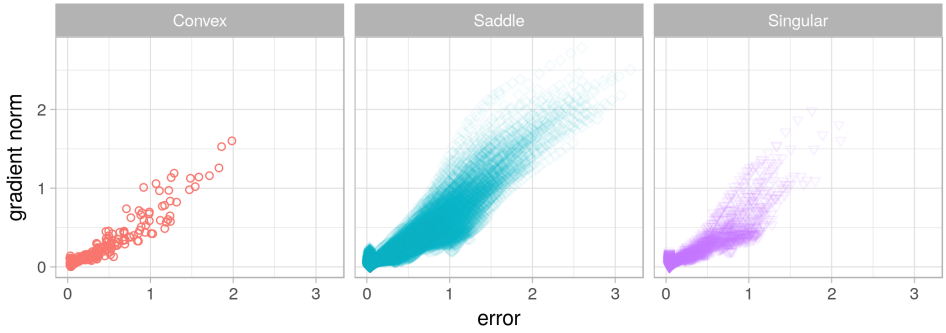
Figure 8.36: L-g clouds for the gradient walks initialised in the $[-10, 10]$ range for the MNIST problem.

Iris

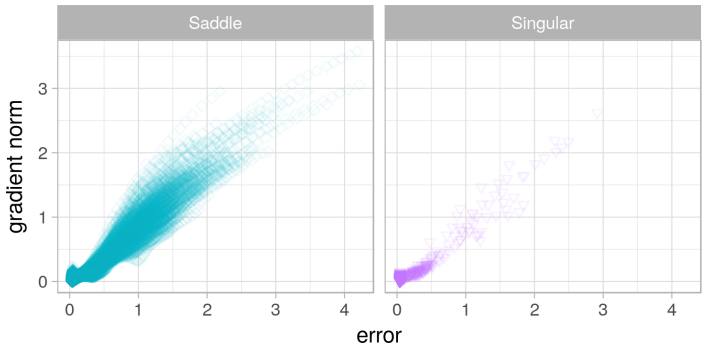
Figure 8.37 shows the l-g clouds obtained for the Iris problem under the $[-1, 1]$ micro setting. Compared to the l-g clouds shown in Figure 8.5, the softmax function yielded



(a) TanH, micro, $[-1, 1]$



(b) ReLU, micro, $[-1, 1]$



(c) ELU, micro, $[-1, 1]$

Figure 8.37: L-g clouds for the gradient walks initialised in the $[-1, 1]$ range for the Iris problem using softmax in the output layer.

l-g clouds of a similar shape, but different curvature. For all hidden neuron activation functions, the number of convex points decreased, and the number of indefinite points increased. Thus, compared to the sigmoid activation, softmax yielded more flatness

and less convexity. Figures 8.5 and 8.37 also show that the softmax function resulted in stronger gradients. Indeed, since softmax is a normalised exponential function, the exponent is expected to produce stronger gradients. Stronger gradients are beneficial to gradient-based algorithms such as backpropagation, which rely on the gradient magnitude to traverse the search space. Reduced convexity is attributed to the fact that individual outputs are combined into a single distribution, thus making individual outputs more dependent on one another. With the softmax function, changing the weights leading into one output neuron will have an effect on the quality of the remaining output neurons. Therefore, the output neuron weights can not be trained independently of one another, making the problem less separable, and thus increasing the complexity of the problem.

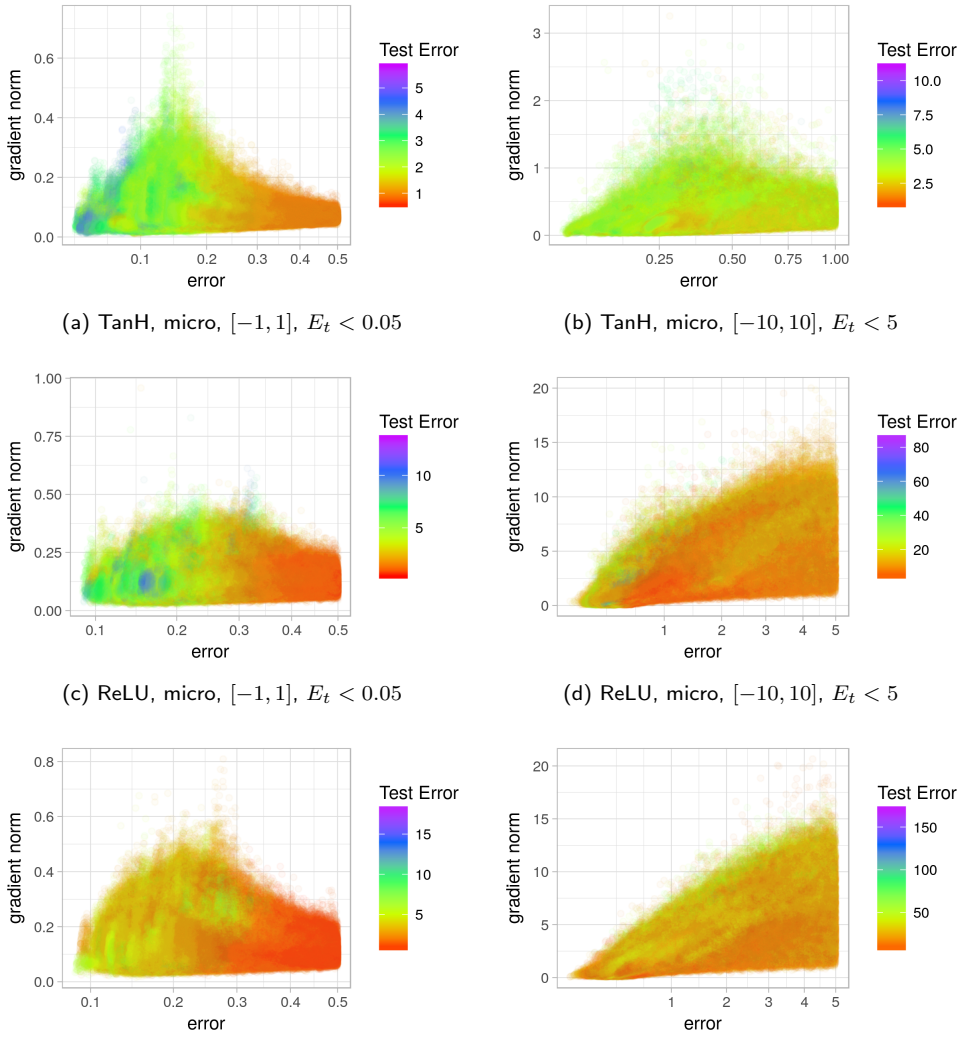
Similar behaviour was observed for the three remaining scenarios ($[-1, 1]$ macro, $[-10, 10]$ micro and macro, not shown for brevity): softmax yielded l-g clouds of the same shape as the ones yielded by sigmoid, but with stronger gradients, and with reduced convexity and an increased number of indefinite points. The classification results presented in Table B.13 show that the gradient walks discovered points of good ($> 80\%$) classification performance across all scenarios, where smaller steps were associated with better performance. Comparing these results to the sigmoid results in Table B.7, the softmax results are somewhat inferior to the sigmoid results. Reduced convexity of the softmax landscapes made the problem less searchable. Given the simplicity of the Iris problem, combining the outputs into a single probability distribution did not introduce a noticeable advantage, and only made the problem harder.

Glass

For the Glass problem, softmax also yielded l-g clouds of the same shape as shown in Section 8.2.4, but with no convex points and more indefinite points for all activation functions. Saddle curvature remained prevalent.

Figure 8.38 shows the l-g clouds for the Glass problem using softmax in the output layer. Compared to the l-g clouds shown in Figure 8.22, the softmax l-g clouds are noisier, with less well-defined structure. The global minimum attractor is wider, with stronger gradients, and more variability in E_g values. For the $[-1, 1]$ setting, deterioration of the

E_g values at the global minimum can be observed. The classification results in Table B.14 show that softmax yielded higher classification accuracy values on the training set than sigmoid (Table B.9). Thus, wider attraction basins with stronger gradients made the softmax loss surface easier to search than the sigmoid loss surface. Therefore, softmax is the preferable output activation choice for the Glass problem.



(a) TanH, micro, $[-1, 1]$, $E_t < 0.05$

(b) TanH, micro, $[-10, 10]$, $E_t < 5$

(c) ReLU, micro, $[-1, 1]$, $E_t < 0.05$

(d) ReLU, micro, $[-10, 10]$, $E_t < 5$

(e) ELU, micro, $[-1, 1]$, $E_t < 0.05$

(f) ELU, micro, $[-10, 10]$, $E_t < 5$

Figure 8.38: L-g clouds colourised according to the corresponding E_g values for the Glass problem using softmax in the output layer.

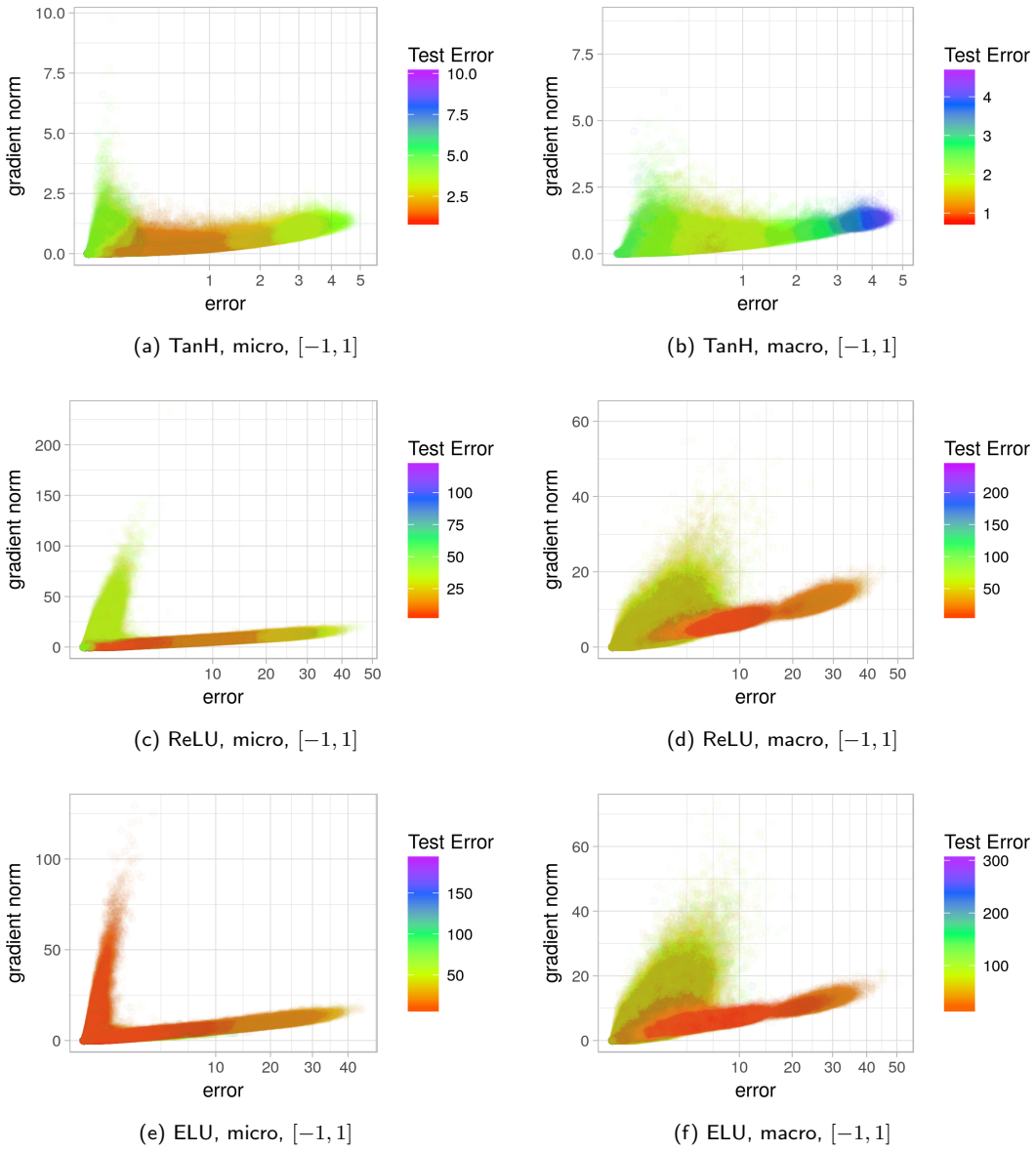
MNIST

Figures 8.39 and 8.40 show the l-g clouds obtained for the MNIST problem with the softmax output neurons, colourised according to the E_g values. Figure 8.39 shows that for all activation functions considered, the gradients yielded by softmax were stronger than the gradients yielded by sigmoid (see Figure 8.35). The split into steep and shallow gradient clusters also manifested stronger for ReLU and ELU when using the softmax activation. Wide and narrow valleys are an inherent feature of the NN loss surfaces, and the stronger gradients imposed by the softmax function have exaggerated this feature. Figure 8.39 also illustrates that the shallow gradient cluster was associated with better generalisation performance than the steep gradient cluster, and that the generalisation performance at the global minimum was generally poor. The classification results reported in Table B.15 show that all the activation functions performed better in terms of both training and generalisation error with the softmax activation compared to the sigmoid activation. The MNIST dataset is complex enough to take advantage of the benefits offered by softmax.

Figure 8.40 shows under the $[-10, 10]$ setting that the l-g cloud shapes were similar to the l-g clouds obtained for the sigmoid activation function, but the gradients once again were much stronger. For ReLU and ELU, the steep gradient attractor was heavier than the shallow gradient attractor, indicating that the loss surface was harder to search under this granularity setting. The classification results in Tables B.12 and B.15 confirm that sigmoid yielded better classification errors on the training dataset under the $[-10, 10]$ setting. Thus, softmax made the error landscape more searchable for fine granularity walks, but made the problem harder when considered on a larger scale. This observation once again highlights the sensitivity of ReLU and ELU to the initialisation range and step size.

8.3 Ruggedness, Gradients, Neutrality

The ruggedness, gradients, and neutrality metrics, as discussed in Section 3.4, were calculated for all problems to gain more insight into the fitness landscape properties associated with the various activation functions. The magnitudes of the numerical gradients were



(a) TanH, micro, $[-1, 1]$

(b) TanH, macro, $[-1, 1]$

(c) ReLU, micro, $[-1, 1]$

(d) ReLU, macro, $[-1, 1]$

(e) ELU, micro, $[-1, 1]$

(f) ELU, macro, $[-1, 1]$

Figure 8.39: L-g clouds for the gradient walks initialised in the $[-1, 1]$ range for the MNIST problem using softmax in the output layer.

used instead of gradient estimates.

Figures 8.41 and 8.42 show for all activation functions that the FEM values for the micro setting were generally smaller than the FEM values for the macro setting,

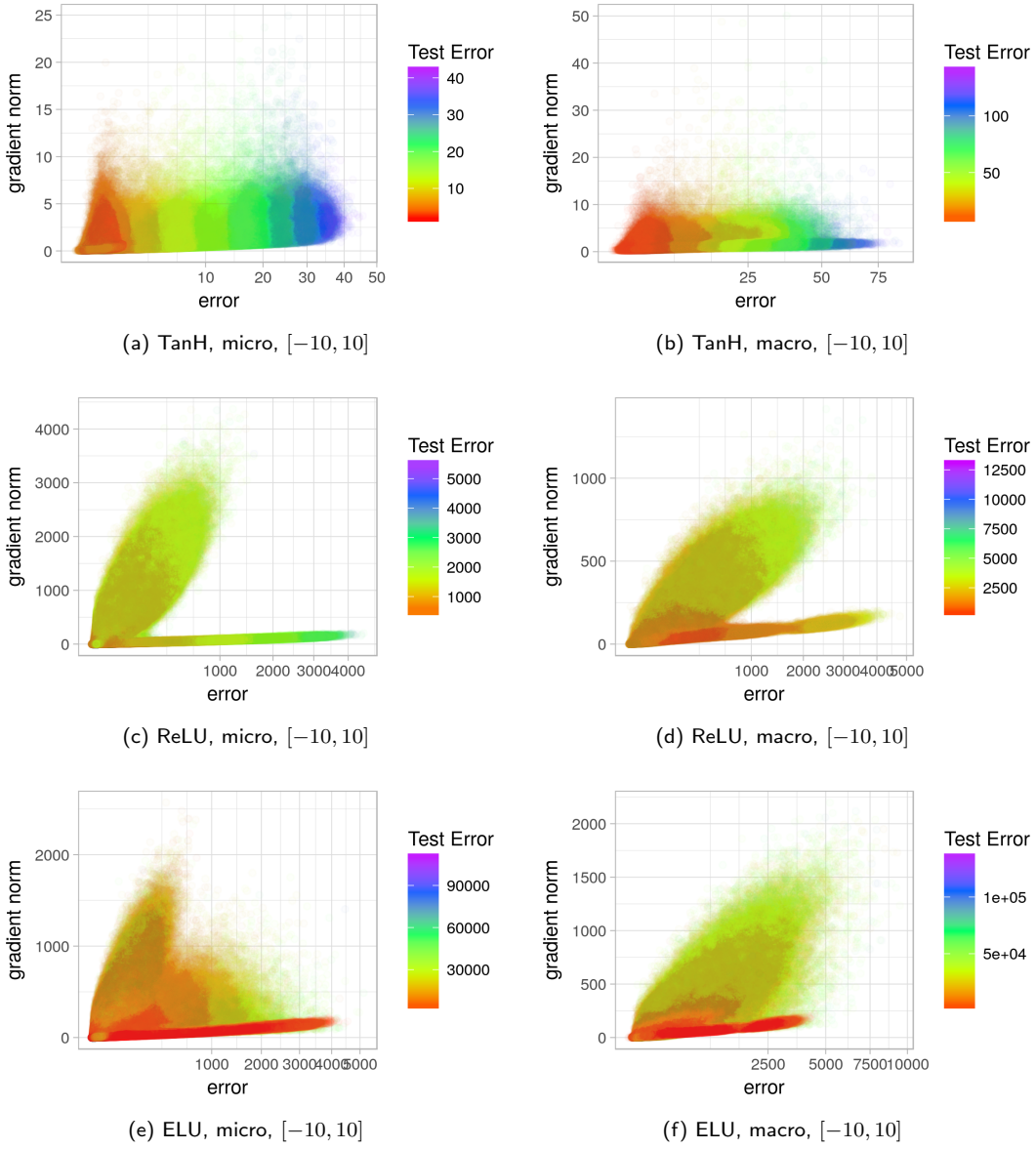


Figure 8.40: L-g clouds for the gradient walks initialised in the $[-10, 10]$ range for the MNIST problem using softmax in the output layer.

irrespective of the initialisation range. The macro $[-1, 1]$ and micro $[-10, 10]$ setting used the same maximum step size (0.2, corresponding to 10% of the first interval and 1% of the second interval), yet yielded different FEM values on all problems considered. Since high

FEM is associated with oscillating behaviour, a smaller FEM for the larger initialisation range indicates that fewer oscillations were observed using the same maximum step size. Therefore, the attraction basins widened as the initialisation range increased. Thus, if a step size related parameter such as the learning rate is optimised for a particular initialisation interval, the same parameter is likely to yield suboptimal performance if applied on a different initialisation range.

Ruggedness under the micro $[-1, 1]$ range was similar for the three activation functions on all problems except for XOR. However, both ReLU and ELU generally exhibited higher ruggedness than TanH for the macro $[-1, 1]$ and micro $[-10, 10]$ settings. This observation confirms that ReLU and ELU were more sensitive to the initialisation range, and became less searchable as the initialisation range increased. For the macro $[-10, 10]$ setting, TanH exhibited higher FEM than ReLU and ELU on most problems. This behaviour is attributed to the saturation exhibited by TanH under the coarse granularity setting. Saturation is associated with narrow valleys, which cause oscillations, and thus high FEM values.

Figures 8.41 and 8.42 show that the gradients steadily increased for all activation functions as the step size and the initialisation range increased. The TanH error landscape gradients were consistently smaller than ReLU and ELU gradients for all problems considered, indicating that the ReLU and ELU landscapes contained more information to guide a gradient-based search. Figures 8.41 and 8.42 show for ReLU and ELU that G_{dev} was larger than G_{avg} for the XOR, Iris, and Cancer problems, i.e. the easiest problems considered in this study. Problems of higher dimensionality and/or higher complexity, such as Glass and MNIST, yielded $G_{avg} > G_{dev}$, i.e. more consistent gradients. Malan [81] made an observation that $G_{dev} >> G_{avg}$ is indicative of step-like, sudden changes in the landscape. According to this observation, and results shown in Figures 8.41 and 8.42, more complex problems yielded simpler gradient walk trajectories, with fewer sudden jumps. This corresponds to the previously made observations that high-dimensional NNs may in fact be easier to optimise than low-dimensional NNs, due to the prevalence of saddle points and other mathematical properties of high-dimensional spaces [28, 33].

Figure 8.43 shows the neutrality measures obtained for the seven benchmark prob-

lems. Once again, no strict neutrality was observed for most problems. The only two problems that exhibited neutrality were Cancer (micro setting) and MNIST (micro $[-1, 1]$ setting). Since gradient walks were used for sampling, neutrality is attributed to convergence to a basin of attraction. Figure 8.42 shows that Cancer and MNIST exhibited low FEM under the settings where neutrality was observed. Thus, the gradient walks discovered attraction basins, and then exploited them with little to no oscillations.

Finally, Figures 8.44 and 8.45 show the FLA measures obtained for the Iris, Glass, and MNIST problems using softmax output units. For Glass and MNIST, higher FEM values were obtained using the softmax activation. Higher gradients were obtained for Iris, Glass, and MNIST. Higher FEM is attributed to steeper and narrower valleys yielded by the softmax activation, resulting in stronger oscillations. MNIST yielded less neutrality with softmax, as shown in Figure 8.45, and Iris resulted in neutrality for the $[-10, 10]$ micro setting. The same setting corresponded to low FEM values, indicating that the gradient walks exploited attraction basins with little to no oscillations.

8.4 Conclusions

This chapter analysed the effect of five different activation functions on the resulting NN error landscapes. TanH, ReLU, and ELU were considered in the hidden layer. Sigmoid and softmax were considered in the output layer. All experiments were conducted under four granularity settings, namely, micro and macro walks using the $[-1, 1]$ and $[-10, 10]$ initialisation ranges.

The choice of activation function did not have an effect on the total number of attractors in the search space, but affected the properties of the discovered basins of attraction. ReLU and ELU yielded steeper attraction basins with stronger gradients than TanH. ReLU exhibited the most convexity out of the activation functions considered, and ELU exhibited the least flatness. The stationary points exhibited by ReLU and ELU were generally more connected than the ones exhibited by TanH, indicating that ReLU and ELU yielded more searchable landscapes. However, ReLU and ELU exhibited stronger sensitivity to the step size and the initialisation range than TanH.

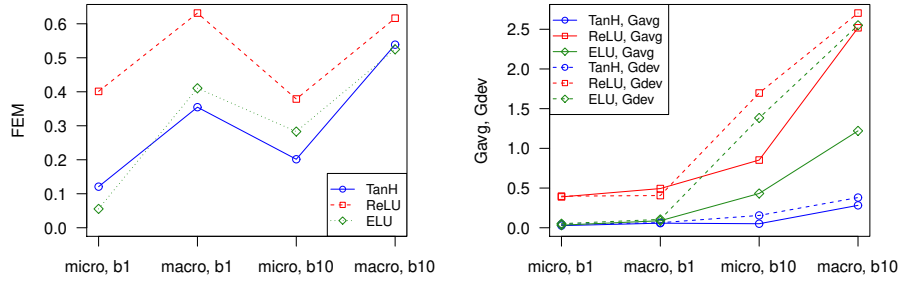
All activation functions exhibited a split into two clusters of steep and shallow gra-

dients. The steep gradient cluster was associated with indefinite, i.e. flat curvature on a selection of problems, caused by inactive, or non-contributing weights. The steep gradient cluster was also often associated with poor generalisation performance. Thus, steep gradients were attributed to narrow valleys, which were associated with saturated neurons. In individual cases, points of high generalisation performance were discovered in the steep gradient clusters. These were attributed to the embedded regularised minima.

Out of the three activation functions considered in the hidden layer, ELU consistently exhibited superior generalisation performance. Thus, the loss surface yielded by the ELU activation function was the most resilient to overfitting.

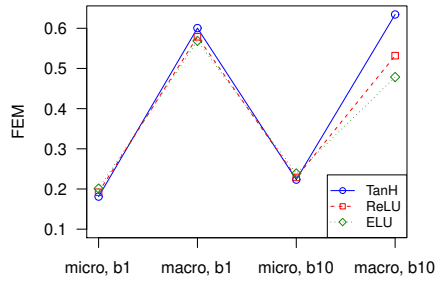
Softmax in the output layer was shown to reduce the convexity of the problem by introducing a higher degree of non-separability. Only the problems of higher dimensionality and complexity were shown to benefit from the softmax activation function. Softmax yielded wider attraction basins with stronger gradients, resulting in more searchable landscapes when considered on a fine granularity scale.

The next chapter considers the effect of the NN architecture parameters on the resulting NN loss surfaces.

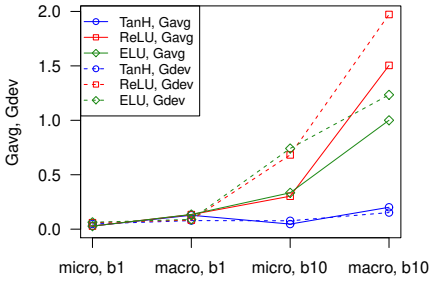


(a) XOR, FEM

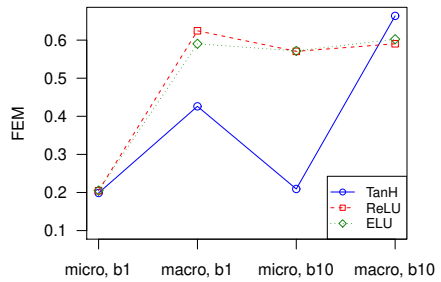
(b) XOR, gradients



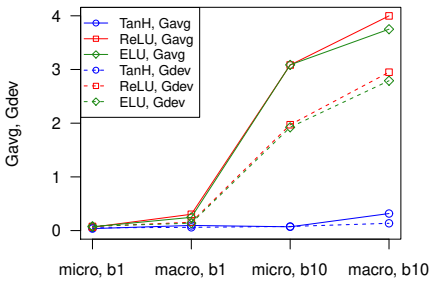
(c) Iris, FEM



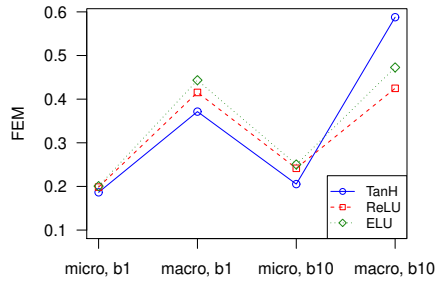
(d) Iris, gradients



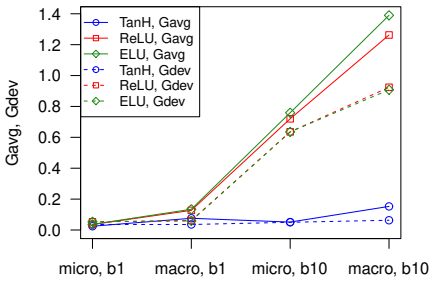
(e) Diabetes, FEM



(f) Diabetes, gradients

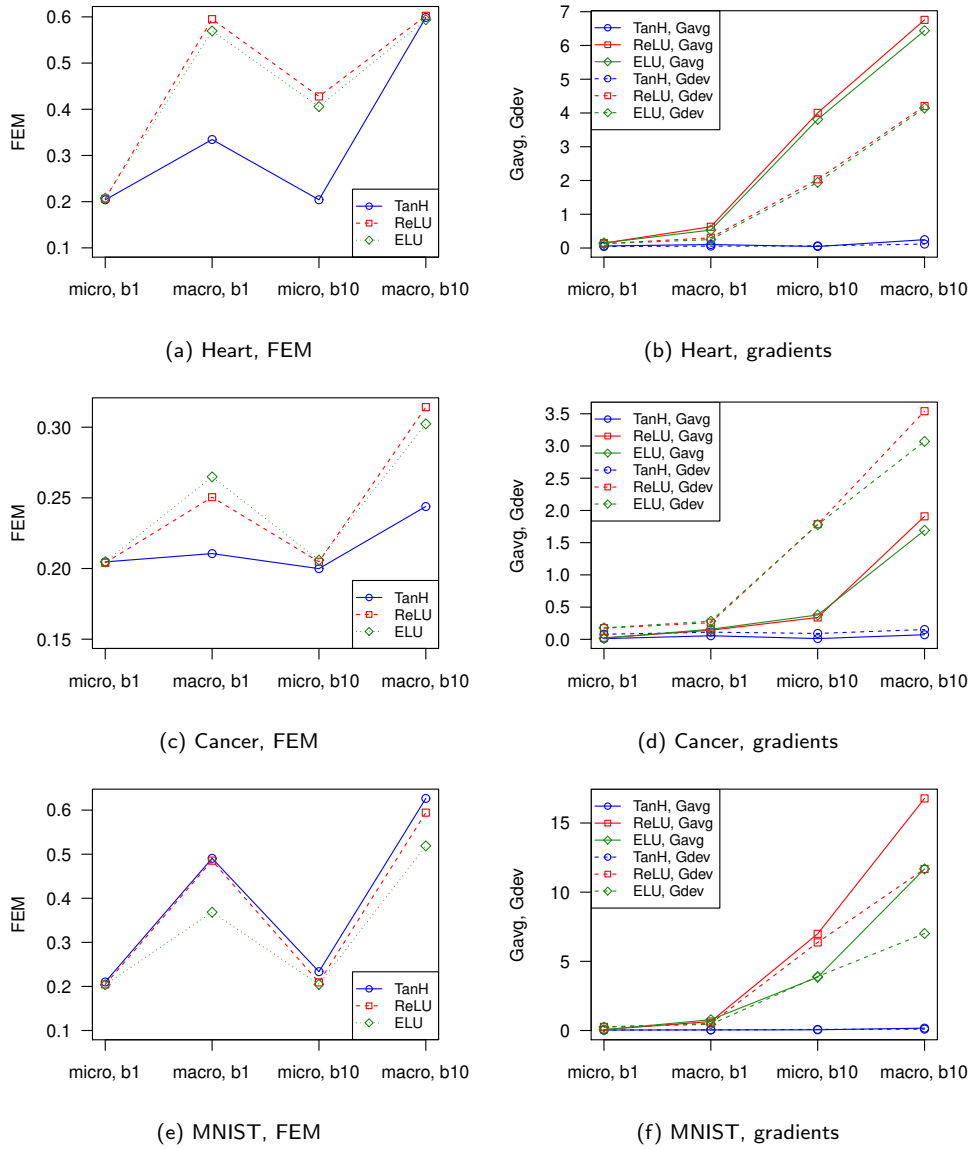


(g) Glass, FEM



(h) Glass, gradients

Figure 8.41: FLA metrics for the XOR, Iris, Diabetes, and Glass problems.



(a) Heart, FEM

(b) Heart, gradients

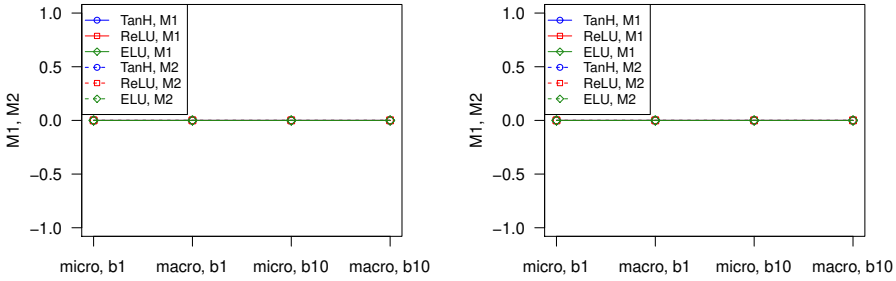
(c) Cancer, FEM

(d) Cancer, gradients

(e) MNIST, FEM

(f) MNIST, gradients

Figure 8.42: FLA metrics for the Heart, Cancer, and MNIST problems.



(a) XOR, neutrality

(b) Iris, neutrality

(c) Diabetes, neutrality

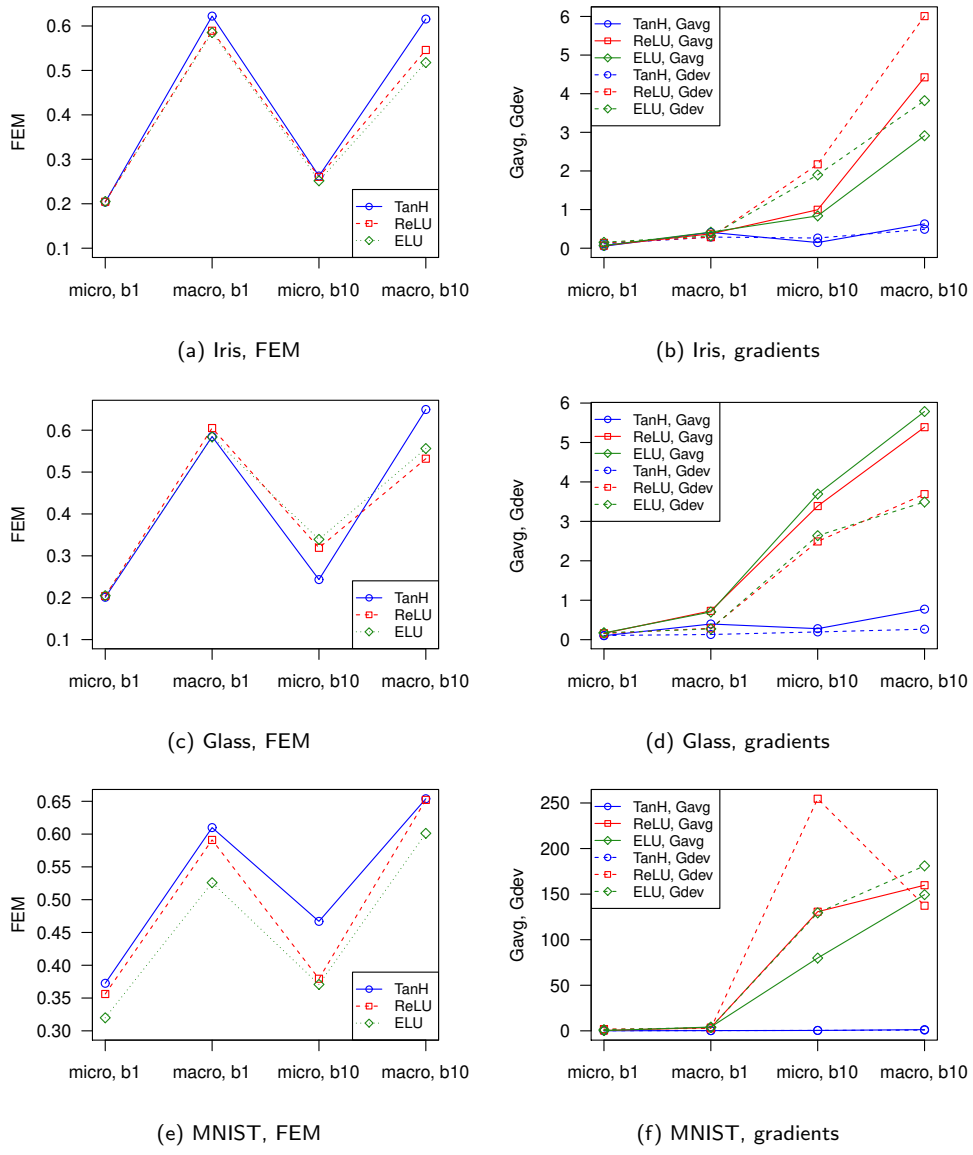
(d) Glass, neutrality

(e) Heart, neutrality

(f) Cancer, neutrality

(g) MNIST, neutrality

Figure 8.43: Neutrality metrics for the various problems considered.



