UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Analysis of catastrophic interference with application to spline neural architectures

by

Heinrich van Deventer

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

14 February 2024

Publication data:

Heinrich van Deventer. Analysis of Catastrophic Interference with Application to Spline Neural Architectures. Master's dissertation, University of Pretoria, Department of Computer Science, Pretoria, South Africa, February 2024.

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

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

Analysis of Catastrophic Interference with Application to Spline Neural Architectures

by

Heinrich van Deventer
E-mail: HPDeventer@gmail.com

Abstract

Continual learning is the sequential learning of different tasks by a machine learning model. Continual learning is known to be hindered by catastrophic interference or forgetting, i.e. rapid unlearning of earlier learned tasks when new tasks are learned. Despite their practical success, artificial neural networks (ANNs) are prone to catastrophic interference. This study analyses how gradient descent and overlapping representations between distant input points lead to distal interference and catastrophic interference. Distal interference refers to the phenomenon where training a model on a subset of the domain leads to non-local changes on other subsets of the domain. This study shows that uniformly trainable models without distal interference must be exponentially large. A novel antisymmetric bounded exponential layer B-spline ANN architecture named *ABEL-Spline* is proposed that can approximate any continuous function, is uniformly trainable, has polynomial computational complexity, and provides *some* guarantees for distal interference. Experiments are presented to demonstrate the theoretical properties of ABEL-Splines. ABEL-Splines are also evaluated on benchmark regression problems. It is concluded that the weaker distal interference guarantees in ABEL-Splines are insufficient for model-only continual learning. It is conjectured that continual learning with polynomial complexity models requires augmentation of the training data or algorithm.
Keywords: continual learning, catastrophic forgetting, catastrophic interference, overlapping representation, sparse distributed representation, regression, spline, artificial neural network.

Supervisors :	Dr. A. S. Bosman
Department :	Department of Computer Science
Degree :	Master of Science

"The only source of knowledge is experience."

Albert Einstein

Acknowledgements

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

- My family and friends, for all your love and support.
- Doctor Anna Bosman, for her guidance as my supervisor.
- Supported by the National Research Foundation (NRF) of South Africa Thuthuka Grant Number 13819413/TTK210316590115.
- Computing resources provided by the South African Centre for High-Performance Computing (CHPC).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humans can learn many tasks sequentially, unlike conventional artificial neural networks (ANNs) trained with gradient descent optimisation [1]. Continual learning entails training a model for many tasks sequentially without forgetting knowledge obtained from the preceding tasks, where the data in the old tasks are no longer available while training on new ones. Continual learning is also known as incremental or life-long learning [2, 3, 4, 5]. Continual learning with ANNs is hindered by catastrophic interference. Catastrophic interference is an emergent phenomenon where a machine learning model such as an ANN learns a new task, and the subsequent parameter updates interfere with the model performance on previously learned tasks [6, 7, 8]. Catastrophic interference is also called catastrophic forgetting. If an ANN cannot effectively learn many tasks, it has limited utility for continual learning [9, 10]. Catastrophic interference is like learning to pick up a cup while forgetting how to drink.

The reviewed studies on catastrophic interference and mitigation techniques focus on **time** or the sequential aspect of the continual learning problem [2, 4]. This study focuses on the input **space** for differentiable models like ANNs. A model f can be trained with gradient descent optimisation to create an updated f'. Let $D(f)$ denote the domain of f and f'. If f is trained at one point $v \in D(f)$, then $f(x)$ may change at some distant point $x \in D(f)$. If one is equipped with some measure of distance $d(x, v) > \delta$ or dissimilarity, it remains to be shown if one can limit the absolute model change $|f(x) - f'(x)| < \varepsilon$ or guarantee that $|f(x) - f'(x)| = 0$.

A differentiable model f trained on data points $v \in D(f)$ with gradient descent optimisation can have non-local or distal changes at points $x \in D(f)$ far away from the training data. Non-local or distal changes to the model are akin to off-target effects, called *distal interference*. Distal interference can cause catastrophic interference in continual learning problems (see Chapter 3). Regression problems commonly require training models over all the data simultaneously [11], to counteract distal interference. Distal interference combined with large model parameter gradients can hinder the convergence and training of large ANN models that are sensitive to hyperparameter and training procedure choices [12, 13]. Distal interference can manifest as seemingly unrelated problems in different contexts, but the same underlying mechanisms disrupt ANN models.

The concept of a 'stability-plasticity' spectrum or trade-off has been used to describe the difference between near-perfect memory models like lookup tables and the adaptive, easy-to-train models such as ANNs with unstable memory [2, 4, 6]. The stability-plasticity spectrum is not precise enough for mathematical analysis. Stable models are immune to the proposed distal interference mechanism and can be called distal orthogonal models due to the geometry of their parameter space (see Section 3.6). It has been noted that orthogonal activity patterns prevent interference, but distributed (overlapping, non-orthogonal) representations are theorised to promote generalisation [14]. Plasticity is related to universal expressiveness, uniform trainability and generalisation. Universally expressive models are universal function approximators with sufficient power to express any continuous function with arbitrarily small error (see Chapter 5). The proposed property of uniform trainability (non-zero parameter gradients for any parameters and input point) eases training with gradient descent optimisation (see Definition 5, in Chapter 3).

Depending on the choice of distance or dissimilarity $d(x, v) > \delta$, and model properties (distal orthogonality, uniform trainability, and universal expressivity), different types of models with varying computational space complexities emerge, as discussed in this study. Using the maximum norm distance $d(x, v) = \max_i(|x_i - v_i|)$ yields an exponentially large model, immune to distal and catastrophic interference, which aligns with the stability-plasticity spectrum (see Chapter 3).

In this study, a partially successful attempt is made to develop an expressive, uniformly trainable model with polynomial space complexity and distal orthogonality using a weaker dissimilarity measure $d(x, v) = \min_i(|x_i - v_i|)$. The proposed ABEL-Spline architecture is designed with many of the mentioned properties but is insufficient for practical model-only continual learning without augmenting the training data or training algorithms. There have been many suggestions to improve continual learning outcomes with augmentation techniques [8, 15, 16]. It is conjectured, but not proven, that polynomial complexity models require augmentation for effective continual learning.

1.1 Objectives

The objectives of this study include analysis of catastrophic interference, the susceptibility of a model to catastrophic interference, and the development of models and techniques to improve continual learning. Some tangential objectives are discussed to explore avenues for future research. The objectives can be summarised as:

1. Review continual learning and catastrophic interference.
   (a) Describe continual or lifelong learning.
   (b) Discuss the emergence of catastrophic interference.
   (c) Review the established mechanisms that cause catastrophic interference.
   (d) Describe techniques in literature that are used to mitigate catastrophic interference and enable continual learning.
2. Study model susceptibility to catastrophic interference.
   (a) Explain why ANNs are susceptible to catastrophic interference.
   (b) Outline why lookup tables are robust to catastrophic interference.
   (c) Investigate the computational complexity for model-only continual learning.
   (d) Discuss the necessity of data or training augmentation to enable continual learning.

3. Determine how gradient descent mechanics contribute to catastrophic interference.
    (a) Use vector calculus and distance metrics to analyse differentiable models (such as ANNs) in general.
    (b) Determine which model properties or training techniques can mitigate catastrophic interference.
4. Develop an efficient and expressive architecture capable of continual learning.
    (a) Design the architecture from first principles to be efficient with polynomial time complexity.
    (b) Prove the architecture is a universal function approximator.
    (c) Prove that the architecture is robust to catastrophic interference.
    (d) Provide an implementation of the architecture in TensorFlow.
    (e) Empirically investigate the function approximation ability of the developed architecture.
    (f) Empirically investigate the robustness of memory retention of the developed architecture.

1.2 Contributions

This dissertation makes five novel contributions, namely:

1. Refined understanding of the causes of catastrophic interference. It is shown that distant **and** overlapping representations cause catastrophic interference. Distant (and probably unrelated) training data points with overlapping representations are susceptible to distal interference and catastrophic interference. Previous works did not emphasise the importance of local versus non-local effects.

2. Mathematical proof that uniformly trainable models robust to distal interference must have exponentially large parameter spaces. The contra-positive is that models with polynomial (which implies non-exponential) space complexity are not uniformly trainable or not robust to distal interference. This theoretical finding is a substantial limitation on the continual learning ability of polynomial complexity models.

3. The formulation of antisymmetric exponentials is provided. The analytical techniques used have applicability to develop provable universal function approximators on domains that are not subsets of $\mathbb{R}^n$, like continuous signals, sets, and multi-sets.

4. The mathematical formulation of the ABEL-Spline ANN architecture with uniform trainability, sparse- and bounded parameter gradients, and a form of min-distal orthogonality is provided. The model is constructed for numerical stability and convergent behaviour as the model size increases. These techniques can be adapted to improve the training stability of other types of ANN models.

5. An efficient TensorFlow implementation of the ABEL-Spline architecture that uses embedding-, convolutional-, and dense layers is developed. The implementation successfully utilises sparsity to compute only non-zero components without wasting time or memory to compute basis functions that evaluate to zero. The implementation can be found in a public GitHub repository[1].

[1] https://github.com/hpdeventer/MSc-Project-Heinrich-van-Deventer

1.3 Derived Publications

Several papers and preprints were drafted during this study over a few years. The final paper written for this study had been submitted for review to a journal.

- **KASAM: Spline Additive Models for Function Approximation:** (Preprint.) Initial attempts at extending Spline (additive) ANNs to universal function approximators used the Kolmogorov-Arnold representation theorem to create the Kolmogorov-Arnold Spline Additive Model (KASAM) [17]. The KASAM model did not inherit min-distal orthogonality from Spline ANNs. In contrast, ABEL-Splines use the general Stone-Weierstrass theorem to prove universal function approximation and inherit min-distal orthogonality from Spline ANNs.

- **ATLAS: Universal Function Approximator for Memory Retention:** (Preprint.) ATLAS ANNs were a precursor to the current version of ABEL-Splines. ABEL-Splines have a more descriptive name and better implementation and computational complexity [18].

- **Distal Interference: Exploring the Limits of Model-Based Continual Learning** (Under review). This paper has been submitted to The Journal of Machine Learning Research (JMLR) for publication. The paper analyses how gradient descent and overlapping or non-orthogonal representations between distant input points lead to distal interference and catastrophic interference. This study formally proves that uniformly trainable models without distal interference must be exponentially large. A novel antisymmetric bounded exponential layer B-spline ANN architecture named *ABEL-Spline* is proposed that can approximate any continuous function, is uniformly trainable, has polynomial computational complexity, and provides *some* guarantees for mitigating distal interference. Experiments indicate that the weaker distal interference guarantees in ABEL-Splines are insufficient for model-only continual learning. It is conjectured that continual learning with polynomial complexity models requires augmentation of the training data or algorithm [19].

1.4 Dissertation Outline

The rest of the dissertation is organised into the following chapters and appendices:

- **Chapter 2** gives an overview and history of ANNs and essential concepts in machine learning.
- **Chapter 3** focuses on continual learning and catastrophic interference. Mechanisms that contribute to catastrophic interference are discussed. A mathematical proof is provided that shows the limits of polynomial complexity ANNs.
- **Chapter 4** explains single-variable cardinal B-splines. Cardinal B-splines are extended to multi-variable functions that are sums of single-variable functions called Spline ANNs. The limited expressive power of Spline ANNs is explained.
- **Chapter 5** defines antisymmetric exponentials to extend Spline ANNs to universal function approximators while retaining desirable properties. Antisymmetric exponentials are proven to be universal function approximators.
- **Chapter 6** outlines the antisymmetric bounded exponential layer spline architecture called ABEL-Spline. The engineering and design choices for numerical stability are explained, and the implementation details are provided.
- **Chapter 7** presents the experiments to test theory-based predictions and to evaluate the proposed ABEL-Spline architecture on a simple continual learning problem and regression benchmarks.
- **Chapter 8** summarises the conclusions of the study. An outline for future work and research directions is discussed.
- **Appendix A** provides a list of relevant or newly defined acronyms.
- **Appendix B** lists and defines the mathematical symbols used in this work, categorised according to the relevant chapter in which they appear.
- **Appendix C** contains raw experimental data.

Chapter 2

Artificial Neural Networks

This chapter gives an overview of artificial neural networks (ANNs). Biological neural networks are discussed in Section 2.1. A history and overview of essential ANN concepts are presented in Section 2.2. Section 2.3 discusses training techniques, loss functions and optimisers. Section 2.4 discusses unexpected problems that emerged with ANNs. Section 2.5 proposes some initial modifications. A summary is given in Section 2.6 that provides an overview of this chapter.

2.1 Biological Neural Networks

Biological neural networks, such as the human brain, are complex organs composed of specialised cells called neurons. These networks are responsible for receiving, processing, and transmitting information in response to external stimuli. Biological neural networks' intricate structure and function give rise to their remarkable computational abilities and adaptability. This complexity hinders understanding and reverse-engineering biological intelligent systems [20].

2.1.1 Structure and Function of Neurons

Neurons are eukaryotic cells with unique features that allow them to transmit information through electrochemical signals. A typical neuron consists of a cell body (also known as soma), dendrites, and an axon covered by a myelin sheath (Figure 2.1). Dendrites are branch-like protrusions that extend from the cell body and receive incoming signals from other neurons. The axon is a long extension responsible for transmitting signals away from the cell body towards other neurons or target cells [21].

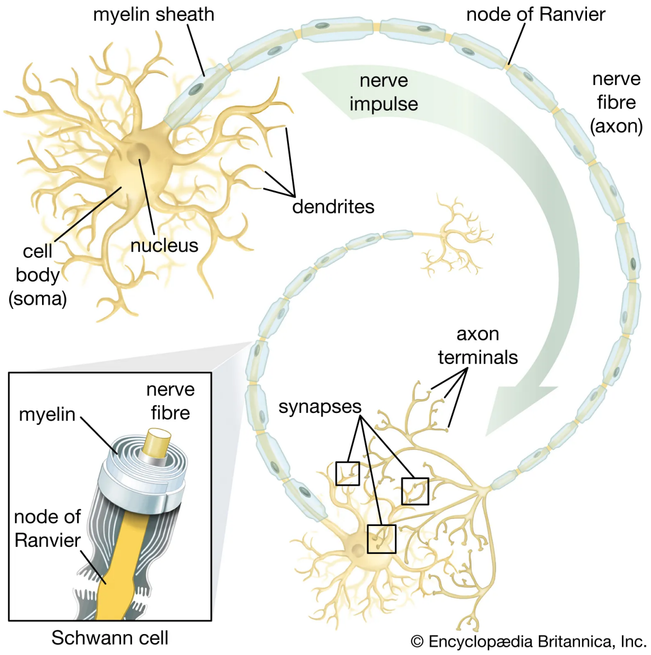


Figure 2.1: Anatomy of a nerve cell. Structural features of a motor neuron include the cell body, nerve fibres, and dendrites. Source: [22].

Electrochemical signals in biological neural networks are transmitted through action potentials and neurotransmitters. Sensory neurons in sensory organs convert external stimuli (e.g., light or pressure) into electrochemical impulses propagating through the network. An impulse reaching a dendrite can cause an action potential spike that travels from the dendrite to the cell body and down the axon. This transient activation state allows for rapid transmission of information along an axon [21].

Neurotransmitters are chemicals that facilitate signal transmission between cells at the junctions between neurons. The junctions between neurons are called synapses. These chemicals are released from the axon terminal of a pre-synaptic neuron and bind to receptors on the post-synaptic neuron's dendrites. This process enables faster and more efficient local communication between neurons than other chemical signals, such as hormones that rely on slow diffusion mechanisms across greater distances [20, 23, 24].

The combination of both modes of transmission allows rapid communication between distant neurons throughout a biological neural network.

2.1.2 Complexity of Biological Neural Networks

One of the main challenges in modelling biological neural networks is their inherent complexity. The human brain contains approximately 100 billion neurons, each one connected to thousands of others through an intricate web of synapses. This vast connectivity gives rise to non-linear and complex behaviour throughout the entire network and individual neuron activity [21]. Several factors contribute to this complexity:

1. *Multiscale organisation*: Biological neural networks operate at multiple spatial and temporal scales. From molecular processes within synapses to large-scale interactions between brain regions, it is unclear which level of resolution is most critical for capturing the primary computational mechanisms [25].

2. *Chemical dynamics*: Neurotransmitters' release, diffusion, and reuptake at the synapses involve complex chemical reactions that can impact signal transmission and synaptic plasticity. Accurately modelling these processes requires accounting for many molecular species, reaction rates, and spatial constraints [21, 26].

3. *Adaptation*: Neurons in biological neural networks can adapt their properties over time based on input patterns, leading to changes in connectivity and function. This plasticity enables learning and memory but adds another complexity layer to modelling efforts [27]

4. *Noise and variability*: Intrinsic noise sources within individual neurons (e.g., stochastic ion channel opening), as well as extrinsic variability across neuronal populations (e.g., heterogeneity in cell properties), can influence network behaviour [21, 28].

5. *Emergent properties*: The collective behaviour of neurons in biological neural networks can give rise to emergent properties that are not easily predicted from the behaviour of individual cells. Identifying these phenomena and their underlying causes is a critical challenge in modelling efforts [29].

Biological neural networks can learn many tasks without catastrophic interference (see Chapter 3) in most cases. However, memory consolidation and retention in biological neural networks is an intricate, not fully understood process. Synaptic plasticity (i.e. changes in synaptic strength) is pivotal in memory. Synaptic plasticity manifests as long-term potentiation (LTP) and long-term depression (LTD) in biological neural networks [30]. The exact mechanisms underlying the transition from short-term to long-term memories remain subjects of active research. Furthermore, the significant role of sleep in memory consolidation, especially during phases like rapid eye movement (REM) sleep, adds another layer of complexity [31].

2.1.3 Modeling Challenges and Outlook

Given biological neural networks' complexity, developing accurate and comprehensive models is a formidable task. Furthermore, determining which aspects of these networks are most crucial for capturing their computational abilities remains an open question. Despite these challenges, artificial intelligence (AI) research has progressed by developing artificial neural networks (ANNs) inspired by their biological counterparts [11, 32, 33]. While ANNs are highly simplified compared to their biological counterparts, they have demonstrated remarkable success in various machine learning tasks [34, 35, 36]. The practical achievements of modern ANNs suggest that the efforts to understand biological neural networks were worthwhile.

Advances in experimental techniques for probing neuronal structure and function at multiple scales will enable better characterisation of biological neural networks. Likewise, theoretical developments in modelling could allow researchers to capture this complexity more accurately. Through a greater understanding of biological neural networks, we can make significant strides towards unravelling the mysteries of these remarkable systems. Harnessing the computational power of the human mind is the dream for advanced artificial intelligence technologies.

2.2 Background on Artificial Neural Networks

ANNs are computational systems inspired by biological neural networks. The development of ANNs has a rich history, with researchers devoting substantial effort to understanding and reverse-engineering biological intelligence over several decades. This section provides an overview of the evolution of ANNs, discussing their essential components and various types and addressing some historical challenges and limitations. This research explores novel ANN architectures that are parameter-efficient, easy to train, universally expressive and do not suffer from catastrophic interference. It is necessary to review all significant developments up to this point.

2.2.1 Early Developments and Limits

ANNs have a storied history, dating back to the early days of artificial intelligence research when McCulloch and Pitts proposed the first mathematical model of a biological neuron called "nerve nets" in 1943. Initial attempts aimed to map propositional logic to neural networks [37]. This foundational work laid the groundwork for the development of artificial neural networks. In the subsequent years, researchers like Hebb [38], Rosenblatt [32], and Werbos [39] contributed significantly to the understanding and refinement of these models.

Hebb's groundbreaking contribution has been summarised as "cells that fire together wire together" in later years [40], providing a framework for understanding how synaptic connections between neurons could be strengthened or weakened based on their activity patterns, leading to learning and memory formation.

The development of the perceptron by Rosenblatt marked a pivotal moment in ANN history, as it established a simple yet powerful algorithm for learning linearly separable patterns. The perceptron was one of the first ANNs [32]. The perceptron model was significant because it demonstrated how simple artificial neurons could be combined into a network to perform basic pattern recognition tasks [41].

However, the perceptron's limitations were exposed by the analysis of Minsky and Papert in 1969 [42]. It was shown that the perceptron model could not compute certain predicates like XOR. This problem demonstrated that single-layer perceptrons were insufficient for solving more complex tasks, highlighting the need for multi-layered neural networks with non-linear activation functions. The XOR problem became historically significant as it led to advances in multi-layer neural networks and backpropagation algorithms for learning in deeper networks [43].

The limitations of perceptrons caused a decline in interest in neural networks until Rumelhart, Hinton, and Williams popularised the backpropagation algorithm in 1986 [44, 45]. This breakthrough revitalised interest in ANNs and spurred further advancements shaping modern deep learning techniques.

2.2.2 Artificial Neurons

Modern ANNs consist of interconnected units called neurons that process and transmit information like a directed graph. These neurons are typically organised into clusters or layers with similar connectivity patterns. Neurons receive input from other neurons, perform some computations, and transmit the results to other neurons in another layer. The parameters associated with connections and neurons within modern ANNs are referred to as weights and biases. The naming convention originates from older statistical models like logistic regression with statistical weights and subtracting bias terms for unbiased estimates. In modern ANNs, weights represent the strength of connections between neurons, while biases are used to adjust the intercept of the hyperplane modelled by each neuron [43]. Artificial neurons are a helpful abstraction from the unfathomable complexity of biological neurons [46, 47].