(a) Iris, FEM

(b) Iris, gradients

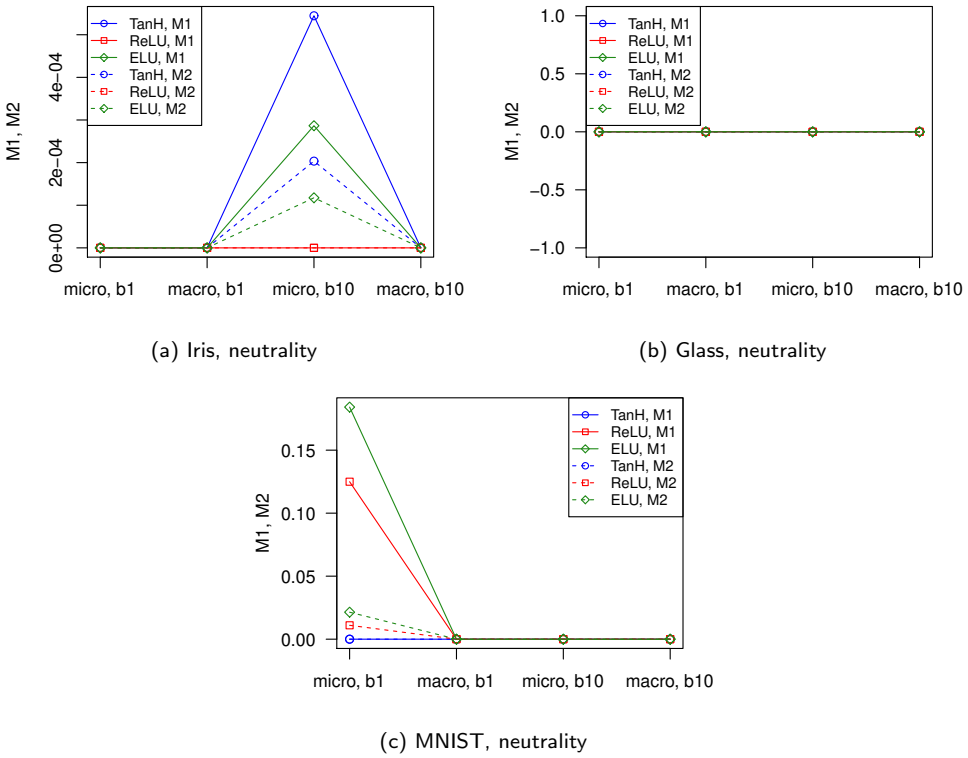
(c) Glass, FEM

(d) Glass, gradients

(e) MNIST, FEM

(f) MNIST, gradients

Figure 8.44: FLA metrics for the Iris, Glass, and MNIST problems using softmax output neurons.



(a) Iris, neutrality

(b) Glass, neutrality

(c) MNIST, neutrality

Figure 8.45: Neutrality metrics for Iris, Glass, and MNIST problems using softmax output neurons.

Chapter 9

Neural Network Architectures

NN performance is known to depend on the chosen NN architecture, i.e. the number of neurons, hidden layers, and the structure of connections [8, 54, 55, 70, 146]. A NN with too few trainable parameters will not be able to fit complex non-linear data, and a NN with an excessive number of trainable parameters was argued to be prone to overfitting [146]. However, empirical studies have also been published where excessively large, i.e. over-parametrised NNs were shown to not be as prone to overfitting as expected [82]. In fact, Caruana et al. [20] provided empirical evidence that the generalisation performance does not deteriorate due to over-parametrisation. Indeed, if the excessive weights of an over-parametrised NN architecture are set to zero by the training algorithm, the NN can be regularised implicitly. An important question to answer is how easily can an optimisation algorithm discover such implicitly regularised solutions. Analysis of the fitness landscapes associated with different architectures can provide some answers.

Kordos and Duch [68] studied the loss surfaces of feed-forward NNs with a varied number of hidden neurons and hidden layers using the PCA projections. NN error landscapes were reported to increase in complexity as more hidden layers were added to the architecture. NNs with more than one hidden layer were shown to contain multiple high-laying plateaus. Under-parametrisation, i.e. too few hidden neurons, induced a flat landscape without the ravines and valleys characteristic of NN error landscapes. A gradual increase in the number of hidden neurons lead to an initial increase in the number of global minima visible on the PCA projections. However, the landscape visibly

flattened as more redundant weights were added to the architecture [68].

Similarly, Sagun et al. [114] performed a Hessian matrix analysis of over-parametrised NNs, and showed that an increase in the dimensionality of the search space due to the addition of hidden neurons yielded an increase in the number of near-zero eigenvalues of the Hessian, i.e. increased flatness. The same authors observed that no high error local minima were detected when the NN architecture was over-parametrised [115]. Theoretical studies have also been published showing that over-parametrised models do not exhibit high error local minima [6, 74]. In fact, recent studies claim that adding an excessively large hidden layer (larger than the number of training points in the dataset) guarantees that almost all local minima will be globally optimal [100]. This correlates with an earlier study by Gallagher [38], where the FDC measure was used to estimate the searchability of NN error landscapes of varied dimensionality, and the increase in the number of inputs and hidden neurons consistently yielded higher searchability scores.

This chapter aims to investigate NN loss surfaces under various NN architecture settings using the FLA techniques. The rest of the chapter is structured as follows: Section 9.1 discusses the experimental procedure. Section 9.2 presents a visual and numerical analysis of stationary points and basins of attraction associated with the various NN architectures. Section 9.3 presents FLA measures of gradients, ruggedness, and neutrality associated with the various architectures. Section 9.4 concludes the chapter.

9.1 Experimental Procedure

The aim of this study is to visually and numerically investigate the local minima and basins of attraction exhibited by a representative set of NN architectures. This section discusses the experimental set-up of the study, and is structured as follows: Section 9.1.1 lists the benchmark problems used, Section 9.1.2 describes the NN hyperparameters and architectures considered in this study, and Section 9.1.3 outlines the sampling algorithm parameters.

9.1.1 Benchmark problems

For the purpose of this study, seven classification problems described in Appendix A were used, namely XOR, Iris, Diabetes, Glass, Cancer, Heart, and MNIST.

9.1.2 Architectures

All experiments employed feed-forward NNs with the ELU activation function in the hidden layers. The ELU activation function was chosen based on the findings presented in Chapter 8, where ELU was shown to exhibit superior generalisation properties. For the binary classification problems, the sigmoid function was used in the output layer. For the multinomial classification problems, the softmax activation function was used in the output layer. The output activation function choices are again based on the findings presented in Chapter 8. The entropic loss was chosen based on the findings presented in Chapter 7, where CE was shown to produce more searchable landscapes with fewer local minima.

To study the influence of the number of hidden neurons on the NN error surfaces, each problem was considered with h minimal number of hidden neurons (specified in Appendix A), with twice as many hidden neurons as prescribed by the minimal architecture ($2 \times h$), and with ten times as many hidden neurons ($10 \times h$). These settings were chosen to simulate a minor increase in the number of hidden neurons ($2 \times h$), as well as a more substantial increase corresponding to the next order of magnitude ($10 \times h$).

To study the influence of the number of hidden layers on the NN error surfaces, 1, 2, and 3 hidden layers were considered for each hidden layer size as discussed in the preceding paragraph. The same number of hidden neurons was used for each successive hidden layer.

9.1.3 Sampling parameters

The same sampling parameters as discussed in Section 7.2.3 were used for the experiments. Progressive gradient walks (refer to Section 6.1.1) were used for the purpose of sampling. The total number of walks was set to be $2 \times m$, where m is the dimensionality of the search space. The walks were unbounded, but two distinct initialisation

ranges were considered, namely $[-1, 1]$ and $[-10, 10]$. Two granularity settings were used throughout the experiments: micro, where the maximum step size was set to 1% of the initialisation range, and macro, where the maximum step size was set to 10% of the initialisation range. Micro walks performed 1000 steps each, and macro walks performed 100 steps each. All datasets except XOR were split into 80% training and 20% test subsets. To calculate the training (E_t) and the generalisation (E_g) errors, the entire train/test subsets were used for all the problems except MNIST. For MNIST, random batches of 100 patterns were sampled from the respective training and test sets. The same data subsets were used to calculate the average classification accuracy obtained by the last step of the gradient walks. The classification accuracy of the training set is further referred to as C_t, and the classification accuracy of the generalisation set is referred to as C_g.

9.2 Empirical Study of Modality

This section presents an analysis of apparent local minima and the corresponding basins of attraction as captured by the progressive gradient walks for the various NN architectures. L-g clouds, proposed in Chapter 6, are employed for the purpose of this study. Sections 9.2.1 to 9.2.7 present the analysis of the various NN architectures for each problem.

9.2.1 XOR

Figure 9.1 summarises the curvature information obtained for the different architectures and sampling settings considered. Each bar in the plot corresponds to a distinct granularity setting, and is colourised proportionally to the curvature of the sampled points. The plots are grouped horizontally according to the total number of hidden neurons ($h, 2 \times h$, and $10 \times h$), and vertically according to the total number of hidden layers (1, 2, and 3). Curvature information was obtained by calculating the eigenvalues of the Hessian for each sampled point.

According to Figure 9.1, an increase in the hidden layer size caused a reduction in convexity and an increase in flatness (singular Hessians). Indeed, the addition of

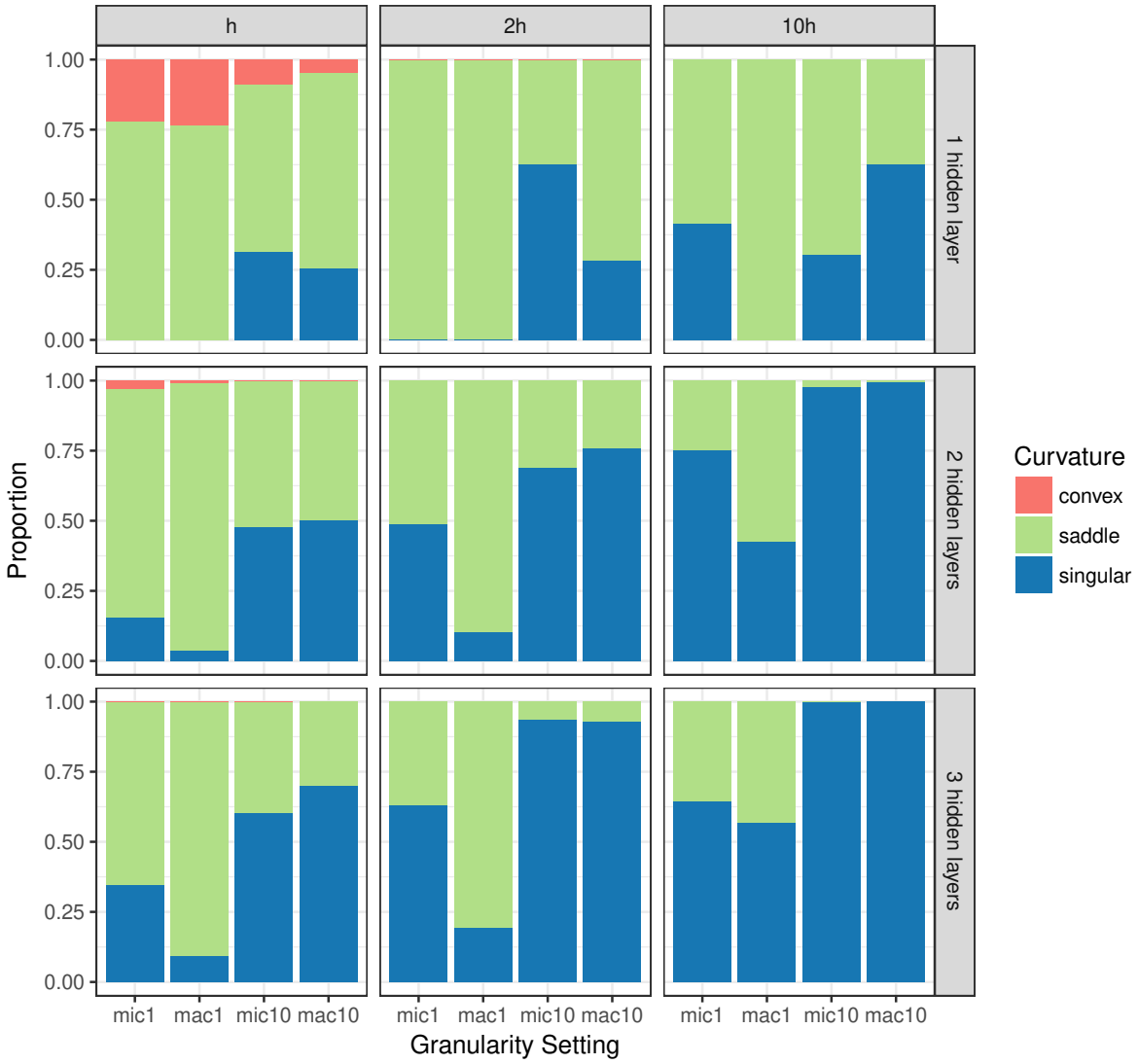


Figure 9.1: Histogram representation of the curvature information sampled by the gradient walks for the XOR problem for various NN architecture settings.

extra neurons to a minimal architecture introduces unnecessary, or redundant weights. Since more compact solutions, i.e. solutions with fewer weights, are embedded in over-parametrised architectures [86], the discovery of such solutions will cause the unnecessary neurons to be disabled, thus introducing flatness.

Figure 9.1 shows that an increase in the number of hidden layers had a similar effect:

Convexity decreased, and flatness increased. In fact, according to Figure 9.1, an increase in the number of hidden layers increased the flatness more rapidly than an increase in the hidden layer size. The rapid increase in flatness associated with deeper architectures is attributed to the inter-dependent variable structure of feed-forward NNs. Since each layer propagates the signals to the next layer, each layer has the ability to set the incoming signals to zero. In other words, if a neuron in a later layer saturates, the effects of saturation will influence the contribution of the weights in all preceding layers.

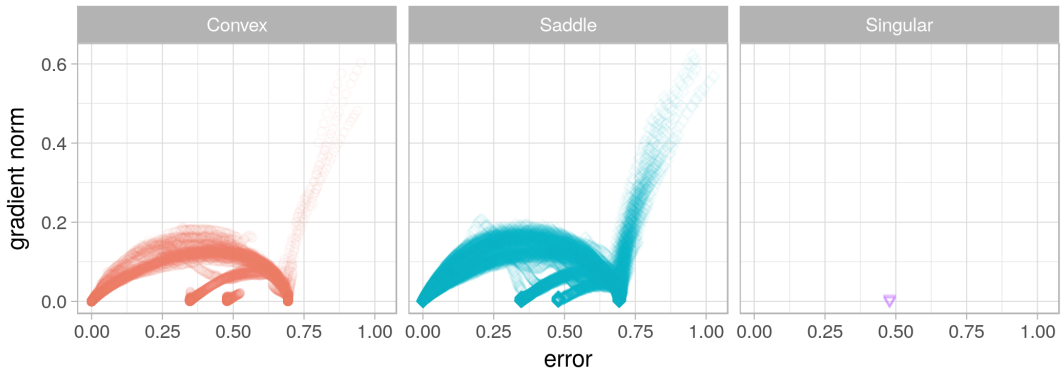
Out of the four granularity settings, macro walks initialised in the $[-1, 1]$ interval yielded the least amount of flatness for most architecture settings considered. This observation indicates that the choice of the initialisation interval and the step size are important parameters to optimise for a NN training algorithm.

An increase in flatness due to an increase in dimensionality, whether by adding extra neurons, or by adding extra layers, agrees with the findings of Sagun et al. [114], where Hessian analysis of over-parametrised NNs was performed. A question that remains to be answered is whether the modality of the NN error surface changes when hidden neurons/hidden layers are added, and whether the effect of adding hidden neurons differs from the effect of adding hidden layers.

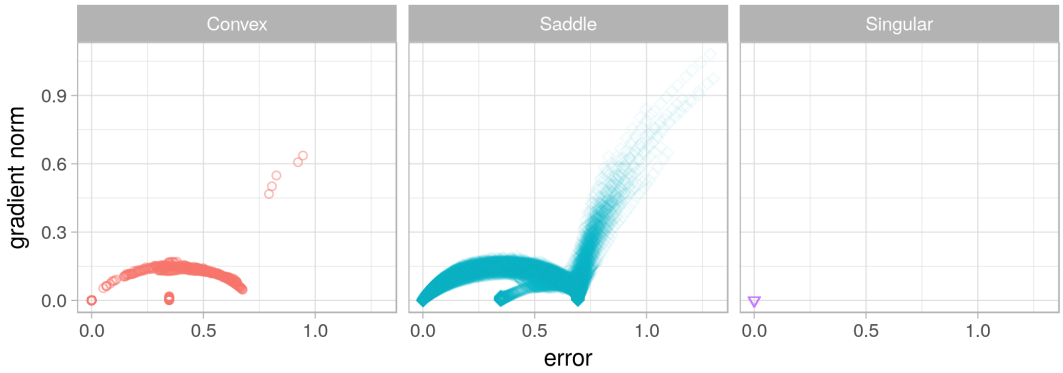
The effect of hidden neurons

Figure 9.2 shows the l-g clouds obtained for the single hidden layer architectures with a different number of hidden neurons. Figure 9.2 clearly shows that the number of convex attractors, i.e. local minima, reduced as more hidden neurons were added. For $h = 2$, four attractors, three of them constituting local minima, were observed (Figure 9.2a). For $h = 4$, only three attractors were detected, two of them constituting local minima (Figure 9.2b). Finally, for $h = 20$, two non-convex attractors were observed, thus local minima were eliminated altogether (Figure 9.2c).

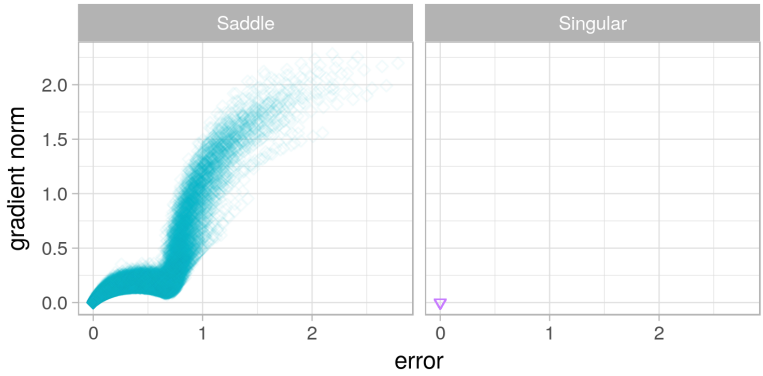
Figure 9.3 shows the l-g clouds obtained for the 2-hidden layer architectures with a different number of hidden neurons. The same trend is evident: The number of attractors decreased as more hidden neurons were added. Figure 9.3c also indicates that the addition of the excessive neurons yielded a split into two clusters, namely the high gradient cluster associated with flat curvature in individual dimensions, and a low gradient



(a) $h = 2$, micro, $[-1, 1]$

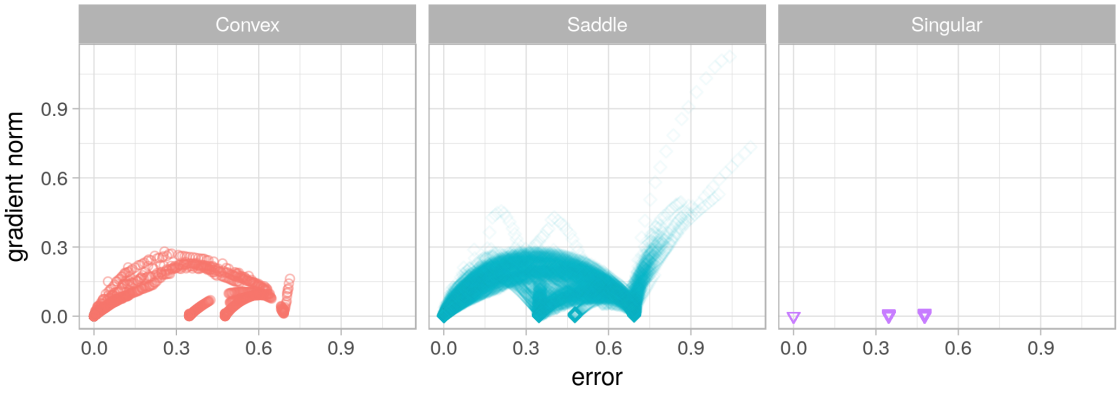


(b) $2 \times h = 4$, micro, $[-1, 1]$

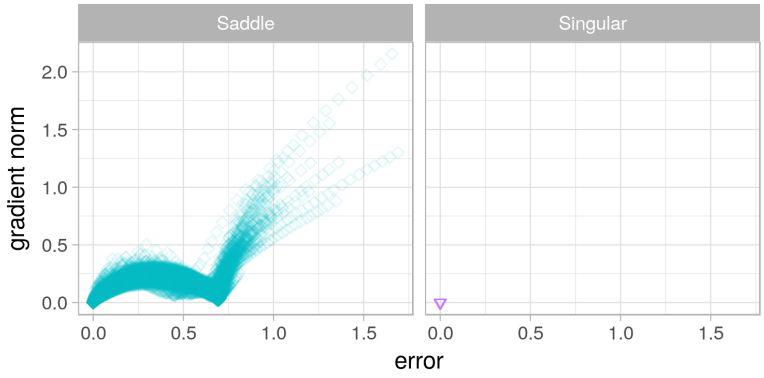


(c) $10 \times h = 20$, micro, $[-1, 1]$

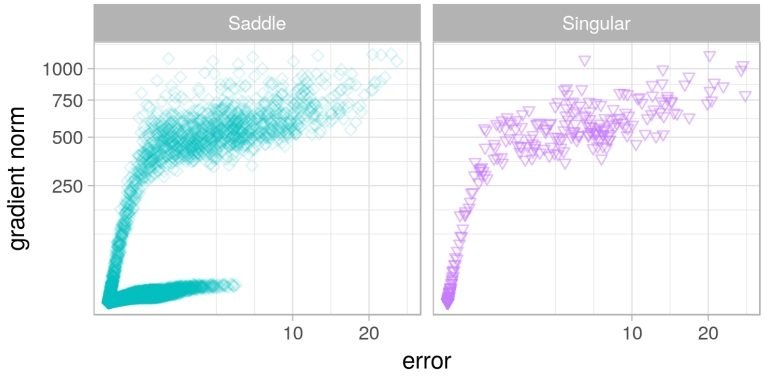
Figure 9.2: L-g clouds for the micro gradient walks initialised in the $[-1, 1]$ range for the XOR problem for the various number of hidden neurons in a single hidden layer.



(a) $h = 2$, micro, $[-1, 1]$



(b) $2 \times h = 4$, micro, $[-1, 1]$



(c) $10 \times h = 20$, micro, $[-1, 1]$

Figure 9.3: L-g clouds for the micro gradient walks initialised in the $[-1, 1]$ range for the XOR problem for the various number of hidden neurons in two consecutive hidden layers.

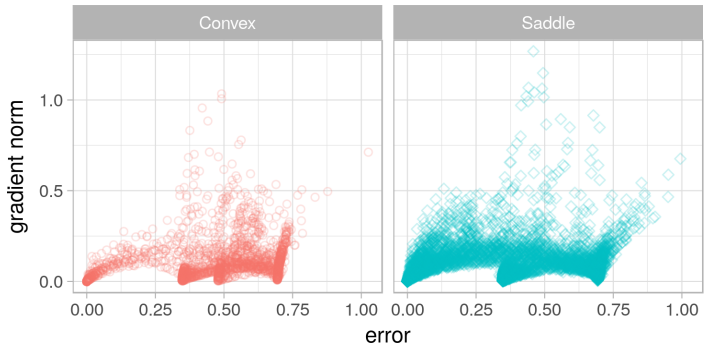
cluster of saddle curvature. In a $2 \times 20 \times 20 \times 1$ feed-forward architecture used for the XOR problem, a large number of neurons can be safely disabled without damaging the quality of the model, since the optimal architecture for XOR is $2 \times 2 \times 1$. Therefore, these unnecessary neurons contribute to the flatness of the resulting NN loss surface. Thus, as previously discussed in Chapter 8, steep gradient clusters, associated with narrow valleys, contain embedded, or implicitly regularised solutions.

Thus, an increase in the number of hidden neurons yielded a decrease in the number of convex attractors for the XOR problem, and thus reduced the number of local minima. This observation correlates with the recent study of Nguyen and Hein [100], where using more hidden neurons than the number of training points in the dataset was theoretically shown to guarantee that almost all local minima would be globally optimal.

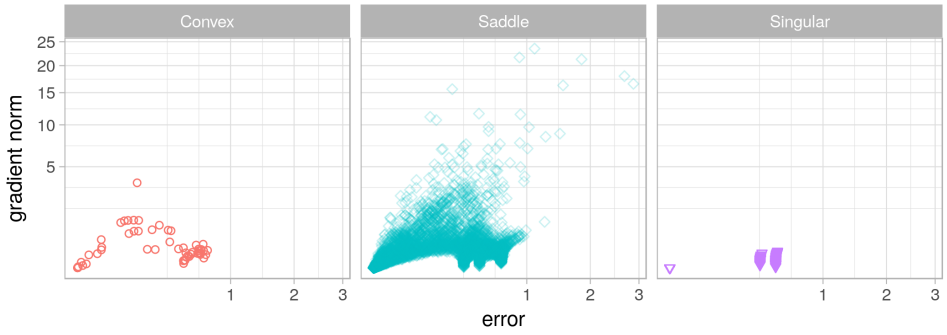
The effect of hidden layers

Figure 9.4 shows the l-g clouds obtained for the $[-1, 1]$ macro walks executed on the loss surface yielded by a NN architecture with a varied number of hidden layers (1, 2, and 3) and two hidden neurons per layer. Figure 9.4a shows that four stationary convex attractors were discovered for the single hidden layer architecture, three of them constituting local minima. The addition of the second hidden layer (Figure 9.4b) decreased the convexity and increased the flatness around the attractors, but the total number of attractors remained the same. The addition of the third hidden layer (Figure 9.4c) also yielded exactly four zero-gradient attractors. Thus, the number of hidden layers did not change the modality of the landscape for the XOR problem, i.e. the number of attractors of a unique error value remained the same. Figure 9.5 shows that the same behaviour was observed for the architectures with four hidden neurons per layer: For the various number of hidden layers, exactly two attractors were observed.

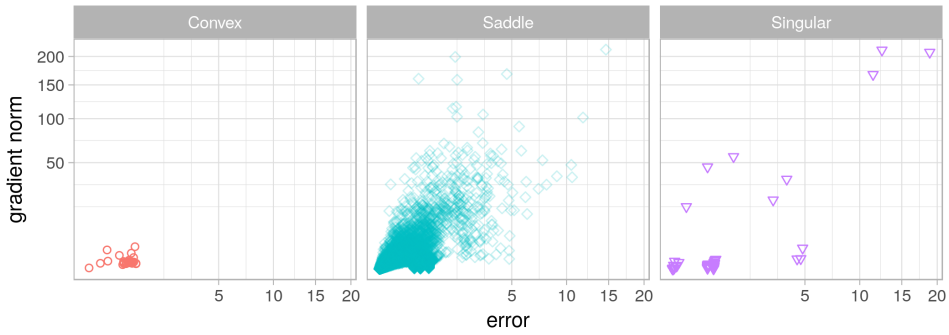
Even though the number of attractors was not altered by the number of hidden layers, the properties of the said attractors changed with a change in the architecture depth. In addition to the decreased convexity and increased flatness, a drastic increase in the gradient magnitudes was observed for deeper architectures. Figures 9.4 and 9.5 show that each new layer increased the maximum gradient by an order of magnitude. Figures 9.2 and 9.3 show that the addition of more hidden neurons also caused an



(a) 1 hidden layer, macro, $[-1, 1]$



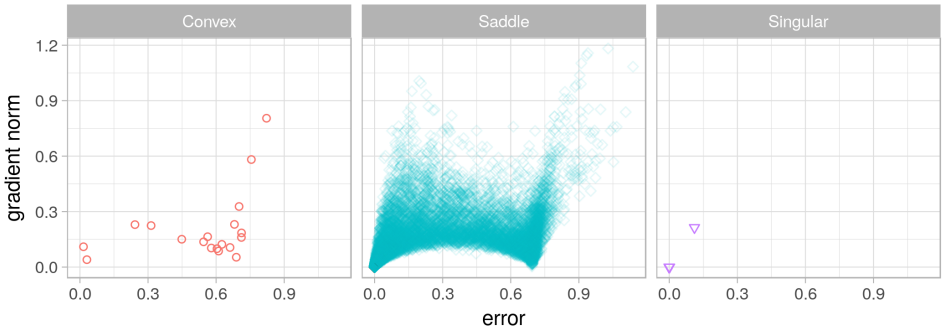
(b) 2 hidden layers, macro, $[-1, 1]$



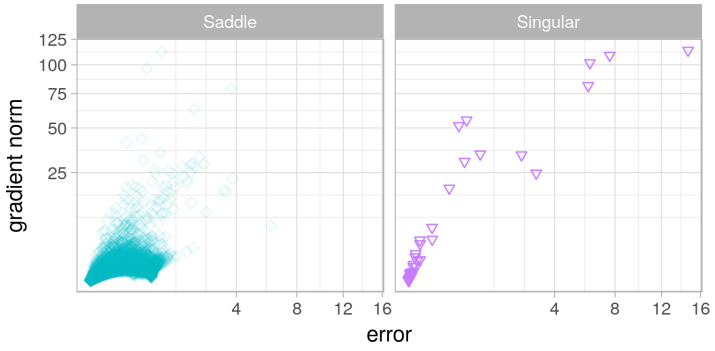
(c) 3 hidden layers, macro, $[-1, 1]$

Figure 9.4: L-g clouds for the macro gradient walks initialised in the $[-1, 1]$ range for the XOR problem for the various number of hidden layers, with $h = 2$ for each layer.

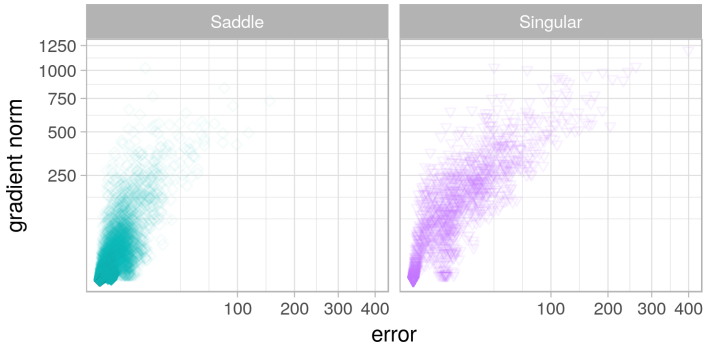
increase in the gradient magnitudes, but not as drastic, especially for the single hidden layer architecture. The range of the error values also increased rapidly for each new layer added.



(a) 1 hidden layer, macro, $[-1, 1]$



(b) 2 hidden layers, macro, $[-1, 1]$



(c) 3 hidden layers, macro, $[-1, 1]$

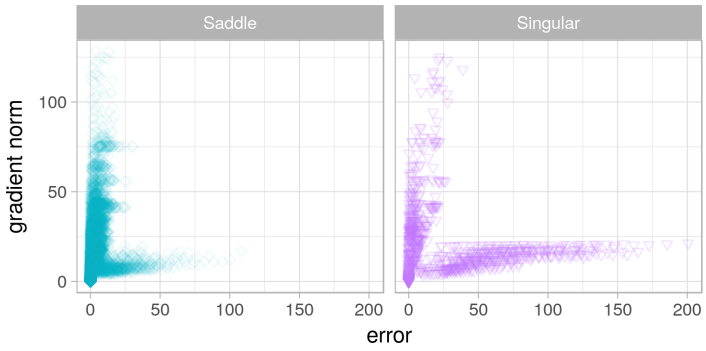
Figure 9.5: L-g clouds for the macro gradient walks initialised in the $[-1, 1]$ range for the XOR problem for the various number of hidden layers, with $h = 4$ for each layer.

Table 9.1 reports the n_{stag} and l_{stag} values calculated for all architectures under the four granularity settings. The n_{stag} values consistently decreased as the hidden layer size increased from h to $10 \times h$, confirming that the number of attractors decreased. The same

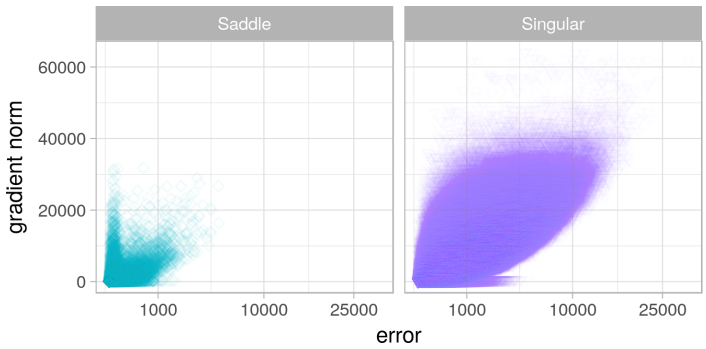
Table 9.1: Basin of attraction estimates calculated for the XOR problem for the various NN architectures. Standard deviation is shown in parenthesis.

	$h = 2$		$2 \times h = 4$		$10 \times h = 20$		
	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}	
$[-1, 1]$,	1.7000	492.9157	1.4059	614.9676	1.0025	884.4772	1 hidden layer
micro	(0.5467)	(210.9447)	(0.4911)	(187.0502)	(0.0496)	(36.3527)	
$[-1, 1]$,	1.1778	47.8111	1.1471	51.3588	1.0840	19.5437	
macro	(0.3823)	(18.3239)	(0.3542)	(15.7978)	(0.6269)	(16.9338)	
$[-10, 10]$,	1.0889	919.2556	1.0412	946.8176	1.3431	879.8992	
micro	(0.3542)	(135.3812)	(0.2735)	(91.5106)	(1.0337)	(231.6628)	
$[-10, 10]$,	1.0889	69.0389	1.0588	56.5343	1.0329	19.0548	
macro	(0.3542)	(22.8096)	(0.4564)	(28.4734)	(0.7276)	(20.1805)	
$[-1, 1]$,	1.7333	526.2489	1.1838	799.7351	1.4380	781.4364	2 hidden layers
micro	(0.4989)	(187.4153)	(0.3873)	(169.7235)	(0.7723)	(264.8623)	
$[-1, 1]$,	1.2267	47.5533	1.0973	62.7447	1.0833	63.4931	
macro	(0.4781)	(19.4510)	(0.4175)	(17.3743)	(0.2764)	(14.0827)	
$[-10, 10]$,	4.0800	447.2633	6.2857	118.2072	7.8147	77.8347	
micro	(3.3337)	(380.1790)	(2.8891)	(104.0755)	(3.2646)	(97.9395)	
$[-10, 10]$,	1.2267	56.3944	1.1739	49.9106	1.3822	30.7981	
macro	(0.5790)	(26.4186)	(0.4490)	(23.4397)	(0.6021)	(15.5443)	
$[-1, 1]$,	1.8952	523.5546	1.3491	787.2199	5.4840	151.4570	3 hidden layers
micro	(0.7675)	(199.3711)	(0.7297)	(225.9290)	(1.8892)	(102.6854)	
$[-1, 1]$,	1.4000	49.2786	1.2474	56.9088	1.1341	56.0215	
macro	(0.6561)	(24.5483)	(0.5836)	(21.9462)	(0.3479)	(16.2139)	
$[-10, 10]$,	4.7524	328.1548	5.6135	152.2001	7.1316	103.3774	
micro	(3.1556)	(329.2022)	(3.1569)	(161.8085)	(2.3415)	(68.1521)	
$[-10, 10]$,	1.3143	51.0778	1.2456	53.4804	1.2727	49.0455	
macro	(0.5906)	(22.2077)	(0.5645)	(21.7501)	(0.4454)	(19.2713)	

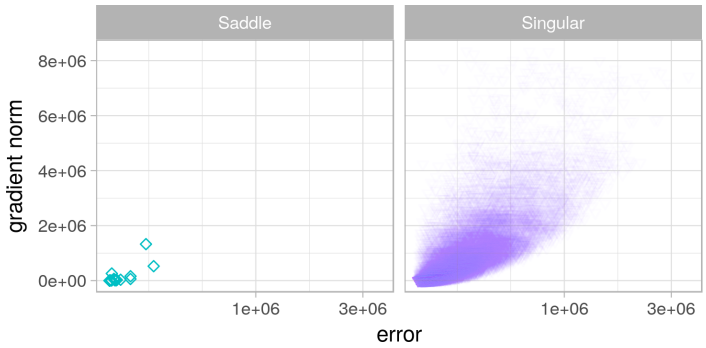
cannot be said about the number of hidden layers: The n_{stag} values did not decrease, and in some cases even increased as more hidden layers were added. In particular, the



(a) 1 hidden layer, micro, $[-10, 10]$



(b) 2 hidden layers, micro, $[-10, 10]$



(c) 3 hidden layers, micro, $[-10, 10]$

Figure 9.6: L-g clouds for the micro gradient walks initialised in the $[-10, 10]$ range for the XOR problem for the various number of hidden layers, with $h = 20$ for each layer.

$[-10, 10]$ micro setting yielded high n_{stag} values for 2 and 3 hidden layer architectures. To better understand this behaviour, l-g clouds were generated for the $[-10, 10]$ micro walks performed on NN architectures with $10 \times h$ hidden neurons, for the various number of hidden layers (see Figure 9.6). Figure 9.6a shows that horizontal clusters of consistent gradients were detected in the vertical cluster of steep gradients. If these clusters correspond to valleys of varied steepness, then high n_{stag} values indicate that transition between various valleys was possible. For all scenarios shown in Figure 9.6, a high degree of flatness was observed. As the number of layers increased, the steep gradient cluster became wider. The high number of stagnation points identified by the n_{stag} values is thus attributed to the increased flatness of the landscape.

The l_{stag} values reported in Table 9.1 also indicate that the average length of the stagnation areas shrunk as the number of layers increased. Thus, more oscillations were observed for deeper architectures, and convergence to a basin of attraction became harder. The l_{stag} values associated with different hidden layer sizes indicate that the addition of more neurons, especially for shallower NNs, reduced oscillations, and made convergence easier (larger l_{stag} values corresponded to larger hidden layer sizes).

9.2.2 Iris

Figure 9.7 summarises the curvature information obtained for the different architectures and sampling settings considered for the Iris problem. Similarly to XOR, an increase in either the hidden layer size, or the number of hidden layers, yielded increased landscape flatness, corresponding to the findings of Sagun et al. [114]. The larger initialisation interval yielded stronger flatness than the smaller initialisation interval. The $[-1, 1]$ macro setting yielded the least amount of flatness, once again. The effect of the hidden layer sizes and the number of hidden layers on the modality of the NN loss surfaces is analysed below.

The effect of hidden neurons

Figure 9.8 shows the l-g clouds obtained using the $[-1, 1]$ macro walks for the single hidden layer NN architectures with a varied hidden layer size. The minimal architecture exhibited the split into higher and lower gradient clusters (see Figure 9.8a). As the

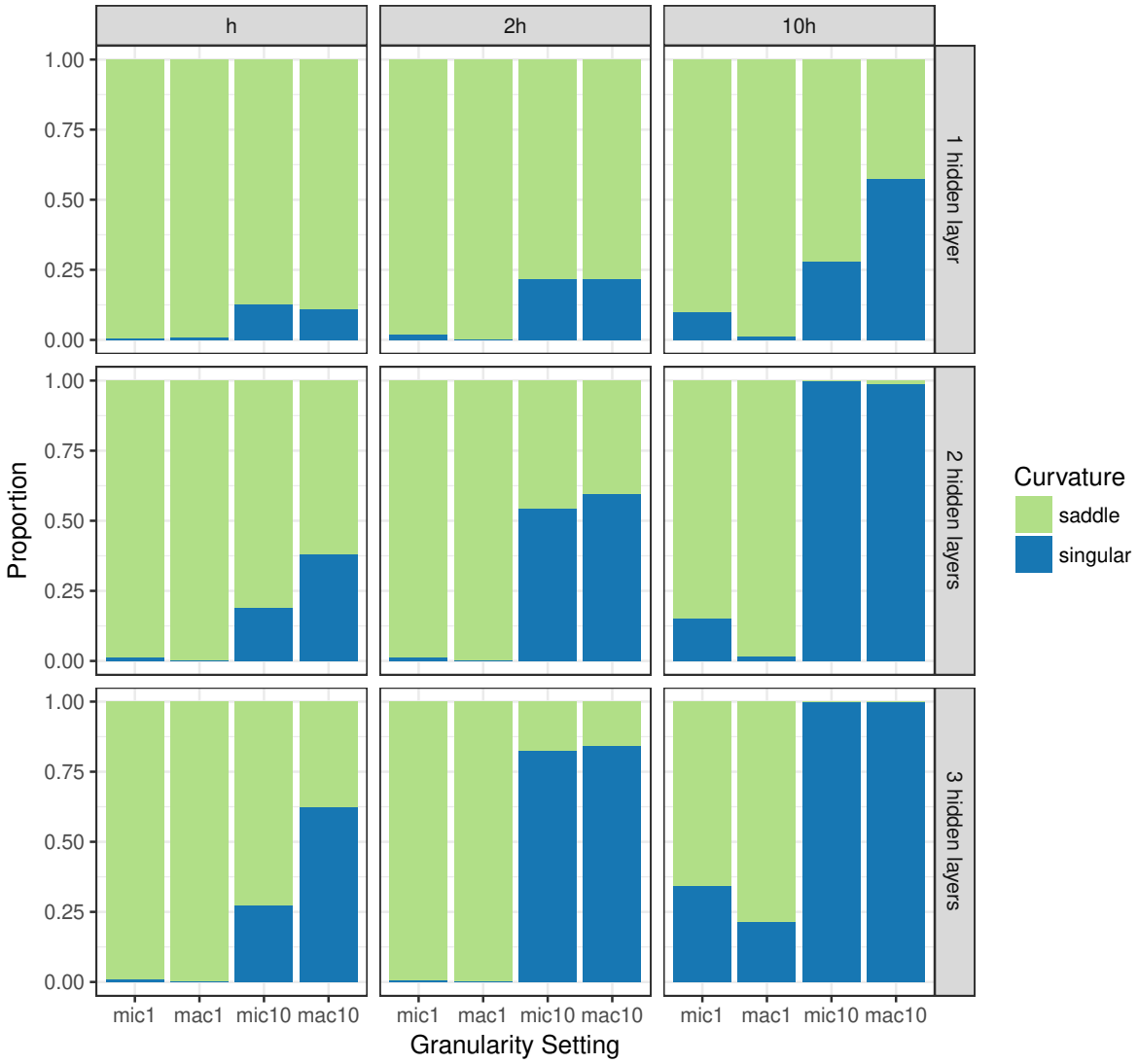
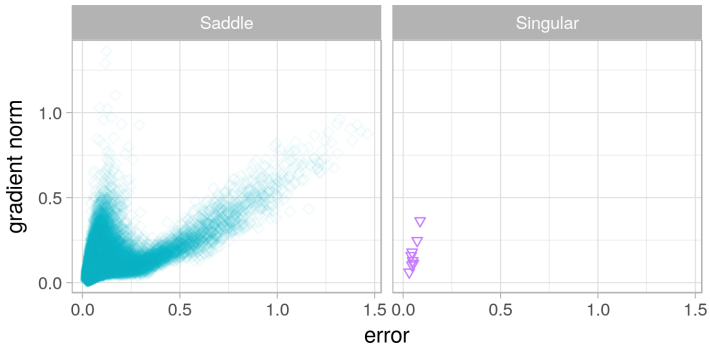
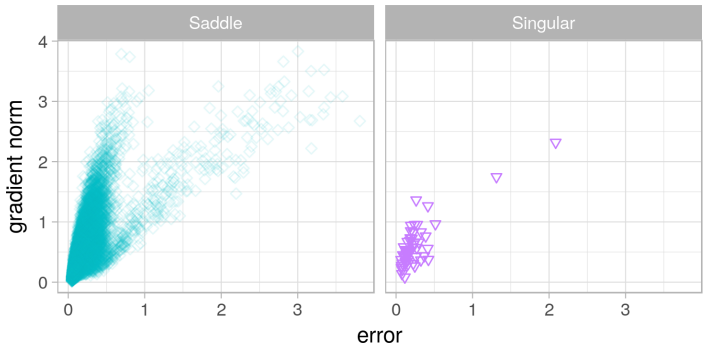


Figure 9.7: Histogram representation of the curvature information sampled by the gradient walks for the Iris problem for various NN architecture settings.

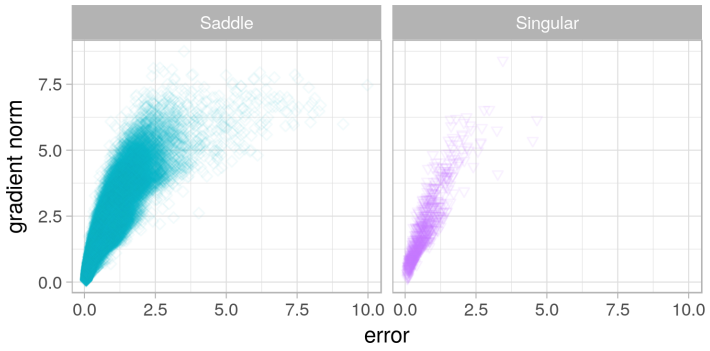
number of hidden neurons increased, the steep gradient cluster became more prominent (see Figure 9.8b). Increasing the size of the hidden layer by an order of magnitude caused the low gradient cluster to completely disappear (see Figure 9.8c). Thus, the total number of differently shaped attractors reduced with an increase of the hidden layer size. The l_{stag} values reported in Table 9.2 confirm that for the single hidden layer



(a) $h = 4$, macro, $[-1, 1]$



(b) $2 \times h = 8$, macro, $[-1, 1]$



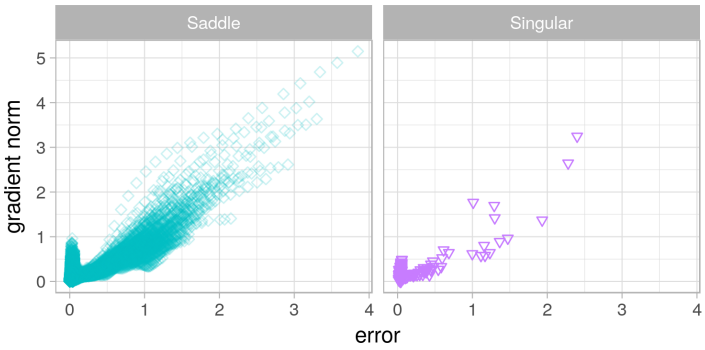
(c) $10 \times h = 40$, macro, $[-1, 1]$

Figure 9.8: L-g clouds for the macro gradient walks initialised in the $[-1, 1]$ range for the Iris problem for the various number of hidden neurons in a single hidden layer.

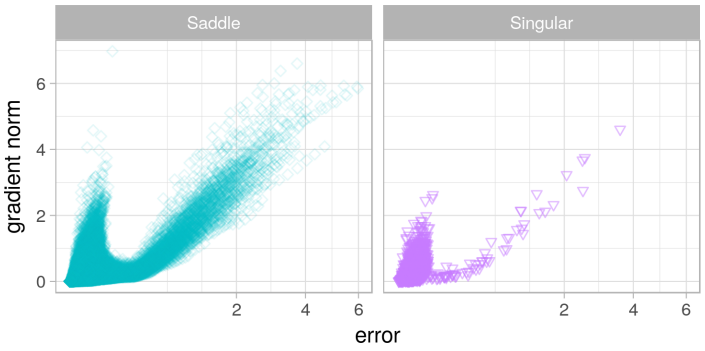
architectures, convergence to an attractor became more rapid as the hidden layer size increased.

Table 9.2: Basin of attraction estimates calculated for the Iris problem for the various NN architectures. Standard deviation is shown in parenthesis.

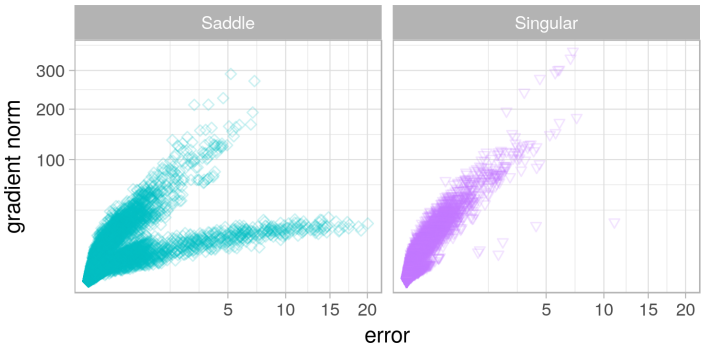
	$h = 4$		$2 \times h = 8$		$10 \times h = 40$		
	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}	
$[-1, 1]$,	1.0000	934.5486	1.0000	956.5746	1.0000	979.9737	1 hidden layer
micro	(0.0000)	(13.2580)	(0.0000)	(7.5175)	(0.0000)	(1.6723)	
$[-1, 1]$,	1.0057	84.3229	1.0075	84.4291	1.0805	61.7025	
macro	(0.0754)	(5.7889)	(0.0861)	(8.4820)	(0.3471)	(27.0304)	
$[-10, 10]$,	1.0000	967.6429	1.0000	971.3284	1.0062	977.9724	
micro	(0.0000)	(6.8482)	(0.0000)	(5.6407)	(0.0962)	(38.7116)	
$[-10, 10]$,	1.0514	78.5826	1.1269	70.7127	1.1039	65.0911	
macro	(0.3255)	(19.1828)	(0.3545)	(24.3757)	(0.3628)	(25.4561)	
$[-1, 1]$,	1.0000	942.1727	1.0000	969.5504	1.1151	931.5000	2 hidden layers
micro	(0.0000)	(10.6863)	(0.0000)	(5.1211)	(0.3993)	(163.3820)	
$[-1, 1]$,	1.0636	76.5030	1.1007	71.6553	1.1009	65.0349	
macro	(0.2789)	(17.0037)	(0.3240)	(21.2979)	(0.3363)	(19.3961)	
$[-10, 10]$,	1.0273	966.2295	1.0000	976.1195	1.3489	903.2885	
micro	(0.2847)	(73.6320)	(0.0000)	(4.7789)	(1.3698)	(217.5959)	
$[-10, 10]$,	1.1455	67.7030	1.1685	66.4667	1.0769	65.2900	
macro	(0.4008)	(22.8917)	(0.4539)	(22.5405)	(0.2985)	(17.2733)	
$[-1, 1]$,	1.0000	947.6133	1.0000	975.3128	2.0556	646.9148	3 hidden layers
micro	(0.0000)	(11.2504)	(0.0000)	(4.8822)	(1.2681)	(336.3288)	
$[-1, 1]$,	1.2200	59.2778	1.2180	57.0990	1.2824	45.9907	
macro	(0.4453)	(23.4992)	(0.5104)	(24.3973)	(0.5717)	(21.9762)	
$[-10, 10]$,	1.6067	853.5418	1.4837	827.4204	8.4442	88.6369	
micro	(1.9558)	(269.8470)	(1.1106)	(270.1939)	(2.9457)	(52.0628)	
$[-10, 10]$,	1.4667	50.5856	1.4416	49.3801	1.4466	44.7563	
macro	(0.7087)	(22.4714)	(0.6505)	(21.0175)	(0.6224)	(17.8009)	



(a) $h = 4$, micro, $[-1, 1]$



(b) $2 \times h = 8$, micro, $[-1, 1]$



(c) $10 \times h = 40$, micro, $[-1, 1]$

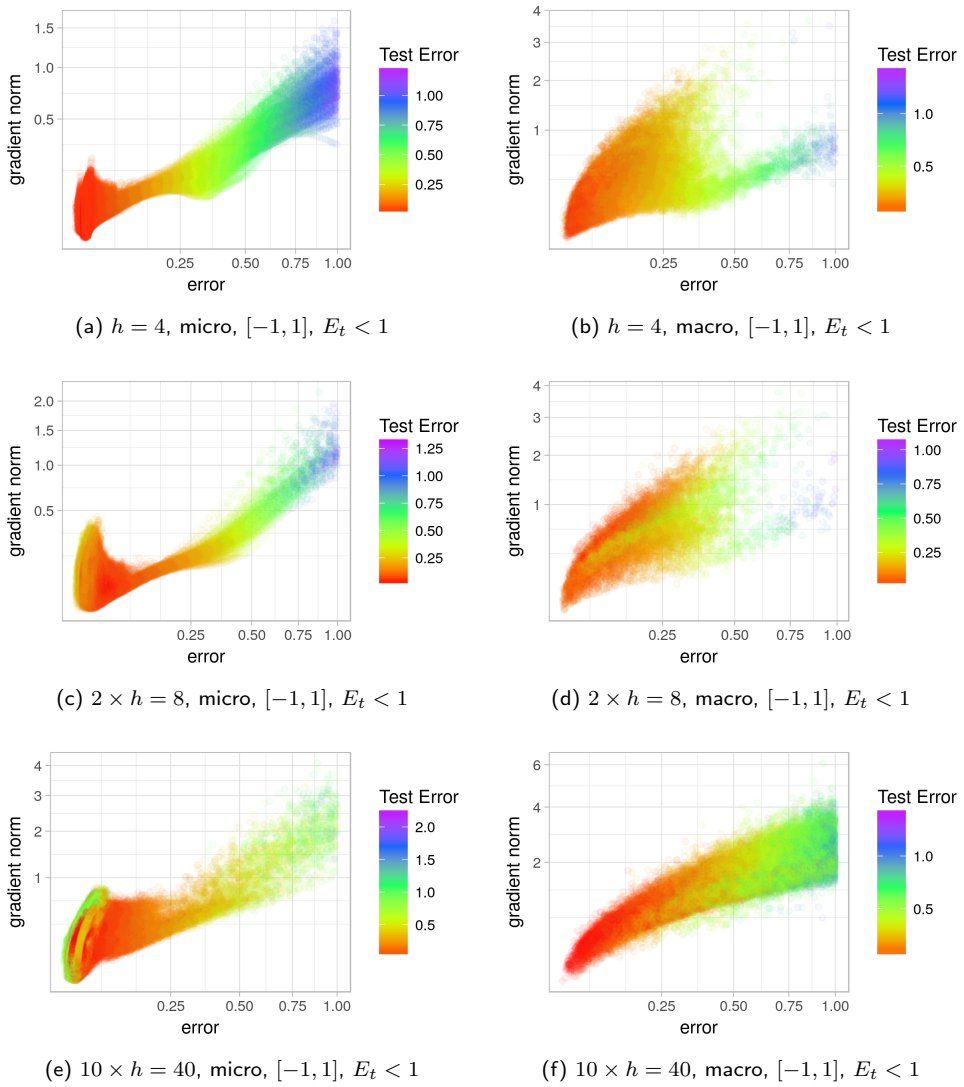
Figure 9.9: L-g clouds for the micro gradient walks initialised in the $[-1, 1]$ range for the Iris problem for the various number of hidden neurons in two consecutive hidden layers.

The same trend is illustrated in Figure 9.9, where the l-g clouds obtained by the $[-1, 1]$ micro walks for the 2-hidden layer NN architectures are shown. The steep gradient cluster

became more and more prominent as the hidden layer size increased, and the low-error attractors associated with the shallow gradient cluster became smoother. An increase in the hidden layer size added unnecessary parameters to the architecture, thus the number of solutions with a proportion of the disabled neurons increased for larger hidden layers.

Thus, the addition of more neurons simplified the loss surface for the Iris problem by creating a simpler attractor. The increased amount of flatness associated with the steep gradient clusters confirms the hypothesis that embedded minima corresponding to smaller architectures were discovered.

Table B.16 in Appendix B shows the average classification accuracy obtained for the training and generalisation sets for the various granularity settings and architectures. According to Table B.16, an increase in the hidden layer size improved the training accuracy, and yielded increased overfitting ($C_t > C_g$) for most micro granularity settings. Excessive NN architectures were argued to be prone to overfitting in the past [146], and the results presented in Table B.16 confirm this hypothesis. However, the tendency to overfit was clearly dependent on the chosen step size: For the $[-1, 1]$ macro setting, $C_g > C_t$ was obtained in multiple cases, thus indicating no overfitting. Figure 9.10 shows the l-g clouds colourised according to the E_g values for the single hidden layer NN architectures of varied hidden layer size. Deterioration of the generalisation quality around the global minimum attractor associated with the increased hidden layer size is evident for the micro walks. However, as indicated by the l_{stag} values in Table 9.2, larger hidden layers caused quicker convergence to the global minimum attractor, thus the evident overfitting can be attributed to the longer exploitation of the global attractor rather than the structure of the loss surface. Larger step sizes, as shown in Figure 9.10 and Table B.16, yielded better generalisation performance on the larger architectures compared to the smaller architectures. Counter-intuitively, the NN architectures with more neurons in the hidden layer required fewer gradient-guided steps than the NN architectures with fewer neurons. This observation confirms the previously made hypothesis that the addition of neurons to a hidden layer simplifies the NN error landscape, and makes global minima easier to find.



(a) $h = 4$, micro, $[-1, 1]$, $E_t < 1$

(b) $h = 4$, macro, $[-1, 1]$, $E_t < 1$

(c) $2 \times h = 8$, micro, $[-1, 1]$, $E_t < 1$

(d) $2 \times h = 8$, macro, $[-1, 1]$, $E_t < 1$

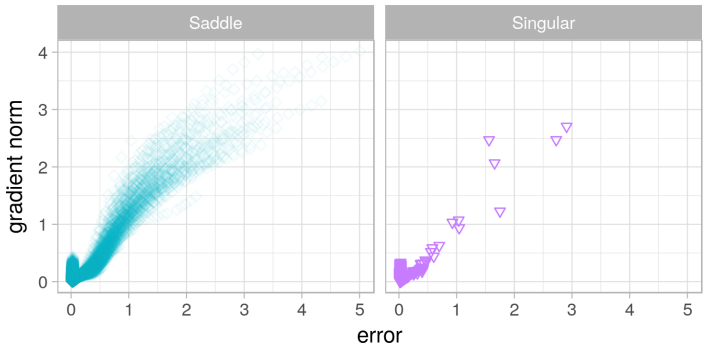
(e) $10 \times h = 40$, micro, $[-1, 1]$, $E_t < 1$

(f) $10 \times h = 40$, macro, $[-1, 1]$, $E_t < 1$

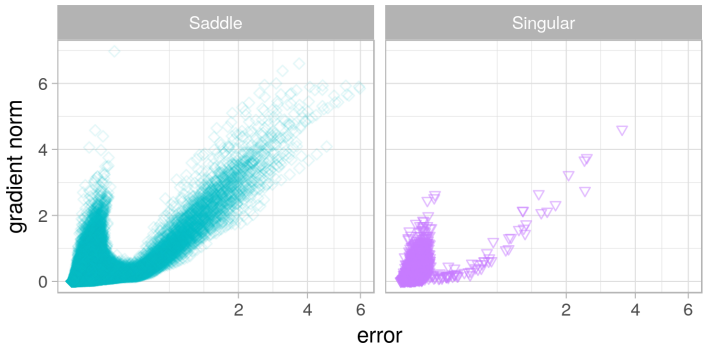
Figure 9.10: L-g clouds colourised according to the corresponding E_g values, obtained for single hidden layer NN architectures of varied hidden layer size for the Iris problem.

The effect of hidden layers

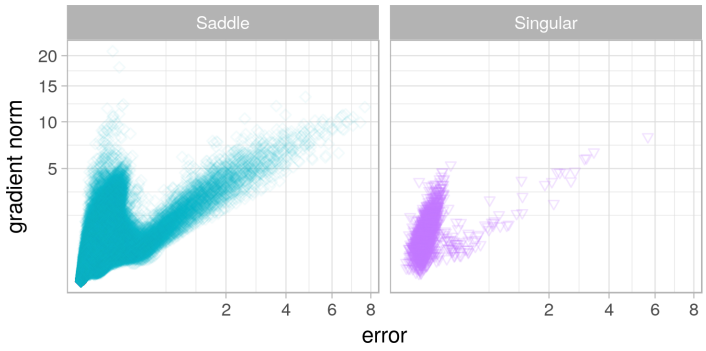
Figure 9.11 shows the l-g clouds obtained using the $[-1, 1]$ micro walks for the NN architectures with a varied number of hidden layers, with exactly eight neurons per hidden layer. According to Figure 9.11, an increase in the number of hidden layers yielded a heavier steep gradient cluster. The steep gradient cluster overlapped with



(a) 1 hidden layer, micro, $[-1, 1]$



(b) 2 hidden layers, micro, $[-1, 1]$



(c) 3 hidden layers, micro, $[-1, 1]$

Figure 9.11: L-g clouds for the micro gradient walks initialised in the $[-1, 1]$ range for the Iris problem for the various number of hidden layers, with $h = 8$ for each layer.

indefinite, i.e. flat points, once again attributed to the embedded minima. Figure 9.11c shows that the shape of the attractors did not change otherwise, and that the addition

of more hidden layers did not have the smoothing effect observed for the addition of more neurons (Figure 9.9). Classification results in Table B.16 confirm that the training accuracy C_t for a particular fixed hidden layer size either did not increase, or decreased as more hidden layers were added for most scenarios considered. The n_{stag} and l_{stag} values reported in Table 9.2 show that convergence to a single attractor took place in most cases, and that the addition of more layers marginally increased the speed of convergence (see micro $[-1, 1]$ setting). However, the deteriorating quality of solutions illustrated in Table B.16 confirms that the problem did not become easier to optimise. Thus, additional layers did not simplify the error landscape for the Iris problem.

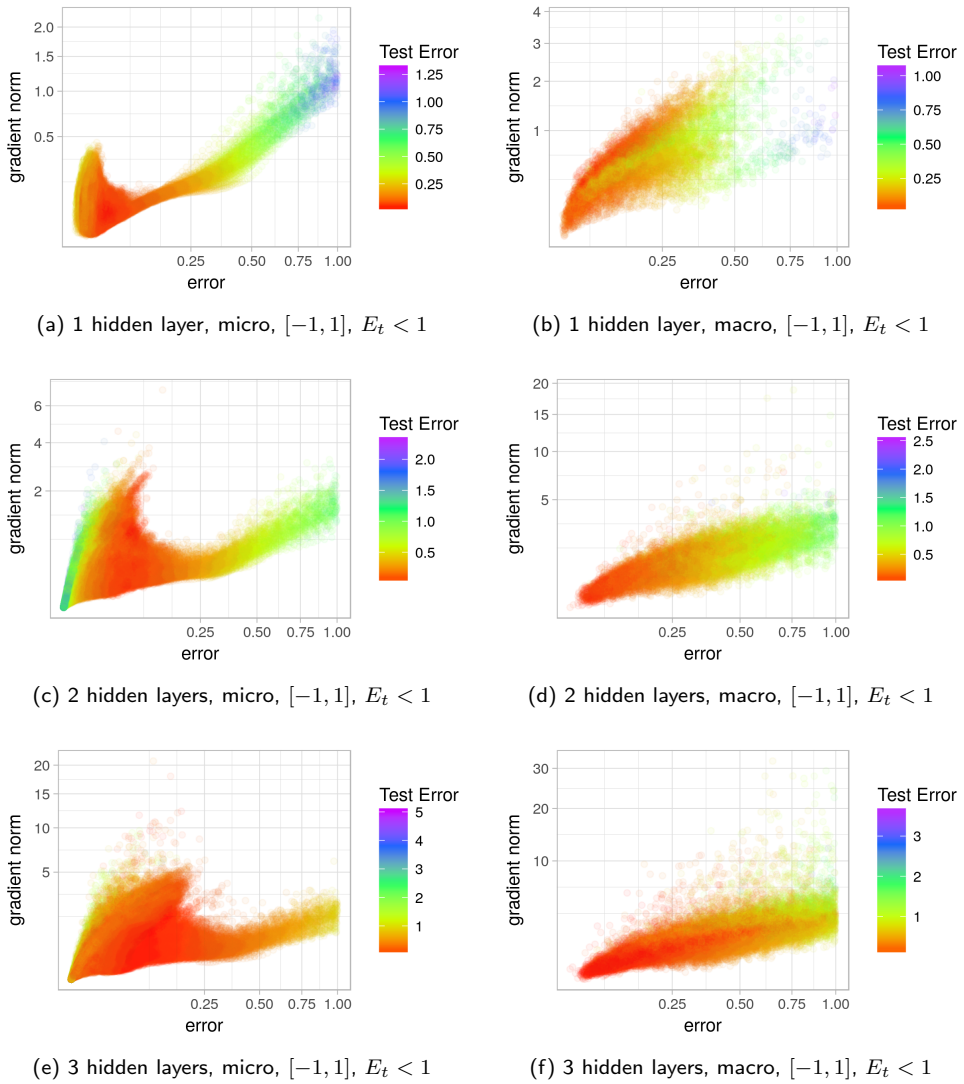
Table B.16 also shows that addition of more layers did not have a consistent effect on the generalisation ability of the discovered solutions. Figure 9.12 shows that the range of E_g values corresponding to the points with $E_t \in [0, 1]$ increased as more hidden layers were added. The same observation applies to the addition of hidden neurons (Figure 9.10), but in case of the hidden layers, the range of E_g values increased more rapidly. Higher deviation in the E_g values confirms that addition of hidden layers to a NN architecture increased the likelihood of discovering solutions of poor generalisation quality, and thus made the problem more difficult to optimise.

9.2.3 Diabetes

Hessian matrices were not computed for problems with dimensionality $m > 3000$ due to computational constraints. The curvature information for the different architectures with $m < 3000$ for the Diabetes problem is presented in Figure 9.13. Figure 9.13 shows that the search space was dominated by saddle curvature. An increase in the hidden layer size did not have a strong effect on the curvature characteristics. An increase in the number of hidden layers, on the other hand, yielded an increased amount of flatness, especially further away from the origin (as sampled by the gradient walks initialised in the $[-10, 10]$ interval). For deeper NN architectures, neurons in the later layers can reduce the influence of the neurons in the earlier layers, thus causing flatness, i.e. non-contributing weights.

Table B.17 in Appendix B reports the average classification accuracy obtained at the last step of the gradient walks for the various architectures. The results in Table B.17

show that an increase in the number of hidden layers yielded a stronger improvement in the C_t values than an increase in the number of hidden neurons. The largest architecture (3 hidden layers with $10 \times h = 80$ neurons per layer) yielded the highest $C_t = 0.9946$, while the smallest (minimal) architecture yielded the highest $C_g = 0.7948$. However, according to Table B.17, an increase in either the hidden layer size or the number of



(a) 1 hidden layer, micro, $[-1, 1]$, $E_t < 1$

(b) 1 hidden layer, macro, $[-1, 1]$, $E_t < 1$

(c) 2 hidden layers, micro, $[-1, 1]$, $E_t < 1$

(d) 2 hidden layers, macro, $[-1, 1]$, $E_t < 1$

(e) 3 hidden layers, micro, $[-1, 1]$, $E_t < 1$

(f) 3 hidden layers, macro, $[-1, 1]$, $E_t < 1$

Figure 9.12: L-g clouds colourised according to the corresponding E_g values, obtained for the various number of hidden layers, with $h = 8$ for each layer on the Iris problem.

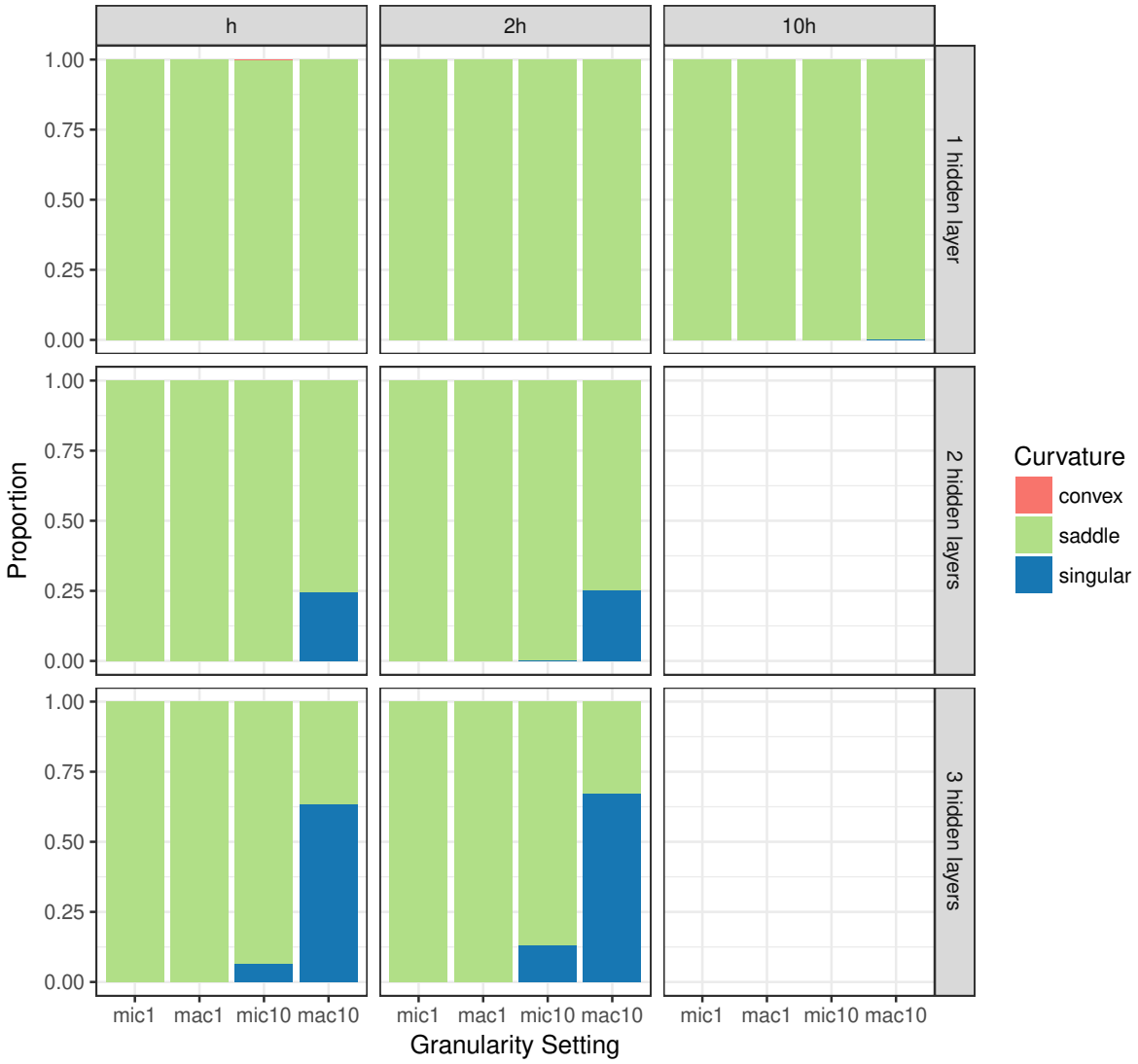
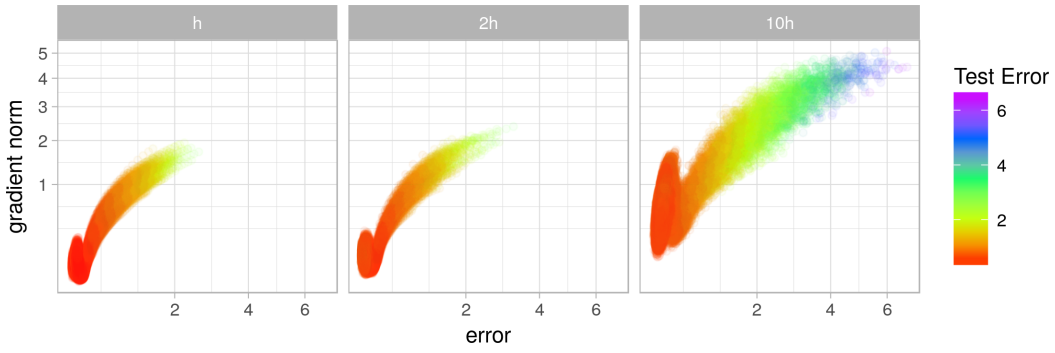
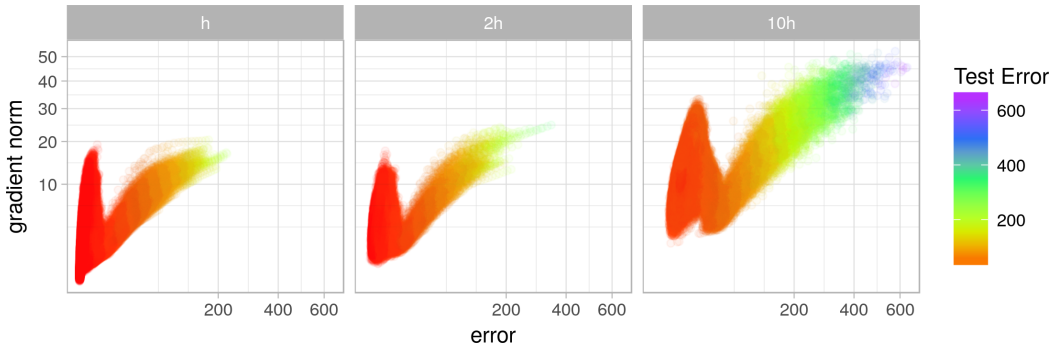


Figure 9.13: Histogram representation of the curvature information sampled by the gradient walks for the Diabetes problem for various NN architectures.

hidden layers did not have a strong negative effect on generalisation. Thus, even though larger architectures exhibited some overfitting, they also exhibited higher searchability.