2.2.3 Diversity of Artificial Neural Networks

Various types of ANNs with different connectivity patterns exist, including feedforward networks, convolutional neural networks (CNNs), recurrent neural networks (RNNs) and transformers with embedding layers [43, 35]. Different architectures were developed for particular problems. ANNs can be applied to numerous tasks such as pattern recognition, machine translation, and game playing [36].

2.2.4 Feedforward Neural Networks

Feedforward neural networks (FFNNs) are a foundational concept in machine learning, characterised by their interconnected layers of neurons and feedforward architecture. They can be described as directed acyclic graphs in graph theory terminology, consisting of input, hidden, and output layers that process and transform input data through non-linear transformations [43, 48, 49]. The feedforward design allows for efficient computation while maintaining a more straightforward structure than other architectures, such as RNNs. Figure 2.2 shows an example of a simple FFNN.

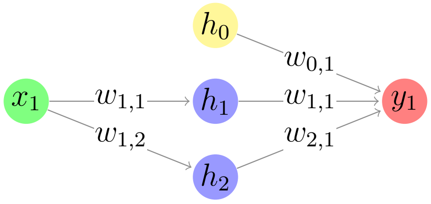


Figure 2.2: A feedforward neural network is a directed acyclic graph.

FFNNs have been successfully applied across various domains, including computer vision, natural language processing, bioinformatics, and finance. Their ability to extract useful features from raw data and learn intricate representations has made them suitable for various applications within these fields. FFNNs remain an essential building block in modern deep learning techniques despite their relative simplicity. FFNNs with hidden layers are also called multi-layer perceptrons (MLPs). FFNNs are ubiquitous in many modern architectures. This study focuses on analysing FFNNs, and new FFNNs are developed, using convolutional layers and embedding layers to perform necessary computations.

2.2.5 Convolutional Neural Networks

CNNs are a class of deep learning algorithms designed to process data with grid-like topology, such as images. They have gained significant attention due to their exceptional performance in various computer vision tasks, including image classification, object detection, and segmentation [50, 43]. Inspired by the human visual cortex, CNNs automatically and adaptively learn spatial hierarchies of features from input data. The architecture of a typical CNN consists of an input layer, multiple hidden layers comprising convolutional layers, pooling layers, MLPs, and an output layer.

The core component of a CNN is the convolutional layer that serves as a feature extractor. Each neuron in this layer is connected to a local receptive field in the previous layer through a set of learnable weights or filters called kernels. These kernels slide over the input data in a process called convolution, generating feature maps representing different aspects of the input [50]. The pooling layers perform down-sampling operations such as max or average pooling to reduce spatial dimensions while preserving important features. This reduces computational complexity and assists in achieving translation invariance by providing an abstracted representation of the input [43]. An example of such an architecture is AlexNet (see Figure 2.3), which was introduced in a 2012 NeurIPS paper and achieved breakthrough results on image classification [50].

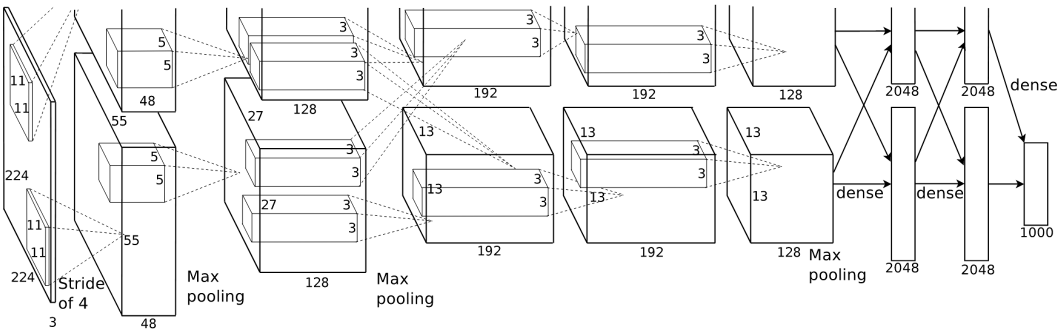


Figure 2.3: AlexNet CNN architecture from the 2012 NeurIPS paper [50].

One key advantage of CNNs is their ability to learn complex hierarchical feature representations without manual feature engineering. As information propagates through successive layers, low-level features such as edges and textures are extracted initially, while higher-level semantic features emerge at deeper layers. This hierarchical structure makes CNNs suitable for handling large-scale image recognition tasks where intricate patterns are discerned from raw pixel values.

One-dimensional convolutional layers can be adapted for single-variable function approximation with cardinal cubic B-splines or Fourier series. This study implements and uses cardinal cubic B-splines computed with convolutional layers.

2.2.6 Embdedding Layers

Embedding layers provide a compact and efficient way of representing high-dimensional categorical data, such as words in natural language processing or users and items in recommendation systems. These layers map discrete input tokens (e.g., word indices) to dense vectors of fixed size, thereby serving as a continuous representation that can be fed into downstream neural network models. The primary purpose of embedding layers is to preserve and learn semantic structure in sequential data to improve the generalisation capability of deep learning models [51].

The structure of an embedding layer consists of a weight matrix with dimensions (V, D), where V is the vocabulary size (number of unique tokens) and D denotes the desired embedding dimension. Each row in this matrix corresponds to an individual token's embedding vector, and they are typically initialised randomly. Embedding vectors are trained with backpropagation to capture meaningful relationships among input tokens. It is worth noting that pre-trained embeddings, such as Word2Vec or GloVe embeddings, can be used to initialise weights [52, 53, 54, 55].

An embedding layer needs $O(V \times D)$ storage for all the weights. However, sparse matrix operations can reduce memory consumption since only a tiny subset of tokens is typically active during training or inference in any given input batch. The time complexity for retrieving an individual token's embedding vector from an embedding layer is $O(1)$, assuming constant-time indexing into a weight matrix. Embedding layers are crucial in modern deep learning, providing a computationally efficient and semantically rich representation for categorical data [51]. Embedding layers have been used extensively in the Transformer architecture [35]. Embedding layers are widely available in modern machine learning libraries and are used in the models developed for this study.

2.3 Training Techniques

One significant advantage of ANNs, compared to symbolic or rule-based AI such as expert systems, is their ability to learn from noisy data. ANNs can "learn" to perform a task by discovering patterns in the training data that map input to output, adjusting the weights and biases accordingly. Supervised training involves modifying these weights and biases to minimise a loss function, which measures the difference between the model-predicted output and the desired output. Typically, an optimisation or training algorithm is employed in this process.

2.3.1 Gradient Descent Algorithms

Gradient descent optimisation algorithms can be efficiently implemented with a message-passing scheme called backpropagation, or backprop [43]. Two widely used optimisers are stochastic gradient descent (SGD) and an adaptive moment estimate variant of stochastic gradient descent called Adam [12]. Alternative training techniques such as Hebbian learning or evolutionary computation have also been explored [38, 56]. Alternative approaches provide different ways of learning and adapting ANN weights and biases based on other principles and criteria.

Stochastic Gradient Descent

SGD is a popular optimisation algorithm for training ANNs that estimates the gradient of the loss function with respect to the model's weights and biases using a random subset, or (mini) batch, of training samples at each iteration. The estimated gradient is then used to update the model's parameters in the direction that minimises the loss function [43, 57].

Formally, given a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i represents the input and y_i represents the corresponding desired output, SGD updates the model's parameters θ iteratively as follows:

$$\theta^{(t+1)} = \theta^{(t)} - \eta\hat{g}^{(t)} = \theta^{(t)} - \eta\left(\frac{1}{B}\sum_{i=1}^{B}\nabla_\theta L(\theta^{(t)}, x_i, y_i)\right), \tag{2.1}$$

where η is the learning rate that determines the step size of each update, $L(\cdot)$ is the loss function that quantifies the difference between the model's predicted output and the desired output for a given input, ∇_θ denotes the gradient with respect to θ, and B is the batch size. The gradient estimate $\hat{g}^{(t)}$ is an average of gradients $\nabla_\theta L(\theta^{(t)}, x_i, y_i)$ over the subset or batch of training data.

SGD is computationally efficient since it only requires a small subset of training samples at each iteration. However, this can introduce high variance in estimating the true gradient, which may lead to slow convergence or oscillation during training. To address some of the issues with SGD, various modifications to SGD have been proposed.

Adam

Adam (from adaptive moment estimation) is an optimisation algorithm combining ideas from momentum-based methods that 'remember' prior gradient updates (see the original backprop paper by Rumelhart and Hinton) and RMSprop [44, 12, 58]. Adam aims to provide adaptive learning rates for different parameters by estimating the gradients' first-order moments (the mean) and second-order moments (the uncentered variance).

The update rule for Adam involves maintaining an exponentially decaying average of past gradients (m_t) and past squared gradients (v_t). These averages are then used to compute bias-corrected estimates of the first-order moment ($\hat{m}_t$) and second-order moment ($\hat{v}_t$). Finally, the model's parameters are updated based on these estimates. Formally, the Adam update rule is given by:

$$m^{(t+1)} \leftarrow \beta_1 m^{(t)} + (1-\beta_1)\hat{g}^{(t)} = \beta_1 m^{(t)} + (1-\beta_1)\left(\frac{1}{B}\sum_{i=1}^{B} \nabla_\theta L(\theta^{(t)}, x_i, y_i)\right) \tag{2.2}$$

$$v^{(t+1)} \leftarrow \beta_2 v^{(t)} + (1-\beta_2)(\hat{g}^{(t)})^2 = \beta_2 v^{(t)} + (1-\beta_2)\left(\frac{1}{B}\sum_{i=1}^{B} \nabla_\theta L(\theta^{(t)}, x_i, y_i)\right)^2 \tag{2.3}$$

$$\hat{m} = \frac{m^{(t+1)}}{1-\beta_1^t} \tag{2.4}$$

$$\hat{v} = \frac{v^{(t+1)}}{1-\beta_2^t} \tag{2.5}$$

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta\frac{\hat{m}}{\sqrt{\hat{v}} + \epsilon} \tag{2.6}$$

where β_1 and β_2 are exponential decay rates for the first and second moments, respectively, and ϵ is a small constant added to the denominator for numerical stability in the Adam update rule. The gradient estimate $\hat{g}^{(t)}$ is an average of gradients $\nabla_\theta L(\theta^{(t)}, x_i, y_i)$ over the subset or batch of training data.

Adam has been shown to perform well in practice and is widely used in deep learning. It combines the advantages of momentum-based methods and adaptive learning rates. However, it has been noted that Adam sometimes leads to poorer generalisation on unseen data than SGD but still converges quickly on training data [59].

2.3.2 Loss Functions and Metrics

Loss functions are fundamental in various machine learning algorithms as they quantify the discrepancy between predicted and actual or target values. Loss functions are also called error functions in some sources [43]. These functions are crucial for optimising model parameters by calculating how well a model fits a dataset. Three commonly used loss functions are Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2 score. The MAE, defined in Equation (2.7), is a straightforward loss function that calculates the average absolute difference between predicted and actual values [43, 60, 61, 62].

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2.7}$$

where n represents the number of data points, y_i denotes the actual value, and $\hat{y}_i$ represents the predicted value. MAE provides an intuitive interpretation of how far off the predictions are from actual values on average without considering their (±) direction. In contrast to MAE, MSE, defined in Equation (2.8), penalises larger errors due to its squared term [43, 60].

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.8}$$

The squaring operation in MSE amplifies larger errors more than smaller ones. Consequently, models optimising MSE prioritise reducing outliers or extreme errors, which may increase sensitivity to anomalies within the dataset. Another commonly used metric for evaluating regression models is the R^2 score, also known as the coefficient of determination. The R^2 score, defined in Equation (2.9), is related to MSE. The R^2 score measures relative error by comparing model predictions with a baseline model that always predicts the mean value of the target variable. It quantifies the proportion of variance in the dependent variable that is explained by the independent variables [63, 64, 65, 66]. The R^2 score is given by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.9}$$

where $\bar{y}$ represents the mean value of the target variable. An R^2 score closer to 1 indicates that the model captures a more significant proportion of data variability. In contrast, an R^2 score equal to or less than zero implies that the model fails to explain any variability beyond what can be achieved by predicting the mean value.

In summary, loss functions such as MAE, MSE, and R^2 provide quantitative measures of how well a model fits a given dataset. Each metric prioritises different facets of how a model falls short in prediction. The choice of loss function depends on specific modelling objectives and priorities.

2.3.3 Data and Training Augmentation

Data augmentation and regularisation techniques attempt to improve model performance and generalisation. The methods are varied and often problem-specific. Manifold Mixup regularisation encourages models to adopt smoother approximations [67, 68, 69]. Pseudo-rehearsal is another training augmentation technique that has been used for continual learning problems [8, 70], see Section 3.3 in Chapter 3 for more details.

2.4 Emergent Challenges with ANNs

During training with gradient descent optimisers, ANNs may face challenges such as exploding or vanishing gradients that limit their practical use. It took many years of research, engineering and fine-tuning to develop architectures and activation functions that sidestep gradient-related problems. For instance, deep learning techniques such as dropout [71], batch normalisation [72], and modern activation functions such as rectified linear unit (ReLU), defined in Equation (2.10), have helped mitigate these issues [73].

$$\mathrm{ReLU}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{2.10}$$

Other emergent phenomena were observed experimentally. It has been noted that ANNs have brittle memories, causing catastrophic interference when learning new tasks sequentially. Chapter 3 discusses continual learning and catastrophic interference in detail.

2.5 Proposed Modifications

The neurons or nodes inside modern ANNs have a ubiquitous mathematical structure. Each neuron executes the same type of computation: multiplying its inputs x_i by their associated weights w_i, adding them together, shifting the weighted sum with a bias term b, and evaluating a non-linear activation function $\sigma(x)$ on the result to obtain the neuron's output or activation [43]. The overall computation is expressed in Equation (2.11)

$$\text{Neuron Output} = \sigma\left(\sum_{i=1}^{n} w_i x_i \; + \; b\right) \tag{2.11}$$

In contrast, the model proposed in this study replaces the linear functions $w_i x_i$ with non-linear single-variable functions $\phi_i(x_i)$ and fixes the activation function $\sigma(x) = \exp(x)$, as shown in Equation (2.12).

$$\text{Neuron Output} = \exp\left(\sum_{i=1}^{n} \phi_i(x_i) \; + \; b\right) \tag{2.12}$$

This modification aims to enhance the expressive power of ANNs by incorporating non-linear single-variable functions instead of linear ones. Further details on the proposed models will be presented in Chapter 4, Chapter 5, and Chapter 6.

2.6 Summary

This chapter outlined the key ideas and research developments on ANNs. ANNs were inspired by biological neural networks. ANNs are computational models that can approximate continuous functions. Section 2.1 reviewed biological neural networks. The history of ANNs was outlined in Section 2.2. Section 2.3 discussed training techniques. The discussed training techniques included loss functions such as MAE, MSE and R^2, optimisers like SGD and Adam, in addition to augmentation techniques. Some problems emerged with ANNs in practice, as recounted in Section 2.4. Section 2.5 outlined some similarities and differences between conventional ANNs and proposed architectures and modifications in this study. The emergent phenomenon of catastrophic interference in ANNs is further examined in Chapter 3.

Chapter 3

Catastrophic Interference and Continual Learning

This chapter examines catastrophic interference or catastrophic forgetting and how it undermines continual learning. Existing explanations for catastrophic interference are reviewed, and a deeper analysis of the underlying dynamics is provided. Section 3.1 explains how trainable models can be susceptible to catastrophic interference. Section 3.2 shows how single-variable function approximators can be vulnerable to catastrophic interference. An augmented training technique to facilitate continual learning called pseudo-rehearsal is described in Section 3.3. Another augmentation technique, orthogonal gradient descent (OGD), is discussed in Section 3.4. Section 3.5 defines and discusses metric spaces and *distance functions*. Section 3.6 explains that lookup tables are robust to catastrophic interference due to *distal orthogonality*. Section 3.7 explains how distal orthogonality leads to learning without *distal interference*. Section 3.8 outlines a mathematical proof that shows the limits of model-only continual learning. A few aims and ideas are drafted following the initial analysis in Section 3.9. A summary of the most important findings is given in Section 3.10.

3.1 Susceptibility to Catastrophic Interference

Catastrophic interference is an emergent phenomenon where a machine learning model, such as an ANN, learns a new task, and the subsequent parameter updates interfere with the model's performance on previously learned tasks [1]. Catastrophic interference is also called catastrophic forgetting in more recent studies to highlight interference over time or tasks [6, 7, 8]. If an ANN cannot effectively learn many tasks, it has limited utility in continual learning [9, 10]. Catastrophic interference is like learning how to jump while forgetting how to land. Even linear functions are susceptible to catastrophic interference, as illustrated in Figure 3.1. A linear model's bias and slope are adjusted to accommodate the first task in blue. The model was modified after training on the second task, as shown in orange. The linear model changed over its entire domain.

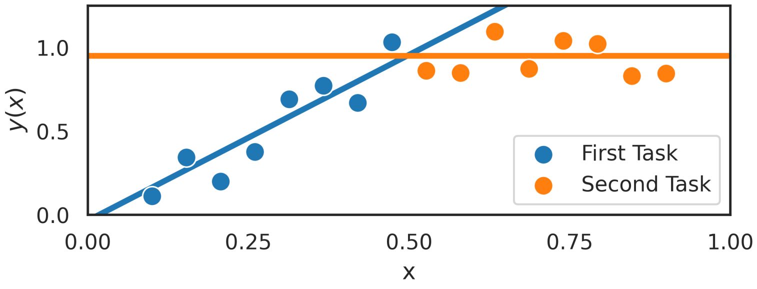


Figure 3.1: A linear function is susceptible to catastrophic interference.

The simple example of catastrophic interference with a linear regression model might be due to the non-linear target function, noise, changing data distributions, or parameter sharing across the input. Sharing parameters across the model's domain means that a gradient update with backpropagation at one input point potentially changes the model's output almost everywhere. The scale of this interference depends on the problem at hand. One can equivalently say that input points have **overlapping representations** when they share the same trainable parameters.

Linear functions are ubiquitous in ANN models, and it is well-known that ANN models are susceptible to catastrophic interference. This study will experimentally demonstrate that ReLU ANNs are sensitive to catastrophic interference [74, 75]. Other function approximation techniques are also susceptible to catastrophic interference.

3.2 Global Parameters in Basis Functions

Basis functions allow one to represent or approximate single-variable functions. Basis functions are linearly independent and constitute a *basis* for the vector space of continuous functions, hence the name. The Fourier series is composed of an additive term $c \in \mathbb{R}$ and sine and cosine basis functions with different frequencies for $2\pi k$, and arbitrary coefficients $a_k, b_k \in \mathbb{R}$ for all $k \in \mathbb{N}$. The Fourier basis for functions on the unit interval $x \in [0, 1]$ is given by:

$$f(x) = c + \sum_{k=1}^{K} a_k \sin(2\pi kx) + b_k \cos(2\pi kx) \tag{3.1}$$

Polynomial functions constitute a set of basis functions on $x \in \mathbb{R}$. A polynomial series is composed of an additive term $c \in \mathbb{R}$ and different powers x^k, and arbitrary coefficients $a_k \in \mathbb{R}$ for all $k \in \mathbb{N}$:

$$f(x) = c + \sum_{k=1}^{K} a_k x^k \tag{3.2}$$

Trigonometric and polynomial functions are susceptible to catastrophic interference. This susceptibility is demonstrated visually in Figure 3.2a and Figure 3.2b. From left to right, one has trigonometric and polynomial functions. Adjusting the coefficients of trigonometric or polynomial functions affects nearly all outputs globally and is not localised. The exception is points where the basis functions are zero at a nodal point.

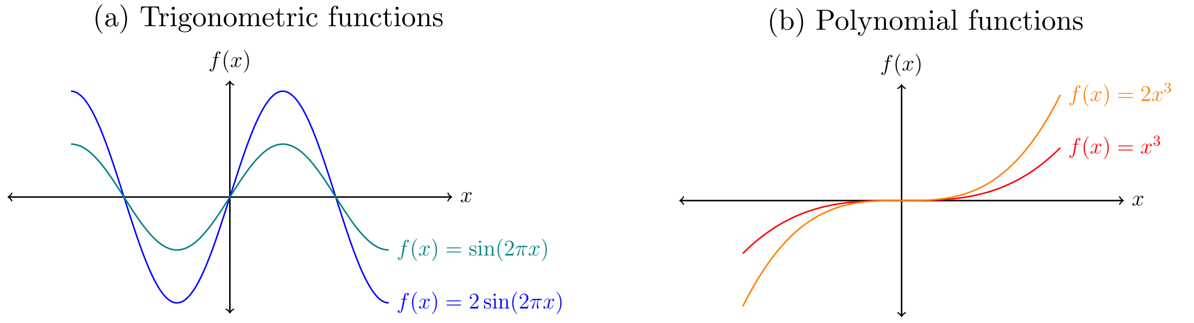


Figure 3.2: Commonly used trigonometric and polynomial basis functions are susceptible to catastrophic interference.

3.3 Pseudo-Rehearsal Training Augmentation

Common techniques for continual learning include reviewing previously seen data. Numerous review and pseudo-rehearsal techniques exist [8, 70]. The main advantage of pseudo-rehearsal is that one does not need to store old training data. Instead, one relies on the model's ability to "memorise" previous training data, which are re-sampled and mixed with training data on new tasks. Some techniques like generative replay use generative models like generative adversarial networks [70]. The expected risk R[f] of a model f is evaluated by integrating the loss function $\ell(f(x), y)$ over the relevant data distribution $P(x, y)$, as shown in Equation (3.3) below:

$$R[f] = \int \ell(f(x), y)\mathrm{d}P(x, y) \tag{3.3}$$

Pseudo-rehearsal explicitly refers back to the model's previous state and is similar to the ideas related to generative replay, except pseudo-rehearsal assumes a uniform distribution over the inputs $x \in D(f)$ [70]. Generative replay estimates the distribution over inputs $x \in D(f)$ with a generative model. Equation (3.4) shows the discrete-time integral equation that governs pseudo-rehearsal.

$$R[f_{t+1}] = \rho \int \ell(f_{t+1}(x), y)\mathrm{d}P(x, y) + (1 - \rho) \int \ell(f_{t+1}(v), f_t(v))\mathrm{d}\mathcal{P}(v) \tag{3.4}$$

The loss on the current task is given by $\ell(f_{t+1}(x), y)$, where x and y denote the input and target values. The distribution $P(x, y)$ represents the data distribution of the current task. The loss $\ell(f_{t+1}(v), f_t(v))$ quantifies the change of the model. The scalar mixing coefficient $\rho \in [0, 1]$ controls the trade-off between novel learning and memory retention. The distribution $\mathcal{P}(v) = \mathcal{P}_{D(f)}$ of the input data with $v \sim \mathcal{P}_{D(f)}$ could be given by a generative model or a uniform distribution $v \sim U_{D(f)}$ over the domain of f denoted $D(f)$. Using a uniform distribution is less computationally demanding than training a generative model but yields poor performance in higher dimensions. Generative replay was developed due to the degraded performance in high-dimensional models [70]. The integral expression $\int \ell(f_{t+1}(v), f_t(v))\mathrm{d}\mathcal{P}(v)$ is similar to $\int_{D'} |f'(x) - f(x)|\, dx$ in the definition of distal interference shown in Equation (3.10).

The key idea is to minimise the loss evaluated on new data, subject to the constraint that the model retains its input-output mapping over the rest of its domain. A simple version of pseudo-rehearsal is demonstrated visually in Figure 3.3.

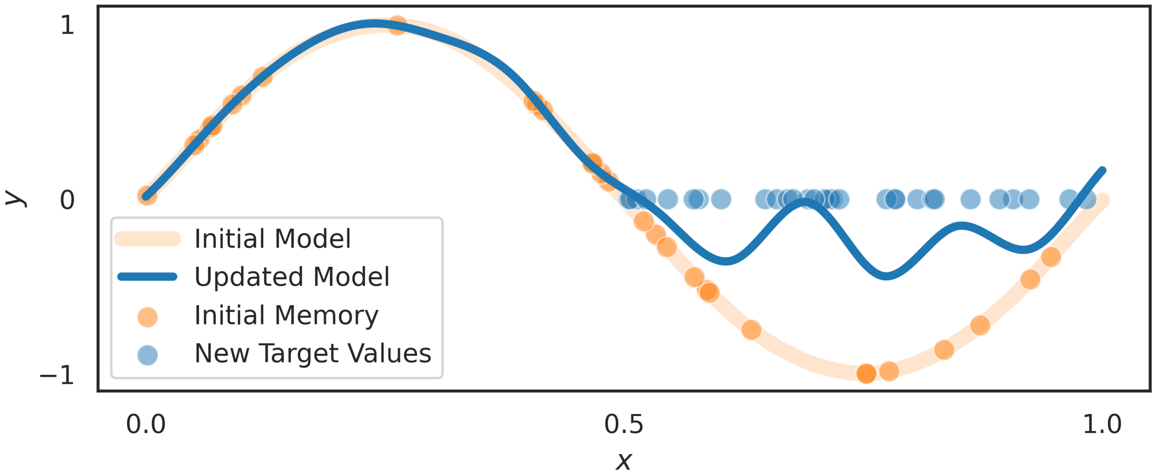


Figure 3.3: Demonstration of pseudo-rehearsal for a one-variable function.

The initial model was trained to fit a sine function on the interval $[0, 1]$. The new task is uniformly sampled on the interval $[0.5, 1]$, and all the target values are zero. The pseudo-rehearsal data is constructed by uniformly sampling over the entire domain $[0, 1]$ with target values equal to the model output **before** training. The mixing coefficient is chosen as $\rho = 0.5$, and the loss function is chosen to be the mean absolute error. Note that the number of data samples is kept low to make it easier to visualise the data and understand the process used in pseudo-rehearsal.

3.4 Orthogonal Gradient Descent

Attempts at combating catastrophic interference and enabling continual learning include augmented training techniques. Orthogonal gradient descent (OGD) is one example [16]. The main idea behind OGD is to process parameter gradient updates to the model so that it is orthogonal to the parameter gradients of previously observed data. The orthogonality ensures minimal change to the model's output at previously learned data.

The explanation follows: Consider a scalar function $f(\theta, v)$ with trainable parameters θ evaluated on the input vector v. The local linear approximation (assuming the tangent plane exists) is given in Equation (3.5).

$$f(\theta, v) \approx f(\theta_0, v) + \nabla_\theta f(\theta_0, v) \cdot (\theta - \theta_0), \tag{3.5}$$

where $\nabla_\theta f(\theta_0, v)$ denotes the parameter gradient evaluated at v, and the initial parameters θ_0. The updated parameters θ are given in terms of the initial parameters and a gradient update at the point v where $\theta = \theta_0 - \eta\hat{g}$ with learning rate η.

Suppose that $\hat{g}$ is **not** orthogonal to $\nabla_\theta f(\theta_0, v)$. The model will hopefully have an improved output value at v after the update, as shown in Equation (3.6).

$$\begin{aligned} f(\theta, v) &\approx f(\theta_0, v) + \nabla_\theta f(\theta_0, v) \cdot (\theta - \theta_0) \\ &= f(\theta_0, v) + \nabla_\theta f(\theta_0, v) \cdot ((\theta_0 - \eta\hat{g}) - \theta_0) \\ &= f(\theta_0, v) - \eta\nabla_\theta f(\theta_0, v) \cdot \hat{g} \\ &\neq f(\theta_0, v) \end{aligned} \tag{3.6}$$

This is an expected finding. The same local linear approximation can be used to explain how OGD can prevent catastrophic interference. A similar approach can be used to estimate how a model will not change at some other point using only the parameter gradients of the model. This is shown in Equation (3.7).

Suppose that x is a previously learned input whose output one seeks to preserve despite the training at v. If $\hat{g}$ is orthogonal to $\nabla_\theta f(\theta_0, x)$, then the model will have a nearly unaltered value at x after the update, as shown in Equation (3.7).

$$\begin{aligned} f(\theta, x) &\approx f(\theta_0, x) + \nabla_\theta f(\theta_0, x) \cdot (\theta - \theta_0) \\ &= f(\theta_0, x) + \nabla_\theta f(\theta_0, x) \cdot ((\theta_0 - \eta\hat{g}) - \theta_0) \\ &= f(\theta_0, x) - \eta \nabla_\theta f(\theta_0, x) \cdot \hat{g} \\ &= f(\theta_0, x) \end{aligned} \tag{3.7}$$

OGD generalises this idea to many training instances instead of just one, but the concept remains the same. Orthogonal parameter gradients can prevent catastrophic interference. The inner product $\nabla_\theta f(\theta_0, x) \cdot \hat{g}$ from Equation (3.7) is similar to $|\nabla_\theta f(x) \cdot \nabla_\theta f(v)|$ in the definition of distal interference shown in Equation (3.10).

3.5 Metric Spaces and Dissimilarity

A *metric space* is a fundamental and general mathematical concept that provides a rigorous framework for studying the notions of distance, convergence, and continuity. It is beneficial for work in computational geometry, pattern recognition, and machine learning, where measuring similarity or dissimilarity between objects is essential.

A metric space consists of a set X and a real-valued function $d : X \times X \to \mathbb{R}$ called the *metric* or (general) *distance function*, which satisfies the following axioms:

1. Non-negativity: $\forall x, y \in X,\ d(x, y) \geq 0$
2. Identity of indiscernibles: $\forall x, y \in X,\ d(x, y) = 0 \iff x = y$
3. Symmetry: $\forall x, y \in X,\ d(x, y) = d(y, x)$
4. Triangle inequality: $\forall x, y, z \in X,\ d(x, z) \leq d(x, y) + d(y, z)$

A critical aspect of metric spaces is their generality. The familiar Euclidean distance in $\mathbb{R}^n$ is just one example of a metric; many other possible metrics can be defined on various sets depending on the application. For instance, consider the discrete metric on an arbitrary set X, defined by $d(x, y) = 1$ if $x \neq y$ and $d(x, y) = 0$ if $x = y$.

The maximum absolute difference or maximum norm distance $\max_i(|x_i - y_i|)$ satisfies all the axioms for a metric. It is based on the maximum or supremum norm.

In contrast, the minimum absolute dissimilarity $\min_i(|x_i - y_i|)$ does not satisfy the identity of indiscernibles nor the triangle inequality. Therefore, it is not a proper metric or distance function but a dissimilarity measure. The minimum absolute dissimilarity is at least symmetric and non-negative.

3.6 Continual Learning with Lookup Tables

This study seeks to identify the properties that endow lookup tables with robust memory and to transfer them to ANNs. Lookup tables in this study are universal function approximators that partition the domain and map inputs from each partition to a (trainable) value or output. Lookup tables can store output values for individual points, but the partition or interval variant is more appropriate for this study. Lookup tables are robust to catastrophic interference [15]. It has been noted that lookup tables and connectionist models like ANNs are on opposite sides of a stability-plasticity spectrum. However, a deeper explanation is needed. The notion of overlapping representations is helpful and can be applied to any trainable model. The overlap of representations corresponds to the inner product of the parameter gradients.

Definition 1 (overlapping representation)**.** *Two points $x, v \in D(f)$ have overlapping representation in a model $f(x)$ with trainable parameters θ if:*

$$\nabla_\theta f(x) \cdot \nabla_\theta f(v) \neq 0$$

Remark 2. *Distance is not considered in the definition of overlapping representations. Heuristically, only points close to each other should have overlapping representations.*

Definition 3 (distal orthogonal model)**.** *Let f(x) be a differentiable model. Given some fixed $\delta > 0$ and non-negative dissimilarity or distance measure $d(x, v)$, a model $f(x)$ is called distal orthogonal w.r.t. $d(x, v)$, if for all $\theta \in \mathbb{R}^p$ and $\forall x, v \in D(f)$:*

$$d(x, v) > \delta \implies \nabla_\theta f(x) \cdot \nabla_\theta f(v) = 0$$

Lookup tables have minimal overlap between distinct points, and this fundamental property is best described as max-distal orthogonality.

Definition 4 (max-distal orthogonal model)**.** *Let f(x) be a differentiable model. Given some $\delta > 0$, a model $f(x)$ is max-distal orthogonal if for any trainable parameters $\theta \in \mathbb{R}^p$ and $\forall x, v \in D(f)$:*

$$\max_i(|x_i - v_i|) > \delta \implies \nabla_\theta f(x) \cdot \nabla_\theta f(v) = 0$$

A lookup table that uniformly partitions the unit hypercube with partition number $z \geq \delta^{-1}$ can distinguish between points $x, v \subset [0, 1]^n$ that differ sufficiently in any **one** of their coordinates and associates independent trainable parameters with no overlapping representation. A consequence of this property is that lookup tables can sequentially learn many input-output mappings without catastrophic interference. This property can be easily visualised for two-dimensional points, as shown in Figure 3.4. The two points x, v have one set of coordinates that differ by more than δ. This property holds for higher dimensions, but two dimensions are easier to visualise.

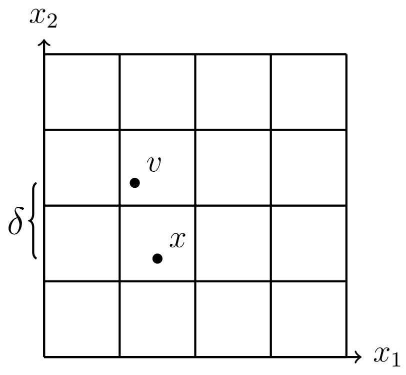


Figure 3.4: Two sufficiently different points to guarantee max-distal orthogonality.

Lookup tables are immune to catastrophic interference because of non-overlapping representations. Lookup tables can continually learn without augmentation like pseudo-rehearsal. Lookup tables have a computational complexity that scales exponentially with the input dimension. Lookup tables generalise poorly with limited training data. Consequently, lookup tables are not practical for high-dimensional problems. Max-distal orthogonality does not require exponentially large models since gradients can be zero.

3.7 Learning Without Distal Interference

Distal interference occurs when parameter updates to change a model output at a specific point affect the model outputs far from the training data point of interest. This non-local interference is an underlying mechanism that can cause catastrophic interference in a continual learning context. This section shows that models with distal orthogonality can learn without distal interference. The mathematical ideas used in this section are similar to neural tangent kernels by Jacot et al. [76]. A model $f(\theta_t, x)$ with initial parameters θ_t can be trained with gradient descent optimisation to create an updated model $f(\theta_{t+1}, x)$. The local linear approximation of a d-distal orthogonal model $f(\theta_{t+1}, x)$ with explicitly shown parameters θ is given by:

$$f(\theta_{t+1}, x) \approx f(\theta_t, x) + \nabla_\theta f(\theta_t, x) \cdot (\theta_{t+1} - \theta_t)$$

Let $\theta_{t+1} = \theta_t - \eta \hat{g}$ be the updated parameters of the model trained at some point $v \in D(f)$, with learning rate η. Let $\hat{g} = (\partial_f L) \nabla_\theta f(\theta_t, v)$, where $\partial_f L$ is the derivative of the loss function w.r.t. the model f, and $\nabla_\theta f(\theta_t, v)$ be the parameter gradient of the model evaluated at v. Then it follows that:

$$\begin{aligned} f(\theta_{t+1}, x) &\approx f(\theta_t, x) + \nabla_\theta f(\theta_t, x) \cdot (\theta_t - \eta \hat{g} - \theta_t) \\ &= f(\theta_t, x) - \eta \, \nabla_\theta f(\theta_t, x) \cdot \hat{g} \\ &= f(\theta_t, x) - \eta \, (\partial_f L) \nabla_\theta f(\theta_t, x) \cdot \nabla_\theta f(\theta_t, v) \end{aligned}$$

Assume that $d(x, v) > \delta$, then for any θ it follows:

$$d(x, v) > \delta \implies \nabla_\theta f(\theta, x) \cdot \nabla_\theta f(\theta, v) = 0$$

Since $\nabla_\theta f_t(x) \cdot \nabla_\theta f_t(v) = 0$, it follows that:

$$f(\theta_{t+1}, x) \approx f(\theta_t, x) \tag{3.8}$$

Thus, a model with distal orthogonality can learn with gradient descent at a point v without affecting the values at a distant point x in the domain. Distal interference can deleteriously affect a model's ability to learn continuously. Distal interference is most troublesome when two distant points have large overlapping representations. The required model size for robustness to distal interference still needs to be determined. Such insight could shed light on the computational complexity bounds for models capable of continual learning.

One can reformulate this idea and drop the explicit dependence on θ. A model f can be trained with gradient descent optimisation to create an updated f'. If f is trained at one point $v \in D(f)$, then $f(x)$ may change at some distant point $x \in D(f)$. Assume that $f'(x) \approx f(x) - \eta \nabla_\theta f(x) \cdot \hat{g}$ from Equation (3.7). One can quantify the difference over the domain with a function norm as shown in Equation (3.9) and the local linear approximation.

$$\begin{aligned} \text{Model Perturbation} := \|f'(x) - f(x)\|_1 &= \int_D |f'(x) - f(x)|\, dx \\ &\approx \int_D |(f(x) - \eta \nabla_\theta f(x) \cdot \hat{g} - f(x)|\, dx \\ &= \int_D \eta |\nabla_\theta f(x) \cdot \hat{g}|\, dx \\ &= \int_D \eta\, |\partial_f L|\, |\nabla_\theta f(x) \cdot\, \nabla_\theta f(v)|\, dx \end{aligned} \tag{3.9}$$

The change in a model over its domain after a parameter update depends on the learning rate η, and the inner product between the parameter gradient $\nabla_\theta f(x)$ of the model, and the loss gradient $\hat{g}$ that depends on the magnitude of loss function derivative $|\partial_f L|$, and the parameter gradient $\nabla_\theta f(v)$ of the training data. This is an expected result. Bounded or sparse parameter gradients can potentially limit the change of the model.

If one chooses a distance or difference measure $d(x, v)$ and some $\delta > 0$, then one can quantify distal interference over some subset of the domain $D' = \{x \mid d(x, v) > \delta\}$, as given in Equation (3.10).

$$\text{Distal Interference} := \int_{D'} |f'(x) - f(x)|\, dx \approx \int_D \eta\, |\partial_f L|\, |\nabla_\theta f(x) \cdot\, \nabla_\theta f(v)|\, dx \tag{3.10}$$

It is possible to construct models that guarantee zero distal interference. The caveat is that one must specify some measure $d(x, v)$ of distance or difference and a fixed $\delta > 0$. This study considers the difference measures $\max_i(|x_i - v_i|)$ and $\min_i(|x_i - v_i|)$. In most of the given model definitions and explanations, one can assume that $z \geq \delta^{-1}$, where $z \in \mathbb{N}$ is the partition number related to the model capacity. Since the choice of δ and $d(x, v)$ is arbitrary, the experimental section uses the simpler $\|f'(x) - f(x)\|_1$ metric.

3.8 Limits on Model-Based Continual Learning

Definition 5 (Uniform Trainability). *A model $f(x)$ is **uniformly trainable** with parameters θ if the parameter gradient of the function is a non-zero vector:*

$$\nabla_\theta f(x) \neq \vec{0},\ \forall x \in D(f),\ \forall \theta \in \mathbb{R}^p$$

Uniform trainability means the model can be adjusted at any point in the domain. This is true for a model with one adjustable parameter that outputs the same value everywhere. It could also be true for more complicated and expressive models. It is possible to have ANN models with zero gradients in some areas of the domain that are not trainable with gradient descent.

Theorem 6. *If a model $f(x)$ with parameters θ is uniformly trainable and max-distal orthogonal for some $\delta \geq z^{-1}$, then it has a parameter space of at least $\Omega(z^n)$ dimensions.*

Proof. Consider a max-distal orthogonal model $f(x)$ with domain $D(f) = [0,1]^n$, and choose $\delta > 0$. Let $x, v \in [0,1]^n$, by Definition 4:

$$\max_i(|x_i - v_i|) > \delta \implies \nabla_\theta f(x) \cdot \nabla_\theta f_t(v) = 0$$

Assume the model is uniformly trainable, then by Definition 5:

$$\nabla_\theta f(x) \neq \vec{0}, \forall x \in [0,1]^n,\ \forall \theta \in \mathbb{R}^p$$

Choose a partition number $z \in \mathbb{N}$ s.t. $z \geq \delta^{-1}$. Consider a set of all $\Theta(z^n)$ grid-points $u^{(i)} \in [0,1]^n$ such that:

$$||u^{(i)} - u^{(j)}||_\infty > \delta \geq z^{-1},\ \forall\ i \neq j \in \mathbb{N}$$

From distal orthogonality, it follows that:

$$\nabla_\theta f(u^{(i)}) \cdot \nabla_\theta f(u^{(j)}) = 0, \forall i \neq j \in \mathbb{N}$$

There are $\Theta(z^n)$ non-zero parameter gradient vectors that are orthogonal to each other. Thus, the set of vectors $\nabla_\theta f(u^{(i)})$ constitute a basis for a vector space with a dimension of $\Theta(z^n)$. It follows that the model requires at least $\Omega(z^n)$ parameters. □

The (logically equivalent) contra-positive of the theorem is: If a model does not have an exponentially large parameter space, then it is not uniformly trainable or not a max-distal orthogonal model. It is necessary to put this into a larger perspective. Uniform trainability means that a model can be trained with gradient descent on any input, i.e. there are no dead zones with no gradient, such as what is possible with ReLU ANNs. A vanishing but non-zero gradient can also hinder trainability. Memory retention guaranteed due to distal orthogonality can counteract catastrophic interference with sequential training on different tasks. The computational complexity of a model (space or time) limits an architecture's scalability. It would seem one cannot have a polynomial complexity model that is easy to train with near-perfect memory retention. It is unsurprising since lookup tables are robust to catastrophic interference but require exponentially many parameters as a function of the input dimension. The trade-offs and unsatisfiability are visualised in Figure 3.5

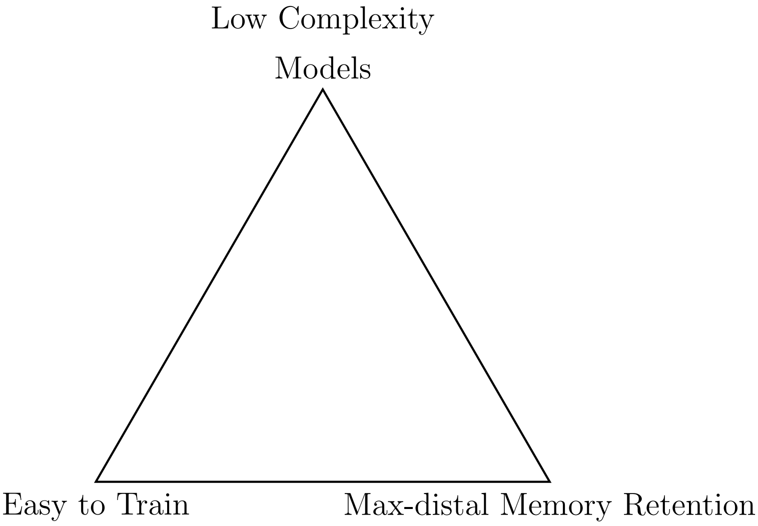


Figure 3.5: The trade-off triangle between low computational complexity, the ease of optimisation with uniform trainability, and memory retention of any machine learning model.