(a) $[-1, 1]$, micro

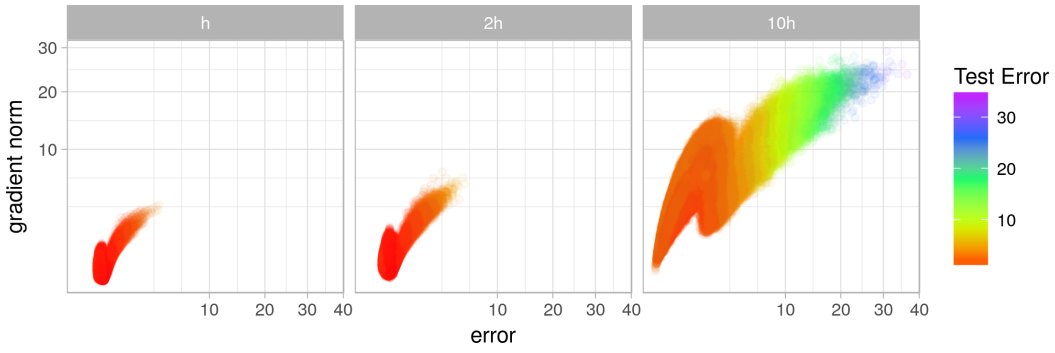


(b) $[-10, 10]$, micro

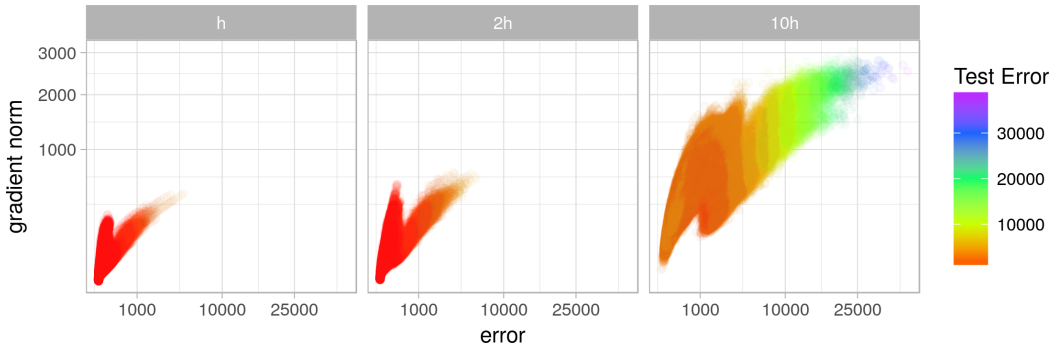
Figure 9.14: L-g clouds for the micro gradient walks initialised in the $[-1, 1]$ and $[-10, 10]$ ranges for the Diabetes problem for the various number of hidden neurons in a single hidden layer.

The effect of hidden neurons

Figure 9.14 shows the l-g clouds obtained by the micro walks for the various hidden layer sizes on a single hidden layer architecture, colourised according to E_g values. Figure 9.14 illustrates that an increase in the hidden layer size increased the range of error values and gradient magnitudes produced. The l-g shapes in Figure 9.14 indicate that a single global attractor was discovered by the smaller architectures. The attractor widened as the number of hidden neurons increased, and visibly split into two attractors for the largest ($10 \times h = 80$) number of neurons. The split into two attractors correlates with the split into two clusters, namely steep gradients around the global minimum, and shallow



(a) $[-1, 1]$, micro



(b) $[-10, 10]$, micro

Figure 9.15: L-g clouds for the micro gradient walks initialised in the $[-1, 1]$ and $[-10, 10]$ ranges for the Diabetes problem for the various number of hidden neurons in two consecutive hidden layers.

gradients further away from the minimum. The clusters are attributed to the wide and narrow valleys exhibited by NN loss surfaces. Thus, the split into two attractors indicates a stronger separation between the wide and narrow valleys exhibited by the loss surfaces of the larger architectures. The n_{stag} values reported in Table 9.3 show that the gradient walks generally did not become stuck more than once for the single-layer architectures. Thus, each walk converged to either a narrow or a wide valley.

Figure 9.15 shows the l-g clouds obtained by the micro walks for the various hidden layer sizes of a 2-hidden layer architecture. An increase in the hidden layer size clearly exaggerated the split into two attractors for the 2-layer architectures. The n_{stag} values in Table 9.3 confirm that two stagnation regions were sometimes detected for larger hidden

Table 9.3: Basin of attraction estimates calculated for the Diabetes problem for the various NN architectures. Standard deviation is shown in parenthesis.

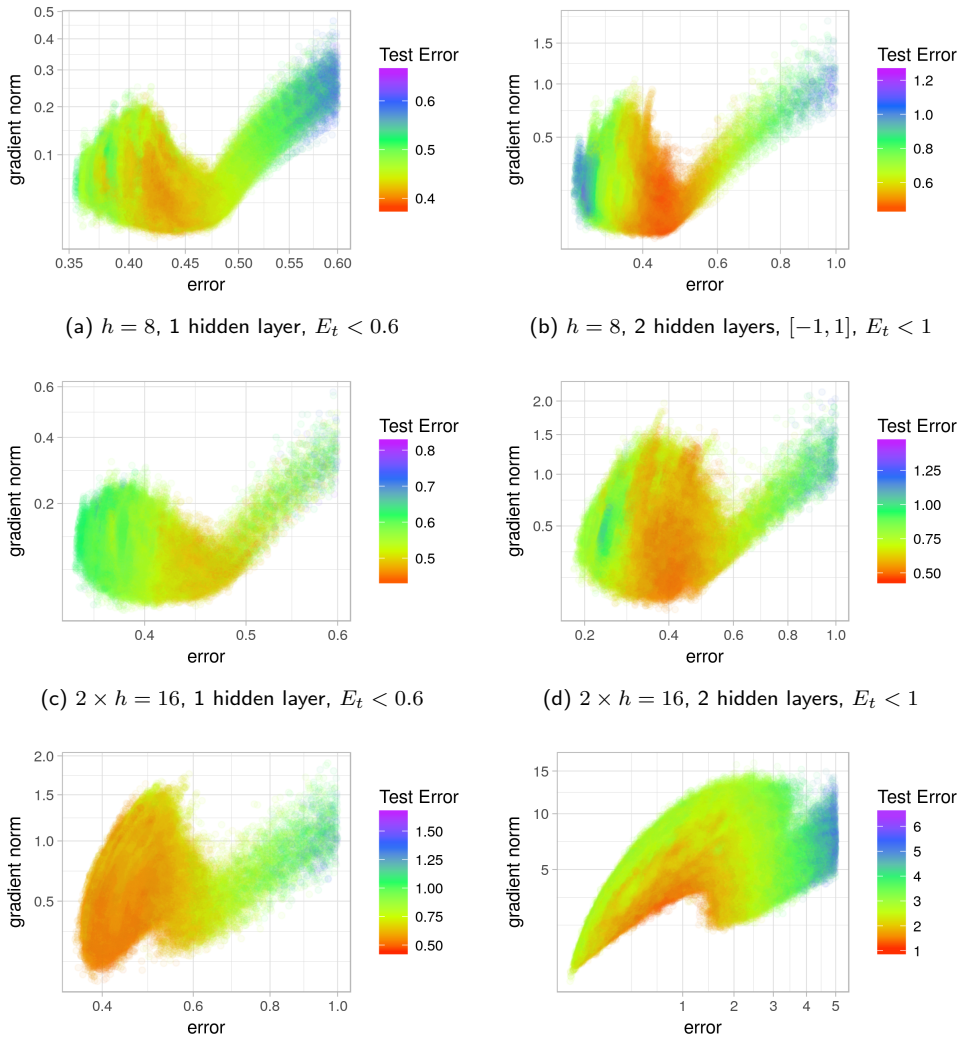
	$h = 8$		$2 \times h = 16$		$10 \times h = 80$		
	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}	
$[-1, 1]$,	1.0000	964.2988	1.0000	972.3447	1.0000	981.4642	1 hidden layer
micro	(0.0000)	(5.1385)	(0.0000)	(3.0608)	(0.0000)	(1.3061)	
$[-1, 1]$,	1.0062	85.9068	1.0435	79.7469	1.1067	42.5487	
macro	(0.0928)	(6.9681)	(0.2324)	(16.9730)	(0.5173)	(30.0841)	
$[-10, 10]$,	1.0000	961.7667	1.0000	969.4596	1.0000	978.6105	
micro	(0.0000)	(5.1574)	(0.0000)	(3.3760)	(0.0000)	(2.8212)	
$[-10, 10]$,	1.0679	77.8084	1.0932	68.8349	1.0662	39.2125	
macro	(0.2705)	(18.3192)	(0.3663)	(25.8064)	(0.5843)	(31.2463)	
$[-1, 1]$,	1.0000	972.1667	1.0000	979.3857	1.0056	974.2042	2 hidden layers
micro	(0.0000)	(4.2551)	(0.0000)	(1.6652)	(0.1003)	(37.9072)	
$[-1, 1]$,	1.0196	81.6520	1.0104	78.3678	1.4435	31.4245	
macro	(0.1386)	(11.8660)	(0.1014)	(10.6638)	(0.7158)	(20.5202)	
$[-10, 10]$,	1.0000	972.9641	1.0000	977.6617	1.0240	962.1185	
micro	(0.0000)	(3.1945)	(0.0000)	(2.8329)	(0.2440)	(74.8544)	
$[-10, 10]$,	1.0098	80.9477	1.0254	72.9863	1.1283	62.2244	
macro	(0.0985)	(8.9638)	(0.1905)	(14.0427)	(0.3961)	(19.1883)	
$[-1, 1]$,	1.0000	975.7356	1.0000	980.9021	1.0025	960.3480	3 hidden layers
micro	(0.0000)	(4.2915)	(0.0000)	(2.0474)	(0.0514)	(33.8877)	
$[-1, 1]$,	1.0444	76.8830	1.0170	78.3746	1.2896	41.0136	
macro	0.2266)	(16.2317)	(0.1449)	(10.1321)	(0.5991)	(22.7162)	
$[-10, 10]$,	1.0244	968.4817	1.0015	978.4544	1.2883	837.0587	
micro	(0.2248)	(72.4814)	(0.0393)	(20.1065)	(0.6836)	(227.2017)	
$[-10, 10]$,	1.2135	65.6555	1.1320	70.0759	1.1522	64.6153	
macro	(0.4528)	(23.0145)	(0.3716)	(18.9212)	(0.4213)	(20.9157)	

layer sizes, indicating the possibility of transition between the attractors. To further study the generalisation performance of the discovered attractors, Figure 9.16 shows the l-g clouds for the subset of points around the global minimum. Figure 9.16 shows that exploitation of the global minimum attractor had a detrimental effect on generalisation, but the detrimental effect weakened as more hidden neurons were added. Specifically, for the 2-hidden layer architecture with $10 \times h = 80$ hidden neurons per layer, the global minimum not only became easier to exploit, but also exhibited a band of good quality solutions at the bottom of the attractor leading to the global minimum. Good quality solutions were also discovered at the bottom of the secondary attractor introduced by additional hidden neurons, indicating that exploitation of the global minimum may be altogether unnecessary. This observation once again correlates with previously made conclusions that wide valleys are likely to exhibit better generalisation properties [21]. Overall, even though the range of errors increased with an increase in the hidden layer size, convergence to the global optimum became easier for the larger hidden layers, especially with more than one hidden layer.

The classification results in Table B.17 indicate that increases in the number of hidden neurons made the quality of the final solution less sensitive to the step size and initialisation interval. For the $[-10, 10]$ initialisation interval, the classification results improved as more neurons were added. Thus, even though low-dimensional NN error landscapes yielded better results in some cases, the higher-dimensional landscapes exhibited better global structure properties. The l-g clouds in Figure 9.14 also indicate that the attractors were wider for larger hidden layers. The l_{stag} results in Table 9.3 confirm that the micro walks discovered wider attraction basins for the larger hidden layers.

The effect of hidden layers

According to the classification results in Table B.17, an increase in the number of hidden layers yielded consistently better C_t results for the $[-1, 1]$ micro walks. In general, micro walks for the $2 \times h$ and $10 \times h$ architectures yielded better results than the macro walks, regardless of the initialisation interval. This observation highlights the interplay between the initialisation range and the step size, and the importance of optimising these two parameters in conjunction.



(a) $h = 8$, 1 hidden layer, $E_t < 0.6$

(b) $h = 8$, 2 hidden layers, $[-1, 1]$, $E_t < 1$

(c) $2 \times h = 16$, 1 hidden layer, $E_t < 0.6$

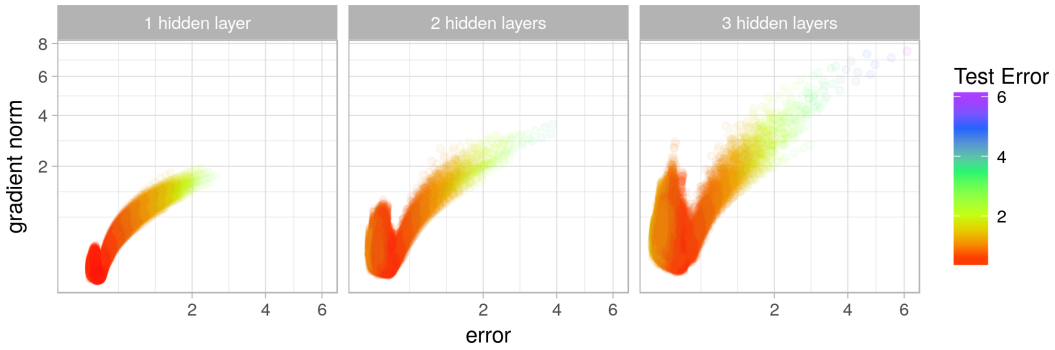
(d) $2 \times h = 16$, 2 hidden layers, $E_t < 1$

(e) $10 \times h = 80$, 1 hidden layer, $E_t < 1$

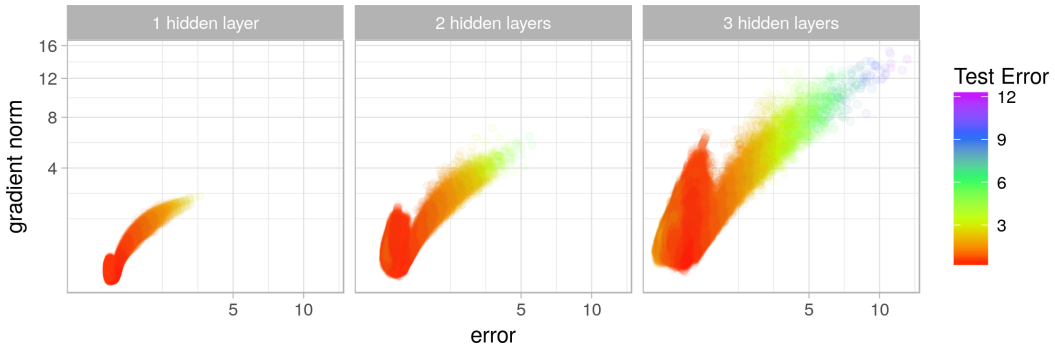
(f) $10 \times h = 80$, 2 hidden layers, $E_t < 5$

Figure 9.16: L-g clouds colourised according to the corresponding E_g values, obtained by the $[-1, 1]$ micro walks for the Diabetes problem.

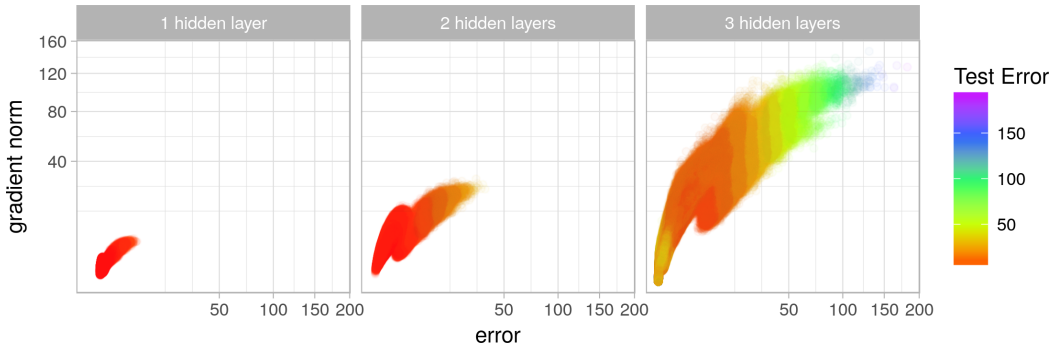
Figure 9.17 shows the l-g clouds obtained by the $[-1, 1]$ micro walks for the NN architectures with the various number of hidden layers. The l-g clouds are shown with a shared scale to emphasise that the shape of the attractors did not change with the addition of more layers, but rather widened in terms of the error and gradient magnitude ranges. Figure 9.17 illustrates that exploitation of the global minimum attractor was



(a) $h = 8$



(b) $2 \times h = 16$



(c) $10 \times h = 80$

Figure 9.17: L-g clouds for the micro gradient walks initialised in the $[-1, 1]$ range for the Diabetes problem for the various number of hidden layers.

not only performed successfully for all architectures, but improved as more hidden layers were added (lower E_t values were obtained). Thus, addition of hidden layers improved

the overall searchability of the landscape. The split into two clusters was observed for most scenarios. Figure 9.17 shows that the separation into two clusters was smoothed by the addition of more neurons per layer, but remained the same as more hidden layers were added for a fixed h.

As far as the generalisation performance is concerned, exploitation of the global minima consistently yielded a deterioration in the generalisation performance. Thus, improved exploitation due to higher searchability of many-layered architectures often resulted in poor generalisation (see Figure 9.16). Figure 9.17c shows that the E_g values obtained around the global minimum attractor were very poor for the 3-hidden layer architecture with $10 \times h$ hidden neurons. However, the same figure illustrates that the E_g performance at the secondary attractor, introduced by the increased hidden layer size, yielded better performance. Thus, convergence to a local attractor associated with wider valleys would have been preferable for the Diabetes problem.

To summarise, the addition of hidden neurons had a strong effect on the shape and number of attractors, and emphasised the split into wide and narrow valleys. The addition of hidden layers did not change the shape or the number of the attractors, but increased the width and steepness of the attractors, thereby making the loss surfaces easier to exploit.

9.2.4 Glass

Figure 9.18 summarises the curvature information obtained for the various NN architectures considered. Similar to all previous problems considered, flatness increased with an increase in dimensionality. The addition of hidden layers yielded a faster increase in flatness than the addition of hidden neurons to a layer. Saddle curvature was prevalent for the $[-1, 1]$ initialisation interval.

The effect of hidden neurons

Table B.18 in Appendix B summarises the classification results obtained at the last step of the gradient walks. According to Table B.18, an increase in the hidden layer size generally improved the C_t values. An interesting trend is observed for the C_g values: Overfitting (deteriorating C_g associated with improved C_t) is sometimes manifested for

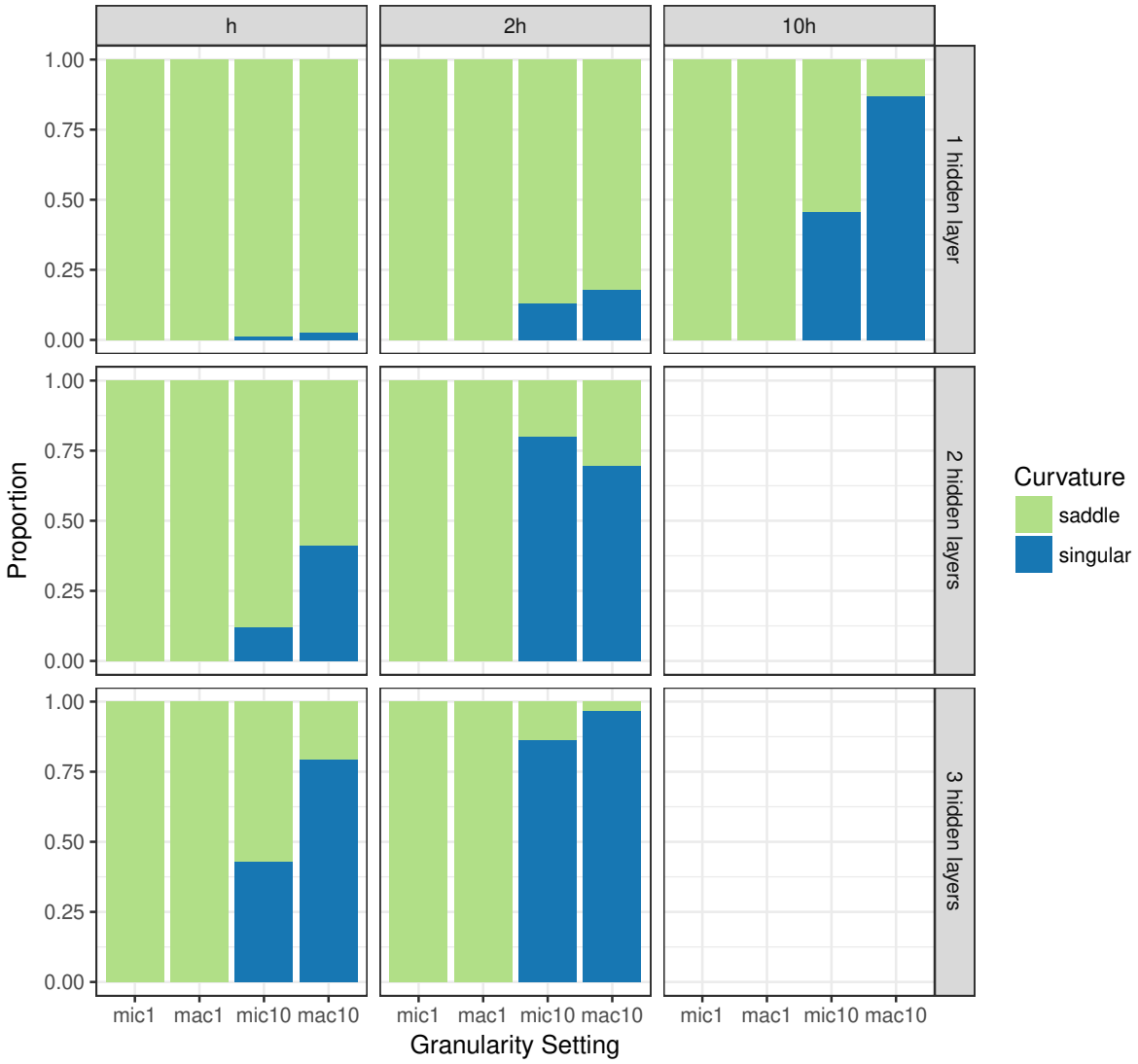
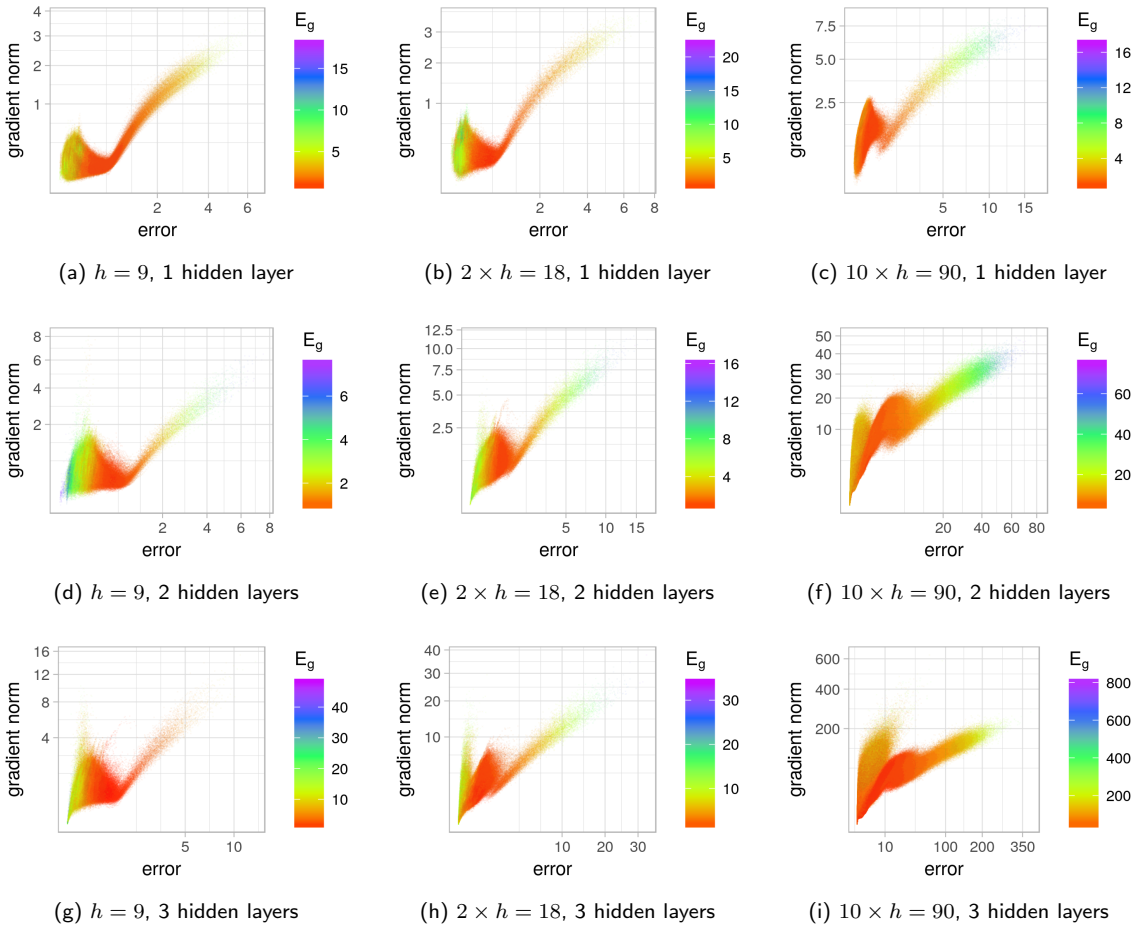


Figure 9.18: Histogram representation of the curvature information sampled by the gradient walks for the Glass problem for various NN architecture settings.

$2 \times h$, but the C_g values increase again as the hidden layer size is further increased to $10 \times h$. Thus, a somewhat larger than minimal hidden layer promoted overfitting in some cases, but a more significant increase in h generally had a positive effect on the generalisation performance for the Glass problem. Overall, improved generalisation performance was associated with an increased hidden layer size, regardless of the number

of the hidden layers. Out of all the architecture settings considered, the highest C_g value was obtained for the single layer architecture with $10 \times h$ hidden neurons. The l_{stag} values in Table 9.4 indicate that an increase in the hidden layer size yielded faster convergence to an attractor.



(a) $h = 9$, 1 hidden layer (b) $2 \times h = 18$, 1 hidden layer (c) $10 \times h = 90$, 1 hidden layer

(d) $h = 9$, 2 hidden layers (e) $2 \times h = 18$, 2 hidden layers (f) $10 \times h = 90$, 2 hidden layers

(g) $h = 9$, 3 hidden layers (h) $2 \times h = 18$, 3 hidden layers (i) $10 \times h = 90$, 3 hidden layers

Figure 9.19: L-g clouds colourised according to the corresponding E_g values, obtained by the $[-1, 1]$ micro walks for the Glass problem for the various NN architectures.

Figure 9.19 shows the l-g clouds obtained for the $[-1, 1]$ micro walks for the various NN architectures. For the single hidden layer architecture, an increase from h to $2 \times h$ did not change the shape of the l-g cloud, but yielded stronger overfitting around the global minimum. An increase of the hidden layer size to $10 \times h$ yielded l-g cloud

Table 9.4: Basin of attraction estimates calculated for the Glass problem for the various NN architectures. Standard deviation is shown in parenthesis.

	$h = 9$		$2 \times h = 18$		$10 \times h = 90$		
	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}	
$[-1, 1]$,	1.0000	955.9787	1.0000	966.9201	1.0000	979.9755	1 hidden layer
micro	(0.0000)	(4.2559)	(0.0000)	(2.2312)	(0.0000)	(0.9297)	
$[-1, 1]$,	1.0000	86.8287	1.0000	86.9371	1.0000	81.3340	
macro	(0.0000)	(0.7009)	(0.0000)	(1.2980)	(0.0000)	(6.3706)	
$[-10, 10]$,	1.0000	963.5053	1.0000	970.1395	1.0000	979.1573	
micro	(0.0000)	(3.8395)	(0.0000)	(2.6690)	(0.0000)	(1.1055)	
$[-10, 10]$,	1.0000	86.9647	1.0000	86.6752	1.0064	79.9905	
macro	(0.0000)	(1.7961)	(0.0000)	(2.8173)	(0.0799)	(9.3428)	
$[-1, 1]$,	1.0000	966.7250	1.0000	977.0220	1.0000	982.3361	2 hidden layers
micro	(0.0000)	(3.5741)	(0.0000)	(1.3398)	(0.0000)	(0.9954)	
$[-1, 1]$,	1.0000	85.6771	1.0000	84.5456	1.0008	75.3811	
macro	(0.0000)	(3.3176)	(0.0000)	(3.4706)	(0.0279)	(5.5302)	
$[-10, 10]$,	1.0000	971.9645	1.0000	976.3668	1.0393	956.2573	
micro	(0.0000)	(2.6558)	(0.0000)	(1.6371)	(0.2300)	(96.0468)	
$[-10, 10]$,	1.0063	84.9705	1.0088	79.5937	1.0024	73.9645	
macro	(0.0792)	(6.1181)	(0.0937)	(8.1220)	(0.0493)	(5.9678)	
$[-1, 1]$,	1.0000	973.6197	1.0000	979.9652	1.0001	980.0378	3 hidden layers
micro	(0.0000)	(3.0230)	(0.0000)	(1.1209)	(0.0092)	(5.9478)	
$[-1, 1]$,	1.0045	82.6591	1.0194	78.9994	1.3958	34.4549	
macro	(0.0673)	(7.6183)	(0.1452)	(10.6144)	(0.5831)	(18.2147)	
$[-10, 10]$,	1.0000	976.1552	1.0400	957.1000	3.1031	463.0445	
micro	(0.0000)	(2.6095)	(0.1960)	(98.5655)	(1.9866)	(314.7356)	
$[-10, 10]$,	1.2601	60.1420	1.2178	64.5984	1.3660	40.0568	
macro	(0.4931)	(23.2638)	(0.4683)	(22.2337)	(0.5609)	(17.8004)	

shapes similar to the Diabetes problem: Two connected attractors were observed. The $10 \times h$ architecture exhibited the least overfitting around the global minimum for the single hidden layer architecture. In general, an increase in the hidden layer size visibly widened the attraction basin, and increased the steepness of the gradients. A single major attractor was observed for most settings. The n_{stag} results in Table 9.4 confirm that a single attractor was detected by the walks for the majority of the settings. The cluster structure appears smoother and more connected for larger hidden layer sizes. The split into two clusters manifested across all scenarios, and inferior generalisation performance was generally associated with the steeper gradient cluster. As before, this behaviour is attributed to the poor generalisation properties of the minima discovered in narrow valleys [21].

The effect of hidden layers

The classification results in Table B.18 indicate that, for the $[-1, 1]$ micro walks, the C_t values generally increased as more hidden layers were added to the architectures, and the corresponding C_g values decreased. Thus, architectures with a higher number of hidden layers were more prone to overfitting. For the macro walks, both C_t and C_g decreased as more hidden layers were added. Thus, deeper architectures were more searchable around the origin when probed with smaller steps, but less searchable otherwise.

Figure 9.19 shows that an increase in the number of hidden layers exaggerated the steep gradient cluster for all hidden layer sizes. Thus, the addition of hidden layers increased the separation into wide and narrow valleys. Additional smaller clusters were also detected for the increased number of hidden layers, indicating that some additional landscape structures were introduced by the deeper architectures.

9.2.5 Cancer

Figure 9.20 summarises the curvature information obtained for the Cancer problem for the various number of hidden layers and hidden layer sizes. Once again, an increase in dimensionality yielded an increase in flatness. More flatness was observed further away from the origin. The $[-1, 1]$ macro setting yielded the least amount of flatness across all

architectures, indicating that the perception of the landscape is strongly dependent on the step size.

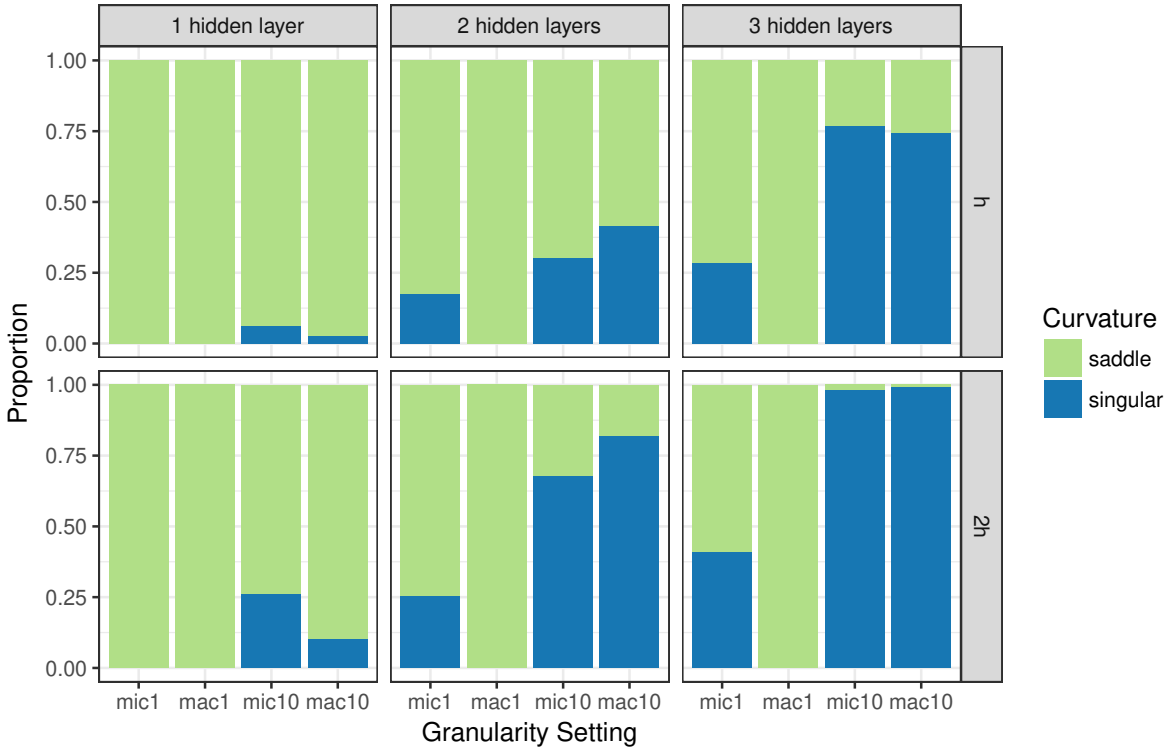
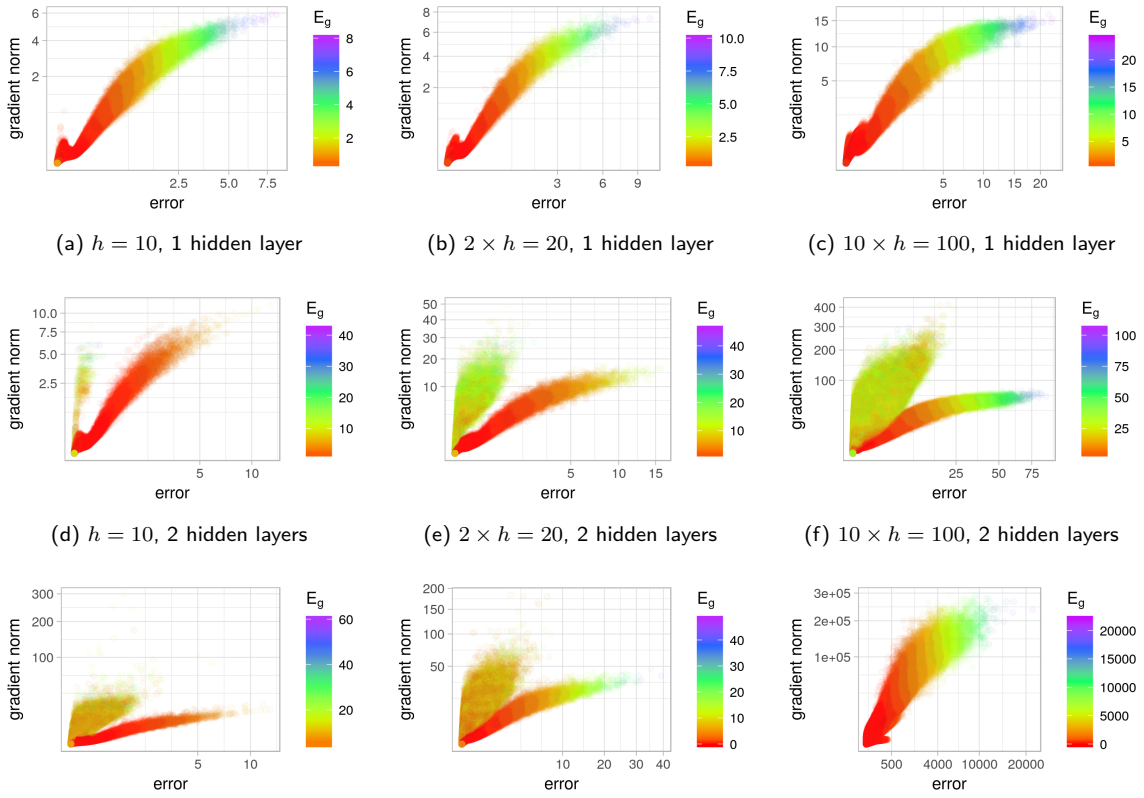


Figure 9.20: Histogram representation of the curvature information sampled by the gradient walks for the Cancer problem for various NN architectures.

The effect of hidden neurons

Table B.19 in Appendix B shows that most architectures obtained $C_t > 90\%$ for most granularity settings considered. For 1-hidden layer architectures, a 100% training accuracy was obtained by the $[-1, 1]$ micro walks for any number of hidden neurons, indicating that the landscape was highly searchable. The C_g values improved as more hidden neurons were added for the same granularity setting. Overall, the C_t and C_g values were not strongly affected by the number of hidden neurons. The Cancer problem is an easy classification problem, and adding more neurons to the hidden layers did not make this

problem harder.



(a) $h = 10$, 1 hidden layer (b) $2 \times h = 20$, 1 hidden layer (c) $10 \times h = 100$, 1 hidden layer

(d) $h = 10$, 2 hidden layers (e) $2 \times h = 20$, 2 hidden layers (f) $10 \times h = 100$, 2 hidden layers

(g) $h = 10$, 3 hidden layers (h) $2 \times h = 20$, 3 hidden layers (i) $10 \times h = 100$, 3 hidden layers

Figure 9.21: L-g clouds colourised according to the corresponding E_g values, obtained by the $[-1, 1]$ micro walks for the Cancer problem for the various NN architectures.

Figure 9.21 shows the l-g clouds obtained by the $[-1, 1]$ micro walks for the various architectures. A single stationary global minimum attractor was discovered for all architectures. For the single hidden layer architecture, the shape of the attractor did not change with an increase in the hidden layer size, but the error range and the gradient range widened. Thus, the global attraction basin widened for larger hidden layers. The l_{stag} values in Table 9.5 confirm that the speed of convergence to the global attractor increased with an increase in the hidden layer size for the $[-1, 1]$ micro setting.

For the 2-hidden layer architectures, a split into steep and shallow gradient clusters was observed. The steep gradient cluster was associated with inferior generalisation

Table 9.5: Basin of attraction estimates calculated for the Cancer problem for the various NN architectures. Standard deviation is shown in parenthesis.

	$h = 10$		$2 \times h = 20$		$10 \times h = 100$		
	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}	
$[-1, 1]$,	1.0000	975.3938	1.0000	979.7598	1.0000	983.6727	1 hidden layer
micro	(0.0000)	(2.6956)	(0.0000)	(1.5779)	(0.0000)	(0.6047)	
$[-1, 1]$,	1.0012	87.7195	1.0031	87.0997	1.0618	77.6422	
macro	(0.0353)	(2.6253)	(0.0683)	(4.7251)	(0.2850)	(18.5088)	
$[-10, 10]$,	1.0000	977.5683	1.0000	980.2099	1.0355	969.1150	
micro	(0.0000)	(2.5137)	(0.0000)	(1.7397)	(0.3280)	(83.4310)	
$[-10, 10]$,	1.0019	87.0460	1.0053	86.3427	1.0130	81.0644	
macro	(0.0466)	(4.4699)	(0.0725)	(6.6630)	(0.1237)	(11.1977)	
$[-1, 1]$,	1.0000	978.7923	1.0316	968.0676	1.3760	872.6577	2 hidden layers
micro	(0.0000)	(1.8208)	(0.2426)	(84.7024)	(1.1119)	(241.2722)	
$[-1, 1]$,	1.0012	87.0389	1.0024	85.1268	1.0396	71.5462	
macro	(0.0340)	(2.7952)	(0.0485)	(5.5808)	(0.2052)	(14.4155)	
$[-10, 10]$,	1.3192	871.7169	3.3058	496.9894	7.1540	142.8222	
micro	(0.8456)	(235.4979)	(2.4903)	(356.7182)	(2.6713)	(110.0526)	
$[-10, 10]$,	1.0980	75.3461	1.0953	71.3317	1.3820	48.7464	
macro	(0.3252)	(16.5105)	(0.3029)	(16.9158)	(0.6076)	(21.0796)	
$[-1, 1]$,	1.1273	913.5485	1.1546	931.3474	4.7934	212.0993	3 hidden layers
micro	(0.3840)	(170.6968)	(0.6819)	(166.8368)	(1.8085)	(123.4397)	
$[-1, 1]$,	1.0665	78.4510	1.0550	76.2268	1.1418	66.2924	
macro	(0.2565)	(14.3946)	(0.2382)	(14.2675)	(0.3752)	(18.5221)	
$[-10, 10]$,	3.6460	376.2874	4.8486	284.8526	5.5062	182.9733	
micro	(2.1179)	(293.3329)	(2.6841)	(250.7712)	(1.7998)	(77.8606)	
$[-10, 10]$,	1.1901	61.8568	1.2756	57.9492	1.2659	56.2615	
macro	(0.4304)	(19.4827)	(0.5306)	(20.5573)	(0.5053)	(19.2546)	

performance, and increased in width and steepness as more hidden neurons were added to the layers. The overlap between the two clusters increased as the hidden layer size increased. The n_{stag} results in Table 9.5 show that the transition between the attractors became more likely as the hidden layer size increased.

For the 3-hidden layer architecture, an increase in the hidden layer size also resulted in the steep gradient cluster becoming wider and heavier than the shallow gradient cluster. The gradient range increased drastically.

The effect of hidden layers

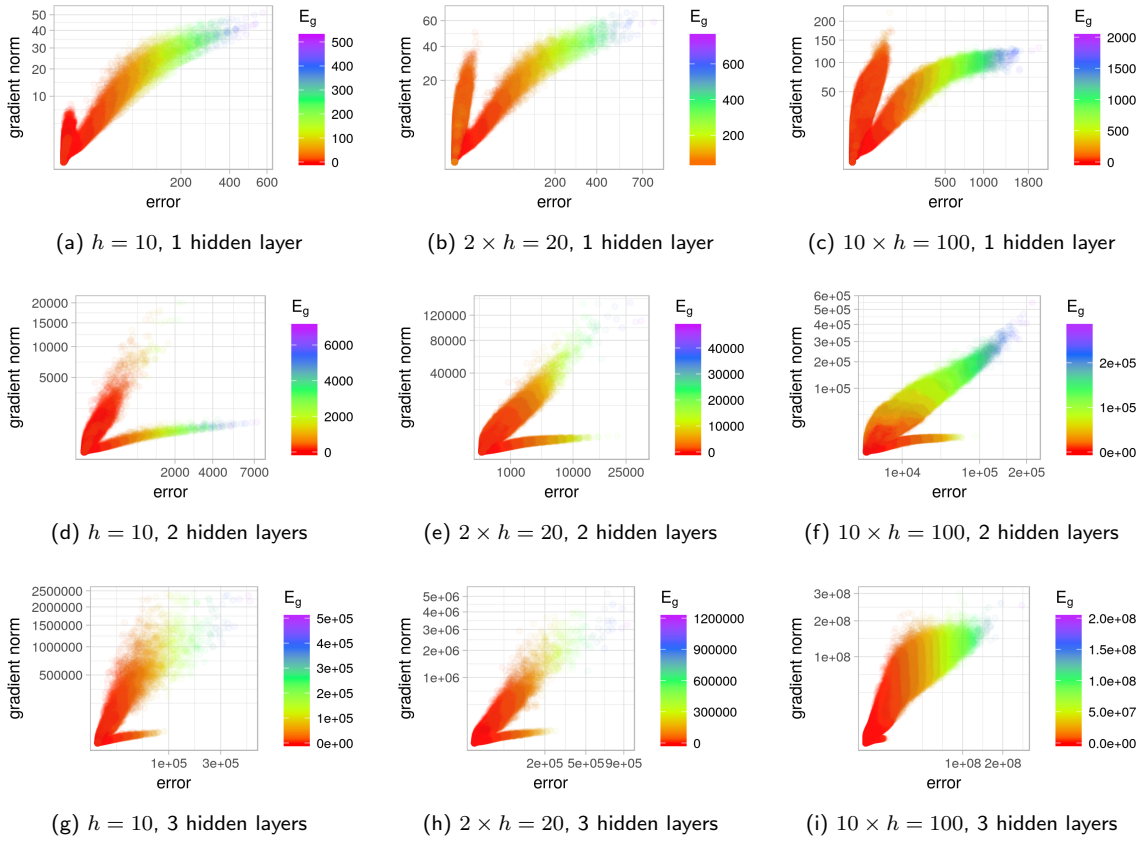
According to the classification results in Table B.19, the C_t values marginally decreased as more hidden layers were added to the architectures. The C_g values did not exhibit a consistent trend for the $[-1, 1]$ walks and generally remained the same, while the $[-10, 10]$ macro walks produced generalisation results of deteriorating quality as the number of hidden layers increased. Thus, the landscape remained searchable around the origin, and the areas further away from the origin became less searchable for deeper NNs.

Figure 9.21 shows that the addition of more hidden layers had a strong impact on the observed cluster structure. For a single hidden layer, the split into steep and shallow gradient clusters was not strongly manifested for the $[-1, 1]$ micro walks. Adding a second hidden layer added a heavy steep gradient cluster to all architectures, regardless of the hidden layer size. The steep gradient cluster became more exaggerated with the addition of a third hidden layer. Figure 9.22 illustrates that the same trend was observed for the $[-10, 10]$ initialisation range: Additional hidden layers reduced the shallow gradient cluster and increased the steep gradient cluster. Thus, deeper architectures were more prone to narrow valleys than the shallow architectures.

Thus, an increase in the hidden layer size was associated with wider attraction basins, and an increase in the architecture depth exaggerated the presence of the narrow valleys in the NN error landscapes.

9.2.6 Heart

Figure 9.23 summarises the curvature information obtained for the Heart problem for the various architectures. The Heart problem yielded loss surfaces dominated by the



(a) $h = 10$, 1 hidden layer (b) $2 \times h = 20$, 1 hidden layer (c) $10 \times h = 100$, 1 hidden layer

(d) $h = 10$, 2 hidden layers (e) $2 \times h = 20$, 2 hidden layers (f) $10 \times h = 100$, 2 hidden layers

(g) $h = 10$, 3 hidden layers (h) $2 \times h = 20$, 3 hidden layers (i) $10 \times h = 100$, 3 hidden layers

Figure 9.22: L-g clouds colourised according to the corresponding E_g values, obtained by the $[-10, 10]$ micro walks for the Cancer problem for the various NN architectures.

saddle curvature. Flatness was observed only for the $[-10, 10]$ initialisation range, and increased with an increase in dimensionality. An increase in the number of hidden layers had a stronger influence on the proportion of flat curvature points than an increase in the hidden layer size.

The effect of hidden neurons

Table B.20 in Appendix B shows that the C_t values generally increased with an increase in the hidden layer size across all granularity settings, including the $[-10, 10]$ micro and macro settings. Thus, an increase in the hidden layer size increased the overall search-ability of the error landscapes. The C_g values also generally improved or remained the

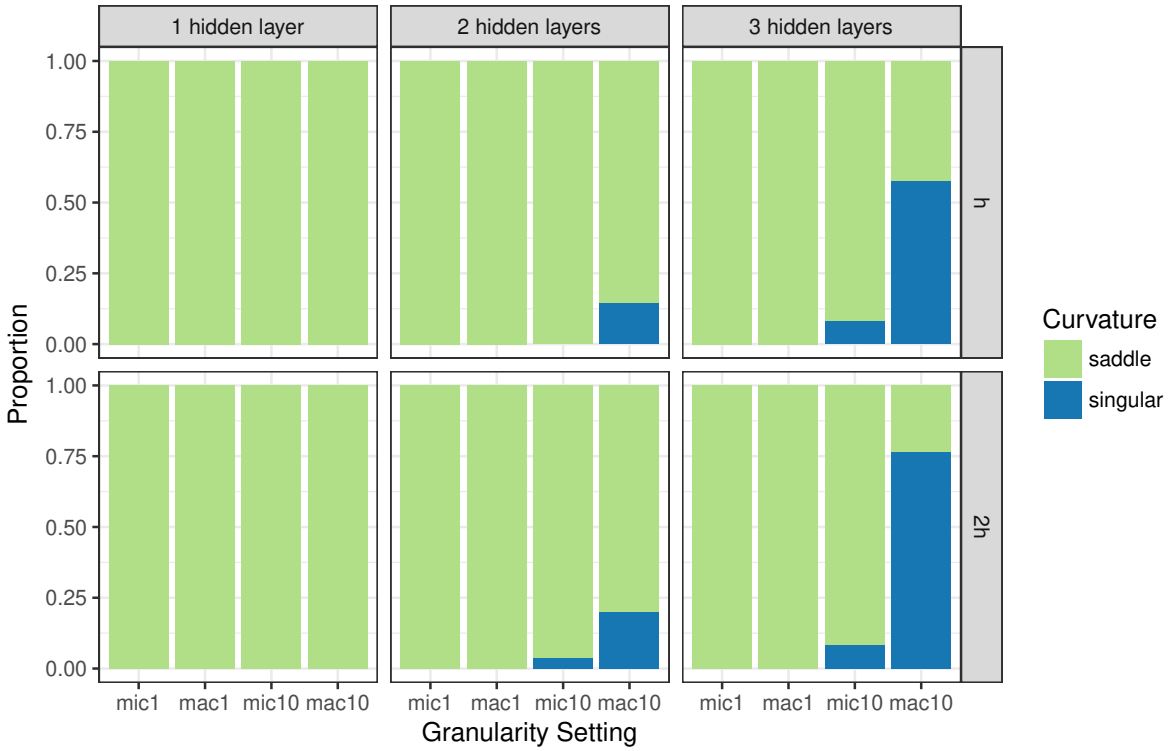
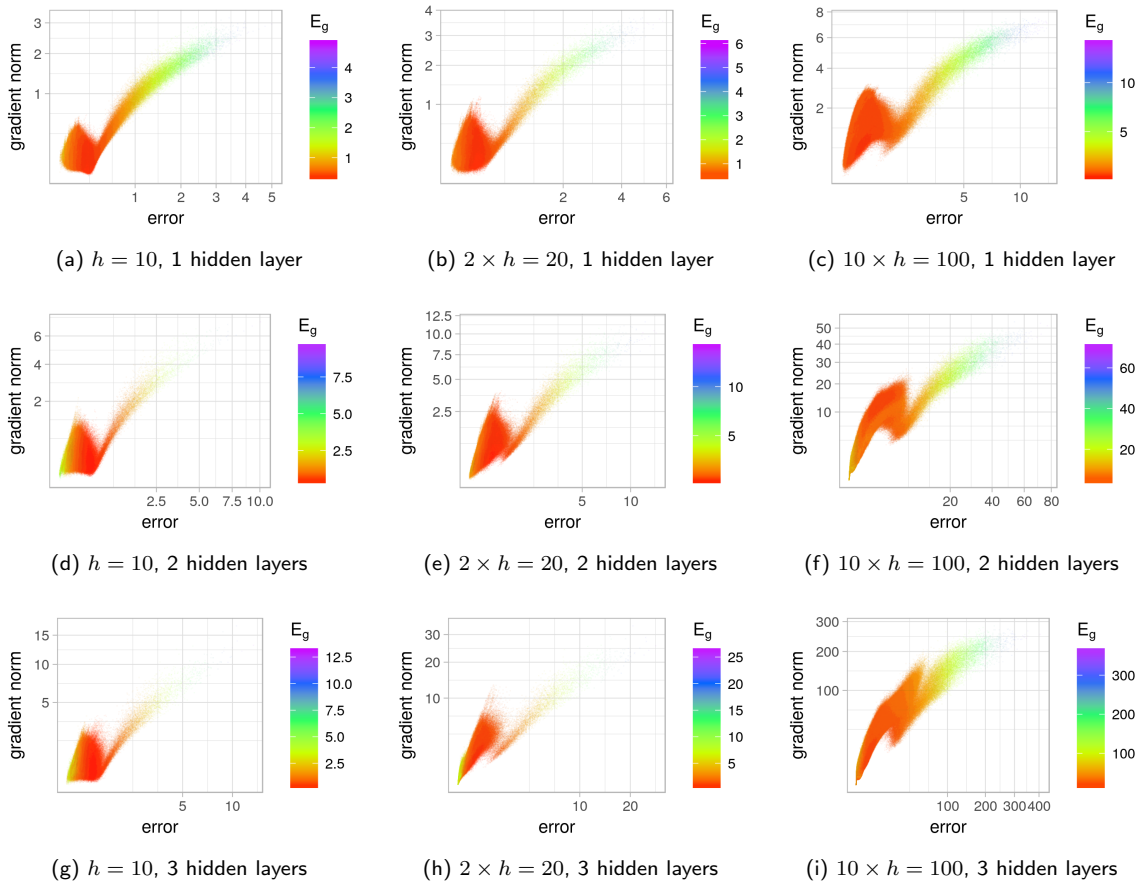


Figure 9.23: Histogram representation of the curvature information sampled by the gradient walks for the Heart problem for various NN architectures.

same as the hidden layer size increased. The highest average C_g value (80.5% accuracy) was observed for the $10 \times h$ architecture with a single hidden layer.

Figure 9.24 shows the l-g clouds obtained by the $[-1, 1]$ micro walks for the various NN architectures. It is evident from Figure 9.24 that the range of error and gradient magnitudes increased with an increase in the hidden layer size. Thus, the width and the steepness of the attractors increased. For the smallest hidden layer size, $h = 10$, two stationary attractors are visible in Figures 9.24d and 9.24g. An increase of the hidden layer size caused the non-global stationary attractor to become non-stationary, as illustrated in Figures 9.24e, 9.24f, 9.24h, and 9.24i. Thus, the search space was simplified by eliminating local minima. Interestingly, the generalisation performance at the local minimum was better than at the global minimum. Thus, convergence to a local minimum may in fact be desirable for NN training.



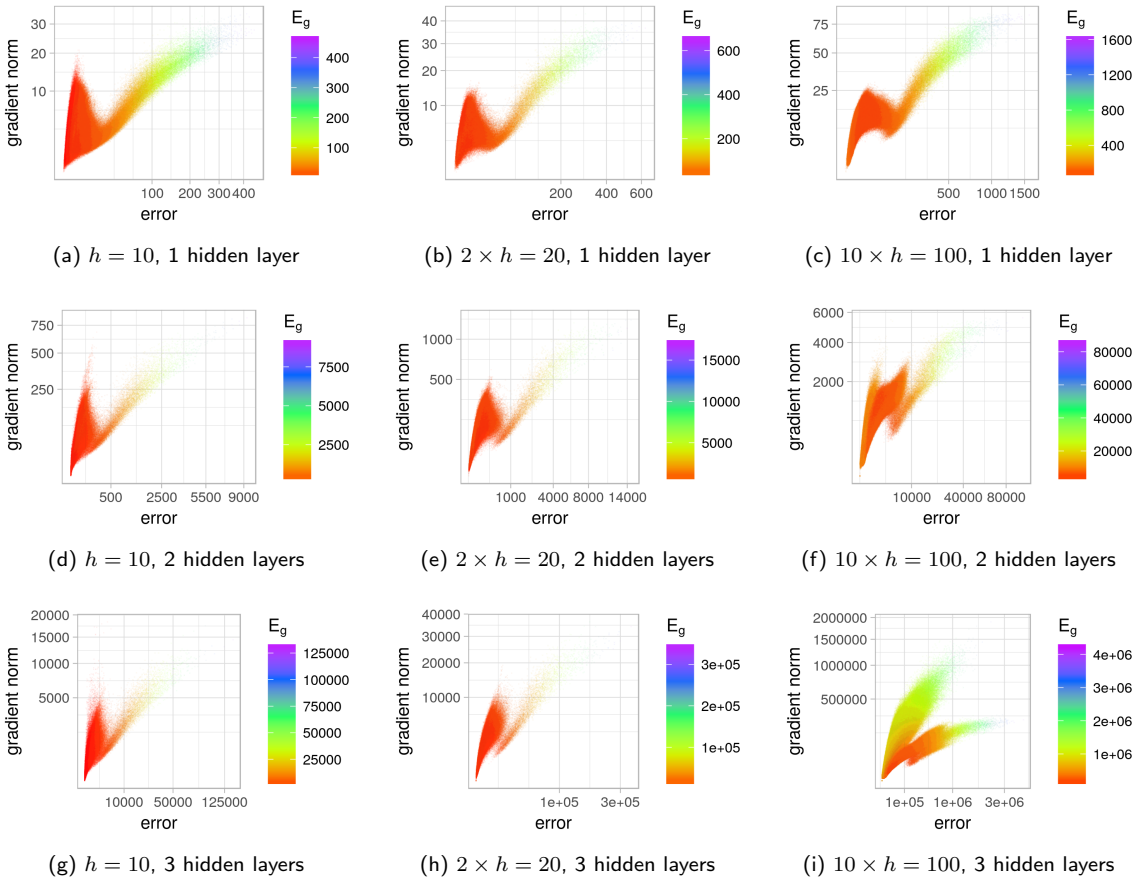
(a) $h = 10$, 1 hidden layer (b) $2 \times h = 20$, 1 hidden layer (c) $10 \times h = 100$, 1 hidden layer

(d) $h = 10$, 2 hidden layers (e) $2 \times h = 20$, 2 hidden layers (f) $10 \times h = 100$, 2 hidden layers

(g) $h = 10$, 3 hidden layers (h) $2 \times h = 20$, 3 hidden layers (i) $10 \times h = 100$, 3 hidden layers

Figure 9.24: L-g clouds colourised according to the corresponding E_g values, obtained by the $[-1, 1]$ micro walks for the Heart problem for the various NN architectures.

Figure 9.25 shows that the $[-10, 10]$ micro walks also discovered two attractors for the Heart problem. The split into steep and shallow gradient clusters was evident, although an increase in the number of hidden layers generally increased the overlap between the two clusters for both $[-1, 1]$ and $[-10, 10]$ walks. The l_{stag} values reported in Table 9.6 indicate that convergence to an attraction basin became easier as the size of the hidden layers increased. The n_{stag} values also indicate that the transition between attractors became more likely for the NN architectures of higher dimensionality.



(a) $h = 10$, 1 hidden layer (b) $2 \times h = 20$, 1 hidden layer (c) $10 \times h = 100$, 1 hidden layer

(d) $h = 10$, 2 hidden layers (e) $2 \times h = 20$, 2 hidden layers (f) $10 \times h = 100$, 2 hidden layers

(g) $h = 10$, 3 hidden layers (h) $2 \times h = 20$, 3 hidden layers (i) $10 \times h = 100$, 3 hidden layers

Figure 9.25: L-g clouds colourised according to the corresponding E_g values, obtained by the $[-10, 10]$ micro walks for the Heart problem for the various NN architectures.

The effect of hidden layers

Table B.20 shows that the C_t values improved for the $[-1, 1]$ walks as more hidden layers were added. However, for the $[-10, 10]$ initialisation range, an increase in the number of hidden layers was associated with decreasing C_t values. Thus, deeper architectures were easier to search around the origin, but harder to exploit further away from the origin. The C_g values, however, consistently decreased with an increase in the architecture's depth, regardless of the initialisation range. Since no regularisation was applied to the gradient walks, the observed overfitting is attributed to the better searchability of the deeper architectures, resulting in stronger global minimum exploitation.

Table 9.6: Basin of attraction estimates calculated for the Heart problem for the various NN architectures. Standard deviation is shown in parenthesis.

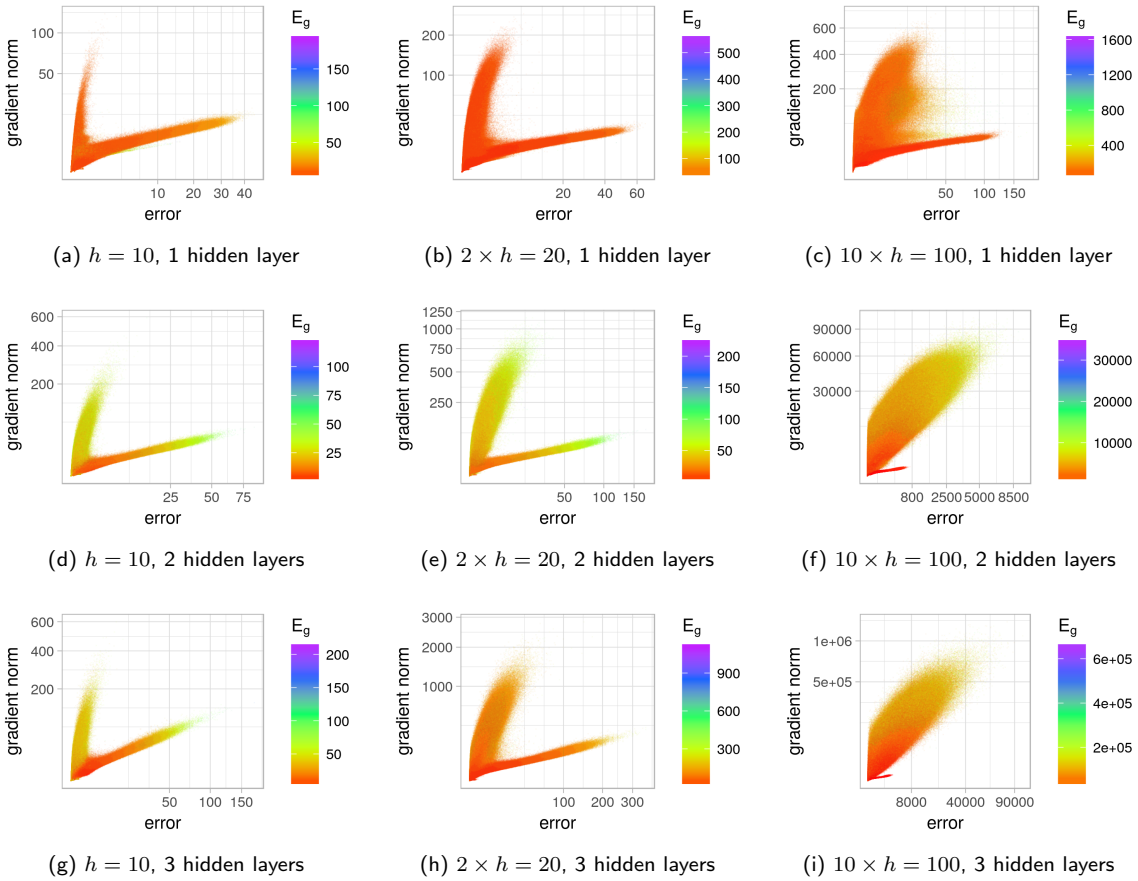
	$h = 10$		$2 \times h = 20$		$10 \times h = 100$		
	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}	
$[-1, 1]$,	1.0000	969.4890	1.0000	974.4604	1.0000	980.5903	1 hidden layer
micro	(0.0000)	(2.6436)	(0.0000)	(2.0019)	(0.0000)	(1.4897)	
$[-1, 1]$,	1.0050	86.0182	1.0191	82.2622	1.3890	25.3449	
macro	(0.0704)	(6.6016)	(0.1472)	(11.9345)	(0.6797)	(20.3909)	
$[-10, 10]$,	1.0000	967.6488	1.0000	972.7808	1.0000	979.1629	
micro	(0.0000)	(3.0297)	(0.0000)	(2.3289)	(0.0000)	(1.9214)	
$[-10, 10]$,	1.0103	84.2049	1.0154	81.8979	1.0769	56.1771	
macro	(0.1144)	(8.4039)	(0.1232)	(10.5917)	(0.3927)	(25.9864)	
$[-1, 1]$,	1.0000	974.6851	1.0000	978.7494	1.0000	978.6013	2 hidden layers
micro	(0.0000)	(2.2263)	(0.0000)	(1.3854)	(0.0000)	(5.2395)	
$[-1, 1]$,	1.0000	84.9390	1.0027	81.2629	1.2582	31.1182	
macro	(0.0000)	(4.6369)	(0.0521)	(6.8659)	(0.5339)	(17.2483)	
$[-10, 10]$,	1.0000	975.5710	1.0000	978.6681	1.0006	972.9974	
micro	(0.0000)	(1.8512)	(0.0000)	(1.8314)	(0.0249)	(16.6906)	
$[-10, 10]$,	1.0022	81.5626	1.0305	75.6269	1.4328	35.0503	
macro	(0.0470)	(7.2358)	(0.1848)	(12.9809)	(0.6165)	(17.4804)	
$[-1, 1]$,	1.0000	977.6052	1.0000	980.3423	1.0000	969.1488	3 hidden layers
micro	(0.0000)	(2.5311)	(0.0000)	(1.9115)	(0.0000)	(10.1859)	
$[-1, 1]$,	1.0027	83.6203	1.0023	82.0634	1.2920	49.2953	
macro	(0.0516)	(5.6872)	(0.0543)	(5.4075)	(0.5641)	(22.0752)	
$[-10, 10]$,	1.0009	978.0705	1.0000	979.9270	3.1999	419.1983	
micro	(0.0306)	(15.3807)	(0.0000)	(2.7779)	(1.8921)	(284.5976)	
$[-10, 10]$,	1.2000	62.2581	1.0757	73.8890	1.1661	55.3564	
macro	(0.4543)	(21.3641)	(0.2979)	(15.3303)	(0.4230)	(18.4677)	

Figures 9.24 and 9.25 show than an increase in the number of hidden layers yielded a single sharp global minimum attractor, that was successfully exploited by the gradient walks. Figure 9.24 also shows that the generalisation behaviour at the global minimum was generally inferior. The width and steepness of the attraction basin increased rapidly as more hidden layers were added. For the $[-10, 10]$ initialisation range, an increase in the number of hidden layers once again resulted in the exaggeration of the steep gradient cluster. The decline in performance for the $[-10, 10]$ initialisation range, associated with an increase of the architecture depth, is thus attributed to the visible split into narrow and wide valleys (see Figure 9.25).

9.2.7 MNIST

Table B.21 in Appendix B summarises the average MNIST classification accuracies obtained at the last step of the gradient walks for the various NN architectures. The results in Table B.21 show that the C_t generally increased as the hidden layer size increased for all granularity settings except the $[-1, 1]$ micro setting. Thus, the error landscapes yielded by wider hidden layers were somewhat harder to exploit with very small steps, but the overall searchability, i.e. global landscape structure, improved. For the 1-hidden layer architecture, the C_t accuracy improved from 61% to 87% for the $[-10, 10]$ macro setting as h increased from 10 to 100. For the 3-hidden layer architecture, the C_t accuracy improved from 15% to 91% as h increased from 10 to 100. This observation corresponds to a recent theoretical study by Johnson [58], where deep and "skinny" NN architectures, i.e. architectures with many hidden layers of a limited size, were shown to not be universal approximators. Table B.21 shows that the generalisation accuracy was also positively affected by an increase in the hidden layer size for the 2- and 3-hidden layer architectures.

Figure 9.26 shows the l-g clouds obtained for the various architectures using the $[-1, 1]$ micro walks. For all architectures considered, the sampled points were split into the steep and shallow gradient clusters, where the steep gradient cluster generally corresponded to poor generalisation performance. An increase in the hidden layer size affected the shape of the steep gradient cluster by making it wider and steeper. Figure 9.26 shows that the transition from h to $2 \times h$, to $10 \times h$, yielded the steep gradient cluster to become



(a) $h = 10$, 1 hidden layer (b) $2 \times h = 20$, 1 hidden layer (c) $10 \times h = 100$, 1 hidden layer

(d) $h = 10$, 2 hidden layers (e) $2 \times h = 20$, 2 hidden layers (f) $10 \times h = 100$, 2 hidden layers

(g) $h = 10$, 3 hidden layers (h) $2 \times h = 20$, 3 hidden layers (i) $10 \times h = 100$, 3 hidden layers

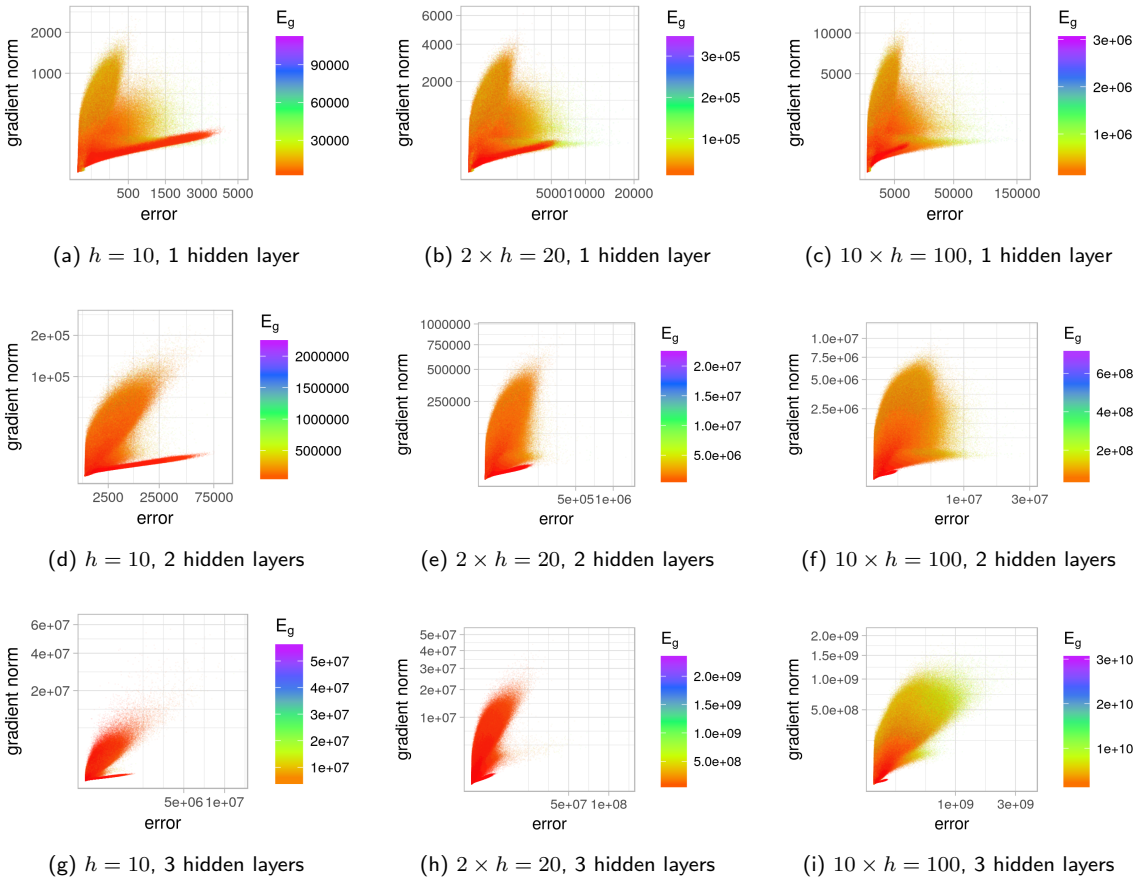
Figure 9.26: L-g clouds colourised according to the corresponding E_g values, obtained by the $[-1, 1]$ micro walks for the MNIST problem for the various NN architectures.

progressively wider. The overlap between the two clusters also increased. Table 9.7 shows that an increase in the hidden layer size resulted in increased n_{stag} values, indicating that the transition between clusters became more likely.