This analysis suggests there are limits to machine learning architectures of practical or polynomial complexity that can be trained sequentially on many tasks without catastrophic interference. Designing machine learning architectures that strike an appropriate balance and trade-offs for effective continual learning may still be possible.

3.9 Aims and Core Ideas

The concepts discussed in previous sections can be visualised with a diagram as shown in Figure 3.6. Topics discussed so far included ANNs, augmented training techniques, lookup tables, memory retention, computational complexity, and ease of training. The aim is to find a reasonable trade-off for effective continual learning, which can be visualised in Figure 3.6.

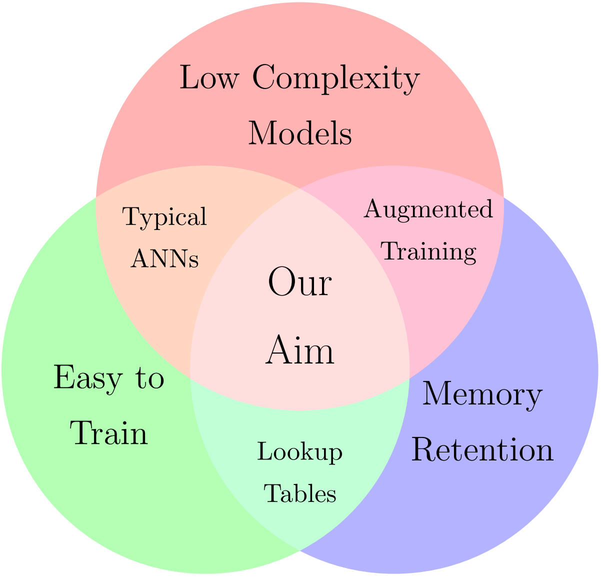


Figure 3.6: A Venn diagram relating key concepts in machine learning related to this study.

Despite the limitations revealed through the analysis, there is a potential workaround to enable continual learning with augmented training techniques like orthogonal gradient descent, pseudo-rehearsal or generative replay. Training- or data augmentation can mitigate the effect of distal interference and hopefully prevent catastrophic interference. Training and data augmentation techniques superficially resemble the replay and memory consolidation mechanisms in human brains, which are facilitated by the hippocampus [30, 77, 78]. It is understood that replay occurs with memory consolidation in human brains. However, the exact learning mechanisms in the human brain are only partially understood. It is conceivable that an analogous machine learning system that is efficiently computable would require a similar form of augmentation, such as pseudo-rehearsal [8].

This study conjectures that: **Continual learning with a polynomial complexity model trained with gradient descent requires augmentation of data or training procedures**.

This is a problematic conjecture to prove directly. Supporting empirical evidence is provided in Chapter 7, but it is not a rigorous and complete proof. Reflecting on all the prior work on continual learning and catastrophic interference did indicate a few avenues for research. There are four possible choices for practical continual learning:

1. Low-dimensional models with max-distal orthogonality and uniform trainability.
2. Retain max-distal orthogonality but sacrifice uniform trainability.
3. Retain uniform trainability and sacrifice max-distal orthogonality.
4. Training augmentation techniques like OGD or pseudo-rehearsal.

It was decided to develop an architecture that retains uniform trainability with a lesser form of distal orthogonality that is less powerful than max-distal orthogonality. The architecture designed for this study (see Chapter 6) also has linear computational complexity and is proven to be a universal function approximator. Experiments are performed to support or invalidate the theoretical claims:

1. Sufficiently large models of the architecture can approximate non-trivial functions.
2. Distal interference can cause catastrophic interference.
3. Distal orthogonal models are robust to distal and catastrophic interference.
4. ReLU ANN models are susceptible to distal and catastrophic interference.

3.10 Summary

This chapter examined continual learning and catastrophic interference in the past. The concepts of overlapping representations, distal interference, and distal orthogonality were introduced. The most significant finding was that models with uniform trainability and guaranteed distal orthogonality must have exponentially large parameter spaces. The contra-positive showed that polynomial complexity models are not uniformly trainable over their domain or do not guarantee distal orthogonality. This theoretical limitation undermines the potential for model-only continual learning.

Given the insight from this chapter, uniformly trainable and efficient single-variable and multi-variable models are outlined in Chapter 4. The multi-variable models are extended to provable universal function approximators in Chapter 5. The computational models are implemented with additional considerations and properties in Chapter 6.

Chapter 4

Spline Artificial Neural Networks

The initial inspiration for this study was to develop new architectures with provable memory guarantees and universal function approximation capabilities based on prior work. The original consideration for cardinal B-splines was born from research by Numenta, Hawkins and Dawkins [79] on Hierarchical temporal memory (HTM) systems with sparse distributed representations [80, 81]. The earliest relevant descriptions of sparse distributed representations or sparse distributed memory models were written by Hinton et al. [14] and Kanerva [82]. Some discussions and comments on HTMs were centred on the absence of an accompanying universal function approximation theorem or proof of it being Turing complete. However, HTM systems do exhibit excellent memory capabilities. After reflection, it became apparent that sparse distributed representations with linear layers are mathematically similar to cardinal B-splines.

Section 4.1 discusses the desired properties of cardinal B-splines. Sparse distributed representations are discussed in Section 4.2. An effort is made to extend the single-variable function approximators to multi-variable functions called z-Spline ANNs to retain some desirable cardinal B-spline properties, discussed in Section 4.3. z-Spline ANNs are defined in Section 4.4, and their analogous properties are given. Section 4.5 provides a summary of this chapter.

4.1 Motivation for B-Splines

Splines are piece-wise defined polynomial functions, often used in computer graphics, function representation and data interpolation [83, 84, 85]. Akin to stitching together (almost) unrelated polynomials together to make a function that is a polynomial locally, but not globally over its domain. A common variant of splines that constitute a basis for a vector space (closed under vector operations) are called B-splines [86, 87, 88].

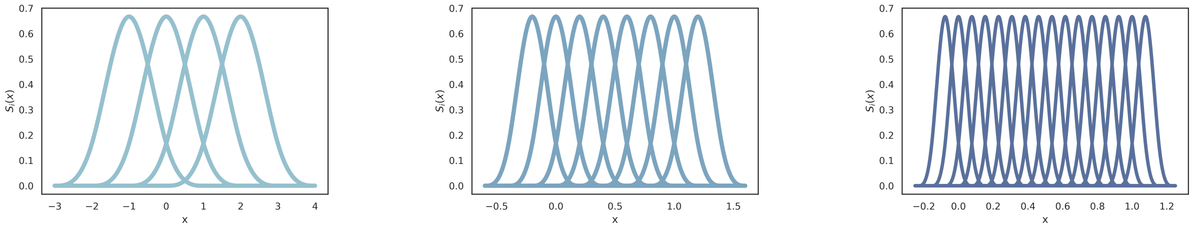
Cardinal B-splines are defined on equally sized sub-intervals or partitions. The boundaries of the partitions are called knots, and knot-insertion algorithms can be used to re-partition into smaller sub-intervals [89]. Splines have been used in ANNs [90, 91, 92].

This study considers cardinal cubic B-splines due to their smooth derivative. Unlike trigonometric and polynomial functions (see Section 3.2), cardinal cubic B-splines have parameters that affect the spline *locally*. Any continuous single-variable function can be approximated with a cardinal cubic B-spline $f(x)$ with basis functions $S_i(x)$, computed from the same activation function $S(x)$.

Definition 7 (z-density B-spline function)**.** *A z-density B-spline function, f, is a cardinal cubic B-spline function of one variable with $4z+3$ basis functions with $z \in \mathbb{N}$ and adjustable parameters θ_i:*

$$f(x) = \sum_{i=1}^{4z+3} \theta_i S_i(x) = \sum_{i=1}^{4z+3} \theta_i S(w_i x + b_i) = \sum_{i=1}^{4z+3} \theta_i S((4z)x + (4-i))$$

Cardinal B-splines uniformly partition the interval $[0,1]$ and each basis function $S_i(x)$ has the same shape for different partition numbers, as shown in Figure 4.1 and Figure 4.2.



(a) Four basis functions. (b) Eight basis functions. (c) Sixteen basis functions.

Figure 4.1: Uniformly spaced cardinal B-splines basis functions have the same shape.

The basis functions extend outside the target interval $[0, 1]$ due to an artefact of the implementation and smoothness requirements. The arguments x of each basis function $S_i(x)$ is simply scaled by $4z$ and shifted by constants b_i before applying the same activation function $S(x)$, which is given by:

$$S(x) = \begin{cases} \frac{1}{6}x^3 & 0 \leq x < 1 \\ \frac{1}{6}\left[-3(x-1)^3 + 3(x-1)^2 + 3(x-1) + 1\right] & 1 \leq x < 2 \\ \frac{1}{6}\left[3(x-2)^3 - 6(x-2)^2 + 4\right] & 2 \leq x < 3 \\ \frac{1}{6}(4-x)^3 & 3 \leq x < 4 \\ 0 & otherwise \end{cases} \tag{4.1}$$

The activation function $S(x)$ is non-zero over the domain $x \in [0, 4]$, and shown in Figure 4.2.

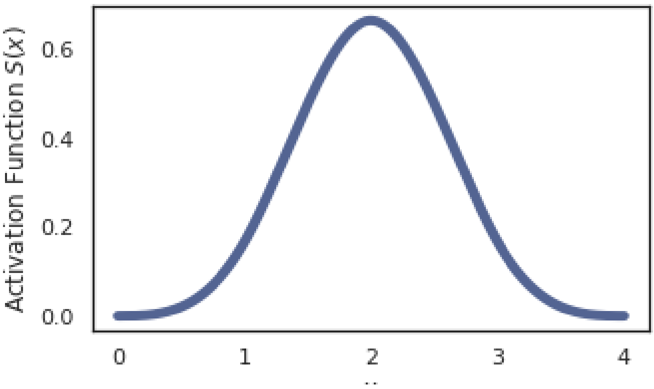


Figure 4.2: Activation function $S(x)$ used to compute cardinal cubic B-splines.

The shape of $S(x)$ resembles a Gaussian, even though the underlying function is piecewise cubic. There are some similarities to using Gaussian kernels for function approximation, with one key difference: Cubic B-spline basis functions are zero almost everywhere, and Gaussian functions are non-zero everywhere.

Cardinal cubic B-splines can be modelled using a two-layer artificial neural network $f(x) = \sum_{i=1}^{K} \theta_i S\left(w_i x + b_i\right)$ with an activation function S corresponding to the shape of each basis function and a pre-defined set of constants w_i and b_i in the first layer. A final trainable linear layer multiplies each basis function with its coefficient parameter θ_i and sums together the results. Keep in mind that the only the coefficients θ_i are trainable thus optimising $f(x) = \sum_{i=1}^{K} \theta_i S\left(w_i x + b_i\right)$ is linear. In contrast, splines that unevenly partition the input space do not permit such a straightforward implementation.

Using cardinal B-splines instead of arbitrary and trainable sub-interval partitions and knots makes optimisation easier [91]. Optimising partitions is non-linear, but optimising only coefficient θ_i (also called control points) is linear and thus convex.

Suppose one changes one parameter θ_i in a cardinal cubic B-spline $f(x)$; then the model only changes on a small region as shown in Figure 4.3. Each parameter of a cardinal B-spline affects a small and local region of the domain, in contrast to other basis function methods discussed in Section 3.2

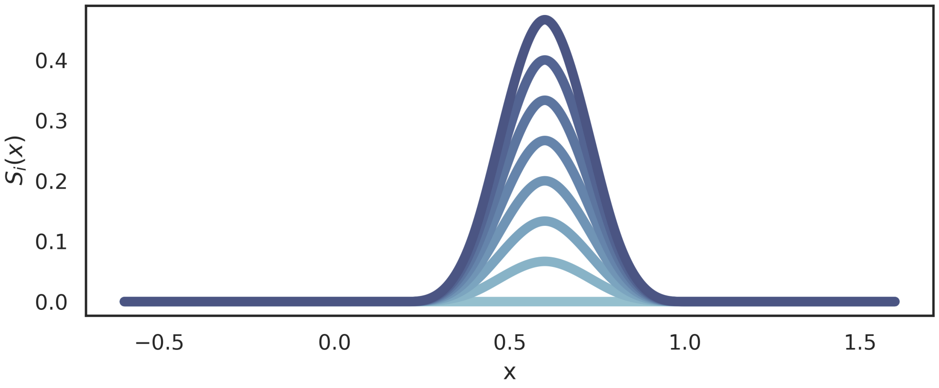


Figure 4.3: Cardinal B-spline basis functions are localised.

Cardinal B-splines were initially considered due to their connection to sparse distributed representations discussed in Section 4.2.

4.2 Sparse Distributed Representation

The initial impetus for B-splines followed from a revelation that sparse distributed representations (SDRs) are zeroth-order cardinal B-splines [14]. Zeroth order splines incidentally resemble single-variable lookup tables with point-wise discontinuities. It is also known that lookup tables are very robust to catastrophic interference [15]. Zeroth-order B-splines have a basis function $B_0(x)$:

$$B_0(x) = \begin{cases} 1, & 0 \leq x < 1 \\ 0, & otherwise \end{cases} \tag{4.2}$$

SDRs are typically a pre-processing step for the input to a model. SDRs encode each scalar variable in a way that superficially resembles a one-hot encoding. SDRs are sparse arrays where most entries are zero. The range of each scalar variable is partitioned into buckets, and the bucket that contains the data point is assigned a value of one. All other entries or buckets in the SDR are zero. A multi-variable SDR is the concatenation of single-variable SDRs. A two-dimensional SDR is shown in Figure 4.4 based on prior explanations of SDRs [14].

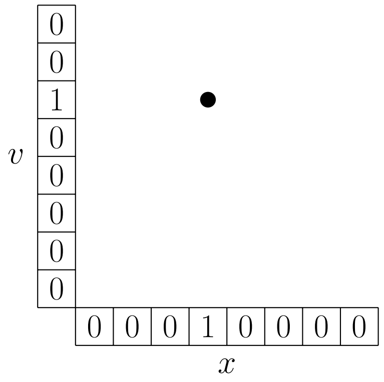


Figure 4.4: SDR encoding of a two-dimensional point.

Different variants of the HTM systems use SDRs to encode information [79, 80, 81]. Applying a linear function to an SDR is equivalent to computing zeroth-order cardinal B-splines, but it is not mentioned in the relevant literature. Cardinal cubic B-splines can be thought of as smoothed SDRs.

4.3 Properties of Cardinal Cubic B-splines

B-splines constitute a flexible and expressive method for function approximation. In addition, analysing B-splines and characterising their general behaviour is simpler than other basis functions. The support interval or domain over which a spline is defined is partitioned into sub-intervals with boundaries called knots. The number of sub-intervals dictates the number of B-spline basis functions. The degree of a B-spline (e.g. zeroth or cubic) is the degree of each locally defined polynomial [86, 87, 88, 89]. Cardinal B-splines exhibit sparse, bounded and non-zero parameter gradients and distal orthogonality.

4.3.1 Sparsity

Proposition 8 (Sparsity). *Let $f(x)$ be a z-density B-spline function, defined on the domain $[0,1] \subset \mathbb{R}$, with trainable parameters θ_i and partition number z. Let $\|\nabla_\theta f(x)\|_0$ denote the number of non-zero components of the gradient vector w.r.t. trainable parameters. For any $x \in D(f)$, the number of non-zero components is bounded:*

$$\|\nabla_\theta f(x)\|_0 \leq 4$$

Proof. Let $f(x)$ be a z-density B-spline function from Definition 7. Consider the components of the vector $\nabla_\theta f(x)$ obtained from the gradient operator, which is simply

$$\frac{\partial}{\partial \theta_i} f(x) = \frac{\partial}{\partial \theta_i} \sum_{j=1}^{4z+3} \theta_j S_j(x) = S_i(x)$$

Inspecting each basis function $S_i(x)$ shows that, at most, four basis functions are non-zero for a fixed $x \in [0,1]$, as visualised in Figure 4.5. It follows that:

$$\|\nabla_\theta f(x)\|_0 \leq 4$$

Thus, the number of non-zero entries is bounded. □

Remark 9. *Sparse gradient vectors leave most weights unchanged and can prevent catastrophic interference. Sparsity also permits efficient model implementations as discussed in Section 6.3, and Chapter 6*

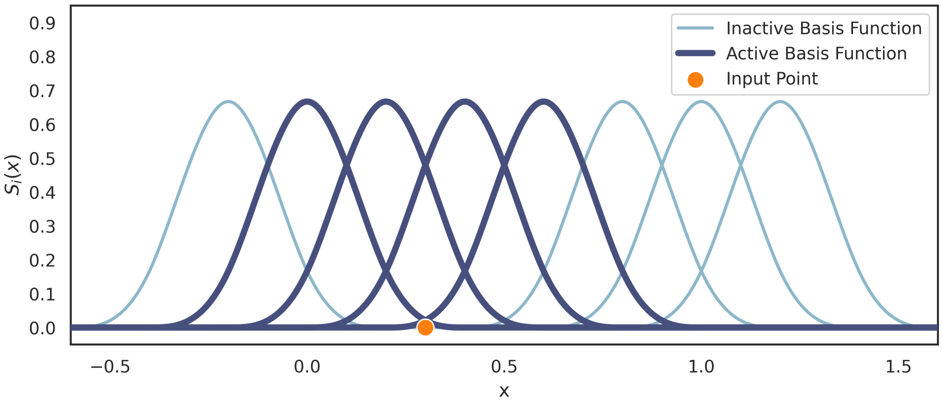


Figure 4.5: Visual proof of sparse and bounded gradients for cardinal B-splines.

4.3.2 Bounded Gradients

Proposition 10 (Bounded Parameter Gradient)**.** *Let $f(x)$ be a z-density B-spline function, defined on the domain $[0,1] \subset \mathbb{R}$, with trainable parameters θ_i and partition number z. For any $x \in D(f)$, the gradient w.r.t trainable parameters, $\nabla_\theta f(x)$, is bounded:*

$$\|\nabla_\theta f(x)\|_1 = \sum_{i=1}^{4z+3} \left| \frac{\partial f}{\partial \theta_i}(x) \right| < 4$$

Proof. Consider the components of the vector $\nabla_\theta f(x)$ obtained from the gradient operator, which is simply:

$$\frac{\partial}{\partial \theta_i} f(x) = \frac{\partial}{\partial \theta_i} \sum_{i=1}^{4z+3} \theta_i S_i(x) = S_i(x)$$

All basis functions $S_i(x)$ are bounded since for any $x \in \mathbb{R}$, the activation function $S(x)$ is always less than 1. Visualised in Figure 4.2 and Figure 4.5. It follows that:

$$\|\nabla_\theta f(x)\|_1 = \sum_{i=1}^{4z+3} \left| \frac{\partial f}{\partial \theta_i}(x) \right| = \sum_{i=1}^{4z+3} |S_i(x)| < 4$$

Thus, the parameter gradient is bounded. □

Remark 11. *Bounded gradients suggest that the model is numerically stable, and small perturbations will not excessively affect the model's learned values. The boundedness of the parameter gradient does not depend on the total number of basis functions.*

4.3.3 Uniform Trainability

Proposition 12 (Trainability of Cardinal B-splines)**.** *Let $f(x)$ be a z-density B-spline function, defined on the domain $[0,1] \subset \mathbb{R}$, with trainable parameters θ_i and partition number z. The gradient w.r.t trainable parameters, $\nabla_\theta f(x)$, is non-zero:*

$$\nabla_\theta f(\theta, x) \neq \vec{0},\ \forall x \in D(f)$$

Proof. Consider the components of the vector $\nabla_\theta f(x)$ obtained from the gradient operator, which is simply

$$\frac{\partial}{\partial \theta_i} f(x) = \frac{\partial}{\partial \theta_i} \sum_{j=1}^{4z+3} \theta_j S_j(x) = S_i(x)$$

If $x \in [0,1]$ is inside the support interval, then at least three basis functions are non-zero. At the boundary of the support interval, it is exactly three instead of the maximum of four. So, at least three components of the parameter gradient are non-zero:

$$\nabla_\theta f(\theta, x) \neq \vec{0},\ \forall x \in D(f),\ \forall \theta \in \mathbb{R}^p$$

Thus, z density B-splines are uniformly trainable on the unit interval. □

Remark 13. *Uniform trainability means a model can be optimised with gradient descent anywhere in its domain. This desirable property is only sometimes guaranteed for conventional ANNs such as ReLU ANNs with "dead" or zeroed neurons.*

4.3.4 Distal Orthogonality

Proposition 14 (absolute distal orthogonality)**.** *Let $f(x)$ be a z-density B-spline function, defined on the domain $[0,1] \subset \mathbb{R}$, with trainable parameters θ_i and partition number z, then $\forall\ x, v \in D(f)$:*

$$|x - v| > z^{-1} \implies \nabla_\theta f(x) \cdot \nabla_\theta f(v) = 0$$

Proof. Let $f(x)$ be a z-density B-spline function, defined on the domain $[0,1] \subset \mathbb{R}$, with trainable parameters θ_i. Consider the points $x, v \in D(f)$. $f(x)$ partitions the unit interval into $4z$ sub-intervals of equal length ($\frac{1}{4z}$).

A visual aid is provided in Figure 4.6. Two distant points are shown in orange. The active or non-zero basis functions associated with each point are shown in dark blue. The inactive or zero basis functions are shown in light blue.

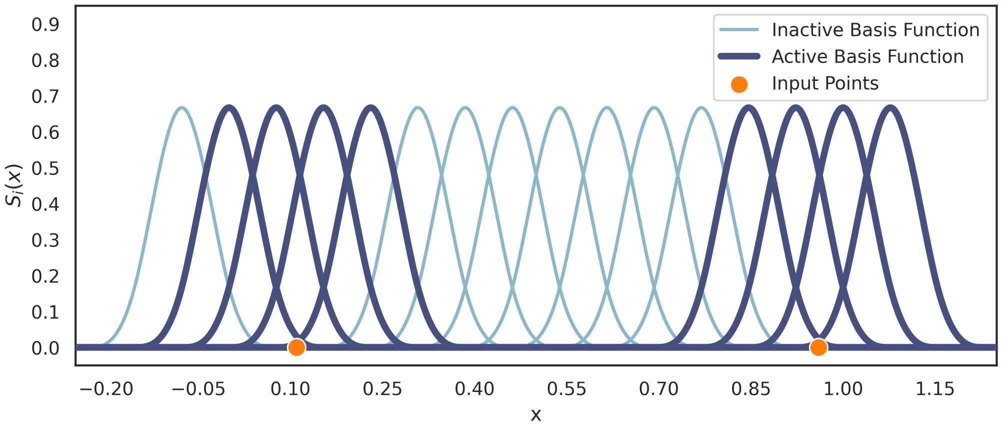


Figure 4.6: Visual proof of distal orthogonality for cardinal B-splines.

If the distance between x and v is larger than *four* sub-intervals, then there are no basis functions in common. Therefore, if $|x - v| > 4(\frac{1}{4z}) = z^{-1}$, then the gradients must be orthogonal with no overlapping parameters. I.e. $\nabla_\theta f(x) \cdot \nabla_\theta f(v) = 0$ □

4.3.5 Stratification

Cardinal cubic B-splines with a large density of basis functions can exhibit what is best described as stratification, illustrated in Figure 4.7, and mentioned in [90]. Stratification resembles overfitting, where the data points are learned precisely, but the model is not adjusted on any regions without training data. The models in Figure 4.7 are initialised to zero. Areas with no training data are never modified and retain their initial value of zero. The training data was sampled from the sine function $\sin(2\pi x)$ to illustrate the effect of increasing the number of basis functions, labelled λ, as shown in Figure 4.7.

Stratification is a consequence of intrinsic memory retention. The flat, unaltered regions between data points are atypical when contrasted with other basis functions. There are a few qualitative differences between stratification and conventional overfitting. Areas with no training data have predictable values (if initialised as constants) and do not exhibit oscillations as seen in the Runge phenomenon with other kinds of basis functions [93, 94, 95]. However, exact interpolation of training data is ill-advised for tasks with noisy training data. The resulting model would have severe variance and oscillate wildly because of overfitting or interpolating with zero error between each consecutive noisy data point on some interval.

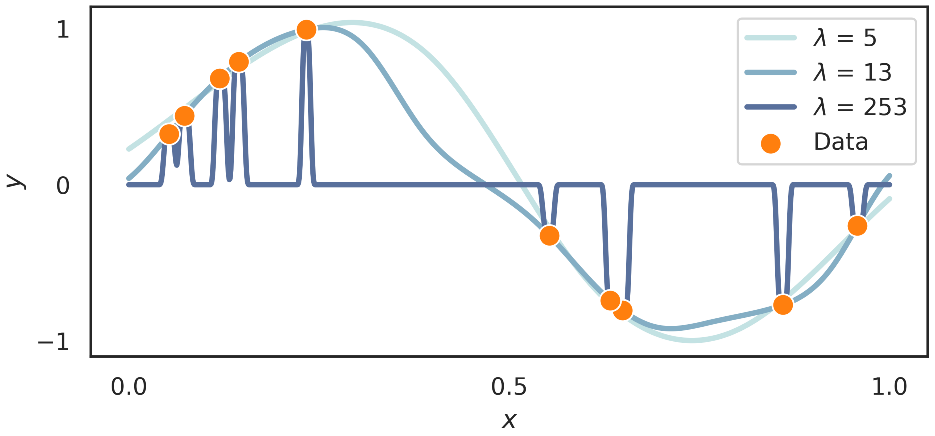


Figure 4.7: Demonstration of stratification in the single-variable case.

Overfitting is task-specific: An over-parametrised model can be ideal if the target function is zero almost everywhere except at a few points. Anomaly detection is a good application for over-parametrised B-spline models. Over-parametrised models might perform poorly on regression problems like estimating a sine function with few training data points, as seen in Figure 4.7. Stratification could be detrimental or advantageous, depending on the application. Manifold Mixup regularisation and data augmentation can potentially counteract stratification when necessary [67, 68, 69].

4.4 Spline Artificial Neural Networks

In this study, Spline ANNs (or z-Spline ANNs) are sums of single-variable z-density B-spline functions. General additive models approximate multi-variable functions as sums of single-variable functions and are often used in statistics [96, 97, 98]. z-Spline ANNs can approximate sums of single-variable functions with z-density B-spline functions.

Incorporating splines into ANNs has been studied to some extent [90, 91, 92]. Lane et al. [90] published a paper at the third NeurIPS conference that employed B-splines to model receptive fields in MLP models. Some investigations used uniformly partitioned splines to implement adaptive and trainable activation functions named spline activation functions (SAFs) in ANNs [99]. Studies have also been on trainable or uneven partition splines that allow the partitions (or knots) to be trained with gradient descent [91]. The relevant studies used shallow ANN architectures with less than three hidden layers.

This study restricts the z-Spline ANN models to evenly spaced partitions for a simpler and more amicable implementation. To the author's knowledge, catastrophic forgetting was not considered in the context of splines.

Definition 15 (z-Spline ANN)**.** *A z-Spline ANN model $f(x)$, defined on $[0,1]^n$, with trainable parameters $\theta \in \mathbb{R}^p$ and partition number $z \in \mathbb{N}$, is a sum of $n \in \mathbb{N}$ single-variable z-density B-splines in each variable:*

$$f(x) = \sum_{j=1}^{n} f_j(x_j) = \sum_{j=1}^{n} \sum_{i=1}^{4z+3} \theta_{i,j} S_{i,j}(x_j) = \sum_{j=1}^{n} \sum_{i=1}^{4z+3} \theta_{i,j} S((4z)x_j + (4-i))$$

where $\theta_{i,j}$ corresponds to trainable parameters or coefficients, and $S_{i,j}(x_j)$ denotes the fixed basis functions for cardinal cubic B-splines. $S(x)$ is the cubic spline activation function. The z-Spline ANNs are uniformly trainable and min-distal orthogonal ($z \geq \delta^{-1}$) models with sparse and bounded parameter gradients. See Section 4.3 for explanations and proofs of the properties of cardinal cubic B-splines.

z-Spline ANNs have linear time and space complexity in the input dimension n. In addition, z-Spline ANNs inherit properties from z-density B-splines: sparsity, bounded gradients, uniform trainability that scale with n and min-distal orthogonality.

4.4.1 Sparsity

Proposition 16 (Sparsity)**.** *Let $f(x)$ be a z-Spline ANN, defined on the domain $x \in [0,1]^n$, with trainable parameters $\theta \in \mathbb{R}^p$. Let $\|\nabla_\theta f(x)\|_0$ denote the number of non-zero components of the gradient vector w.r.t trainable parameters is bounded:*

$$\|\nabla_\theta f(x)\|_0 \leq 4n \; \forall x \in D(f)$$

Proof. Let $f(x)$ be a z-Spline ANN, defined on the domain $x \in [0,1]^n$, with trainable parameters $\theta \in \mathbb{R}^p$. Let $\|\nabla_\theta f(x)\|_0$ denote the number of non-zero components of the gradient vector w.r.t trainable parameters. From definition 15:

$$\|\nabla_\theta f(x)\|_0 = \|\nabla_\theta \sum_{j=1}^{n} f_j(x_j)\|_0 = \sum_{j=1}^{n} \|\nabla_\theta f_j(x_j)\|_0$$

From proposition 10,

$$\|\nabla_\theta f(x)\|_0 < \sum_{j=1}^{n} 4 = 4n$$

□

4.4.2 Bounded Gradients

Proposition 17 (Bounded Parameter Gradient)**.** *Let $f(x)$ be a z-Spline ANN, defined on the domain $x \in [0,1]^n$, with trainable parameters $\theta \in \mathbb{R}^p$. For any $x \in D(f)$, the gradient w.r.t trainable parameters, $\nabla_\theta f(x)$, is bounded:*

$$\|\nabla_\theta f(x)\|_1 = \sum_{j=1}^{n} \sum_{i=1}^{4z+3} |S_{i,j}(x_j)| < 4n$$

Proof. Let $f(x)$ be a z-Spline ANN, defined on the domain $x \in [0,1]^n$, with trainable parameters $\theta \in \mathbb{R}^p$. From definition 15:

$$\|\nabla_\theta f(x)\|_1 = \|\nabla_\theta \sum_{j=1}^{n} f_j(x_j)\|_1 = \sum_{j=1}^{n} \|\nabla_\theta f_j(x_j)\|_1$$

From proposition 8,

$$\|\nabla_\theta f(x)\|_1 \leq \sum_{j=1}^{n} 4 = 4n$$

□

4.4.3 Uniform Trainability

Proposition 18 (Trainability of z-Spline ANNs)**.** *Let $f(x)$ be a z-Spline ANN, defined on the domain $x \in [0,1]^n$, with trainable parameters $\theta \in \mathbb{R}^p$:*

$$\nabla_\theta f(x) \neq \vec{0}$$

Proof. Let $f(x)$ be a z-Spline ANN, defined on the domain $x \in [0,1]^n$, with trainable parameters $\theta \in \mathbb{R}^p$. From definition 15:

$$\nabla_\theta f(x) = \sum_{j=1}^{n} \nabla_\theta f_j(x_j)$$

From proposition 12, it follows that $\nabla_\theta f_j(x_j)$ are non-zero. Since f_j are independent it follows that $\nabla_\theta f(x) \neq \vec{0}$. □

Unsurprisingly, z-Spline ANNs inherit sparsity, bounded parameter gradients and uniform trainability. The overall model is convex since the model is linear w.r.t. trainable parameters. This makes global optimisation easy with gradient descent algorithms.

4.4.4 Min-Distal Orthogonality

Min-distal orthogonality is similar to the single-variable case. However, **each** coordinate must sufficiently differ as shown in Figure 4.8, unlike lookup tables that are discussed in Section 3.6, and shown in Figure 3.4.

Proposition 19 (min-distal orthogonality)**.** *Let $f(x)$ be a z-Spline ANN, defined on the domain $x \in [0,1]^n$, with trainable parameters $\theta \in \mathbb{R}^p$. For any $x, v \in D(f)$:*

$$\min_i(|x_i - v_i|) > z^{-1} \implies \nabla_\theta f(x) \cdot \nabla_\theta f(v) = 0$$

Remark 20. *Min-distal orthogonality is similar to max-distal orthogonality (see Definition 4). If a model is a max-distal orthogonal model, then it is a min-distal orthogonal model. However, not all min-distal orthogonal models are max-distal orthogonal models.*

Proof. Let $f(x)$ be a z-Spline ANN, defined on the domain $x \in [0,1]^n$, with trainable parameters $\theta \in \mathbb{R}^p$. Choose $x, v \in D(f)$. From definition 15, it follows that:

$$\nabla_\theta f(x) \cdot \nabla_\theta f(v) = \left(\sum_{j=1}^{n} \nabla_\theta f_j(x_j)\right) \cdot \left(\sum_{j=1}^{n} \nabla_\theta f_j(v_j)\right)$$

Since each function f_j is independent for each input dimension,

$$\left(\sum_{j=1}^{n} \nabla_\theta f_j(x_j)\right) \cdot \left(\sum_{j=1}^{n} \nabla_\theta f_j(v_j)\right) = \sum_{j=1}^{n} \nabla_\theta f_j(x_j) \cdot \nabla_\theta f_j(v_j)$$

Assume that $\min_i(|x_i - v_i|) > \frac{1}{z}$, then from proposition 14 it follows that each inner product is zero. Finally, $\nabla_\theta f(x) \cdot \nabla_\theta f(v) = 0$ □

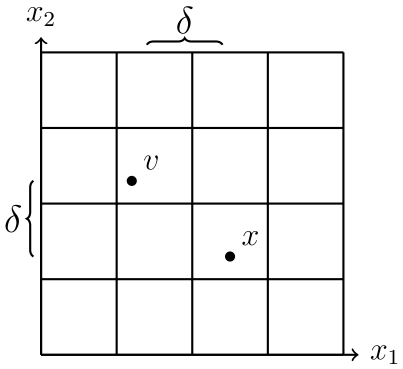


Figure 4.8: Two sufficiently different points to guarantee min-distal orthogonality.

Note that z-Spline ANNs can be helpful for specific applications. Optimising their parameters with a convex loss function yields a convex optimisation problem with a global optimum that can be reached with gradient descent optimisation methods. z-Spline ANNs have **some** guarantees regarding non-overlapping representations due to lesser orthogonality, but it is unclear how useful min-distal orthogonality is for continual learning. A major theoretical problem with z-Spline ANNs is their limited expressivity: z-Spline ANNs cannot approximate any multi-variable continuous function. If the target function is a sum of single-variable functions, then z-Spline ANNs would be ideal, but many problems have more complicated target functions.

One of the goals of this study is to extend z-Spline ANNs to an architecture expressive enough to approximate any multi-variable continuous function while retaining the desirable properties of sparsity, bounded gradients, uniform trainability, min-distal orthogonality and polynomial computational complexity.

4.5 Summary

This chapter introduced cardinal B-splines and their desirable properties for single-variable function approximation. Cardinal B-splines were extended to multi-variable functions called z-Spline ANNs. It was shown that z-Spline ANNs are linear w.r.t. model parameters and are easy to train. z-Spline ANNs have limited expressive power and are not universal function approximators. Functions called *antisymmetric exponentials* are introduced in Chapter 5 to extend z-Spline ANNs to universal function approximators.

Chapter 5

Antisymmetric Exponentials

This chapter outlines a set of continuous functions called *antisymmetric exponentials*. This study aims to extend z-spline ANNs to universal function approximators while retaining min-distal orthogonality and other properties discussed in Chapter 4. Initial attempts in this study to extend z-Spline ANN using the Kolmogorov-Arnold representation theorem did not transfer their properties, in contrast to the Stone-Weierstrass theorem and exponentials. Antisymmetric exponentials guarantee universal function approximation for the model proposed in Chapter 6. Antisymmetric exponentials are named for a sign-swap property:

$$-\Bigg(\exp(x) - \exp(y)\Bigg) = \exp(y) - \exp(x)$$

Developing antisymmetric exponentials can reveal new function approximation methods. Functions on domains different than $[0, 1]^n$ or $\mathbb{R}^n$ are notoriously unwieldy and difficult to approximate in practice. Functions defined on continuous signals, sets, sequences, graphs and point clouds are challenging to model and approximate from data alone [100, 101, 102]. ANNs that can distinguish (see Definition 24) between any two potential inputs are necessary for universal function approximation.

Section 5.1 provides necessary background definitions and theorems. Antisymmetric exponentials are defined in Section 5.2. Section 5.3 and Section 5.4 prove that antisymmetric exponentials constitute a unital sub-algebra. Section 5.5 proves that antisymmetric exponentials can separate points. Section 5.6 proves that antisymmetric exponentials are universal function approximators. A summary is given in Section 5.7.

5.1 Preliminary Definitions and Theorems

This section gives standard definitions and theorems necessary to prove universal function approximation that can be found in analysis courses and textbooks [103]. The general and abstract formulation can help to evaluate or develop function approximation methods on different domains. However, this project is restricted to the domain $[0,1]^n$.

Definition 21. *Let (X, d) be a metric space. Then $C_b(X)$ is the normed subspace of $B(X)$ comprising all continuous, bounded functions $f : X \to \mathbb{R}$.*

Definition 22. *A subset $S \subseteq C_b(X)$ is called an **algebra** if*

1. *it is a linear subspace of S, i.e. it is closed under addition and scalar multiplication (by arbitrary scalars $\alpha \in \mathbb{R}$); and*

2. *it is closed under point-wise multiplication, i.e. for all $f, g \in S$, we have $fg \in S$.*

Remark 23. *A unital sub-algebra is an algebra of a subset $S \subseteq C_b(X)$ that contains the multiplicative identity or unity like the constant function $x \mapsto 1$*

Definition 24. *We say that a set $S \subseteq C_b(X)$ **separates points in** X if for all $x, y \in X$ with $x \neq y$, there exists $f \in S$ such that $f(x) \neq f(y)$.*

Definition 25. *Let (X, d) be a metric space. A set $Y \subseteq X$ is called **dense** in X if for every $x \in X$ and every $\varepsilon > 0$, there exists $y \in Y$ such that $d(x, y) < \varepsilon$.*

Remark 26. *Assume the supremum norm $\|f\|_{sup} = \sup_{x\in X} |f(x)|$ for function spaces, and $d(f, g) = \sup_{x\in X} |f(x) - g(x)|$.*

Theorem 27 (Stone–Weierstrass)**.** *Let (X, d) be a compact metric space. Suppose that $S \subseteq C_b(X)$ is an algebra that separates points in X. Furthermore, suppose that the constant function $x \mapsto 1$ belongs to S. Then S is dense in $C_b(X)$.*

5.2 Definition of Antisymmetric Exponentials

Definition 28. *Let $(X, d) \subseteq \mathbb{R}^n$ be a compact metric space. For all ψ in the set of* ***antisymmetric exponentials*** *$\Psi \subseteq C_b(X)$, there exists $M \in \mathbb{N}$, and $g_{k,j}(x_j)$, $h_{k,j}(x_j) \in C(\mathbb{R})$ such that:*

$$\psi(x) = \sum_{k=1}^{M} \left(\exp\left(\Sigma_{j=1}^{n} g_{k,j}(x_j) \right) - \exp\left(\Sigma_{j=1}^{n} h_{k,j}(x_j) \right) \right)$$

It can be shown that the set of antisymmetric exponentials is closed under scalar multiplication, addition, and point-wise multiplication. If one multiplies an antisymmetric exponential with a scalar, the result is still an antisymmetric exponential. If one adds two antisymmetric exponentials, the result is also an antisymmetric exponential. If one multiplies two antisymmetric exponentials, the result is still an antisymmetric exponential. This is one of the reasons the number of positive and negative functions is chosen to be the same.

5.3 Antisymmetric Exponentials are an Algebra

Lemma 29. *Suppose that $\Psi \subseteq C_b(X)$ is the set of antisymmetric exponentials, then Ψ is closed under scalar multiplication for any $\rho \in \mathbb{R}$.*

Proof. Let $(X, d) \subseteq \mathbb{R}^n$ be a compact metric space, with $x \in (X, d)$, and components $x = (x_1, .., x_n)$. Let $\psi, \psi' \in \Psi$ be antisymmetric exponentials. From Definition 28:

$$\psi = \sum_{k=1}^{M} \left(\exp\left(\Sigma_{j=1}^{n} g_{k,j}(x_j) \right) - \exp\left(\Sigma_{j=1}^{n} h_{k,j}(x_j) \right) \right)$$
$$\psi' = \sum_{k=1}^{M'} \left(\exp\left(\Sigma_{j=1}^{n} g'_{k,j}(x_j) \right) - \exp\left(\Sigma_{j=1}^{n} h'_{k,j}(x_j) \right) \right)$$

where $g_{k,j}(x_j)$, $h_{k,j}(x_j)$, $g'_{k,j}(x_j)$, $h'_{k,j}(x_j) \in C(\mathbb{R})$ are continuous single-variable functions.