Figure 9.27 shows the l-g clouds obtained by the $[-10, 10]$ micro walks. For the wider initialisation range, the same split into two clusters is observed, although the two clusters appear more connected. The $[-10, 10]$ micro walks used a larger maximum step size ($\varepsilon = 0.2$) than the $[-1, 1]$ micro walks ($\varepsilon = 0.02$), which enabled the transition between the attractors. The n_{stag} results in Table 9.7 confirm that the $[-10, 10]$ micro setting yielded more transitions than the $[-1, 1]$ micro setting. Overall, Figure 9.27

Table 9.7: Basin of attraction estimates calculated for the MNIST problem for the various NN architectures. Standard deviation is shown in parenthesis.

	$h = 10$		$2 \times h = 20$		$10 \times h = 100$		
	n_{stag}	l_{stag}	n_{stag}	l_{stag}	n_{stag}	l_{stag}	
$[-1, 1]$,	1.0002	972.4753	1.0003	976.5304	1.2534	869.2570	1 hidden layer
micro	(0.0137)	(7.2004)	(0.0231)	(8.1524)	(0.6305)	(214.7664)	
$[-1, 1]$,	1.0000	87.1719	1.0000	86.8179	1.1187	58.1058	
macro	(0.0000)	(1.0704)	(0.0056)	(1.9766)	(0.3262)	(14.0348)	
$[-10, 10]$,	1.2259	897.1252	1.9853	736.3558	6.3124	209.9105	
micro	(0.7907)	(196.0541)	(1.7754)	(328.4335)	(3.0031)	(217.9327)	
$[-10, 10]$,	1.1188	64.2060	1.1204	62.5981	1.2399	39.0601	
macro	(0.3251)	(15.2903)	(0.3263)	(14.8605)	(0.4636)	(13.9766)	
$[-1, 1]$,	1.0000	975.1188	1.0003	975.3782	5.4535	132.0095	2 hidden layers
micro	(0.0000)	(1.6518)	(0.0186)	(10.0241)	(1.5912)	(50.7952)	
$[-1, 1]$,	1.0055	82.4143	1.0347	72.3031	1.2228	47.4801	
macro	(0.0749)	(6.7373)	(0.1836)	(11.2519)	(0.4494)	(16.1408)	
$[-10, 10]$,	1.5375	823.3736	5.7235	189.3396	7.3454	89.0696	
micro	(1.2952)	(272.9467)	(2.2454)	(153.6532)	(1.9763)	(34.2737)	
$[-10, 10]$,	1.1399	61.4209	1.2137	48.3679	1.2899	42.3394	
macro	(0.3581)	(17.5909)	(0.4350)	(16.1845)	(0.5078)	(16.0460)	
$[-1, 1]$,	1.0000	977.4182	1.0016	976.3500	6.0636	113.1507	3 hidden layers
micro	(0.0000)	(1.4157)	(0.0440)	(20.0490)	(1.7757)	(46.5121)	
$[-1, 1]$,	1.0639	75.6586	1.2234	51.7492	1.1649	54.7344	
macro	(0.2607)	(15.4679)	(0.4740)	(18.9694)	(0.3830)	(15.8819)	
$[-10, 10]$,	2.2607	669.4512	5.6652	156.0723	6.4024	119.4655	
micro	(1.9355)	(347.9560)	(1.9549)	(83.7339)	(1.7800)	(46.3345)	
$[-10, 10]$,	1.3552	47.1306	1.3467	46.0160	1.1937	51.7768	
macro	(0.5909)	(22.2128)	(0.6033)	(20.0852)	(0.4126)	(16.1017)	



(a) $h = 10$, 1 hidden layer (b) $2 \times h = 20$, 1 hidden layer (c) $10 \times h = 100$, 1 hidden layer

(d) $h = 10$, 2 hidden layers (e) $2 \times h = 20$, 2 hidden layers (f) $10 \times h = 100$, 2 hidden layers

(g) $h = 10$, 3 hidden layers (h) $2 \times h = 20$, 3 hidden layers (i) $10 \times h = 100$, 3 hidden layers

Figure 9.27: L-g clouds colourised according to the corresponding E_g values, obtained by the $[-1, 1]$ micro walks for the MNIST problem for the various NN architectures.

shows that an increase in the hidden layer size increased the overlap between the two clusters, effectively blending the two clusters into one wide cluster with a single global attractor at the error of zero. Thus, an increase in the hidden layer size simplified the error landscapes.

Table B.21 shows that the addition of more layers without an increase in the hidden layer size generally resulted in deteriorating C_t as well as C_g values. This observation once again correlates with the theoretical findings in [58]. The l-g clouds in Figures 9.26 and 9.27 show that an increase in the number of hidden layers increased the gradient and error magnitude ranges, but otherwise did not affect the degree of separation between

the steep and the shallow gradient clusters. As the depth of the architecture increased, the steeper cluster became visibly heavier, and the shallow cluster diminished. In Figure 9.27i, the steep gradient cluster exhibited a few layers, indicating that the global basin of attraction comprised of valleys of varied steepness. Overall, an increase in the number of hidden layers did not simplify the MNIST error landscape.

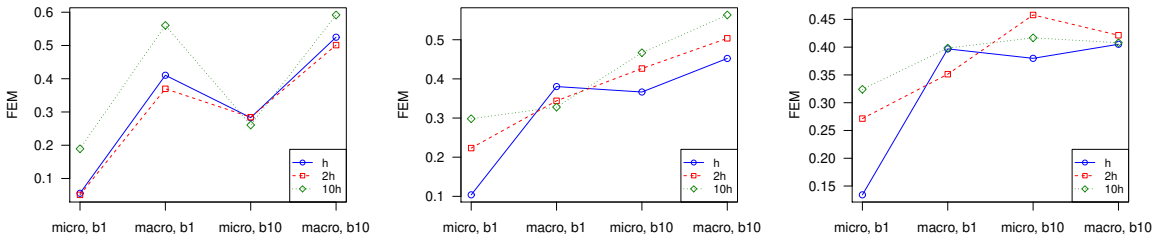
9.3 Ruggedness, Gradients, Neutrality

The ruggedness, gradients, and neutrality metrics, as discussed in Section 3.4, were calculated for all problems to gain more insight into the fitness landscape properties associated with the various NN architecture settings. The magnitudes of the numerical gradients were used instead of gradient estimates.

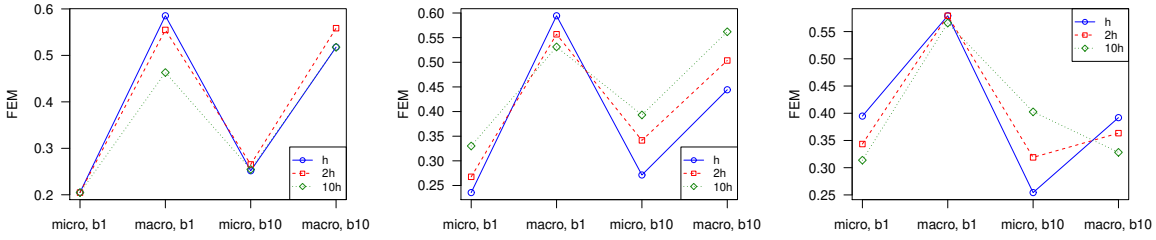
Figures 9.28 and 9.29 show the FEM values obtained for the various problems. Across most problems considered and across the various architectures, the $[-1, 1]$ macro setting consistently yielded the highest ruggedness. It was also observed in Sections 9.2.1 to 9.2.7 that the $[-1, 1]$ macro setting was associated with the least amount of flat curvature, i.e. indefinite Hessians. These observations indicate that across all problems, the $[-1, 1]$ macro setting yielded oscillatory behaviour, and the maximum step size typically exceeded the width of the attraction basin. The same maximum step for the $[-10, 10]$ initialisation interval yielded less ruggedness, indicating that the attraction basins, hypothesised to have a valley shape, became wider further away from the origin.

Figures 9.28 and 9.29 show that an increase in the hidden layer size often yielded a smoother FEM curve across the various granularity settings, i.e. a more consistent degree of ruggedness. This correlates with the observations made in this chapter that an increase in the hidden layer size makes the NN loss surfaces more searchable, and reduces the sensitivity of the gradient-guided search to the chosen step size and initialisation range. An increase in the number of hidden layers also yielded a decrease in the maximal FEM values for some of the problems, indicating that deeper architectures have indeed exhibited smoother landscapes for some scenarios.

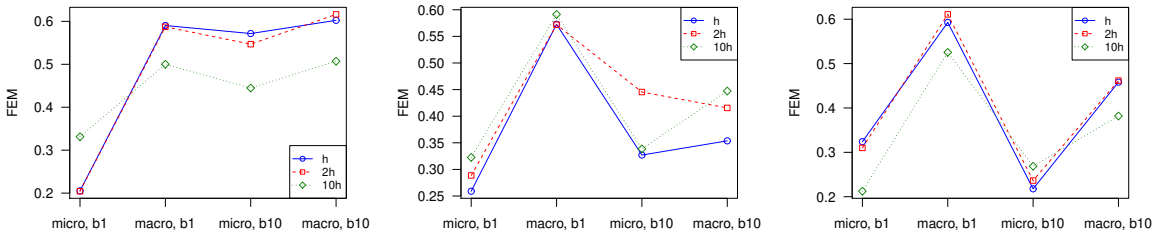
Figures 9.30 and 9.31 summarise the gradient averages and standard deviations obtained for the various problems. The gradients exhibited a consistently upward trend



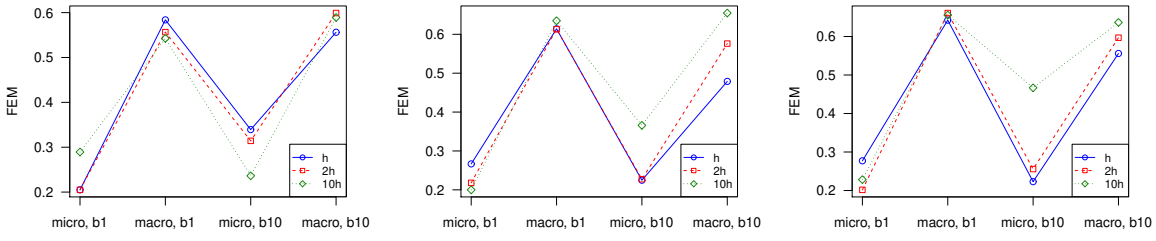
(a) XOR, FEM, 1 hidden layer (b) XOR, FEM, 2 hidden layers (c) XOR, FEM, 3 hidden layers



(d) Iris, FEM, 1 hidden layer (e) Iris, FEM, 2 hidden layers (f) Iris, FEM, 3 hidden layers



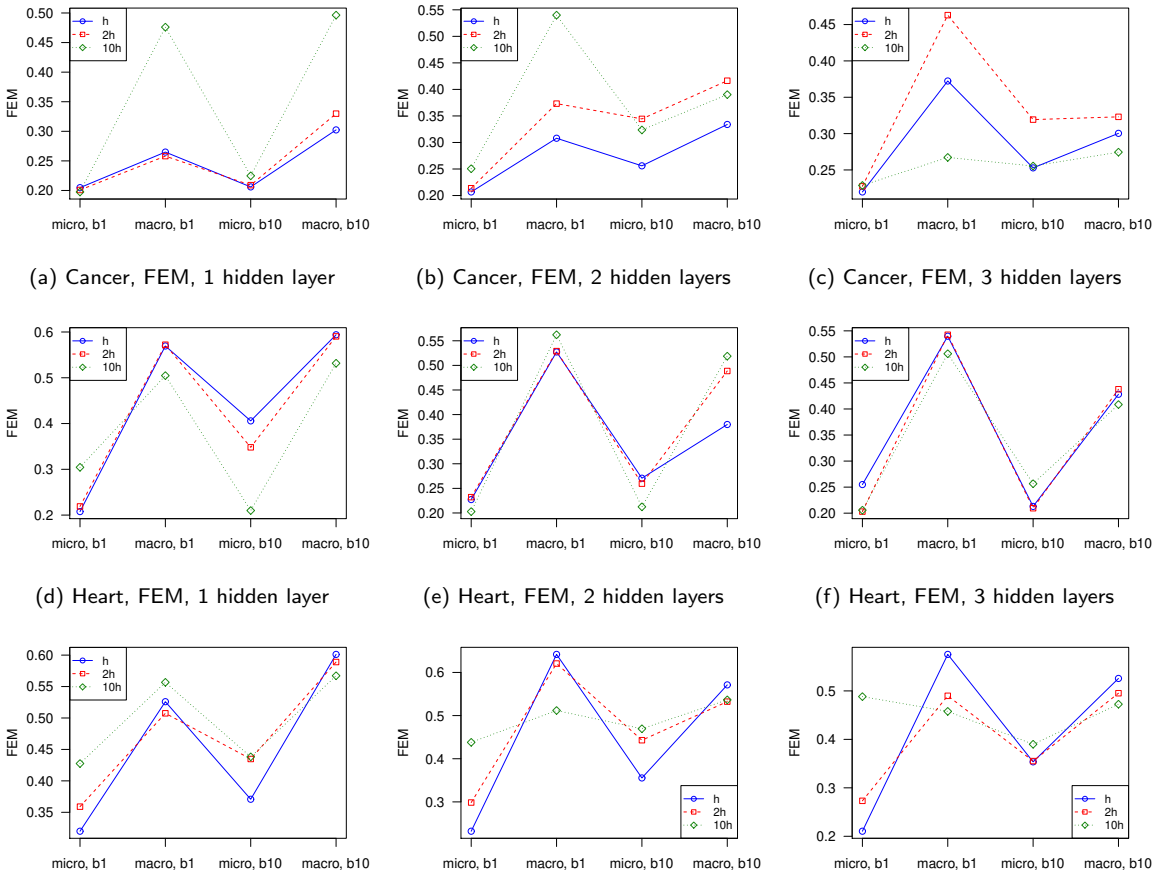
(g) Diabetes, FEM, 1 hidden layer (h) Diabetes, FEM, 2 hidden layers (i) Diabetes, FEM, 3 hidden layers



(j) Glass, FEM, 1 hidden layer (k) Glass, FEM, 2 hidden layers (l) Glass, FEM, 3 hidden layers

Figure 9.28: FEM metrics for the XOR, Iris, Diabetes, and Glass problems.

associated with an increase in the step size, initialisation range, hidden layer size, and the total number of hidden layers. An increase in the number of hidden layers had a more drastic effect on the maximal gradient magnitudes than an increase in the hidden



(a) Cancer, FEM, 1 hidden layer (b) Cancer, FEM, 2 hidden layers (c) Cancer, FEM, 3 hidden layers

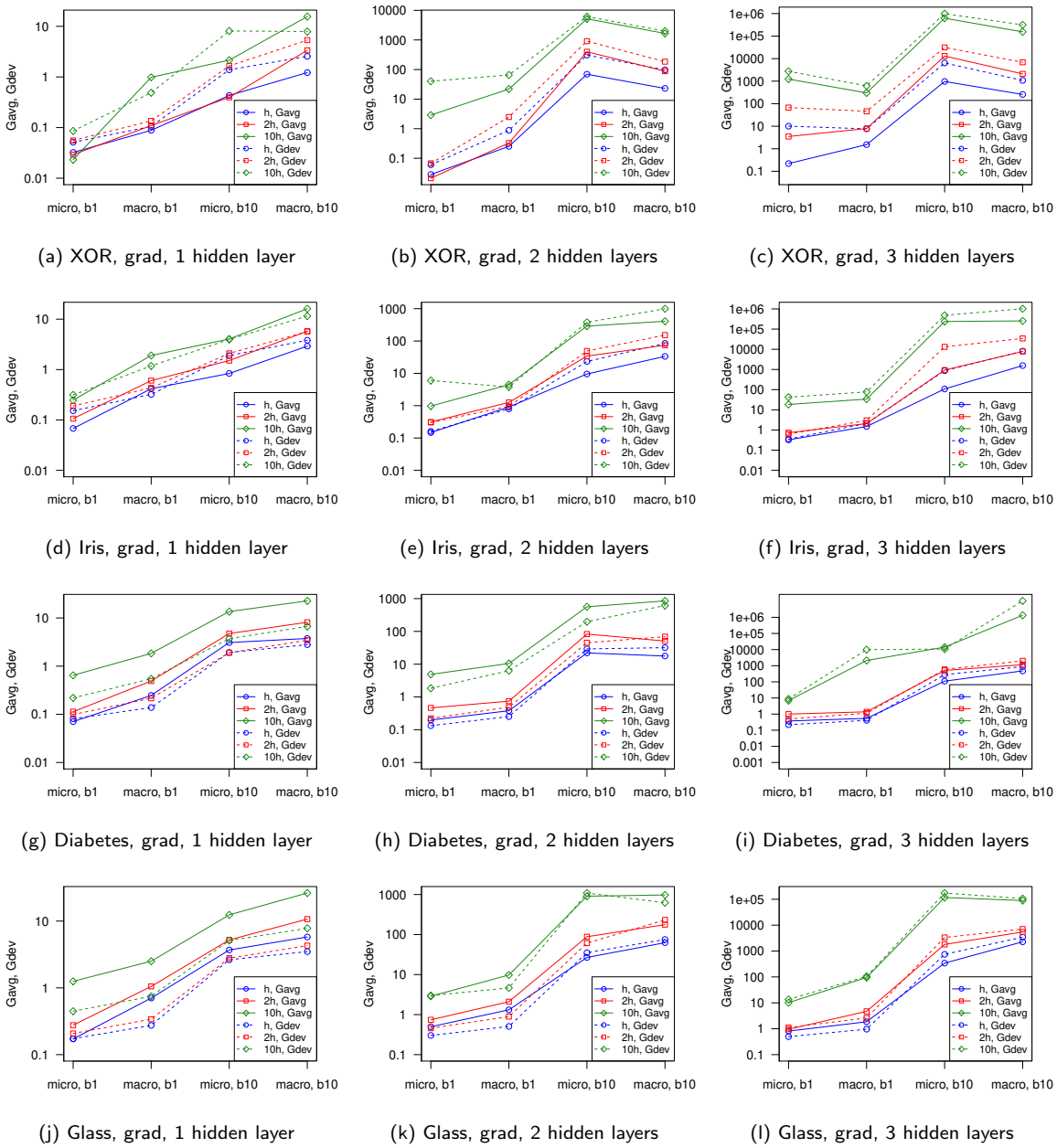
(d) Heart, FEM, 1 hidden layer (e) Heart, FEM, 2 hidden layers (f) Heart, FEM, 3 hidden layers

(g) MNIST, FEM, 1 hidden layer (h) MNIST, FEM, 2 hidden layers (i) MNIST, FEM, 3 hidden layers

Figure 9.29: FEM metrics for the Cancer, Heart, and MNIST problems.

layer size. For most problems, $G_{avg} < G_{dev}$ was observed, indicating step-like transitions between fitness levels. For problems where $G_{avg} > G_{dev}$ was observed for smaller architectures, an increase in the total number of hidden layers yielded G_{dev} to gradually become larger than the corresponding G_{avg}. This correlates with the observation that deep and over-parametrised architectures induce flatness and high lying plateaus [68], thus yielding more drastic transitions between the various landscape structures.

Figures 9.32 and 9.33 show the neutrality metrics obtained for the various problems. Out of the seven problems considered, only XOR and Cancer exhibited a significant amount of neutrality. XOR has a very limited number of data patterns, and Cancer is



(a) XOR, grad, 1 hidden layer

(b) XOR, grad, 2 hidden layers

(c) XOR, grad, 3 hidden layers

(d) Iris, grad, 1 hidden layer

(e) Iris, grad, 2 hidden layers

(f) Iris, grad, 3 hidden layers

(g) Diabetes, grad, 1 hidden layer

(h) Diabetes, grad, 2 hidden layers

(i) Diabetes, grad, 3 hidden layers

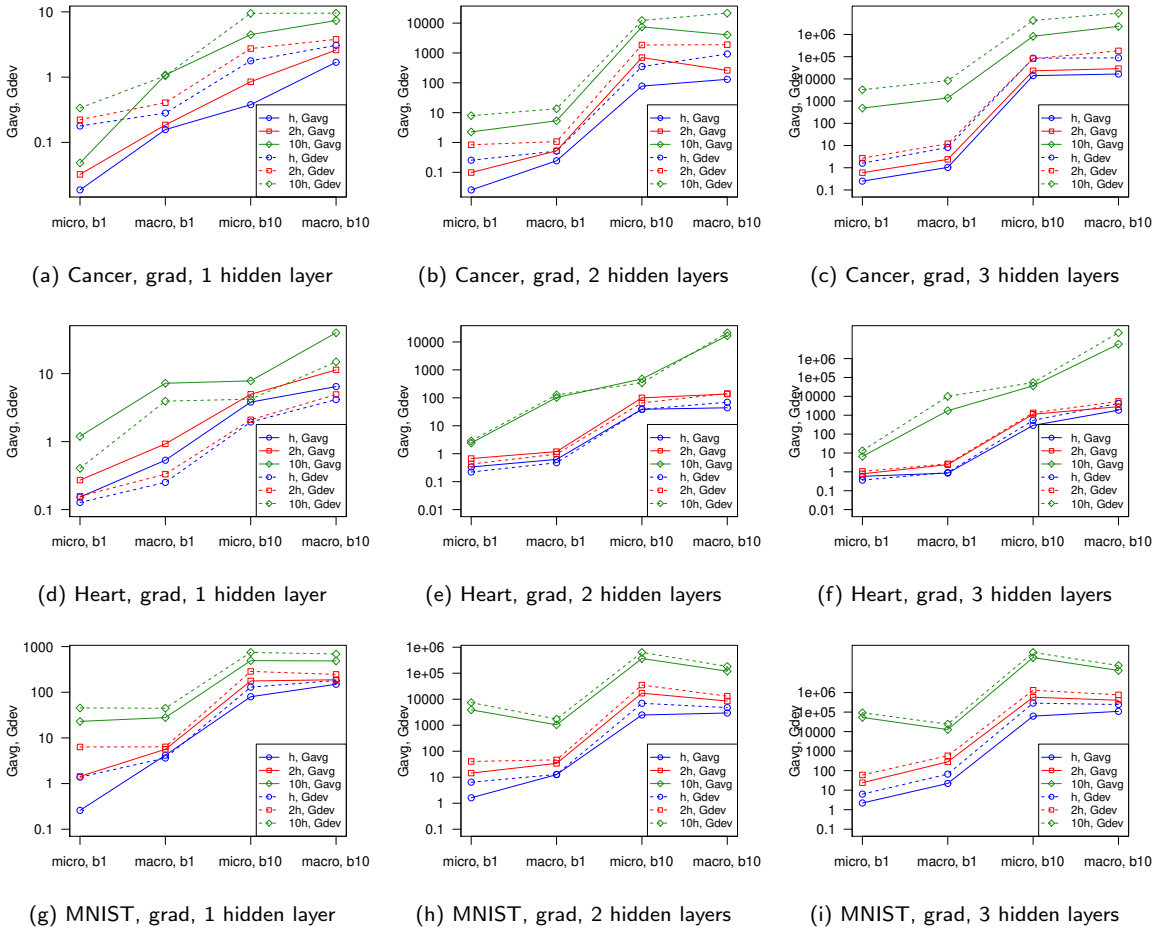
(j) Glass, grad, 1 hidden layer

(k) Glass, grad, 2 hidden layers

(l) Glass, grad, 3 hidden layers

Figure 9.30: Gradient metrics for the XOR, Iris, Diabetes, and Glass problems.

known to be a trivial classification problem. Thus, the observed neutrality is attributed to the comparative simplicity of these two problems. Other problems such as MNIST exhibited a minor degree of neutrality for the smallest initialisation range and step size



(a) Cancer, grad, 1 hidden layer (b) Cancer, grad, 2 hidden layers (c) Cancer, grad, 3 hidden layers

(d) Heart, grad, 1 hidden layer (e) Heart, grad, 2 hidden layers (f) Heart, grad, 3 hidden layers

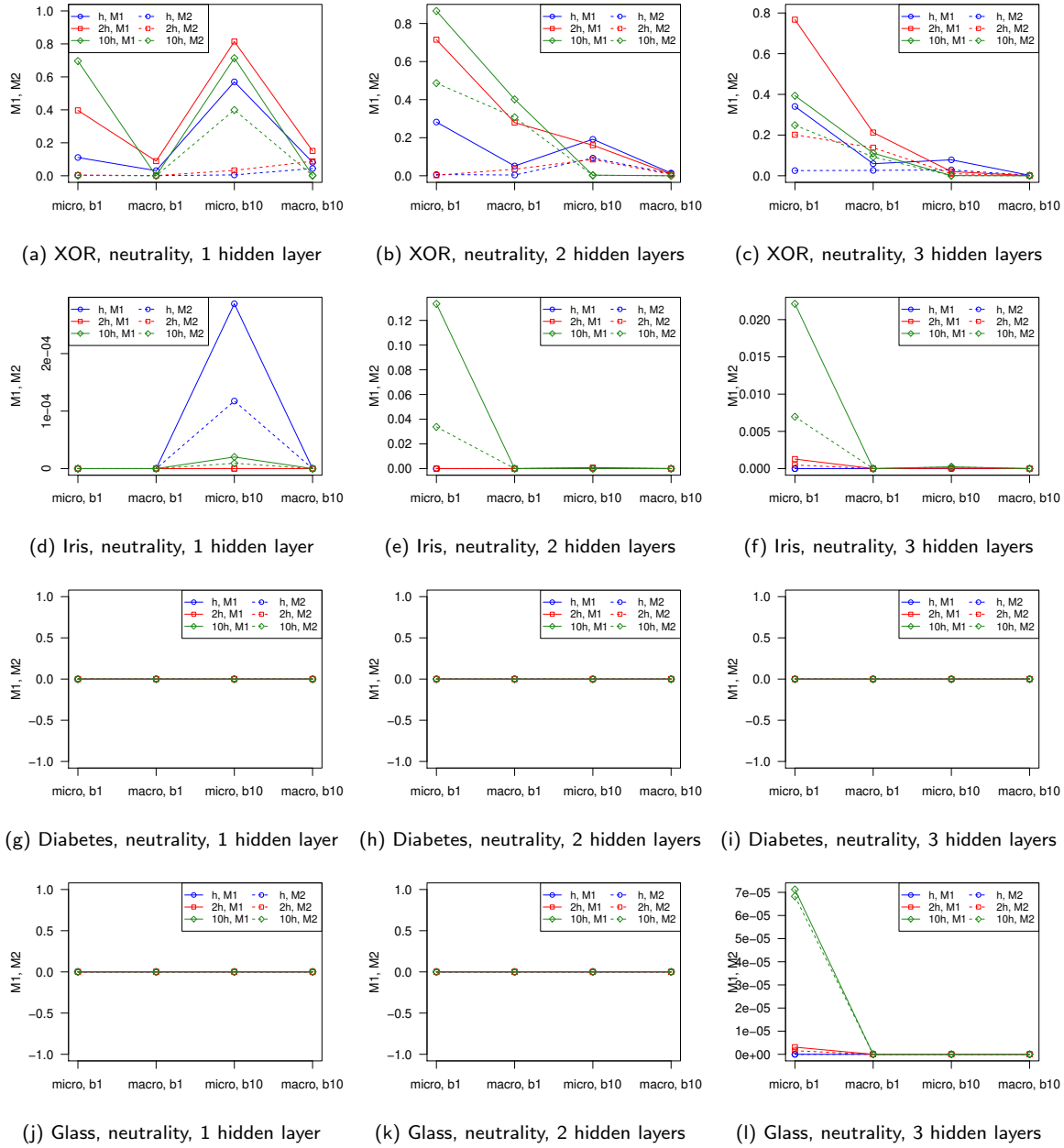
(g) MNIST, grad, 1 hidden layer (h) MNIST, grad, 2 hidden layers (i) MNIST, grad, 3 hidden layers

Figure 9.31: Gradient metrics for the Cancer, Heart, and MNIST problems.

considered. The observed neutrality is attributed to the successful exploitation of the global minima. The Heart and the Glass problems exhibited neutrality only for the 3-hidden layer architectures, which corresponds to the increased flatness associated with an increase in dimensionality, as well as the improved searchability, i.e. easier exploitation.

9.4 Conclusions

This chapter presented a visual and numerical analysis of the NN loss surfaces associated with various NN architectures. Seven different classification problems were considered.



(a) XOR, neutrality, 1 hidden layer (b) XOR, neutrality, 2 hidden layers (c) XOR, neutrality, 3 hidden layers

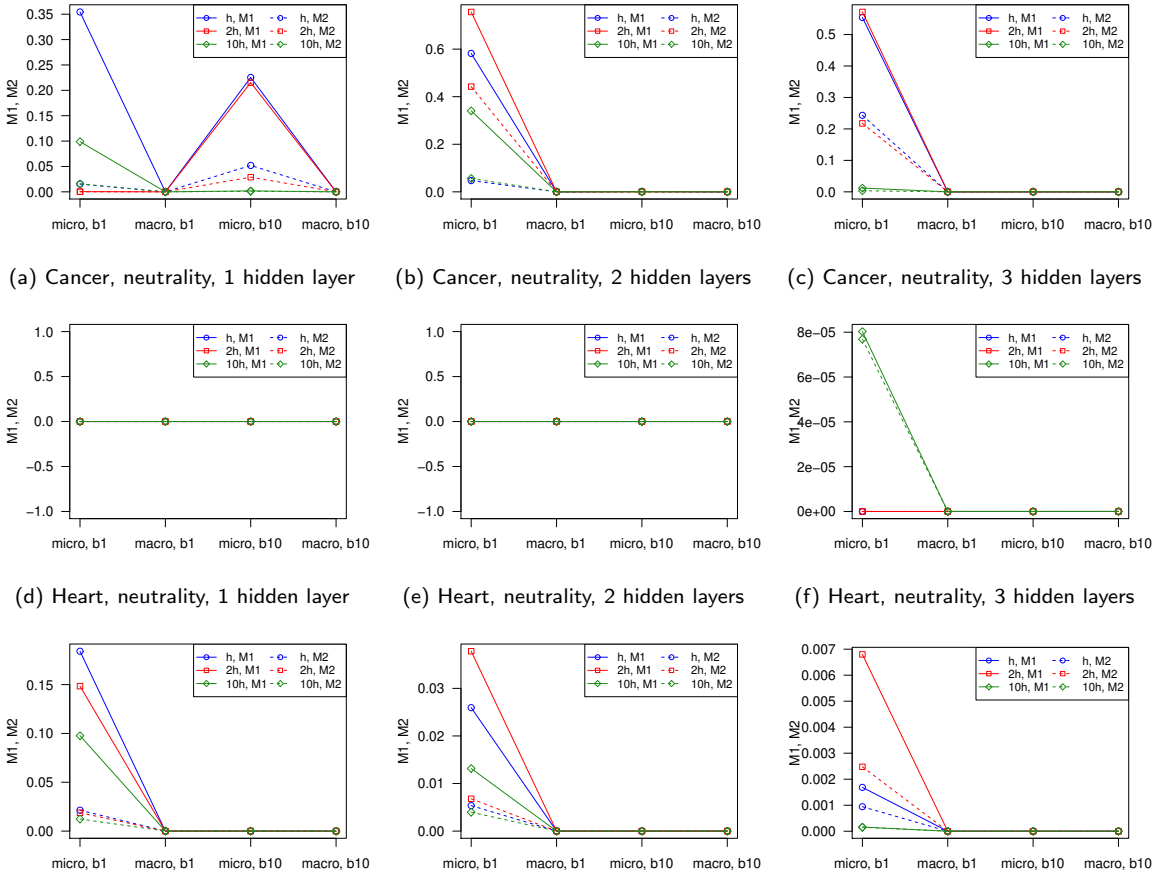
(d) Iris, neutrality, 1 hidden layer (e) Iris, neutrality, 2 hidden layers (f) Iris, neutrality, 3 hidden layers

(g) Diabetes, neutrality, 1 hidden layer (h) Diabetes, neutrality, 2 hidden layers (i) Diabetes, neutrality, 3 hidden layers

(j) Glass, neutrality, 1 hidden layer (k) Glass, neutrality, 2 hidden layers (l) Glass, neutrality, 3 hidden layers

Figure 9.32: Neutrality metrics for the XOR, Iris, Diabetes, and Glass problems.

For each problem, h, $2 \times h$, and $10 \times h$ hidden layer sizes were considered, where h corresponded to the minimal number of hidden neurons for a 1-hidden layer architecture. Further, for each hidden layer size, 1-, 2-, and 3-hidden layer architectures were



(a) Cancer, neutrality, 1 hidden layer (b) Cancer, neutrality, 2 hidden layers (c) Cancer, neutrality, 3 hidden layers

(d) Heart, neutrality, 1 hidden layer (e) Heart, neutrality, 2 hidden layers (f) Heart, neutrality, 3 hidden layers

(g) MNIST, neutrality, 1 hidden layer (h) MNIST, neutrality, 2 hidden layers (i) MNIST, neutrality, 3 hidden layers

Figure 9.33: Neutrality metrics for the Cancer, Heart, and MNIST problems.

considered. Each architecture was studied under four different granularity settings in order to capture the landscape features present at different parts of the search space.

The results presented in this chapter indicated that an increase in problem dimensionality yielded an increase in indefinite, or flat curvature, as previously observed by Sagun et al. [114]. The flat curvature was especially prevalent further away from the origin for all problems considered. An increase in the number of hidden layers yielded a more rapid increase in flatness than an increase in the hidden layer size. This behaviour was attributed to the inter-variable dependency between the hidden layers in a feed-forward architecture.

For the XOR problem, an increase in the number of hidden neurons was shown to reduce the number of local minima. For the same problem, an increase in the number of hidden layers was shown to reduce the convexity and increase the amount of saddle curvature. However, an increase in the number of hidden layers with a fixed hidden layer size had no effect on the total number of stationary attractors. Thus, an increase in the size of the hidden layer was shown to change the shape of the attractor in a more meaningful way than the addition of extra hidden layers. This observation correlates with the theoretical studies in [58], where deep and "skinny" NNs were shown to not exhibit the universal approximator properties.

For most problems, a single major attractor at the global minimum was observed. An increase in the hidden layer size, as well as an increase in the number of hidden layers, yielded an increase in the width and steepness of the observed attractor. Most problems exhibited a split into two clusters of steep and shallow gradients. The clusters were attributed to the narrow and wide valleys exhibited by the NN loss surfaces. An increase in the hidden layer size was shown to increase the overlap between the two clusters, up to a complete merge of the two clusters into a single cluster. An increase in the number of hidden layers did not exhibit the same effect. Instead, the steep cluster generally became heavier as more hidden layers were added, up to a complete disappearance of the shallow cluster. Thus, an increase in the number of hidden layers was shown to exaggerate the narrow valleys as a landscape feature.

In general, both an increase in the hidden layer size, as well as an increase in the number of hidden layers, were shown to improve the searchability of the resulting error landscapes. Quicker convergence to an attraction basin was often observed as a result of increased dimensionality. However, larger hidden layers were shown to be more instrumental in the overall improvement of the landscape structure. Sensitivity to the step size and the initialisation range was reduced for larger hidden layers. Superior classification quality was also associated with larger hidden layer sizes.