Let $0 < \alpha \in \mathbb{R}^+$. Any real number $\rho \in \mathbb{R}$ can be expressed as $\alpha, -\alpha$, or zero. This is necessary since logarithms are not real or undefined for non-positive numbers. The three cases are considered:

1. Choose the interior functions for ψ' such that $g'_{k,1}(x_1) = g_{k,1}(x_1) + \log(\alpha)$, and $h'_{k,1}(x_1) = h_{k,1}(x_1) + \log(\alpha)$. Choose $g'_{k,j}(x_j) = g_{k,j}(x_j)$, and $h'_{k,j}(x_j) = h_{k,j}(x_j)$ for all indices $j > 1 \in \mathbb{N}$:

$$\alpha\psi = \sum_{k=1}^{M} \Bigg(\exp\Bigg(\log(\alpha) + \Sigma_{j=1}^{n} g_{k,j}(x_j) \Bigg) - \exp\Bigg(\log(\alpha) + \Sigma_{j=1}^{n} h_{k,j}(x_j) \Bigg) \Bigg) = \psi'$$

2. Choose the interior functions for ψ' such that $g'_{k,1}(x_1) = h_{k,1}(x_1) + \log(\alpha)$, and $h'_{k,1}(x_1) = g_{k,1}(x_1) + \log(\alpha)$. Choose $g'_{k,j}(x_j) = h_{k,j}(x_j)$, and $h'_{k,j}(x_j) = g_{k,j}(x_j)$ for all indices $j > 1 \in \mathbb{N}$:

$$-\alpha\psi = \sum_{k=1}^{M} \Bigg(\exp\Bigg(\log(\alpha) + \Sigma_{j=1}^{n} h_{k,j}(x_j) \Bigg) - \exp\Bigg(\log(\alpha) + \Sigma_{j=1}^{n} g_{k,j}(x_j) \Bigg) \Bigg) = \psi'$$

3. Choose the interior functions for ψ' such that $g'_{k,j}(x_j) = 0$, and $h'_{k,j}(x_j) = 0$ for all indices $j, k \in \mathbb{N}$:

$$0\psi = 0 = \sum_{k=1}^{M'} \Bigg(\exp(0) - \exp(0) \Bigg) = \psi'$$

One must simply define single-variable interior functions as zero or absorb a constant. Since this is true for any $\rho \in \mathbb{R}$, it follows that Ψ is closed under scalar multiplication. □

The following requirement is to show that the set of antisymmetric exponentials is closed under addition. I.e. adding togther two antisymmetric exponentials yields a third antisymmetric exponential. Sets of functions closed under scalar multiplication and addition satisfy the axioms for a vector space.

Lemma 30. *Suppose that $\Psi \subseteq C_b(X)$ is the set of antisymmetric exponentials, then Ψ is closed under addition.*

Proof. Let $(X,d) \subseteq \mathbb{R}^n$ be a compact metric space, with $x \in (X,d)$, and components $x = (x_1, .., x_n)$. Let $\psi, \psi', \psi'' \in \Psi$ be antisymmetric exponentials. From Definition 28:

$$\psi = \sum_{k=1}^{M}\left(\exp\left(\Sigma_{j=1}^{n} g_{k,j}(x_j)\right) - \exp\left(\Sigma_{j=1}^{n} h_{k,j}(x_j)\right)\right)$$
$$\psi' = \sum_{k=1}^{M'}\left(\exp\left(\Sigma_{j=1}^{n} g'_{k,j}(x_j)\right) - \exp\left(\Sigma_{j=1}^{n} h'_{k,j}(x_j)\right)\right)$$
$$\psi'' = \sum_{k=1}^{M''}\left(\exp\left(\Sigma_{j=1}^{n} g''_{k,j}(x_j)\right) - \exp\left(\Sigma_{j=1}^{n} h''_{k,j}(x_j)\right)\right)$$

A more compact notation that fits on the page for the sums of single-variable functions is $G'_k := \Sigma_{j=1}^{n} g'_{k,j}(x_j)$, $H'_k := \Sigma_{j=1}^{n} h'_{k,j}(x_j)$, similarly for G_k, H_k and G''_k, H''_k. Choose the interior functions for ψ'' such that $G''_k = G_k$, $H''_k = H_k$ for all indices $k \leq M$. Choose $G''_k = G'_{k-M}$, $H''_k = H'_{k-M}$, $\forall k > M$ such that the indices are in range. With $M'' = M + M'$, it follows that:

$$\psi + \psi' = \left(\sum_{k=1}^{M}\left(\exp G_k - \exp H_k\right)\right) + \left(\sum_{k=1}^{M'}\left(\exp G'_k - \exp H'_k\right)\right)$$
$$\psi + \psi' = \sum_{k=1}^{M+M'}\left(\exp G''_k - \exp H''_k\right) = \sum_{k=1}^{M''}\left(\exp G''_k - \exp H''_k\right) = \psi''$$

One can 'find' a $\psi'' \in \Psi$ such that $\psi'' = \psi + \psi'$ by choosing appropriate interior functions in ψ''. Since this is true for any $\psi, \psi' \in \Psi$ it follows that Ψ is closed under addition. □

Antisymmetric exponentials do constitute a vector space. What distinguishes an algebra from a vector space is some way to 'multiply' vectors from the space such that the result is also an element of the vector space. This is not true in general for the inner product that maps vectors to scalars. The following requirement is to show that antisymmetric exponentials are closed under point-wise multiplication.

Lemma 31. *Suppose that $\Psi \subseteq C_b(X)$ is the set of antisymmetric exponentials, then Ψ is closed under point-wise multiplication.*

Proof. Let $(X, d) \subseteq \mathbb{R}^n$ be a compact metric space, with $x \in (X, d)$, and components $x = (x_1, .., x_n)$. Let $\psi, \psi', \psi'' \in \Psi$ be antisymmetric exponentials. From Definition 28:

$$\psi = \sum_{p=1}^{M} \left(\exp\left(\Sigma_{j=1}^{n} g_{p,j}(x_j)\right) - \exp\left(\Sigma_{j=1}^{n} h_{p,j}(x_j)\right)\right)$$
$$\psi' = \sum_{q=1}^{M'} \left(\exp\left(\Sigma_{j=1}^{n} g'_{q,j}(x_j)\right) - \exp\left(\Sigma_{j=1}^{n} h'_{q,j}(x_j)\right)\right)$$
$$\psi'' = \sum_{k=1}^{M''} \left(\exp\left(\Sigma_{j=1}^{n} g''_{k,j}(x_j)\right) - \exp\left(\Sigma_{j=1}^{n} h''_{k,j}(x_j)\right)\right)$$

A more compact notation that fits on the page for the sums of single-variable functions is $G'_k := \Sigma_{j=1}^{n} g'_{k,j}(x_j)$, $H'_k := \Sigma_{j=1}^{n} h'_{k,j}(x_j)$, similarly for G_k, H_k and G''_k, H''_k. Multiplying out the expressions yields $4MM'$ terms in total, with $2MM'$ positive and $2MM'$ negative exponential functions, such that $M'' = 2MM'$. The key is that the result has equal numbers of positive and negative exponentials. The choice of indexing is arbitrary, and k, p, q are used to distinguish different functions, with k being dependent on (p, q). The interior functions are closed under addition and $G''_k = G_p + G'_q$ or $G''_k = H_p + H'_q$ for positive terms. For negative terms one has $H''_k = H_p + G'_q$ or $H''_k = G_p + H'_q$ such that:

$$\begin{aligned}
\psi\psi' &= \left(\sum_{p=1}^{M}\left(\exp G_p - \exp H_p\right)\right)\left(\sum_{q=1}^{M'}\left(\exp G'_q - \exp H'_q\right)\right) \\
&= \sum_{p=1}^{M}\sum_{q=1}^{M'}\left(\exp G_p - \exp H_p\right)\left(\exp G'_q - \exp H'_q\right) \\
&= \sum_{p=1}^{M}\sum_{q=1}^{M'}\left(\exp\left(G_p + G'_q\right) - \exp\left(H_p + G'_q\right) - \exp\left(G_p + H'_q\right) + \exp\left(H_p + H'_q\right)\right) \\
&= \sum_{k=1}^{M''}\left(\exp G''_k - \exp H''_k\right) = \psi''
\end{aligned}$$

There exists a $\psi'' \in \Psi$ such that $\psi'' = \psi\psi'$ since interior functions are closed under addition. This is true for any $\psi, \psi' \in \Psi$. Therefore, Ψ is closed under multiplication. □

Lemma 32. *Suppose that $\Psi \subseteq C_b(X)$ is the set of antisymmetric exponentials, then Ψ is an algebra.*

Proof. Antisymmetric exponentials $\Psi \subseteq C_b(X)$ are closed under scalar multiplication (by lemma 29), addition (by lemma 30), and point-wise multiplication (by lemma 31). The set of antisymmetric exponentials $\Psi \subseteq C_b(X)$ is an algebra by definition. □

5.4 Antisymmetric Exponentials Contain Unity

Containing a constant function is necessary so that the set of functions can represent a constant function. Regardless of the tautology, universal function approximation of continuous functions also includes constant functions.

Lemma 33. *Let (X, d) be a compact metric space with $x \in X$. Suppose that $\Psi \subseteq C_b(X)$ is the set of antisymmetric exponentials, then the constant function $x \mapsto 1$ is an element of Ψ*

Proof. Let $(X, d) \subseteq \mathbb{R}^n$ be a compact metric space, with $x \in (X, d)$, and components $x = (x_1, .., x_n)$. Let $\psi, \psi' \in \Psi$ be antisymmetric exponentials. From Definition 28:

$$\psi = \sum_{k=1}^{M} \left(\exp\left(\Sigma_{j=1}^{n} g_{k,j}(x_j) \right) - \exp\left(\Sigma_{j=1}^{n} h_{k,j}(x_j) \right) \right)$$

Choose $g_{1,1}(x_1) = \log(2)$, and all other functions $g_{k,j}(x_j)$, $h_{k,j}(x_j) = 0$, by substitution it follows that:

$$\begin{aligned}
\psi &= \exp(\log 2) - \exp(0) + \sum_{k=2}^{M} \left(\exp\left(\Sigma_{j=1}^{n} g_{k,j}(x_j) \right) - \exp\left(\Sigma_{j=1}^{n} h_{k,j}(x_j) \right) \right) \\
\psi &= \exp(\log 2) - \exp(0) + \sum_{k=2}^{M} \left(\exp(0) - \exp(0) \right) \\
\psi &= 2 - 1 + \sum_{k=2}^{M} \left(1 - 1 \right) \\
\psi &= 1
\end{aligned}$$

The constant function $x \mapsto 1$ is in the set of antisymmetric exponentials Ψ. □

5.5 Antisymmetric Exponentials Separate Points

For any two distinct elements from the domain, an antisymmetric exponential exists that maps to two different output values. This is necessary to show that the set of functions is not limited to constant functions and has enough expressive power to distinguish between different points from the domain.

Lemma 34. *Let (X, d) be a compact metric space. Suppose that $\Psi \subseteq C_b(X)$ is the set of antisymmetric exponentials. Ψ separates points in X such that or all $x, y \in X$ with $x \neq y$, there exists a $f \in \Psi$ such that $f(x) \neq f(y)$.*

Proof. Ψ separates points x, y in X. Suppose $x \neq y$, without loss of generality, that the qth components differ: $x_q \neq y_q$. Let $\psi \in \Psi$ such that:

$$\psi(x) = \sum_{k=1}^{M} \left(\exp\left(\Sigma_{j=1}^{n} g_{k,j}(x_j) \right) - \exp\left(\Sigma_{j=1}^{n} h_{k,j}(x_j) \right) \right)$$

Choose $g_{1,q}(x_q) = x_q$, and all other single-variable functions $g_{k,j}(x_j)$, $h_{k,j}(x_j) = 0$. It follows that: $\psi(x) = \exp(x_q) - 1$. Similarly $\psi(y) = \exp(y_q) - 1$. The exponential function is strictly monotone, so it follows: $\psi(x) \neq \psi(y)$. □

5.6 Antisymmetric Exponentials are Universal Function Approximators

Theorem 35. *Let (X, d) be a compact metric space. Suppose that $\Psi \subseteq C_b(X)$ is the set of antisymmetric exponentials, then Ψ is dense in $C_b(X)$*

Proof. Let $(X, d) \subset \mathbb{R}^n$ be a compact metric space. Suppose that $\Psi \subseteq C_b(X)$ is the set of antisymmetric exponentials. By lemma 32, Ψ is an algebra. By lemma 33, Ψ contains the constant function $x \mapsto 1$, and by lemma 34, Ψ separates points in X. By the Stone-Weierstrass theorem 27, Ψ is dense in $C_b(X)$. □

Ψ is dense in the metric space of continuous bounded functions $(C_b(X), d)$. It follows from the definition that for any function $f \in C_b(X)$ and every $\varepsilon > 0$, there exists an antisymmetric exponential function $\psi \in \Psi$ such that $d(f, \psi) < \varepsilon$.

5.7 Summary

This chapter introduced antisymmetric exponentials, and proved that antisymmetric exponentials are universal function approximators. The construction and definition of antisymmetric exponentials were chosen to simplify the required proofs. In Chapter 6, antisymmetric exponentials are modified and implemented so that many of the desirable properties of Spline ANNs are transferred to universal function approximators.

Chapter 6

ABEL-Splines: Antisymmetric Bounded Exponential Layer Splines

The ABEL-Spline architecture is designed for efficient, expressive, robust and numerically stable function approximation models. ABEL-Splines stands for "Antisymmetric Bounded Exponential Layer Splines." ABEL-Splines combine Spline ANNs, antisymmetric exponentials, and absolutely convergent series to yield models with desirable properties aimed at being practical and useful. Section 6.1 gives a mathematical definition of ABEL-Splines. Section 6.2 details the desirable properties of the architecture. Section 6.3 explains the implementation of ABEL-Splines in TensorFlow. A summary of this chapter is presented in Section 6.4.

6.1 Definition of ABEL-Splines

Definition 36 (ABEL-Spline)**.** *Let $A(x)$ be an ABEL-Spline function, defined on the domain $x \in [0,1]^n$, with trainable parameters $\theta \in \mathbb{R}^p$ and partition number $z \in \mathbb{N}$. Then, $\exists\, \mathcal{K} \in \mathbb{N}$, and multi-variable z-Spline ANN functions $F(x), G_k(x), H_k(x)$ such that:*

$$A(x) := F(x) + \sum_{k=1}^{\mathcal{K}} \frac{1}{k^2}\Big(\exp(G_k(x)) - \exp(H_k(x)) \Big)$$

ABEL-Splines can equivalently be given in terms of single-variable z-density B-spline functions $f_j(x_j), g_{k,j}(x_j)$, and $h_{k,j}(x_j)$ such that:

$$A(x) = \sum_{j=1}^{n} f_j(x_j) + \sum_{k=1}^{\mathcal{K}} \frac{1}{k^2}\Bigg(\exp(\Sigma_{j=1}^{n} g_{k,j}(x_j)) - \exp(\Sigma_{j=1}^{n} h_{k,j}(x_j))\Bigg)$$

Remark 37. *Note that multi-dimensional outputs are treated as separate scalar functions, approximated with the outlined schema, and are independent of each other. The absolutely convergent series of scale factors k^{-2} was chosen for numerical stability and to ensure the model is absolutely convergent, as $\mathcal{K} \to \infty$. A series $\sum_i a_i$ is absolutely convergent if $\sum_i |a_i|$ converges to a well-defined limit [104]. Another feature is that the series of scale factors also breaks the symmetry that would otherwise exist between terms.*

6.2 Properties of ABEL-Splines

ABEL-Splines are universal function approximators with some inherent memory retention. Universal function approximation follows from antisymmetric exponentials[1]. ABEL-Splines possess properties atypical of most universal function approximators:

1. **Sparse activity** – most neural units are zero and inactive.
2. **Bounded parameter gradients** regardless of model size.
3. **Uniformly trainable** anywhere in the domain
4. **Min-distal orthogonality**

6.2.1 Sparsity

Proposition 38 (Sparsity)**.** *Let $A(x)$ be an ABEL-Spline function, defined on the domain $x \in [0,1]^n$, with trainable parameters $\theta \in \mathbb{R}^p$ and partition number $z \in \mathbb{N}$. Let $\|\nabla_\theta A(x)\|_0$ denote the number of non-zero components of the gradient vector w.r.t. trainable parameters, then for any $x \in D(f)$, the number of non-zero components is bounded:*

$$\|\nabla_\theta A(x)\|_0 \leq 4n(2\mathcal{K}+1)$$

[1] ABEL-Splines are very *able*.

Proof. Let $A(x)$ be a z-ABEL-Spline from Definition 36. From the triangle inequality and the pseudo-norm property: $\|\alpha x\|_0 = \|x\|_0,\ \forall\ x \in [0,1]^n,\ \ \forall\ \alpha \neq 0 \in \mathbb{R}$, it follows that:

$$\|\nabla_\theta A(x)\|_0 \leq \|\nabla_\theta F(x)\|_0 + \sum_{k=1}^{\mathcal{K}} \left\| \frac{1}{k^2}\Big(\exp(G_k(x))\nabla_\theta G_k(x) - \exp(H_k(x))\nabla_\theta H_k(x) \Big) \right\|_0$$

$$\|\nabla_\theta A(x)\|_0 \leq \|\nabla_\theta F(x)\|_0 + \sum_{k=1}^{\mathcal{K}} \Big(\|\nabla_\theta G_k(x)\|_0 - \|\nabla_\theta H_k(x)\|_0 \Big)$$

From Proposition 16 it follows: $\|\nabla_\theta F(x)\|_0 = \|\nabla_\theta G_k(x)\|_0 = \|\nabla_\theta H_k(x)\|_0 = 4n$, so

$$\|\nabla_\theta A(x)\|_0 \leq 4n + \sum_{k=1}^{\mathcal{K}} \big(4n + 4n\big) = 4n(2\mathcal{K} + 1)$$

The model has a total of $n(4z+3)(2\mathcal{K}+1)$ trainable parameters. The gradient vector has a maximum of $4n(2\mathcal{K}+1)$ non-zero entries, independent of z. At most, the fraction of active basis functions is $\frac{4}{4z+3}$. □

6.2.2 Bounded Gradients

Proposition 39 (Bounded gradient)**.** *Let $A(x)$ be an ABEL-Spline function, defined on the domain $x \in [0,1]^n$, with partition number $z \in \mathbb{N}$ and* ***bounded*** *trainable parameters $|\theta_i| < \mu$, $\forall \theta \in \mathbb{R}^p$. Then the parameter gradient $\nabla_\theta A(x)$ is bounded $\forall x \in D(f)$:*

$$\|\nabla_\theta A(x)\|_1 < 4n + \big(8n\exp(4\mu n)\big)\frac{\pi^2}{6}$$

Proof. Let $A(x)$ be a z-ABEL-Spline. From Definition 36 and the triangle inequality:

$$\|\nabla_\theta A(x)\|_1 \leq \|\nabla_\theta F(x)\|_1 + \sum_{k=1}^{\mathcal{K}} \frac{1}{k^2}\Big(\exp(G_k(x))\|\nabla_\theta G_k(x)\|_1 + \exp(H_k(x))\|\nabla_\theta H_k(x)\|_1 \Big)$$

From Proposition 17 it follows: $\|\nabla_\theta F(x)\|_1, \|\nabla_\theta G_k(x)\|_1, \|\nabla_\theta H_k(x)\|_1 < 4n$, so

$$\|\nabla_\theta A(x)\|_1 < 4n + \sum_{k=1}^{\mathcal{K}} \frac{1}{k^2}\Big(\exp(G_k(x))4n + \exp(H_k(x))4n \Big)$$

Since $|\theta_i| < \mu, \ \forall \theta \in \mathbb{R}^p$, it follows that $|G_k|, |H_k| < 4\mu n$. Consequently:

$$\|\nabla_\theta A(x)\|_1 < 4n + \sum_{k=1}^{\mathcal{K}} \frac{1}{k^2} 4n \Big(\exp(4\mu n) + \exp(4\mu n) \Big) = 4n + \big(8n \exp(4\mu n)\big) \sum_{k=1}^{\mathcal{K}} \frac{1}{k^2}$$

By substitution, the absolutely convergent series $\sum_k k^{-2}$ gives:

$$\|\nabla_\theta A(x)\|_1 < 4n + \big(8n \exp(4\mu n)\big) \sum_{k=1}^{\infty} \frac{1}{k^2} = 4n + \big(8n \exp(4\mu n)\big) \frac{\pi^2}{6}$$

Thus, $\|\nabla_\theta A(x)\|_1$ is bounded $\forall x \in [0,1]^n$ and bounded parameters $|\theta_i| < \mu, \ \forall \theta \in \mathbb{R}^p$ □

Remark 40. *The factor of k^{-2} inside the expression for ABEL-Spline is necessary to ensure the sum converges in the limit of infinitely many exponential terms $\mathcal{K} \to \infty$. This technique could be used on other ANN models to stabilise training.*