The steep gradient cluster was associated with inferior generalisation performance for most problems considered. The increased searchability due to the additional hidden layers/hidden neurons often resulted in better exploitation of the global minimum, but worse generalisation performance.

Thus, an increase in the problem dimensionality was shown to yield a more searchable and more exploitable error landscape. An increase in the hidden layer size was shown to effectively reduce the number of local minima, and simplify the shape of the global attractor. An increase in the number of hidden layers was shown to sharpen the attractor, thus making it more exploitable.

The next chapter concludes the thesis by summarising the obtained results, and discusses potential topics for future research.

Chapter 10

Conclusions

This chapter concludes the thesis by summarising the main findings and contributions in line with the research objectives (Section 10.1), and proposing some topics for future research (Section 10.2).

10.1 Summary of Conclusions

The main objective of this work was to study NN loss surfaces using FLA techniques, and to determine the relationship between various NN hyperparameters and the resulting landscape features. NNs and the associated hyperparameters that influence the error landscapes were discussed in Chapter 2. In order to apply FLA to NNs, a survey of the existing FLA methods was performed in Chapter 3. It was determined that the sampling methods typically used to perform FLA are defined for bounded spaces. Thus, the first sub-objective of this thesis was to investigate the sensitivity of the FLA metrics to the chosen search space boundaries, and to determine whether NN loss surfaces exhibit different characteristics in different subspaces.

The FLA sensitivity to the search space boundaries was studied in Chapter 4. All FLA metrics used in this study exhibited a sensitivity to the boundaries chosen. NN loss surfaces exhibited high gradient values on both small and large search subspaces, indicating that steep gradients constitute an inherent NN error landscape property. Gradient magnitudes increased with an increase in problem dimensionality. An increase in

search space boundaries increased the variance of gradients, indicating that the step-like transitions between plateaus become increasingly drastic further away from the origin. NN weights with absolute values within the $[0.1, 1]$ interval were shown to capture information potentially useful to an optimisation algorithm. Asymmetric regions of the search space were shown to be less searchable than the symmetric regions.

The second sub-objective of the thesis was to establish FLA as a viable method for NN error landscape analysis. For this purpose, a selection of existing FLA techniques were used to study the influence of a regularisation term on the error landscape in Chapter 5. Weight elimination was considered for the purpose of this study. The weight elimination term was shown to smooth the error landscape while introducing additional minima. The backpropagation algorithm was shown to perform robustly on very rugged landscapes. However, step-like landscapes with rare and sudden fitness changes affected the backpropagation performance negatively. Additionally, optimisation ranges for the w_0 and λ regularisation parameters were suggested.

The third sub-objective of the thesis was to identify weaknesses in the existing FLA techniques, and to propose new FLA metrics and algorithms accordingly. Based on the studies performed in Chapters 3, 4, and 5, the following three main weaknesses of the existing techniques were identified:

1. The existing metrics do not provide a means to quantify the modality of the NN error landscapes. Existing modality metrics rely on the Euclidean distance calculations, which was shown to be unreliable in high-dimensional spaces [95].

2. Sampling algorithms such as the random walk and the progressive random walk provide no guarantees of discovering high fitness solutions, and thus cannot be employed for the purpose of modality quantification.

3. The existing adaptive sampling algorithms for continuous spaces use population-based mechanics to perform hill climbing, which is inefficient in high-dimensional spaces.

To combat these weaknesses, the following methods were proposed in Chapter 6:

1. The progressive gradient walk was proposed as an efficient alternative to population-based hill climbing. The progressive gradient walk uses the gradient information

to determine the direction of each step. The magnitude of the step is randomised per dimension within the given bounds, thus adding stochasticity and preventing convergence.

2. An intuitive visualisation of the local minima and the associated basins of attraction was proposed, namely *loss-gradient clouds*. The loss-gradient clouds are constructed by plotting the loss values against the corresponding gradient vector magnitudes as sampled by the progressive gradient walk. Points of zero gradient are identified as the stationary points. To further classify the stationary points as minima, maxima, or saddles, Hessian matrix information is used to identify the curvature of each sampled point.

3. Two simple metrics to quantify the number and extent of attraction basins as sampled by the progressive gradient walks were proposed. The proposed metrics quantify the connectedness of the various basins, as well as the likelihood of escape from the discovered basins.

The second main objective was to study the NN loss surfaces with FLA techniques under various hyperparameter settings, namely loss functions, activation functions, and NN architectures. Chapter 7 presented a study of the loss surfaces associated with the quadratic (SSE) and entropic (CE) loss functions. In Chapter 8, FLA was applied to NN problems with various combinations of the sigmoid, TanH, ReLU, ELU, and softmax activation functions employed in the hidden and output layers. Chapter 9 concluded the study by presenting an analysis of various NN architectures in the FLA context. The main findings of these three chapters are summarised as follows:

1. Loss functions study, Chapter 7:

   - Both SSE and CE were shown to exhibit convex local minima for the XOR problem. The majority of the classification problems considered exhibited a single main attractor around the global optimum, indicating that no local minima was detected.
   - Saddle curvature was the most prevalent curvature observed, and some of the higher-dimensional problems exhibited only saddle curvature for all sampled

points.

- SSE was shown to consistently exhibit more local stationary points and associated attractors than CE.
- CE exhibited a more consistent and searchable landscape structure with steeper gradients for all problems considered.
- SSE was shown to yield superior generalisation performance in some cases.
- The presence of valley-shaped optima in NN error landscapes was confirmed. The FLA results suggested that the discovered valleys became wider further away from the origin. For the majority of the problems, descending into a valley was easily accomplished by the gradient walks. Travelling down the bottom of the valley towards a global minimum yielded a decrease in the generalisation performance for both SSE and CE.
- The sampled points were often split into two major clusters: points of low error and high gradients, and points of higher error and low gradients. These are hypothesised to represent narrow and wide valleys, respectively. Superior generalisation performance was exhibited by the points in the wide valleys.

2. Activation functions study, Chapter 8:

- The choice of activation function did not have an effect on the total number of attractors in the search space, but affected the properties of the discovered basins of attraction.
- ReLU was shown to exhibit the most convexity out of all the activation functions considered, and ELU was shown to exhibit the least flatness.
- The stationary points exhibited by ReLU and ELU were generally more connected than those exhibited by TanH, indicating that ReLU and ELU yielded more searchable landscapes. However, ReLU and ELU were shown to exhibit stronger sensitivity to the step size and the initialisation range than TanH.
- All activation functions exhibited a split into two clusters of steep and shallow gradients, associated with narrow and wide valleys. Narrow valleys were associated with saturated neurons and embedded regularised minima.

- The ELU activation function exhibited superior generalisation properties compared to the other hidden neuron activation functions.
- Softmax in the output layer reduced the convexity of the problem by introducing a higher degree of non-separability. Only the problems of higher dimensionality and complexity were shown to benefit from the softmax activation function.

3. NN architectures study, Chapter 9:

- An increase in problem dimensionality yielded an increase in indefinite, or flat curvature. The flat curvature was especially prevalent further away from the origin for all problems considered.
- An increase in the number of hidden layers yielded a more rapid increase in flatness than an increase in the hidden layer size.
- Despite the flatness, an increase in the problem dimensionality was shown to yield a more searchable and more exploitable error landscape.
- An increase in the hidden layer size was shown to effectively reduce the number of local minima, and to simplify the shape of the global attractor by widening the attraction basin.
- An increase in the number of hidden layers was shown to sharpen the global attractor, thus making it more exploitable.
- An increase in the number of hidden layers had no effect on the total number of stationary attractors, indicating that depth without width does not guarantee a good final solution.
- An increase in the number of hidden layers was shown to exaggerate the steep gradient cluster, associated with inferior generalisation performance for most problems considered. Thus, deeper architectures promoted narrow valley structures.

Overall, the FLA techniques employed in this thesis successfully confirmed the existing theoretical insights, and provided intuitive, visual means of improving and refining the current understanding of NN error landscapes.

10.2 Future Work

This thesis yielded a number of interesting research directions that can be explored in future. The proposed research topics are discussed below.

10.2.1 Loss surfaces of neural networks for regression

This study was limited to classification problems only. It would be interesting to perform a similar analysis of NNs for regression tasks, and determine whether regression problems yield different landscape properties. Regression problems require higher precision than classification problems, and may therefore exhibit more varied landscape features. It would be especially interesting to determine whether regression problems exhibit the same lack of local minima as the classification problems.

10.2.2 Gravitational search

Across all experiments conducted in this study, the samples initialised and/or confined to a smaller range around the origin typically exhibited superior performance. From the FLA perspective, wider regions of the search space were classified as highly rugged, with extremely steep gradients and little information to guide a training algorithm. This observation not only highlights the importance of the weight initialisation range for the success of a training algorithm, but also indicates that a search algorithm that gravitates towards the origin may prove to be a viable search strategy. The design and testing of such an algorithm is proposed as further research.

10.2.3 Fitness landscape analysis for regularisation

The regularisation study presented in this thesis was limited to the weight elimination approach. It will be interesting to compare weight elimination error landscapes to other regularised error landscapes, such as weight decay and dropout. FLA can potentially be used to optimise the regularisation parameters involved, since FLA metrics provide a handy visualisation tool for the corresponding error landscapes.

10.2.4 Multi-objective regularisation

The regularisation coefficient λ was shown to have a significant effect on the resulting error landscape. The necessity to optimise λ can be eliminated altogether by employing a multi-objective algorithm to optimise both the loss function and the regularisation term separately, and to find a suitable trade-off solution thereof.

10.2.5 Exploitative population-based search

The presence of a single global attractor in the majority of the problems considered suggests that an exploitative rather than an exploratory approach should be taken for the purpose of NN training. This observation has strong implications for population-based training algorithms, which so far failed to be effectively applied to high-dimensional NN training problems. A population-based approach designed with exploitation rather than exploration in mind may perform competitively, especially if gradient information is used as one of the population guides. This hypothesis is further supported by a recent study of PSO in high-dimensional spaces [102], were the efficacy of exploitation over exploration in high-dimensional spaces was observed. Investigation of exploitative population-based techniques applied to NNs is an interesting topic for future research.

10.2.6 Stochastic search

The impressive ability of the randomised progressive gradient walks to find the global optimum suggests that stochastic gradient-guided approaches can be considered for the purpose of NN training. A hybrid algorithm can be developed that switches between the classic gradient descent and the stochastic gradient-guided search based on the magnitude of the gradient, or some other measure of stagnation.

10.2.7 New loss functions

The observation that the SSE landscape may have superior generalisation properties suggests that a hybrid of SSE and CE may produce a landscape that combines the searchability of CE with the robustness of SSE. FLA properties of such a loss function

can be evaluated using the techniques employed in this study.

10.2.8 Properties of the wide and narrow valleys

The results of this study confirmed previously made observations of the presence of valley-shaped optima in NN error landscapes. Further, the presence of wide and narrow valleys was established. Narrow valleys were associated with saturated neurons and embedded local minima. However, no explicit differentiation between the saturated points and the embedded minima was made. Further studies can be performed to uncover the properties of the narrow valleys associated with embedded minima. Identification of the embedded minima in the narrow valleys may be used to design new training algorithms biased towards the discovery of minimal architectures within the over-parametrised search spaces.

10.2.9 Other neural network architectures

This study was limited to standard, fully connected feed-forward architectures. FLA analysis of more complex modern NN architectures, such as convolutional NNs, recurrent NNs, and NNs with residual connections, can be carried out. Larger, more realistic datasets than the ones used in this study should also be considered to confirm the various observations and conclusions made in this thesis.

10.2.10 Loss-gradient cloud analysis of continuous optimisation problems

The loss-gradient cloud visualisation proposed in this study was designed for the analysis of NN loss surfaces. However, the same principle can be successfully applied to other continuous optimisation problems. When numerical gradient is unavailable, an estimate of the gradient can be used. The applicability of the loss-gradient clouds to other optimisation domains is an interesting topic for future research.

10.2.11 Improved modality metrics

This study did not attempt to quantify the number of optima. An improved modality metric can be designed to not only quantify the presence or absence of the stationary points, but also to map the relationship between the various attractors. Local optima networks (LONs), originally designed for combinatorial problems [101], can be adapted for NN problems. The stationary points discovered by the progressive gradient walks can be mapped onto a LON structure, and LON techniques can be further used to gain more insight into the structure of the NN error landscapes.

10.2.12 Discrete event stochastic simulation metamodeling for FLA

This study employed FLA methods to analyse the features of the loss surfaces. An alternative approach would be to employ discrete event stochastic simulation metamodeling [66, 151] for the purpose of understanding the search space. Discrete event simulation is typically used in the engineering applications when the system output is not directly observable. A metamodel is constructed in order to simplify the system that cannot be feasibly sampled. Therefore, a metamodel can potentially be created to capture the features of a NN loss surface in a lower-dimensional space.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from https://www.tensorflow.org.

[2] L. Altenberg. The evolution of evolvability in genetic programming. In K. E. Kinnear, editor, *Advances in Genetic Programming*, chapter 3, pages 47–74. MIT Press, Cambridge, Massachusetts, 1994.

[3] K. Alyahya and J. E. Rowe. Simple random sampling estimation of the number of local optima. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*, pages 932–941, Edinburgh, Scotland, 2016. Springer.

[4] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee. Understanding deep neural networks with rectified linear units. In *Proceedings of the International Conference on Learning Representations*, pages 1–17, Vancouver, Canada, 2018.

[5] P. Auer, M. Herbster, and M. K. Warmuth. Exponentially many local minima for single neurons. In *Advances in Neural Information Processing Systems*, pages 316–322, Denver, Colorado, United States, 1996.

[6] A. J. Ballard, R. Das, S. Martiniani, D. Mehta, L. Sagun, J. D. Stevenson, and D. J. Wales. Energy landscapes for machine learning. *Physical Chemistry Chemical Physics*, 19(20):12585–12603, 2017.

[7] D. S. Bassett and M. S. Gazzaniga. Understanding complexity in the human brain. *Trends in Cognitive Sciences*, 15(5):200–209, 2011.

[8] P. G. Benardos and G.-C. Vosniakos. Optimizing feedforward artificial neural network architecture. *Engineering Applications of Artificial Intelligence*, 20(3):365–382, 2007.

[9] K. P. Bennett and O. L. Mangasarian. Neural network training via linear programming. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 1990.

[10] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK, 1995.

[11] Y. Borenstein and R. Poli. Information landscapes. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, pages 1515–1522, Washington DC, USA, 25-29 June 2005. ACM Press.

[12] A. S. Bosman, A. P. Engelbrecht, and M. Helbig. Search space boundaries in neural network error landscape analysis. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pages 1–8, Piscataway, USA, 2016. IEEE.

[13] A. S. Bosman, A. P. Engelbrecht, and M. Helbig. Fitness landscape analysis of weight-elimination neural networks. *Neural Processing Letters*, 48(1):353–373, 2018.

[14] A. S. Bosman, A. P. Engelbrecht, and M. Helbig. Progressive gradient walk for neural network fitness landscape analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1473–1480. ACM, 2018.

[15] A. S. Bosman, A. P. Engelbrecht, and M. Helbig. Visualising basins of attraction for the cross-entropy and the squared error neural network loss functions. *arXiv e-prints*, page arXiv:1901.02302, January 2019.

[16] H. Bourlard and N. Morgan. *Connectionist speech recognition: a hybrid approach*, volume 247 of *The Kluwer international series in engineering and computer science*. Boston: Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[17] J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. F. Soulié and J. Hérault, editors, *Neurocomputing*, pages 227–236, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[18] P. Caamaño, F. Bellas, J. A. Becerra, and R. J. Duro. Evolutionary algorithm characterization in real parameter optimization problems. *Applied Soft Computing*, 13(4):1902–1921, 2013.

[19] P. Caamaño, A. Prieto, J. A. Becerra, F. Bellas, and R. J. Duro. Real-valued multimodal fitness landscape characterization for evolution. In *Proceedings of the International Conference on Neural Information Processing*, pages 567–574. Springer, 2010.

[20] R. Caruana, S. Lawrence, and C. L. Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems*, pages 402–408, 2001.

[21] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina. Entropy-SGD: Biasing gradient descent into wide valleys. In *Proceedings of The International Conference on Learning Representations*, pages 1–19, 2017.

[22] A. M. Chen, H.-m. Lu, and R. Hecht-Nielsen. On the geometry of feedforward neural network error surfaces. *Neural computation*, 5(6):910–927, 1993.

[23] A. Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 192–204, 2015.

[24] A. Choromanska, Y. LeCun, and G. B. Arous. Open problem: The landscape of the loss surfaces of multilayer networks. In *Proceedings of The 28th Conference on Learning Theory*, pages 1756–1760, 2015.

[25] W. S. Cleveland and S. J. Devlin. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American statistical association*, 83(403):596–610, 1988.

[26] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *Proceedings of the International Conference on Learning Representations*, pages 1–14, 2016.

[27] A. Czarn, C. MacNish, K. Vijayan, and B. Turlach. The detrimentality of crossover. In *Australasian Joint Conference on Artificial Intelligence*, pages 632–636, Berlin, Heidelberg, 2007. Springer, Springer Berlin Heidelberg.

[28] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, pages 2933–2941, 2014.

[29] Y. Davidor. Epistasis variance: Suitability of a representation to genetic algorithms. *Complex Systems*, 4(4):369–383, 1990.

[30] L. Deng, G. Hinton, and B. Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *Proceedigs of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603, Piscataway, USA, 2013. IEEE.

[31] J. Denker, D. Schwartz, B. Wittner, S. Solla, R. Howard, L. Jackel, and J. Hopfield. Large automatic learning, rule extraction, and generalization. *Complex systems*, 1(5):877–922, 1987.

[32] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028, 2017.

[33] D. L. Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. *AMS math challenges lecture*, pages 1–33, 2000.

[34] F. Draxler, K. Veschgini, M. Salmhofer, and F. Hamprecht. Essentially no barriers in neural network energy landscape. In J. Dy and A. Krause, editors, *Proceedings of the International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1309–1318, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[35] G. Dreyfus. *Neural networks: methodology and applications*. Springer, Berlin, Germany, 2005.

[36] C. H. Edwards. *Advanced calculus of several variables*. Academic Press, New York, USA, 1973.

[37] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[38] M. Gallagher. Fitness distance correlation of neural network error surfaces: A scalable, continuous optimization problem. In *Proceedings of the 12th European Conference on Machine Learning*, pages 157–166. Springer-Verlag, 2001.

[39] M. R. Gallagher. *Multi-layer perceptron error surfaces: Visualization, structure and modelling*. PhD thesis, University of Queensland, St Lucia 4072, Australia, 2000.

[40] M. R. Gallagher and T. Downs. Visualization of learning in multilayer perceptron networks using principal component analysis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 33(1):28–34, 2003.

[41] J. Garnier and L. Kallel. How to detect all maxima of a function. In *Theoretical aspects of evolutionary computing*, pages 343–370. Springer, 2001.

[42] J. Garnier and L. Kallel. Efficiency of local search with multiple local optima. *SIAM Journal on Discrete Mathematics*, 15(1):122–141, 2001.

[43] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural computation*, 7(2):219–269, 1995.

[44] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[45] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[46] P. Golik, P. Doetsch, and H. Ney. Cross-entropy vs. squared error training: a theoretical and experimental comparison. In *Interspeech*, volume 13, pages 1756–1760, 2013.

[47] G. Grinstein, M. Trutschl, and U. Cvek. High-dimensional visualizations. In *Proceedings of the Visual Data Mining Workshop*. Citeseer, 2001.

[48] L. G. C. Hamey. XOR has no local minima: A case study in neural network error surface analysis. *Neural Networks*, 11(4):669–681, 1998.

[49] L. D. Harmon. Studies with artificial neurons, I: Properties and functions of an artificial neuron. *Biological Cybernetics*, 1(3):89–101, 1961.

[50] G. E. Hinton. Learning translation invariant recognition in a massively parallel networks. In *Proceedings of the Parallel Architectures and Languages Europe*, pages 1–13. Springer, 1987.

[51] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[52] S. Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991.

[53] T. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[54] G.-B. Huang. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks*, 14(2):274–281, 2003.

[55] G.-B. Huang, L. Chen, and C. K. Siew. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 17(4):879–892, 2006.

[56] J. Humeau, A. Liefooghe, E.-G. Talbi, and S. Verel. Paradiseo-mo: From fitness landscape analysis to efficient local search algorithms. *Journal of Heuristics*, 19(6):881–915, 2013.

[57] D. R. Hush, B. Horne, and J. M. Salas. Error surfaces for multilayer perceptrons. *IEEE Transactions on Systems, Man and Cybernetics*, 22(5):1152–1161, October 1992.

[58] J. Johnson. Deep, skinny neural networks are not universal approximators. In *Proceedings of the International Conference on Learning Representations*, 2019.

[59] T. Jones. *Evolutionary algorithms, fitness landscapes and search.* PhD thesis, The University of New Mexico, 1995.

[60] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann Publishers Inc., 1995.

[61] L. Kallel, B. Naudts, and C. R. Reeves. Properties of fitness functions and search landscapes. In *Theoretical aspects of evolutionary computing*, pages 175–206. Springer, 2001.

[62] S. Kauffman and S. Levin. Towards a general theory of adaptive walks on rugged landscapes. *Journal of Theoretical Biology*, 128(1):11–45, 1987.

[63] K. Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pages 586–594, 2016.

[64] K. Kawaguchi, L. P. Kaelbling, and Y. Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

[65] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *Proceedings of the International Conference for Learning Representations*, 2017.

[66] R. A. Kilmer, A. E. Smith, and L. J. Shuman. Neural networks as a metamodeling technique for discrete event stochastic simulation. *Intelligent Engineering Systems Through Artificial Neural Networks*, 4(1), 1994.

[67] D. M. Kline and V. L. Berardi. Revisiting squared-error and cross-entropy functions for training neural network classifiers. *Neural Computing & Applications*, 14(4):310–318, 2005.

[68] M. Kordos and W. Duch. A survey of factors influencing MLP error surface. *Control and Cybernetics*, 33(4):611–631, 2004.

[69] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[70] A. S. Lapedes and R. M. Farber. How neural nets work. In *Advances in Neural Information Processing Systems*, pages 442–456, 1988.

[71] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[72] Y. LeCun, C. Cortes, and C. Burges. MNIST handwritten digit database. *AT&T Labs [Online]. Available:* http://yann.lecun.com/exdb/mnist, 2010.

[73] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[74] H. Lu and K. Kawaguchi. Depth creates no bad local minima. *arXiv e-prints*, page arXiv:1702.08580, February 2017.

[75] K. M. Malan and A. P. Engelbrecht. Quantifying ruggedness of continuous landscapes using entropy. In *IEEE Congress on Evolutionary Computation*, pages 1440–1447. IEEE, 2009.

[76] K. M. Malan and A. P. Engelbrecht. Steep gradients as a predictor of pso failure. In *Proceedings of the Conference Companion on Genetic and Evolutionary Computation*, pages 9–10. ACM, 2013.

[77] K. M. Malan and A. P. Engelbrecht. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148–163, 2013.

[78] K. M. Malan and A. P. Engelbrecht. Characterising the searchability of continuous optimisation problems for PSO. *Swarm Intelligence*, 8(4):275–302, 2014.

[79] K. M. Malan and A. P. Engelbrecht. A progressive random walk algorithm for sampling continuous fitness landscapes. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2507–2514. IEEE, 2014.

[80] K. M. Malan and A. P. Engelbrecht. Ruggedness, funnels and gradients in fitness landscapes and the effect on PSO performance. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 963–970. IEEE, 2013.

[81] K. M. Malan. *Characterising continuous optimisation problems for particle swarm optimisation performance prediction*. PhD thesis, University of Pretoria, 2014.

[82] G. Martin and J. A. Pittman. Recognizing hand-printed letters and digits. In *Advances in Neural Information Processing Systems*, pages 405–414, 1990.

[83] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[84] R. McGill, J. W. Tukey, and W. A. Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, 1978.

[85] S. McLoone and G. Irwin. Improving neural network training solutions using regularisation. *Neurocomputing*, 37(14):71–90, 2001.

[86] D. Mehta, X. Zhao, E. A. Bernal, and D. J. Wales. Loss surface of XOR artificial neural networks. *Physical Review E*, 97(5):052307, 2018.

[87] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 829–836. ACM, 2011.

[88] P. Merz and B. Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE Transactions on Evolutionary Computation*, 4(4):337–352, 2000.

[89] T. Milne. Piecewise strong convexity of neural networks. *arXiv e-prints*, page arXiv:1810.12805, October 2018.

[90] M. Minsky and S. A. Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 1969.

[91] J. Moody, S. J. Hanson, and R. P. Lippmann. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. *Advances in neural information processing systems*, 4:847–854, 1992.

[92] J. E. Moody, S. J. Hanson, A. Krogh, and J. A. Hertz. A simple weight decay can improve generalization. *Advances in Neural Information Processing Systems*, 4:950–957, 1995.

[93] N. Morgan and H. Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. In *Advances in Neural Information Processing Systems*, pages 630–637, 1990.

[94] R. Morgan and M. Gallagher. Length scale for characterising continuous optimization problems. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*, pages 407–416. Springer, 2012.

[95] R. Morgan and M. Gallagher. Sampling techniques and distance metrics in high dimensional continuous landscape analysis: Limitations and improvements. *IEEE Transactions on Evolutionary Computation*, 18(3):456–461, 2014.

[96] I. Moser, M. Gheorghita, and A. Aleti. Identifying features of fitness landscapes and relating them to problem difficulty. *Evolutionary computation*, 25(3):407–437, 2017.

[97] M. A. Muñoz, Y. Sun, M. Kirley, and S. K. Halgamuge. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences*, 317:224–245, 2015.

[98] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th international conference on machine learning*, pages 807–814, 2010.

[99] B. Naudts and J. Naudts. The effect of spin-flip symmetry on the performance of the simple ga. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*, pages 67–76. Springer, 1998.

[100] Q. Nguyen and M. Hein. The loss surface of deep and wide neural networks. *arXiv e-prints*, page arXiv:1704.08045, April 2017.

[101] G. Ochoa, M. Tomassini, S. Vérel, and C. Darabos. A study of NK landscapes' basins and local optima networks. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, pages 555–562, New York, NY, USA, 2008. ACM.

[102] E. T. Oldewage. The perils of particle swarm optimization in high dimensional problem spaces. Master's thesis, University of Pretoria, 2018.

[103] T. E. Oliphant. A guide to numpy, 2006. Software available from http://www.numpy.org.

[104] K. Pearson. The problem of the random walk. *Nature*, 72(1867):342, 1905.

[105] E. Pitzer and M. Affenzeller. A comprehensive survey on fitness landscape analysis. In *Recent Advances in Intelligent Engineering Systems*, pages 161–191. Springer, 2012.

[106] L. Prechelt. Proben1 – a set of neural network benchmark problems and benchmarking rules. Technical report, Universität Karlsruhe, Karlsruhe, Germany, September 1994.

[107] A. Rakitianskaia and A. Engelbrecht. Weight regularisation in particle swarm optimisation neural network training. In *Proceedings of the IEEE Symposium on Swarm Intelligence*, pages 1–8, Florida, USA, 2014. IEEE.

[108] A. Rakitianskaia and A. Engelbrecht. Measuring saturation in neural networks. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pages 1423–1430. IEEE, 2015.

[109] A. Rakitianskaia and A. Engelbrecht. Saturation in PSO neural network training: Good or evil? In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 125–132, Sendai, Japan, 2015. IEEE.

[110] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti. Detecting novel associations in large data sets. *Science*, 334(6062):1518–1524, 2011.

[111] B. Rister and D. L. Rubin. Piecewise convexity of artificial neural networks. *Neural Networks*, 94:34 – 45, 2017.

[112] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[113] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, California University, San Diego La Jolla Institute for Cognitive Science, 1985.

[114] L. Sagun, U. Evci, V. U. Guney, Y. Dauphin, and L. Bottou. Empirical analysis of the hessian of over-parametrized neural networks. In *Proceedings of the International Conference on Learning Representations*, pages 1–15, 2018.

[115] L. Sagun, V. U. Guney, G. B. Arous, and Y. LeCun. Explorations on high dimensional landscapes. In *Proceedings of the International Conference on Learning Representations*, pages 1–11, 2015.

[116] H. Sak, A. Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*, pages 338–342, 2014.

[117] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[118] H. Shen. Towards a mathematical understanding of the difficulty in learning with feedforward neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 811–820, 2018.

[119] T. Smith, P. Husbands, and M. O'Shea. Not measuring evolvability: Initial investigation of an evolutionary robotics search space. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 9–16. IEEE, 2001.

[120] S. Solla, E. Levin, and M. Fleisher. Accelerated learning in layered neural networks. *Complex systems*, 2:625–640, 1988.

[121] D. Soudry and Y. Carmon. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016.

[122] F. Spitzer. *Principles of random walk*, volume 34. Springer Science & Business Media, 2013.

[123] I. G. Sprinkhuizen-Kuyper and E. J. Boers. The local minima of the error surface of the 2-2-1 XOR network. *Annals of Mathematics and Artificial Intelligence*, 25(1-2):107, 1999.

[124] I. G. Sprinkhuizen-Kuyper and E. J. Boers. A local minimum for the 2-3-1 XOR network. *IEEE Transactions on Neural Networks*, 10(4):968–971, 1999.

[125] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[126] P. F. Stadler. Fitness landscapes. In *Biological evolution and statistical physics*, pages 183–204. Springer, 2002.

[127] Y. Sun, S. K. Halgamuge, M. Kirley, and M. A. Munoz. On the selection of fitness landscape analysis metrics for continuous optimization problems. In *Proceedings of the IEEE International Conference on Information and Automation for Sustainability*, pages 1–6. IEEE, 2014.

[128] Y. Sun, M. Kirley, and S. K. Halgamuge. Quantifying variable interactions in continuous optimization problems. *IEEE Transactions on Evolutionary Computation*, 21(2):249–264, 2017.

[129] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[130] G. Swirszcz, W. M. Czarnecki, and R. Pascanu. Local minima in training of neural networks. *arXiv e-prints*, page arXiv:1611.06310, November 2016.

[131] T. Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the International Conference on Machine learning*, pages 1064–1071. ACM, 2008.

[132] J. W. Tukey. *Exploratory data analysis*, volume 2. Reading, Mass., 1977.

[133] P. D. Turney. Increasing evolvability considered as a large-scale trend in evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference Workshop Program*, pages 43–46. ACM, 1999.

[134] W. A. van Aardt, A. S. Bosman, and K. M. Malan. Characterising neutrality in neural network error landscapes. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1374–1381. IEEE, 2017.

[135] F. Van Den Bergh and A. P. Engelbrecht. Cooperative learning in neural networks using particle swarm optimizers. *South African Computer Journal*, 26:84–90, 2000.

[136] A. B. Van Wyk and A. P. Engelbrecht. Overfitting by PSO trained feedforward neural networks. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.

[137] V. K. Vassilev, T. C. Fogarty, and J. F. Miller. Information characteristics and the structure of landscapes. *Evolutionary computation*, 8(1):31–60, 2000.

[138] J. Wang, Z. Ye, W. Gao, and J. M. Zurada. Boundedness and convergence analysis of weight elimination for cyclic training of neural networks. *Neural Networks*, 82:49–61, 2016.

[139] E. J. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85(411):664–675, 1990.

[140] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman. Generalization by weight elimination with application to forecasting. *Advances in Neural information processings systems*, 3, 1991.

[141] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman. Generalization by weight-elimination applied to currency exchange rate prediction. In *Proceedings of the International Joint Conference on Neural Networks*, volume 1, pages 837–841, Seattle, 1991. IEEE.

[142] E. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological cybernetics*, 63(5):325–336, 1990.

[143] P. J. Werbos. *Beyond regression: New tools for prediction and analysis in the behavioural sciences*. PhD thesis, Harvard University, Boston, USA, 1974.

[144] D. Whitley, S. Rana, J. Dzubera, and K. E. Mathias. Evaluating evolutionary algorithms. *Artificial intelligence*, 85(1-2):245–276, 1996.

[145] H. Wickham. ggplot2: Elegant graphics for data analysis, 2016. Software available from https://ggplot2.tidyverse.org.

[146] B. M. Wilamowski. Neural network architectures and learning algorithms. *IEEE Industrial Electronics Magazine*, 3(4), 2009.

[147] P. R. Winters. Forecasting sales by exponentially weighted moving averages. *Management science*, 6(3):324–342, 1960.

[148] S. Wright. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In *Proceedings of the International Congress on Genetics*, pages 356–366, 1932.

[149] C. Xing, D. Arpit, C. Tsirigotis, and Y. Bengio. A walk with sgd. *arXiv e-prints*, page arXiv:1802.08770, February 2018.

[150] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[151] B. Yu and K. Popplewell. Metamodels in manufacturing: a review. *The International Journal of Production Research*, 32(4):787–796, 1994.

[152] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, et al. On rectified linear units for speech processing. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3517–3521. IEEE, 2013.

Appendix A

Benchmark problems

A selection of well-known real world classification problems of varied dimensionality were used in this study. Table A.1 summarises the minimal NN architecture parameters used for each dataset, as well as the total dimensionality of the weight space. The specified sources point to publications from which each dataset and/or NN architectures were adopted.

Table A.1: Benchmark Problems

Problem	Input	Hidden	Output	Dimensionality	Source
XOR	2	2	1	9	[48]
Iris	4	4	3	35	[37]
Diabetes	8	8	1	81	[106]
Glass	9	9	6	150	[106]
Cancer	30	10	1	321	[106]
Heart	32	10	1	341	[106]
MNIST	784	10	10	7960	[72]

The properties of each dataset are briefly discussed below:

1. **XOR:** XOR (exclusive-or) is a simple, but linearly non-separable problem that can be solved by a feedforward NN with at least two hidden neurons. As such, XOR

is often used to analyse basic properties of artificial neural networks. The dataset consists of 4 binary patterns.

2. **Iris:** The famous Iris flower data set [37] contains 50 specimens from each of the three species of iris flowers, i.e. *Iris Setosa*, *Iris Versicolor*, and *Iris Virginica*. There are 150 patterns in the dataset. The iris data set, even though relatively low-dimensional and simple, is not altogether trivial, as two of the three output classes significantly overlap across two of the four input variables, and two inputs have low correlation with the class labels [47].

3. **Diabetes:** The diabetes dataset [106] captures personal data of 768 Pima Indian patients, classified as diabetes positive or diabetes negative. All inputs are continuous, and 65.1% of the examples are diabetes negative. The data set contains noise [106].

4. **Glass:** The glass dataset [106] captures chemical components of glass shards. Each glass shard belongs to one of six classes: float processed or non-float processed building windows, vehicle windows, containers, tableware, or head lamps. There are 214 patterns in the dataset, all inputs are continuous, and two of the inputs have very low correlation with the class labels. The frequency of the 6 classes are 70, 76, 17, 13, 9, and 29 instances, respectively.

5. **Cancer:** The breast cancer Wisconsin (diagnostic) dataset [106] consists of 699 patterns, each containing tumor descriptors, and a binary classification into benign or malignant.

6. **Heart:** The heart disease prediction dataset [106] contains 920 patterns, each describing various patient descriptors. The goal is to correctly predict whether at least one of four major vessels is reduced in diameter by more than 50%.

7. **MNIST:** The MNIST dataset of handwritten digits [72] contains 70,000 examples of grey scale handwritten digits from 0 to 9, where each digit is stored as a 28×28 grey scale image. For the purpose of this study, the 2-dimensional input is treated as a 1-dimensional vector.

Input values for all problems except XOR have been standardised by subtracting the mean per input dimension, and scaling every input variable to unit variance. All outputs were binary encoded for problems with two output classes, and one-hot binary encoded for problems with more than two output classes.

Appendix B

Classification Accuracy

The average classification accuracy values arrived at by the gradient walks are reported in this appendix. Averages are calculated across the accuracy values as observed at the last step of each walk. The classification accuracy of the training set is referred to as C_t, and the classification accuracy of the test set is referred to as C_g. Tables B.1, B.2, B.3, B.4, B.5, and B.6 list the average C_t and C_g values obtained for the SSE and CE losses for the Iris, Diabetes, Glass, Cancer, Heart, and MNIST problems, respectively. Tables B.7, B.8, B.9, B.10, B.11, and B.12 list the average C_t and C_g values obtained for the various activation functions for the Iris, Diabetes, Glass, Cancer, Heart, and MNIST problems, respectively. Tables B.13, B.14, and B.15 list the average C_t and C_g values obtained using the softmax output activation function for the Iris, Glass, and MNIST problems, respectively. Tables B.16, B.17, B.18, B.19, B.20, and B.21 list the average C_t and C_g values obtained for the various NN architectures for the Iris, Diabetes, Glass, Cancer, Heart, and MNIST problems, respectively. Standard deviation is shown in parenthesis.

Table B.1: Iris, classification accuracy.

	micro				macro			
	SSE		CE		SSE		CE	
	C_t	C_g	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$	1.00000	0.91819	0.98352	0.93333	0.96638	1.00000	0.97557	0.96667
	(0.00000)	(0.01660)	(0.00125)	(0.00000)	(0.00881)	(0.00000)	(0.00646)	(0.00000)
$[-10, 10]$	0.97252	0.97105	0.99245	0.90581	0.92155	0.92829	0.92857	0.92457
	(0.07622)	(0.08790)	(0.00734)	(0.05097)	(0.09578)	(0.10521)	(0.05844)	(0.05806)

Table B.2: Diabetes, classification accuracy.

	micro				macro			
	SSE		CE		SSE		CE	
	C_t	C_g	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$	0.91094	0.66913	0.85453	0.73725	0.81141	0.73586	0.81187	0.74165
	(0.01002)	(0.02400)	(0.00991)	(0.02684)	(0.00959)	(0.01712)	(0.01017)	(0.02307)
$[-10, 10]$	0.85434	0.74521	0.83494	0.69911	0.79669	0.68657	0.71915	0.66424
	(0.01441)	(0.02648)	(0.01480)	(0.03019)	(0.02970)	(0.03580)	(0.06485)	(0.05338)

Table B.3: Glass, classification accuracy.

	micro				macro			
	SSE		CE		SSE		CE	
	C_t	C_g	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$	0.94400	0.55738	0.93769	0.62740	0.79600	0.68398	0.79793	0.67744
	(0.01636)	(0.04912)	(0.01551)	(0.04766)	(0.02738)	(0.04128)	(0.02285)	(0.05012)
$[-10, 10]$	0.79578	0.60513	0.90388	0.62657	0.71373	0.58626	0.69585	0.55828
	(0.08762)	(0.08170)	(0.02785)	(0.05711)	(0.06541)	(0.06897)	(0.07784)	(0.08112)

Table B.4: Cancer, classification accuracy.

	micro				macro			
	SSE		CE		SSE		CE	
	C_t	C_g	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$	0.99944	0.97298	1.00000	0.97322	0.99451	0.97656	0.99633	0.97487
	(0.00096)	(0.00788)	(0.00000)	(0.00861)	(0.00227)	(0.00759)	(0.00275)	(0.00887)
$[-10, 10]$	0.99813	0.96206	1.00000	0.96408	0.99539	0.96574	0.99357	0.97335
	(0.00150)	(0.01170)	(0.00000)	(0.00685)	(0.00279)	(0.01006)	(0.00612)	(0.00961)

Table B.5: Heart, classification accuracy.

	micro				macro			
	SSE		CE		SSE		CE	
	C_t	C_g	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$	0.97447	0.78274	0.97918	0.77466	0.91038	0.83086	0.90601	0.82772
	(0.00477)	(0.02250)	(0.00648)	(0.02138)	(0.00925)	(0.01538)	(0.00915)	(0.01743)
$[-10, 10]$	0.95409	0.76148	0.93496	0.80585	0.85821	0.83363	0.80135	0.74857
	(0.00910)	(0.02149)	(0.01096)	(0.02425)	(0.01829)	(0.02063)	(0.05700)	(0.05340)

Table B.6: MNIST, classification accuracy.

	micro				macro			
	SSE		CE		SSE		CE	
	C_t	C_g	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$	0.98922	0.56534	0.99846	0.57332	0.96611	0.60834	0.98404	0.61831
	(0.01097)	(0.04874)	(0.00395)	(0.04828)	(0.01829)	(0.04600)	(0.01334)	(0.04547)
$[-10, 10]$	0.87408	0.49537	0.95444	0.52401	0.74988	0.46981	0.70077	0.44259
	(0.05040)	(0.05590)	(0.02668)	(0.05063)	(0.06487)	(0.06232)	(0.07736)	(0.06666)

Table B.7: Iris, classification accuracy for different activations.

	TanH		ReLU		ELU	
	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$,	1.0000	0.9664	0.9990	0.9612	0.9834	1.0000
micro	(0.0004)	(0.0031)	(0.0027)	(0.0123)	(0.0013)	(0.0000)
$[-1, 1]$	0.9663	0.9783	0.9691	0.9453	0.9703	0.9603
macro	(0.0121)	(0.0161)	(0.0125)	(0.0318)	(0.0130)	(0.0192)
$[-10, 10]$	0.9899	0.9618	0.9767	0.9672	0.9766	0.9616
micro	(0.0074)	(0.0258)	(0.0232)	(0.0355)	(0.0179)	(0.0267)
$[-10, 10]$	0.9351	0.8746	0.8597	0.8735	0.8721	0.8963
macro	(0.0720)	(0.0714)	(0.1113)	(0.1344)	(0.0980)	(0.1098)

Table B.8: Diabetes, classification accuracy for different activations.

	TanH		ReLU		ELU	
	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$,	0.8511	0.7488	0.8251	0.7456	0.8040	0.7948
micro	(0.0107)	(0.0269)	(0.0100)	(0.0214)	(0.0106)	(0.0200)
$[-1, 1]$,	0.8217	0.7141	0.7605	0.7737	0.7767	0.7612
macro	(0.0103)	(0.0216)	(0.0192)	(0.0502)	(0.0159)	(0.0231)
$[-10, 10]$,	0.8229	0.7175	0.7459	0.6680	0.7439	0.6818
micro	(0.0156)	(0.0276)	(0.0473)	(0.0471)	(0.0456)	(0.0438)
$[-10, 10]$,	0.7222	0.6751	0.6585	0.6498	0.6494	0.6356
macro	(0.0510)	(0.0514)	(0.0780)	(0.0919)	(0.0743)	(0.0727)

Table B.9: Glass, classification accuracy for different activations.

	TanH		ReLU		ELU	
	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$,	0.9242	0.6804	0.9106	0.6902	0.8913	0.6950
micro	(0.0190)	(0.0496)	(0.0241)	(0.0478)	(0.0202)	(0.0370)
$[-1, 1]$,	0.7875	0.6342	0.7765	0.6708	0.7323	0.7015
macro	(0.0297)	(0.0668)	(0.0342)	(0.0502)	(0.0361)	(0.0579)
$[-10, 10]$,	0.8430	0.6209	0.7779	0.5605	0.7660	0.6433
micro	(0.0445)	(0.0730)	(0.0580)	(0.0735)	(0.0586)	(0.0635)
$[-10, 10]$,	0.6671	0.6045	0.5437	0.4696	0.5158	0.4310
macro	(0.0727)	(0.0849)	(0.0625)	(0.0808)	(0.1082)	(0.0970)

Table B.10: Cancer, classification accuracy for different activations.

	TanH		ReLU		ELU	
	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$,	1.0000	0.9681	1.0000	0.9685	1.0000	0.9586
micro	(0.0000)	(0.0107)	(0.0000)	(0.0133)	(0.0000)	(0.0105)
$[-1, 1]$,	0.9958	0.9578	0.9919	0.9753	0.9942	0.9661
macro	(0.0030)	(0.0159)	(0.0062)	(0.0151)	(0.0048)	(0.0104)
$[-10, 10]$,	1.0000	0.9669	0.9992	0.9565	0.9990	0.9790
micro	(0.0002)	(0.0174)	(0.0020)	(0.0190)	(0.0022)	(0.0132)
$[-10, 10]$,	0.9825	0.9643	0.9827	0.9604	0.9847	0.9668
macro	(0.0473)	(0.0453)	(0.0190)	(0.0274)	(0.0134)	(0.0210)

Table B.11: Heart, classification accuracy for different activations.

	TanH		ReLU		ELU	
	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$,	0.9765	0.7883	0.9442	0.7727	0.9304	0.8040
micro	(0.0067)	(0.0234)	(0.0108)	(0.0219)	(0.0100)	(0.0205)
$[-1, 1]$,	0.9211	0.7645	0.8506	0.7917	0.8485	0.8041
macro	(0.0088)	(0.0246)	(0.0251)	(0.0281)	(0.0213)	(0.0282)
$[-10, 10]$,	0.9308	0.7581	0.8700	0.7593	0.8788	0.7540
micro	(0.0108)	(0.0237)	(0.0256)	(0.0321)	(0.0242)	(0.0299)
$[-10, 10]$,	0.7832	0.7737	0.7394	0.7175	0.7487	0.6998
macro	(0.0551)	(0.0611)	(0.0678)	(0.0661)	(0.0648)	(0.0633)

Table B.12: MNIST, classification accuracy for different activations.

	TanH		ReLU		ELU	
	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$,	0.8974	0.4929	0.9629	0.5769	0.9732	0.5816
micro	(0.1576)	(0.0878)	(0.1252)	(0.1034)	(0.1159)	(0.0985)
$[-1, 1]$	0.9487	0.5714	0.8851	0.5165	0.9705	0.5747
macro	(0.0249)	(0.0505)	(0.0608)	(0.0668)	(0.0261)	(0.0527)
$[-10, 10]$	0.4144	0.2062	0.7478	0.2944	0.8442	0.4058
micro	(0.0706)	(0.0425)	(0.0940)	(0.0551)	(0.0774)	(0.0599)
$[-10, 10]$	0.2486	0.1501	0.4901	0.2826	0.4531	0.2701
macro	(0.0616)	(0.0424)	(0.0876)	(0.0690)	(0.0994)	(0.0739)

Table B.13: Iris, classification accuracy using the softmax activation in the output layer.

	TanH		ReLU		ELU	
	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$,	0.9892	0.9602	0.9835	0.9624	0.9828	0.9855
micro	(0.0038)	(0.0148)	(0.0016)	(0.0236)	(0.0036)	(0.0175)
$[-1, 1]$	0.9524	0.9684	0.9544	0.9551	0.9538	0.9510
macro	(0.0284)	(0.0529)	(0.0238)	(0.0307)	(0.0237)	(0.0282)
$[-10, 10]$	0.9968	0.9615	0.9647	0.9730	0.9812	0.9132
micro	(0.0058)	(0.0155)	(0.0353)	(0.0442)	(0.0211)	(0.0247)
$[-10, 10]$	0.8446	0.8419	0.8256	0.8247	0.8323	0.8305
macro	(0.0996)	(0.0980)	(0.1253)	(0.1368)	(0.1038)	(0.1141)

Table B.14: Glass, classification accuracy using the softmax activation in the output layer.

	TanH		ReLU		ELU	
	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$,	0.9617	0.5913	0.9285	0.7332	0.9293	0.6787
micro	(0.0137)	(0.0560)	(0.0176)	(0.0442)	(0.0168)	(0.0511)
$[-1, 1]$	0.8014	0.6417	0.7800	0.5738	0.7585	0.6063
macro	(0.0344)	(0.0513)	(0.0422)	(0.0699)	(0.0455)	(0.0577)
$[-10, 10]$	0.8928	0.6095	0.8038	0.6256	0.7998	0.6105
micro	(0.0397)	(0.0722)	(0.0524)	(0.0755)	(0.0561)	(0.0701)
$[-10, 10]$	0.6752	0.6064	0.5814	0.4733	0.5896	0.4951
macro	(0.0673)	(0.1025)	(0.0718)	(0.0746)	(0.0888)	(0.0745)

Table B.15: MNIST, classification accuracy using the softmax activation in the output layer.

	TanH		ReLU		ELU	
	C_t	C_g	C_t	C_g	C_t	C_g
$[-1, 1]$,	0.9994	0.5699	0.9980	0.5807	0.9990	0.5899
micro	(0.0025)	(0.0474)	(0.0056)	(0.0465)	(0.0047)	(0.0470)
$[-1, 1]$	0.9833	0.6041	0.9707	0.6143	0.9683	0.5904
macro	(0.0143)	(0.0468)	(0.0211)	(0.0488)	(0.0248)	(0.0492)
$[-10, 10]$	0.5589	0.2501	0.5668	0.3230	0.8019	0.4001
micro	(0.0757)	(0.0462)	(0.0736)	(0.0493)	(0.0819)	(0.0618)
$[-10, 10]$	0.2942	0.1654	0.5151	0.3212	0.6131	0.3446
macro	(0.0591)	(0.0406)	(0.0867)	(0.0701)	(0.0742)	(0.0649)

Table B.16: Iris, classification accuracy for the various NN architectures.

	$h = 4$		$2 \times h = 8$		$10 \times h = 40$		
	C_t	C_g	C_t	C_g	C_t	C_g	
$[-1, 1]$,	0.9828	0.9855	0.9925	0.9667	0.9791	0.9683	1 hidden layer
micro	(0.0036)	(0.0175)	(0.0038)	(0.0000)	(0.0059)	(0.0071)	
$[-1, 1]$,	0.9538	0.9510	0.9377	0.9470	0.8817	0.8970	
macro	(0.0237)	(0.0282)	(0.0468)	(0.0382)	(0.0682)	(0.0800)	
$[-10, 10]$,	0.9812	0.9132	0.9622	0.9731	0.9639	0.9448	
micro	(0.0211)	(0.0247)	(0.0290)	(0.0303)	(0.0242)	(0.0353)	
$[-10, 10]$,	0.8323	0.8305	0.8924	0.9005	0.8939	0.8508	
macro	(0.1038)	(0.1141)	(0.0793)	(0.0773)	(0.0653)	(0.0785)	
$[-1, 1]$,	0.9829	0.9621	0.9973	0.9618	0.9900	0.8650	2 hidden layers
micro	(0.0074)	(0.0282)	(0.0059)	(0.0128)	(0.0251)	(0.0352)	
$[-1, 1]$,	0.9185	0.9321	0.9025	0.8980	0.8965	0.9520	
macro	(0.0695)	(0.0543)	(0.0596)	(0.0685)	(0.0573)	(0.0376)	
$[-10, 10]$,	0.9383	0.9179	0.9473	0.9069	0.9669	0.9461	
micro	(0.0720)	(0.0766)	(0.0532)	(0.0607)	(0.0285)	(0.0453)	
$[-10, 10]$,	0.6922	0.6415	0.7225	0.7650	0.8741	0.8849	
macro	(0.1206)	(0.1363)	(0.1286)	(0.1202)	(0.0927)	(0.0959)	
$[-1, 1]$,	0.9823	0.9511	0.9907	0.9657	0.9727	0.9278	3 hidden layers
micro	(0.0072)	(0.0251)	(0.0126)	(0.0100)	(0.0209)	(0.0389)	
$[-1, 1]$,	0.9010	0.8836	0.8851	0.9156	0.8399	0.8244	
macro	(0.0653)	(0.0796)	(0.0722)	(0.0891)	(0.1126)	(0.1205)	
$[-10, 10]$,	0.8553	0.8160	0.8630	0.8093	0.8987	0.8015	
micro	(0.1325)	(0.1437)	(0.1100)	(0.1019)	(0.0985)	(0.1029)	
$[-10, 10]$,	0.4441	0.4362	0.4811	0.4707	0.6510	0.6555	
macro	(0.1666)	(0.1989)	(0.1787)	(0.2227)	(0.1979)	(0.1907)	

Table B.17: Diabetes, classification accuracy for the various NN architectures.

	$h = 8$		$2 \times h = 16$		$10 \times h = 80$		
	C_t	C_g	C_t	C_g	C_t	C_g	
$[-1, 1]$,	0.8040	0.7948	0.8326	0.7174	0.7996	0.7329	1 hidden layer
micro	(0.0106)	(0.0200)	(0.0091)	(0.0179)	(0.0140)	(0.0238)	
$[-1, 1]$,	0.7767	0.7612	0.7609	0.7343	0.7115	0.6742	
macro	(0.0159)	(0.0231)	(0.0222)	(0.0293)	(0.0471)	(0.0459)	
$[-10, 10]$,	0.7439	0.6818	0.7559	0.6948	0.7878	0.6978	
micro	(0.0456)	(0.0438)	(0.0288)	(0.0391)	(0.0355)	(0.0368)	
$[-10, 10]$,	0.6494	0.6356	0.6914	0.6979	0.7088	0.6676	
macro	(0.0743)	(0.0727)	(0.0565)	(0.0584)	(0.0578)	(0.0543)	
$[-1, 1]$,	0.8556	0.7195	0.8769	0.7171	0.9281	0.7025	2 hidden layers
micro	(0.0160)	(0.0287)	(0.0170)	(0.0257)	(0.0333)	(0.0301)	
$[-1, 1]$,	0.7750	0.7433	0.7701	0.6872	0.7135	0.6707	
macro	(0.0223)	(0.0365)	(0.0327)	(0.0513)	(0.0679)	(0.0756)	
$[-10, 10]$,	0.7348	0.6488	0.8067	0.6679	0.8939	0.6744	
micro	(0.0548)	(0.0462)	(0.0420)	(0.0355)	(0.0384)	(0.0339)	
$[-10, 10]$,	0.5696	0.5650	0.5673	0.5673	0.7597	0.6720	
macro	(0.1103)	(0.1229)	(0.1172)	(0.0847)	(0.0643)	(0.0569)	
$[-1, 1]$,	0.8751	0.7391	0.9193	0.7307	0.9946	0.6534	3 hidden layers
micro	(0.0189)	(0.0239)	(0.0260)	(0.0317)	(0.0043)	(0.0278)	
$[-1, 1]$,	0.7718	0.7103	0.7264	0.6670	0.6896	0.6161	
macro	(0.0350)	(0.0442)	(0.0608)	(0.0492)	(0.1248)	(0.0730)	
$[-10, 10]$,	0.6405	0.6338	0.8267	0.6825	0.9485	0.6977	
micro	(0.0791)	(0.1161)	(0.0767)	(0.0492)	(0.0389)	(0.0355)	
$[-10, 10]$,	0.5183	0.5132	0.5421	0.5514	0.6510	0.5989	
macro	(0.1618)	(0.1002)	(0.1213)	(0.1526)	(0.1173)	(0.1502)	

Table B.18: Glass, classification accuracy for the various NN architectures.

	$h = 9$		$2 \times h = 18$		$10 \times h = 90$		
	C_t	C_g	C_t	C_g	C_t	C_g	
$[-1, 1]$,	0.9293	0.6787	0.9313	0.6516	0.8664	0.7602	1 hidden layer
micro	(0.0168)	(0.0511)	(0.0154)	(0.0456)	(0.0293)	(0.0383)	
$[-1, 1]$,	0.7585	0.6063	0.7506	0.6371	0.7649	0.5609	
macro	(0.0455)	(0.0577)	(0.0466)	(0.0639)	(0.0388)	(0.0470)	
$[-10, 10]$,	0.7998	0.6105	0.8824	0.6104	0.9147	0.6288	
micro	(0.0561)	(0.0701)	(0.0395)	(0.0561)	(0.0300)	(0.0534)	
$[-10, 10]$,	0.5896	0.4951	0.6978	0.6069	0.7587	0.6373	
macro	(0.0888)	(0.0745)	(0.0559)	(0.0667)	(0.0480)	(0.0646)	
$[-1, 1]$,	0.9576	0.7004	0.9799	0.6447	0.9930	0.7012	2 hidden layers
micro	(0.0237)	(0.0501)	(0.0183)	(0.0493)	(0.0087)	(0.0431)	
$[-1, 1]$,	0.7524	0.5880	0.7640	0.6733	0.7475	0.5932	
macro	(0.0481)	(0.0696)	(0.0500)	(0.0676)	(0.0584)	(0.0784)	
$[-10, 10]$,	0.7359	0.5790	0.9044	0.6537	0.9225	0.5561	
micro	(0.1027)	(0.0945)	(0.0508)	(0.0619)	(0.0447)	(0.0538)	
$[-10, 10]$,	0.3455	0.3077	0.4285	0.3679	0.7927	0.6215	
macro	(0.1334)	(0.1466)	(0.1299)	(0.1391)	(0.0584)	(0.0745)	
$[-1, 1]$,	0.9783	0.5311	0.9985	0.6407	0.9804	0.6704	3 hidden layers
micro	(0.0189)	(0.0471)	(0.0035)	(0.0545)	(0.0197)	(0.0436)	
$[-1, 1]$,	0.6815	0.6060	0.6543	0.5783	0.6688	0.5189	
macro	(0.0566)	(0.0778)	(0.0774)	(0.0977)	(0.0673)	(0.0774)	
$[-10, 10]$,	0.4969	0.4147	0.7174	0.5126	0.8629	0.6021	
micro	(0.1332)	(0.0986)	(0.1229)	(0.1049)	(0.0754)	(0.0639)	
$[-10, 10]$,	0.2222	0.2184	0.3189	0.2678	0.7321	0.5744	
macro	(0.1303)	(0.1494)	(0.1400)	(0.1267)	(0.0696)	(0.0820)	

Table B.19: Cancer, classification accuracy for the various NN architectures.

	$h = 10$		$2 \times h = 20$		$10 \times h = 100$		
	C_t	C_g	C_t	C_g	C_t	C_g	
$[-1, 1]$,	1.0000	0.9586	1.0000	0.9725	1.0000	0.9783	1 hidden layer
micro	(0.0000)	(0.0105)	(0.0001)	(0.0116)	(0.0000)	(0.0071)	
$[-1, 1]$,	0.9942	0.9661	0.9981	0.9502	0.9875	0.9560	
macro	(0.0048)	(0.0104)	(0.0026)	(0.0172)	(0.0120)	(0.0292)	
$[-10, 10]$,	0.9990	0.9790	0.9982	0.9589	0.9952	0.9590	
micro	(0.0022)	(0.0132)	(0.0027)	(0.0131)	(0.0048)	(0.0104)	
$[-10, 10]$,	0.9847	0.9668	0.9868	0.9717	0.9874	0.9555	
macro	(0.0134)	(0.0210)	(0.0094)	(0.0170)	(0.0104)	(0.0158)	
$[-1, 1]$,	1.0000	0.9365	0.9999	0.9605	0.9985	0.9347	2 hidden layers
micro	(0.0001)	(0.0151)	(0.0006)	(0.0079)	(0.0024)	(0.0140)	
$[-1, 1]$,	0.9939	0.9673	0.9942	0.9457	0.9904	0.9620	
macro	(0.0059)	(0.0157)	(0.0060)	(0.0148)	(0.0120)	(0.0154)	
$[-10, 10]$,	0.9853	0.9586	0.9850	0.9532	0.9821	0.9575	
micro	(0.0254)	(0.0290)	(0.0120)	(0.0136)	(0.0117)	(0.0197)	
$[-10, 10]$,	0.9490	0.9336	0.9754	0.9524	0.9772	0.9407	
macro	(0.0784)	(0.0777)	(0.0299)	(0.0308)	(0.0357)	(0.0342)	
$[-1, 1]$,	0.9994	0.9549	0.9993	0.9548	0.9860	0.9780	3 hidden layers
micro	(0.0018)	(0.0160)	(0.0016)	(0.0158)	(0.0279)	(0.0262)	
$[-1, 1]$,	0.9855	0.9618	0.9869	0.9576	0.9480	0.9312	
macro	(0.0387)	(0.0388)	(0.0236)	(0.0275)	(0.1023)	(0.1058)	
$[-10, 10]$,	0.9629	0.9431	0.9713	0.9494	0.9745	0.9723	
micro	(0.0648)	(0.0658)	(0.0380)	(0.0371)	(0.0354)	(0.0327)	
$[-10, 10]$,	0.8415	0.8476	0.9588	0.9318	0.9209	0.9004	
macro	(0.1673)	(0.1764)	(0.0824)	(0.0827)	(0.1267)	(0.1331)	

Table B.20: Heart, classification accuracy for the various NN architectures.

	$h = 10$		$2 \times h = 20$		$10 \times h = 100$		
	C_t	C_g	C_t	C_g	C_t	C_g	
$[-1, 1]$,	0.9304	0.8040	0.9419	0.8033	0.9382	0.8050	1 hidden layer
micro	(0.0100)	(0.0205)	(0.0108)	(0.0215)	(0.0166)	(0.0179)	
$[-1, 1]$,	0.8485	0.8041	0.8235	0.7921	0.7914	0.7207	
macro	(0.0213)	(0.0282)	(0.0286)	(0.0406)	(0.0570)	(0.0538)	
$[-10, 10]$,	0.8788	0.7540	0.9269	0.7595	0.9782	0.7660	
micro	(0.0242)	(0.0299)	(0.0182)	(0.0264)	(0.0082)	(0.0218)	
$[-10, 10]$,	0.7487	0.6998	0.8090	0.7687	0.8218	0.7849	
macro	(0.0648)	(0.0633)	(0.0446)	(0.0444)	(0.0448)	(0.0430)	
$[-1, 1]$,	0.9751	0.7648	0.9853	0.7944	0.9971	0.7511	2 hidden layers
micro	(0.0110)	(0.0234)	(0.0090)	(0.0229)	(0.0027)	(0.0218)	
$[-1, 1]$,	0.8822	0.7687	0.8905	0.7675	0.8403	0.7454	
macro	(0.0251)	(0.0325)	(0.0255)	(0.0332)	(0.0703)	(0.0670)	
$[-10, 10]$,	0.8923	0.7081	0.9637	0.7515	0.9830	0.7642	
micro	(0.0377)	(0.0378)	(0.0179)	(0.0259)	(0.0118)	(0.0211)	
$[-10, 10]$,	0.5906	0.5806	0.7180	0.6748	0.6723	0.6334	
macro	(0.0624)	(0.0758)	(0.0812)	(0.0727)	(0.1335)	(0.1193)	
$[-1, 1]$,	0.9775	0.7637	0.9987	0.7259	0.9973	0.7735	3 hidden layers
micro	(0.0108)	(0.0278)	(0.0014)	(0.0240)	(0.0030)	(0.0219)	
$[-1, 1]$,	0.8597	0.7606	0.8278	0.7403	0.8193	0.7126	
macro	(0.0387)	(0.0434)	(0.0530)	(0.0484)	(0.1189)	(0.0940)	
$[-10, 10]$,	0.7502	0.6597	0.9770	0.7519	0.9519	0.7443	
micro	(0.0970)	(0.0755)	(0.0141)	(0.0281)	(0.0505)	(0.0500)	
$[-10, 10]$,	0.5403	0.5273	0.6374	0.6091	0.6205	0.5675	
macro	(0.0745)	(0.0659)	(0.0770)	(0.0732)	(0.1417)	(0.0916)	

Table B.21: MNIST, classification accuracy for the various NN architectures.

	$h = 10$		$2 \times h = 20$		$10 \times h = 100$		
	C_t	C_g	C_t	C_g	C_t	C_g	
$[-1, 1]$,	0.9990	0.5899	0.9945	0.5757	0.9816	0.5815	1 hidden layer
micro	(0.0047)	(0.0470)	(0.0143)	(0.0472)	(0.0260)	(0.0506)	
$[-1, 1]$,	0.9683	0.5904	0.9755	0.6066	0.9653	0.5898	
macro	(0.0248)	(0.0492)	(0.0211)	(0.0470)	(0.0278)	(0.0468)	
$[-10, 10]$,	0.8019	0.4001	0.8507	0.4056	0.8802	0.4067	
micro	(0.0819)	(0.0618)	(0.0670)	(0.0584)	(0.0567)	(0.0566)	
$[-10, 10]$,	0.6131	0.3446	0.7860	0.4387	0.8697	0.4973	
macro	(0.0742)	(0.0649)	(0.0819)	(0.0723)	(0.0641)	(0.0618)	
$[-1, 1]$,	0.9933	0.5559	0.9816	0.5796	0.9617	0.5747	2 hidden layers
micro	(0.0142)	(0.0538)	(0.0267)	(0.0527)	(0.0334)	(0.0455)	
$[-1, 1]$,	0.7812	0.4756	0.9300	0.5691	0.9374	0.5792	
macro	(0.0993)	(0.0770)	(0.0591)	(0.0594)	(0.0409)	(0.0501)	
$[-10, 10]$,	0.5427	0.2776	0.7185	0.3298	0.7818	0.3660	
micro	(0.1094)	(0.0703)	(0.0898)	(0.0681)	(0.0716)	(0.0624)	
$[-10, 10]$,	0.3525	0.2272	0.7839	0.4593	0.8808	0.5199	
macro	(0.0928)	(0.0652)	(0.1337)	(0.0925)	(0.0638)	(0.0628)	
$[-1, 1]$,	0.9926	0.4867	0.9783	0.5651	0.9417	0.5213	3 hidden layers
micro	(0.0129)	(0.0594)	(0.0264)	(0.0527)	(0.0407)	(0.0480)	
$[-1, 1]$,	0.4005	0.2680	0.5536	0.3474	0.9276	0.5792	
macro	(0.1234)	(0.0844)	(0.1892)	(0.1198)	(0.0495)	(0.0539)	
$[-10, 10]$,	0.3789	0.2203	0.7015	0.3227	0.7136	0.3456	
micro	(0.1415)	(0.0750)	(0.1029)	(0.0740)	(0.0790)	(0.0607)	
$[-10, 10]$,	0.1488	0.1289	0.4847	0.3052	0.9097	0.5519	
macro	(0.0627)	(0.0530)	(0.1841)	(0.1132)	(0.0558)	(0.0580)	

Appendix C

Implementation Details

This appendix discusses the implementation details of the experiments conducted for this thesis. Section C.1 describes the hardware used. Section C.2 discusses the software used, and provides a reference to the original code developed for the study.

C.1 Hardware

All experiments were run on a single node of a computing cluster with 24 Intel 5th generation CPUs and 128 GB RAM. The computing facilities were provided by the The Centre for High Performance Computing (CHPC) in Cape Town, South Africa. A detailed description of the CHPC facilities is available at the following URL: https://chpc.ac.za/

C.2 Software

This study used the Python programming language (Python Software Foundation, https://www.python.org/) to set-up and conduct all the experiments. The TensorFlow [1] library was used to implement the neural networks and the various sampling algorithms. The NumPy [103] package was used to implement the FLA measures used in this study. All graphs presented in the thesis were generated using the ggplot2 package [145].

The code developed for the purpose on this study is publicly available at the following

URL: https://github.com/annabosman/fla-in-tf.

Appendix D

Acronyms

BP	Backpropagation
CE	Cross-Entropy Error
ELU	Exponential Linear Unit
EWMA	Exponentially Weighted Moving Average
FDC	Fitness Distance Correlation
FEM	First Entropic Measure of Ruggedness
FLA	Fitness Landscape Analysis
LOESS	Locally Estimated Scatterplot Smoothing
NN	Artificial Neural Network
PCA	Principal Component Analysis
PSO	Particle Swarm Optimisation
ReLU	Rectified Linear Unit
SSE	Sum Squared Error
TanH	Hyperbolic Tangent
XOR	Exclusive OR

Appendix E

Symbols

This appendix lists the mathematical symbols used throughout the thesis, and their definitions. The symbols used within each chapter are listed under separate sections. Each section lists only newly introduced or redefined symbols.

E.1 Chapter 2: Artificial Neural Networks

α	Momentum coefficient
Δw_{ab}	Adjustment of the weight between neurons a and b
η	Learning rate
θ	Bias threshold
λ	Penalty coefficient
a, b	Indices of neurons in successive layers
D	Dataset
D_T	Training set
D_G	Generalisation set
E	Error metric
E_{sse}	Sum squared error
E_{ce}	Cross-entropy error
f	Activation function

f_{o_k}	Activation functions of the k'th output neuron
f_{u_j}	Activation functions of the j^{th} hidden neuron
I	Size of the input layer
i	Index of an input neuron
J	Size of the hidden layer
j	Index of a hidden neuron
K	Size of the output layer
k	Index of an output neuron
M	Size of a successive layer
n	Number of NN inputs
net	Weighted input signal
o_k	The output of the k'th neuron in the output layer
$o_{k,p}$	The k'th output obtained for data point p
P	The number of data points, or patterns
p	A single data point, or pattern
t	Iteration, or epoch count
$t_{k,p}$	The k'th target value for data point p
u_j	The j^{th} hidden layer signal
v_{ji}	A weight connecting the j'th hidden neuron and the i'th input neuron
W	The total number of weights in a NN
w_0	Sensitivity threshold for weight elimination
w_{ab}	The weight between neurons a and b
w_{kj}	A weight connecting the k'th output neuron and the j'th hidden neuron
w_i	The i^{th} weight corresponding to the input x_i
x_i	The i^{th} input to a neuron
y	Output of a neuron
z_i	The i^{th} input layer signal

E.2 Chapter 3: Fitness Landscape Analysis

Δe_l	Difference between the fitness values of $\mathbf{x}_l$ and $\mathbf{x}_{l+1}$
$\Delta \mathbf{x}_l$	Step vector
ϵ	FEM sensitivity threshold
ε	Radius of the neighbourhood $\mathcal{N}_\varepsilon$
ξ	Sensitivity threshold for neutrality metrics calculation
$\omega_{\mathbf{max}}$	The longest sub-walk of $\mathcal{W}$ consisting of neutral 3-point objects only.
$\mathbf{b}$	Bit mask
d	Distance metric
F	Fitness landscape
$\mathbf{FDC}_s$	Estimated fitness distance correlation
$\mathcal{G}$	Set of all fitness values in a sample
G_{avg}	Estimated average gradient
G_{dev}	Standard deviation of G_{avg}.
g	Fitness function
$\overline{g}$	The mean of $\mathcal{G}$
l	Index of a step/point in a sample
L	Sample length
M_1	First neutrality metric
M_2	Second neutrality metric
m	Dimension of the search space
$\mathcal{N}_\varepsilon$	Neighbourhood with radius ε
$\mathbb{R}$	Real numbers
$r^{\max}, r^{\min}$	Indices of the maximum and minimum values in a 3-point object
$\mathcal{S}$	3-point object
s_l	Symbolic entropy representation of step l
$\mathcal{W}$	Progressive random walk sample
X	Set of solutions

X'	Approximated set of solutions
$\mathbf{x}$	Candidate solution
$\mathbf{x}^*$	Global minimum
$\tilde{\mathbf{x}}$	Local minimum
$\mathbf{x}_l$	The l^{th} point in the sample
$\mathbf{x}_o$	The o^{th} neighbour of $\mathbf{x}_l$

E.3 Chapter 4: Search Space Boundaries

$fanin$	The number of connections leading into a neuron
N	Width of the search space boundaries

E.4 Chapter 6: Modality Quantification

β	Decay coefficient for the EWMA smoothing
$\mathbf{g}_l$	The gradient vector calculated for a point $\mathbf{x}_l$
l_{stag}	The average length of the stagnant regions
n_{stag}	The average number of stagnant regions encountered per sample
T	A sequence of length L
T'	An EWMA-smoothed version of sequence T
w	Window size for the EWMA smoothing

E.5 Chapter 9: Neural Network Architectures

C_g	Classification accuracy of the generalisation set
C_t	Classification accuracy of the training set

Appendix F

Derived Publications

This section provides a list of all publications derived from this thesis. The following conference and journal articles were published:

- Anna Sergeevna Bosman, Andries Engelbrecht, and Mardé Helbig. Search space boundaries in neural network error landscape analysis. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pages 1–8. IEEE, 2016.
- Willem Abraham van Aardt, Anna Sergeevna Bosman, and Katherine Mary Malan. Characterising neutrality in neural network error landscapes. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1374–1381. IEEE, 2017.
- Anna Sergeevna Bosman, Andries Engelbrecht, and Mardé Helbig. Fitness landscape analysis of weight-elimination neural networks. *Neural Processing Letters*, 48(1):353–373, 2018.
- Anna Sergeevna Bosman, Andries Engelbrecht, and Mardé Helbig. Progressive gradient walk for neural network fitness landscape analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1473–1480. ACM, 2018.

The following preprint was submitted to the *Neural Networks* journal, and is currently under review:

- Anna Sergeevna Bosman, Andries Engelbrecht, and Mardé Helbig. Visualising Basins of Attraction for the Cross-Entropy and the Squared Error Neural Network Loss Functions. *arXiv e-prints*, page arXiv:1901.02302, January 2019.