6.2.3 Uniform Trainability

Proposition 41 (Trainability of ABEL-Spline)**.** *Let $A(x)$ be an ABEL-Spline function, defined on the domain $x \in [0,1]^n$, with trainable parameters $\theta \in \mathbb{R}^p$ and partition number $z \in \mathbb{N}$. For any $x \in D(A)$, the gradient w.r.t. trainable parameters, $\nabla_\theta A(x)$, is non-zero:*

$$\nabla_\theta A(\theta, x) \neq \vec{0}$$

Proof. Let $A(x)$ be a z-ABEL-Spline function. From the Definition 36, it follows that:

$$\nabla_\theta A(x) = \nabla_\theta \Bigg(F(x) + \sum_{k=1}^{\mathcal{K}} \frac{1}{k^2} \Big(\exp(G_k(x)) - \exp(H_k(x)) \Big) \Bigg)$$

$$\nabla_\theta A(x) = \nabla_\theta F(x) + \sum_{k=1}^{\mathcal{K}} \frac{1}{k^2} \Big(\exp(G_k(x)) \nabla_\theta G_k(x) - \exp(H_k(x)) \nabla_\theta H_k(x) \Big)$$

From Proposition 18 it follows that:

$$\nabla_\theta F(x) \neq \vec{0}\ , \ \nabla_\theta G_k(x) \neq \vec{0}\ , \ \nabla_\theta H_k(x) \neq \vec{0}$$

The functions F, G_k, H_k are independently parameterised, so $\nabla_\theta F, \nabla_\theta G_k, \nabla_\theta H_k$ must be linearly independent. Any linear combination of linearly independent vectors with non-zero coefficients ($\exp(z) \neq 0, \ \forall z \in \mathbb{R}$) must be non-zero. □

Remark 42. *The ABEL-Spline architecture can be adjusted with gradient descent on any part of the domain. This, in combination with bounded gradients, can aid training in practice.*

6.2.4 Min-Distal Orthogonality

Proposition 43 (min-distal orthogonal ABEL-Spline)**.** *Let $A(x)$ be an ABEL-Spline function, defined on the domain $x \in [0,1]^n$, with trainable parameters $\theta \in \mathbb{R}^p$ and partition number $z \in \mathbb{N}$. then for any $x, v \in D(A)$:*

$$\min_i(|x_i - v_i|) > z^{-1} \implies \nabla_\theta A(x) \cdot \nabla_\theta A(v) = 0$$

Proof. Let $A(x)$ be a z-ABEL-Spline function. From the Definition 36, it follows that:

$$\nabla_\theta A(x) = \nabla_\theta F(x) + \sum_{k=1}^{\mathcal{K}} \frac{1}{k^2}\Big(\exp(G_k(x))\nabla_\theta G_k(x) - \exp(H_k(x))\nabla_\theta H_k(x)\Big)$$

Assume that $\min_i(|x_i - v_i|) > z^{-1}$. It follows from proposition 19:

$$\nabla_\theta \Phi(x) \cdot \nabla_\theta \Phi(v) = 0, \; \forall \; \Phi \in \{F, G_1, .., G_k, H_1, .., H_k\}$$

The functions are independently parameterised, so $\forall x, v \in [0,1]^n$ the cross terms are:

$$\nabla_\theta \Psi(x) \cdot \nabla_\theta \Phi(v) = 0, \; \forall \; \Psi \neq \Phi \in \{F, G_1, .., G_k, H_1, .., H_k\}$$

Substituting $\nabla_\theta \Phi(x) \cdot \nabla_\theta \Phi(v) = 0$, and not counting zeroed cross-terms it follows:

$$\nabla_\theta A(x) \cdot \nabla_\theta A(v) = 0 + \sum_{k=1}^{\mathcal{K}} \frac{1}{k^4}\Big(\exp(G_k(x) + G_k(v))\,0 - \exp(H_k(x) + H_k(v))\,0\Big)$$

Finally, $\nabla_\theta A(x) \cdot \nabla_\theta A(v) = 0$, so ABEL-Splines are min-distal orthogonal models. □

Remark 44. *Two sufficiently different points in each input variable have orthogonal parameter gradients. Min-distal orthogonality means ABEL-Splines can be robust to catastrophic interference without other regularisation and training techniques depending on the data distribution, i.e. if min-distal orthogonality holds over sequential tasks. However, memory retention can still be improved in conjunction with other methods.*

6.3 Computational Complexity of ABEL-Splines

Implementation details for TensorFlow can be found in the public GitHub repository[2]. Some minor changes in the order of operations to avoid multiplication during training and inference can be seen in Equation (6.1). The k^{-2} factors are absorbed into the exponentials using logarithms. The logarithm values are computed and stored as bias values in the relevant layers upon creation with exponential activation functions.

$$A(x) := F(x) + \left(\sum_{k=1}^{\mathcal{K}} \exp(G_k(x) - 2\log k)\right) - \left(\sum_{k=1}^{\mathcal{K}} \exp(H_k(x) - 2\log k)\right) \tag{6.1}$$

The interior z-Spline ANN functions $F(x), G_k(x), H_k(x)$ are sums of single-variable cubic spline functions. z denotes the partition number or density of cardinal B-spline intervals, $\mathcal{K}$ denotes the number of (positive and negative) exponentials, and n denotes the input dimension of the model. Each single-variable function has space and time complexity $\mathcal{O}(z)$ and $\mathcal{O}(1)$, respectively. This means the z-Spline ANN functions $F(x), G_k(x), H_k(x)$ have space and time complexity of $\mathcal{O}(nz)$ and $\mathcal{O}(n)$, respectively. Consequently, ABEL-Splines with a one-dimensional output must have a space and time complexity of $\mathcal{O}(nz\mathcal{K})$ and $\mathcal{O}(n\mathcal{K})$, respectively. Finally, for a multi-dimensional z-ABEL-Spline model that maps $[0,1]^n$ to $\mathbb{R}^m$ one has the computational complexities:

$$\text{Space: } \Theta(nz\mathcal{K}m), \text{ and Time: } \Theta(n\mathcal{K}m)$$

ABEL-Spline can be implemented with 1D convolution, reshaping, embedding, multiplication and dense layers in TensorFlow. The same basis functions must be computed for each input variable, hence 1D convolutions. One can compute only the non-zero basis functions with embedding layers by correctly scaling, shifting, and rounding inputs. The implementation was improved throughout the study towards the current version. The ABEL-Spline architecture has exactly $n(4z+3)(2\mathcal{K}+1)m$ trainable parameters.

[2] https://github.com/hpdeventer/MSc-Project-Heinrich-van-Deventer

6.4 Summary

This chapter defined the antisymmetric bounded exponential layer spline (ABEL-Spline) ANN architecture. ABEL-Splines combine z-Spline ANNs, antisymmetric exponentials, and absolutely convergent series to yield models with the desired properties. Section 6.1 provided the mathematical definition of ABEL-Splines. Section 6.2 listed and proved properties of ABEL-Splines. Properties include sparsity, bounded parameter gradients, uniform trainability, universal function approximation, and min-distal orthogonality. Section 6.3 discussed the computational complexity and implementation of ABEL-Splines in TensorFlow. ABEL-Splines are evaluated against more traditional ReLU ANNs in Chapter 7 with a few experiments.

Chapter 7

Experimentation

This chapter presents the experiments conducted to verify the theoretical claims made in Chapter 6 and evaluate the function approximation power of the proposed spline-based models. Section 7.1 describes the perturbation experiments, which test the theoretical predictions for distal interference and orthogonality with standard and proposed models. The two-dimensional experiment in Section 7.2 allows one to observe the performance of spline and standard ANN models and the effect of pseudo-rehearsal in a visual setting. Section 7.3 presents benchmark experiments that estimate the expressive power of the spline-based ANN architectures and function approximation capabilities. Section 7.4 concludes the chapter.

7.1 Perturbation Experiments

This experimental section aims to answer the question: How does a model change over its domain if one modifies or 'perturbs' the model parameters at one point with a gradient descent optimiser? This is a smaller and more tractable problem compared with the more general problem of evaluating how any model changes over its domain during training on any problem and any input dimension using any optimiser. Studying the general case with experimentation is not feasible. Instead, a smaller, more tractable set of experiments are performed.

Consider a trainable model $f(x)$ prior to training. One training data point denoted $v \in D(f)$ is randomly sampled from a uniform distribution over $D(f)$. The target value τ (for $v \in D(f)$) is sampled from a normal distribution. The model is trained on the randomly sampled data with input $v \in D(f)$ and target value τ. The model is evaluated for changes after training. The metric from Equation (3.9) in Chapter 3 is used to quantify how much a model output changed over its domain:

$$\text{Model Perturbation} := \|f' - f\|_1 = \int_D |f'(x) - f(x)|\, dx$$

The integral is estimated using random sampling Monte Carlo integration [105, 106] with points $x^j \in D(f)$ randomly sampled from a uniform distribution over the domain $D(f)$. Since $D(f) = [0,1]^n$, the volume of the domain of integration is exactly 1, so the Monte Carlo estimate reduces to calculating the mean absolute difference. The estimates of the integral are insufficient to show if distal orthogonality prevents distal interference. One also needs to calculate the difference measures between the points $x^j \in D(f)$ used to estimate the integral and the single training data point $v \in D(f)$, and inspect if $|f(x^j) - f'(x^j)| = 0$ whenever $d(x^j, v) > z^{-1}$ for any $x^j \in D(f)$.

For the sake of clarity, let $AP(x) := |f(x) - f'(x)|$ for any $x \in D(f)$ denote the absolute perturbation (at one point). Let $D_{\text{max}}(x) := \max_i(|x_i^{(j)} - v_i|)$ denoted the maximum absolute difference and similarly define the measure $D_{\text{min}}(x) := \min_i(|x_i^{(j)} - v_i|)$ to be the minimum absolute difference. The functions $D_{\text{max}}(x)$ and $D_{\text{min}}(x)$ measure the dissimilarity between any point $x \in D(f)$ and the single training data point $v \in D(f)$.

One can calculate $AP(x^j)$, and the dissimilarity measures $D_{\text{min}}(x^j)$ and $D_{\text{max}}(x^j)$ for all the randomly sampled points sampled from a uniform distribution over $D(f)$. One can use the pair of values to make scatter plots to visualise the distribution. If $D_{\text{min}}(x^j) > z^{-1}$, then $AP(x^j) = 0$ for min-distal orthogonal models. If $D_{\text{max}}(x^j) > z^{-1}$, then $AP(x^j) = 0$ for max-distal orthogonal models.

The rest of the section is organised as follows: Section 7.1.1 outlines the models chosen for the perturbation experiment. Section 7.1.2 outlines the procedure and methods used to obtain the results. Section 7.1.3 shows that ABEL-Splines and lookup tables are min-distal orthogonal models, but deep and wide ReLU ANNs are not min-distal orthogonal models. Section 7.1.4 demonstrates that lookup tables are max-distal orthogonal models, but ABEL-Splines are not max-distal orthogonal models under the maximum norm, as expected. Section 7.1.5 shows that an increased partition number does decrease absolute model perturbations. Section 7.1.6 outlines the interaction between partition number and input dimension and their combined effect on absolute model perturbation. Section 7.1.7 demonstrates the effect of optimisers (Adam and SGD) on absolute model perturbation. Section 7.1.8 inspects the effect of pseudo-rehearsal on absolute model perturbation. Section 7.1.9 discusses the interaction between using Adam and pseudo-rehearsal combined. Section 7.1.10 presents a summary of the perturbation experiments. Mean absolute perturbations and standard deviations for the experiments can be viewed in Table C.1 found in Appendix C.

7.1.1 Considered Models

This study investigates six different types of machine learning models. These models can be broadly categorised into two groups: (1) linear and conventional ReLU ANN models and (2) spline-based and lookup table models. The experiments are performed for different input dimensions n and models with partition numbers $z \in [1, 2, 4, 8, 10]$. Table 7.1 and Table 7.2 show the models and number of parameters.

Table 7.1: Trainable parameters of the considered models.

Model	Number of Parameters
Linear model	$n + 1$
Wide ReLU ANN	$1000n + 2001$
Deep ReLU ANN	$16n + 1937$
ABEL-Splines	$13n(4z + 3)$
z-Spline ANNs	$n(4z + 3)$
Lookup Tables	z^n

Table 7.2: Maximum number of model parameters for different input dimensions.

Model	1D	2D	3D	4D	5D	6D
Linear model	2	3	4	5	6	7
Wide ReLU ANN	3 001	4 001	5 001	6 000	7 001	8 001
Deep ReLU ANN	1 953	1 969	1 985	2 001	2 017	2 033
ABEL-Spline ($z = 10$)	559	1 118	1 677	2 236	2 795	3 354
z-Spline ANN ($z = 10$)	43	86	129	172	215	258
Lookup Table ($z = 10$)	10	100	1 000	10 000	100 000	1000 000

The critical features of these models are as follows:

Linear, Wide- and Deep ReLU ANNs

The linear model $f(x) = w_i x_i + b$ provides a simple baseline for comparison with more complex ANNs. The wide ReLU ANN consists of a single hidden layer with 1000 rectified linear unit (ReLU) neurons connecting to the output layer. In contrast, the deep ReLU ANN has eight hidden layers with 16 ReLU neurons each. Both ANN models include a shift and scaling input layer to map values from $[0, 1]$ to $[-1, +1]$. All trainable parameters are initialised with the uniform Glorot weight initialisation technique [13].

Lookup Tables and Spline-Based ANNs

The lookup table model partitions the input space into z^n cubes with length z^{-1} and trainable parameters and output values for each cube. The lookup table model is a max-distal orthogonal model used for comparison. The z-Spline ANN and ABEL-Spline models are more sophisticated approaches that incorporate spline functions, as discussed in Chapter 4 and Chapter 6. Trainable parameters are initialised with random values sampled from a uniform distribution over $[-1, +1]$. Different versions of each model are considered with partition numbers $z \in [1, 2, 4, 8, 10]$. Additionally, the ABEL-Spline models has $K = 6$ positive and negative exponentials. The ABEL-Splines have exactly $n(4z + 3)(2K + 1)$ trainable parameters for n input dimensions.

7.1.2 Methodology

This section describes the procedures for perturbation experiments in detail, including data generation, model training, evaluation, and experimental configuration.

Training Data Generation (with No Pseudo-Rehearsal)

One training data point denoted $v \in D(f)$ is randomly sampled from a uniform distribution over $D(f) = [0,1]^n$. The target value τ (for $v \in D(f)$) is sampled from a normal distribution.

Training Data Generation with Pseudo-Rehearsal

One training data point denoted $v \in D(f)$ is randomly sampled from a uniform distribution over $D(f) = [0,1]^n$. The target value τ (for $v \in D(f)$) is sampled from a normal distribution. The training data is augmented with 3500 points $u^q \in D(f)$ sampled from a uniform distribution over $D(f)$, and target values $f(u^q)$ are sampled from the model itself. The mixing coefficient ρ from Equation (3.4) is chosen to be $\rho = \frac{7}{17} < 0.5$.[1]

Procedure

Each model f is trained on the randomly sampled training data for each experiment configuration to yield an updated model f'. For each model f before and f' after training (on the point $v \in D(f)$), and for each of the 5000 points $x^j \in D(f)$ sampled from a uniform distribution over $D(f)$. The following are calculated:

1. Absolute point-wise perturbation $AP(x^j) := |f(x^j) - f'(x^j)|$
2. Model perturbation $\int_D |f'(x) - f(x)|\, dx \approx \frac{1}{5000} \sum_j AP(x^j)$
3. Maximum absolute difference $D_{\max}(x^{(j)}) := \max_i(|x_i^{(j)} - v_i|)$
4. Minimum absolute difference $D_{\min}(x^{(j)}) := \min_i(|x_i^{(j)} - v_i|)$

The procedure is repeated for 30 independent trials.

[1] Instead of applying statistical weighting to training data, the point $v \in D(f)$ is duplicated such that the ratio of instances is $\frac{10}{17} > 0.5$ of the entire training data set.

Model Training

All models are trained with mean squared error (MSE), and batch sizes of 128 are used for all 20 training epochs. The optimisers are either SGD or Adam, depending on the experiment configuration.

Experiment Configurations

For each experiment configuration, all models are evaluated in 30 separate trials with different points $v \in [0, 1]^n$ sampled for training in each trial. The following hyperparameters were varied, yielding a total of 24 experimental configurations:

- Input dimensions: One to six
- Optimisers: Adam or SGD
- Pseudo-rehearsal: True or False

7.1.3 Min-Distal Orthogonal Models

The effect of min-distal orthogonality is demonstrated with absolute point-wise perturbations $AP(x^j) := |f(x^j) - f'(x^j)|$, in addition to the minimum absolute differences $D_{\min}(x^{(j)}) := \min_i(|x_i^{(j)} - v_i|)$. The data are shown in scatter plots. The horizontal axes correspond to the minimum absolute difference between input points. The vertical axes represent absolute point-wise perturbations.

ABEL-Splines and Lookup Tables

In Figure 7.1, one can see the absolute point-wise perturbation $AP(x^j) := |f(x^j) - f'(x^j)|$ for the three-dimensional ABEL-Spline and lookup table models on the vertical axis. The minimum absolute difference $D_{\text{min}}(x^{(j)}) := \min_i(|x_i^{(j)} - v_i|)$ is shown on the horisontal axis. The ABEL-Spline and lookup table have the same partition number $z = 2$. One can see that the distributions drop to zero before z^{-1}, which corresponds to 0.5 on the horizontal axis.

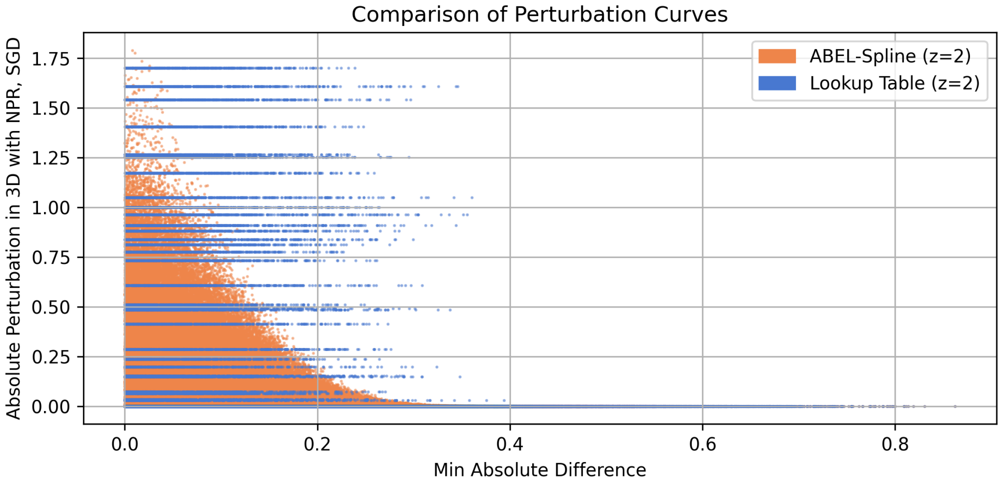


Figure 7.1: Lookup tables and ABEL-Splines are min-distal orthogonal models.

The distributions that drop to zero mean that ABEL-Splines and lookup tables have *zero* distal interference when measuring dissimilarity with $\min_i(|x_i^{(j)} - v_i|)$. The lookup table and ABEL-Spline models exhibit min-distal orthogonality, as expected from the mathematical analysis and proofs of this study.

Deep- and Wide ReLU ANNs

In Figure 7.2, one can see the absolute point-wise perturbation $AP(x^j) := |f(x^j) - f'(x^j)|$ for the three-dimensional deep and wide ReLU ANN models on the vertical axis. The minimum absolute difference $D_{\min}(x^{(j)}) := \min_i(|x_i^{(j)} - v_i|)$ is shown on the horisontal axis. One can see that the distributions do not drop to zero in contrast to the ABEL-Spline and lookup table models in Figure 7.1.

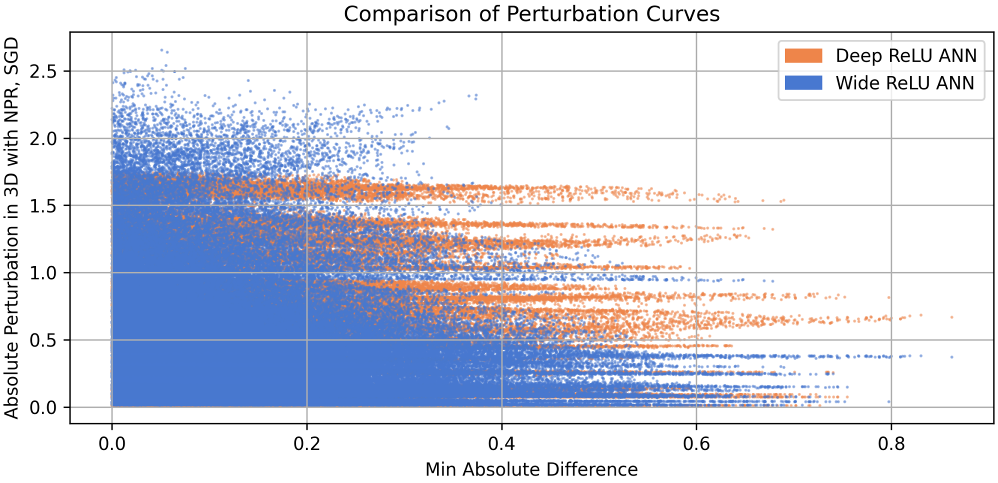


Figure 7.2: ReLU ANNs are not min-distal orthogonal models.

The distributions that do not drop to zero mean that ReLU ANNs have *non-zero* distal interference when measuring dissimilarity with $\min_i(|x_i^{(j)} - v_i|)$. Consequently, ReLU ANNs are not min-distal orthogonal models, which logically implies that ReLU ANNs are also not max-distal orthogonal models. This is an expected result since conventional ReLU ANNs have no mathematically provable guarantees regarding distal orthogonality and distal interference.

7.1.4 Max-Distal Orthogonal Models

In Figure 7.3, one can see the absolute point-wise perturbation $AP(x^j) := |f(x^j) - f'(x^j)|$ for the three-dimensional ABEL-Spline and lookup table models on the vertical axis. The maximum absolute difference $D_{\max}(x^{(j)}) := \max_i(|x_i^{(j)} - v_i|)$ is shown on the horisontal axis. The ABEL-Spline and lookup table have the same partition number $z = 2$.

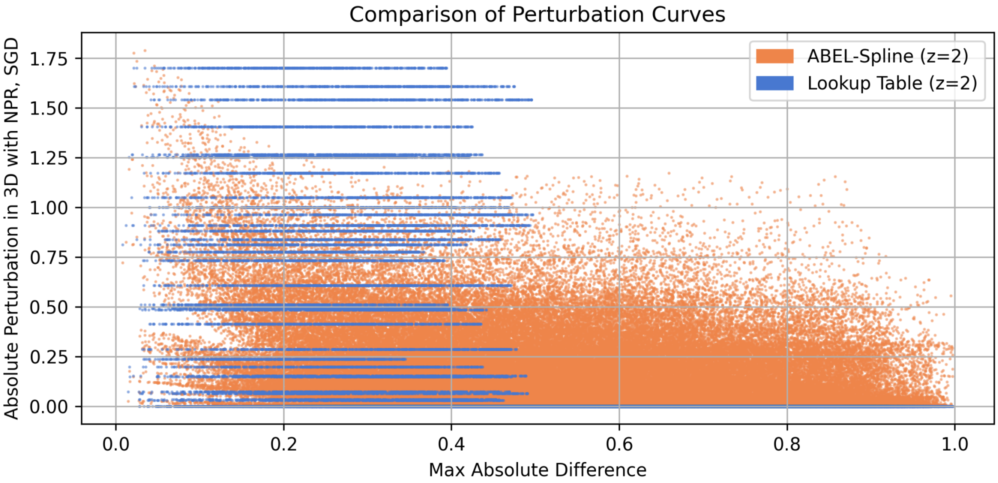


Figure 7.3: Lookup tables are max-distal orthogonal models, but ABEL-Splines are not.

One can see in Figure 7.3 that the lookup table's distribution drops to zero before $D_{\max}(x^{(j)}) = z^{-1}$, which corresponds to 0.5 on the horisontal axis. From the definition of distal interference in Equation (3.10), consider the set of points $D' = \{x \mid d(x, v) > \delta\}$ that sufficiently differ from the training data v. This supports the analysis in this study: For any $\delta > 0$ and difference measure $d(x, v) = \max_i(|x_i - v_i|)$ one can find a lookup table f with partition number $z > \delta^{-1}$ that exhibits no distal interference:

$$\text{Distal Interference} := \int_{D'} |f'(x) - f(x)|\, dx = 0$$

The ABEL-Spline model perturbation does not fall to zero and thus suffers max-distal interference. One can infer that ABEL-Splines are not max-distal orthogonal models. Inspecting deep- and wide ReLU ANN models is unnecessary since ReLU ANNs already failed to exhibit min-distal orthogonality, which logically implies that the ReLU ANNs are also not max-distal orthogonal models.

7.1.5 Influence of Partition Number

In Figure 7.4, one can see the absolute point-wise perturbation $AP(x^j) := |f(x^j) - f'(x^j)|$ for the three-dimensional ABEL-Splines for different partition numbers $z \in [1, 2, 4, 8, 10]$ on the vertical axis. The minimum absolute difference $D_{\min}(x^{(j)}) := \min_i(|x_i^{(j)} - v_i|)$ is shown on the horisontal axis. The ABEL-Splines are colour-coded and labelled according to partition numbers.

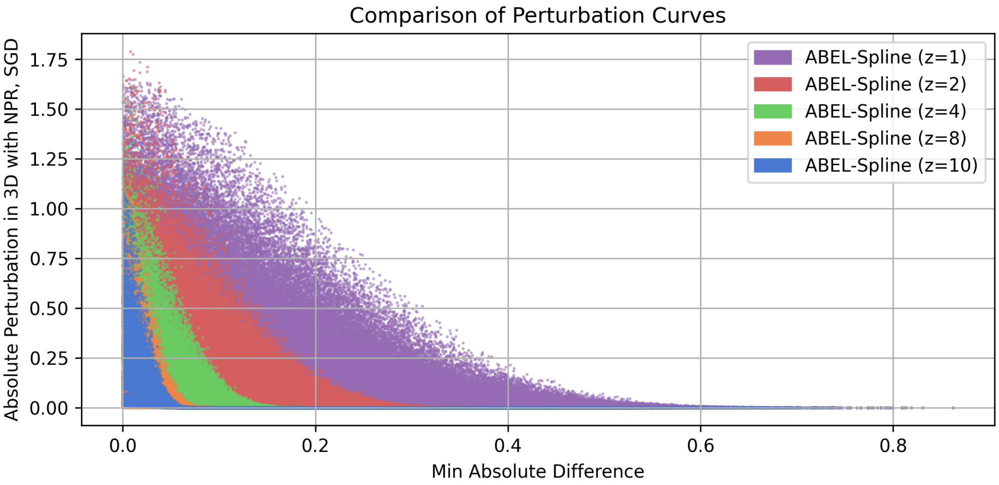


Figure 7.4: Perturbation distributions of ABEL-Splines with different partition numbers.

The distributions in Figure 7.4 drop to *zero* before z^{-1}, which corresponds to the numbers 1, 0.5, 0.25, 0.125, and 0.1 on the horizontal axis. Referring to the definition of distal interference in Equation (3.10), consider the set of points $D' = \{x \mid d(x, v) > \delta\}$ that sufficiently differ from the training data v. This supports the analysis in this study: For any $\delta > 0$ and difference measure $d(x, v) = \min_i(|x_i - v_i|)$ one can find an ABEL-Spline model A with partition number $z > \delta^{-1}$ that exhibits no distal interference:

$$\text{Distal Interference} := \int_{D'} |A'(x) - A(x)|\, dx \approx \int_{D'} \eta\, |\partial_A L|\, |\nabla_\theta A(x) \cdot \nabla_\theta A(y)|\, dx = 0$$

ABEL-Splines exhibit min-distal orthogonality in a very predictable fashion.

The following results inspect the distributions of absolute point-wise perturbation $AP(x^j) := |f(x^j) - f'(x^j)|$ for the three-dimensional ABEL-Splines. A box plot (or whisker plot) of the distributions for different partition numbers is presented in Figure 7.5. The bulk of the distribution mass tends toward zero for a larger partition number. There are outliers at the tail ends of the distribution that remain as the partition number increases due to points where min-distal orthogonality does not hold.

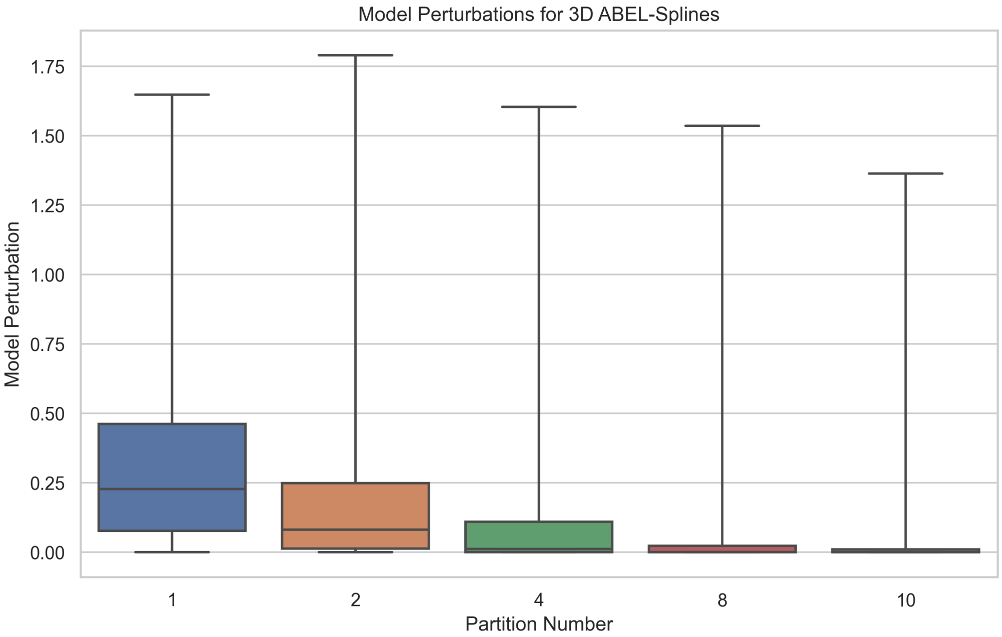


Figure 7.5: Box plot of perturbations for ABEL-Splines with different partition numbers in three dimensions.

The mean model perturbation $\int_D |f'(x) - f(x)|\, dx \approx \frac{1}{5000} \sum_j AP(x^j)$ for ABEL-Splines is shown in Figure 7.6, with different input dimensions as a function of partition number averaged over 30 trials. The mean model perturbation tends to decrease as the partition number increases. This is an expected result, since increased partition number should limit distal interference. What is curious is the lack of an evident trend for increased input dimension, which warrants further scrutiny in Section 7.1.6. For comparison, in Figure 7.7, one can see the line plot and scatter plot of mean model perturbation for lookup tables. There is a clear and consistent trend in Figure 7.7 of smaller mean model perturbation for increased partition number $z \geq 2$ and input dimensions. Lookup tables with $z = 1$ have one parameter regardless of dimension.

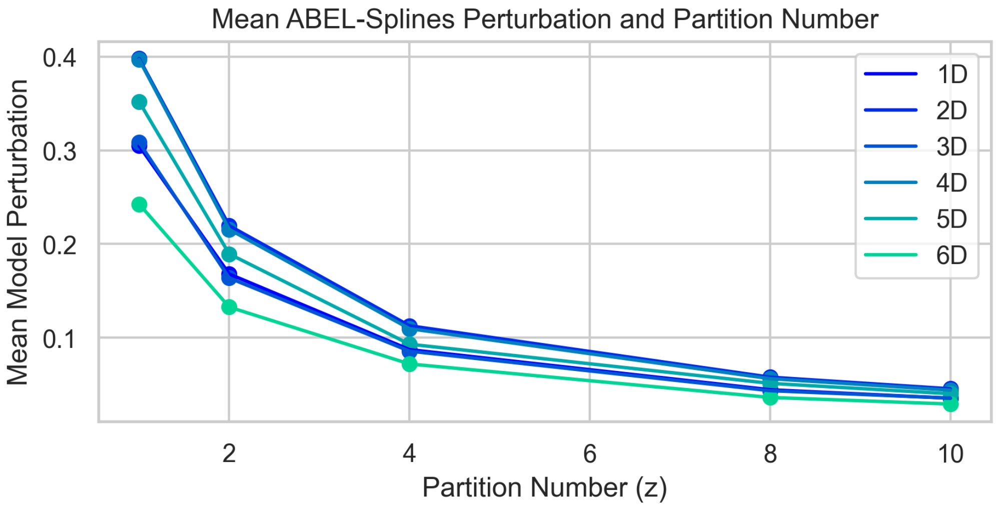


Figure 7.6: Line and scatter plot of mean ABEL-Spline perturbations for different partition numbers and input dimensions.

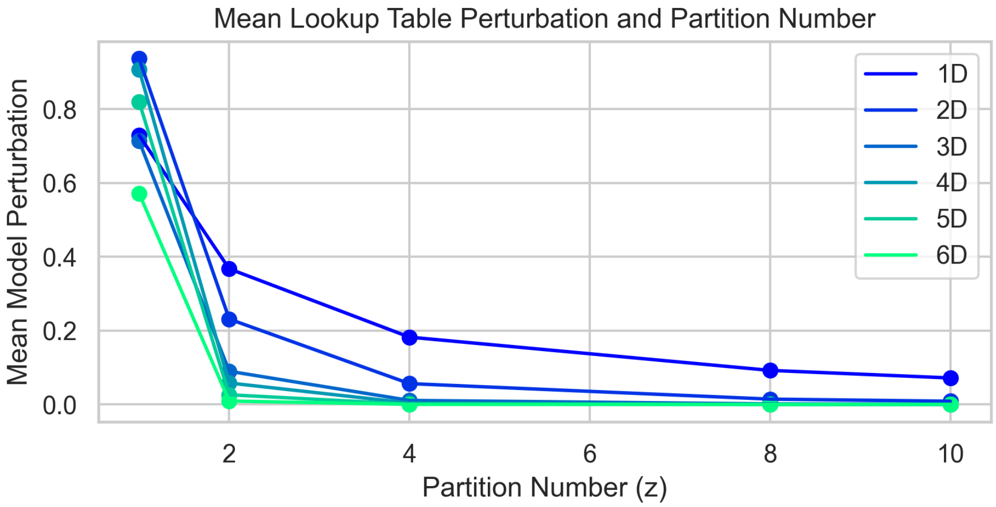


Figure 7.7: Line and scatter plot of mean lookup table perturbations for different partition numbers and input dimensions.

7.1.6 Distal Orthogonality for Different Dimensions

In Figure 7.8, one can see the absolute point-wise perturbation $AP(x^j) := |f(x^j) - f'(x^j)|$ on the vertical axis for the $(z = 2)$ ABEL-Splines for different input dimensions $n \in [1, 2, 3, 4, 5, 6]$. The minimum absolute difference $D_{\min}(x^{(j)}) := \min_i(|x_i^{(j)} - v_i|)$ is shown on the horisontal axis. The ABEL-Splines are colour-coded and labelled according to input dimension. As expected, min-distal orthogonality persists over different dimensions for a specific partition number, $z = 2$. The min-distal orthogonality of ABEL-Splines was proven mathematically in Chapter 6 for any dimension and partition number $z \in \mathbb{N}$, and the perturbation experiment demonstrates empirically that this property holds.

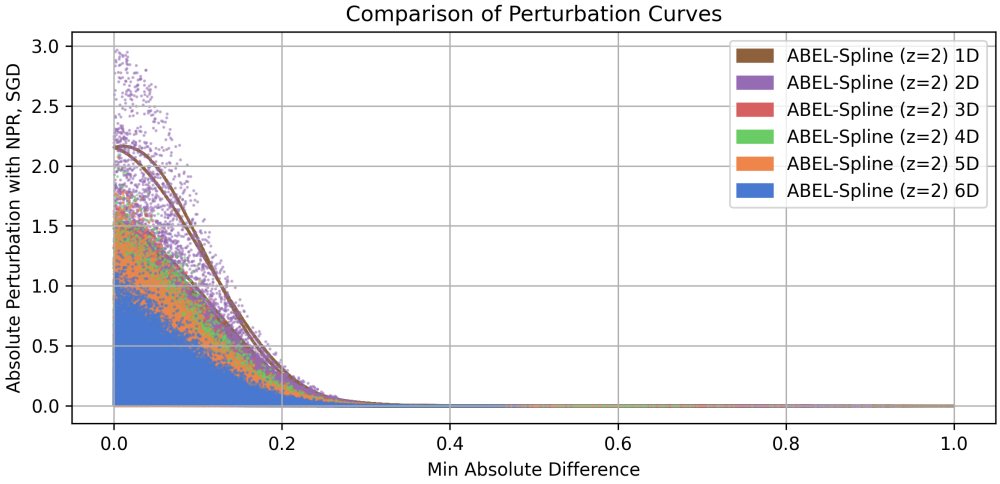


Figure 7.8: Perturbation distributions of ABEL-Splines with different input dimensions.

The mean model perturbation $\int_D |f'(x) - f(x)|\, dx \approx \frac{1}{5000} \sum_j AP(x^j)$ for ABEL-Splines is shown in Figure 7.9, with different partition numbers as a function of input dimension averaged over 30 trials. Figure 7.9 is notable for the absence of a clear trend for increased input dimension. The oscillations seen in Figure 7.9 are a spurious result. All the models are trained and evaluated on the same data for each trial and input dimension. Therefore, if the 2D ABEL-Spline model with $z = 2$ has a larger mean perturbation value, then the 2D ABEL-Spline model with $z = 4$ probably has a larger mean perturbation value. For comparison, in Figure 7.10, one can see the mean model perturbation for lookup tables as a function of input dimension. There is a clear and consistent trend in

Figure 7.10 of smaller mean model perturbation for increased partition number $z \geq 2$ and input dimensions. Lookup tables with $z = 1$ have one parameter regardless of dimension and are more susceptible to distal interference.

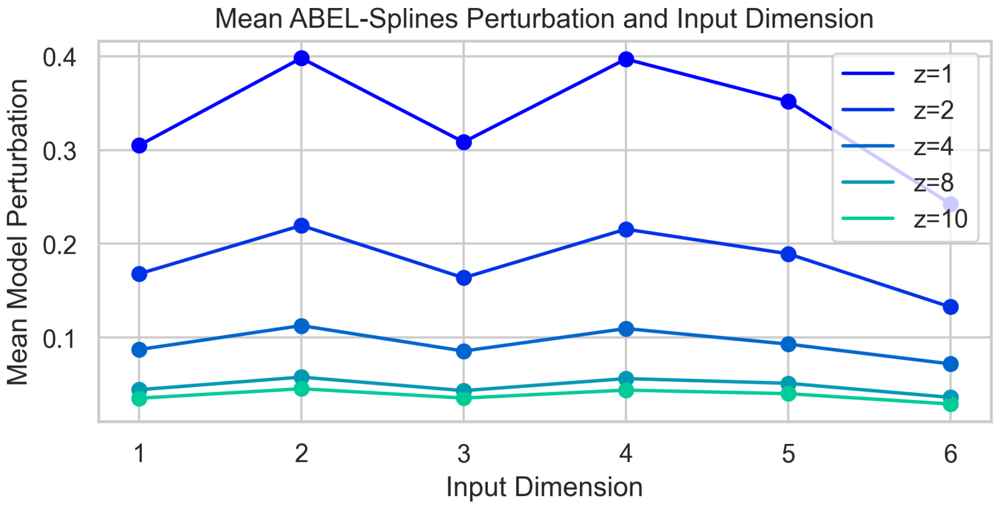


Figure 7.9: Mean perturbation of ABEL-Splines as a function of dimension for different partition numbers.

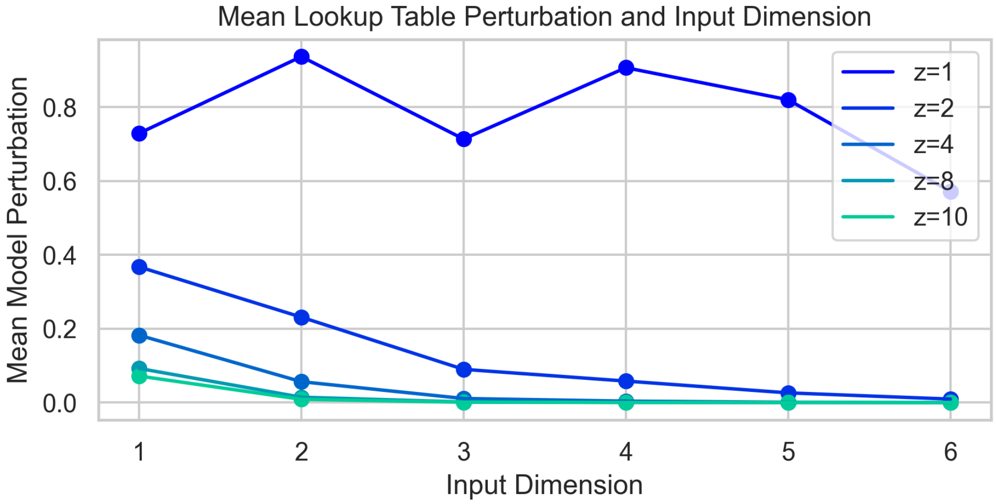


Figure 7.10: Mean perturbation of lookup table models as a function of dimension for different partition numbers.

7.1.7 Optimiser Effect on Model Perturbation

Figure 7.11 shows the absolute point-wise perturbation $AP(x^j) := |f(x^j) - f'(x^j)|$ on the vertical axis for the three-dimensional ABEL-Splines trained using stochastic gradient descent (SGD) or the Adam optimiser. The horisontal axis denotes the minimum absolute difference $D_{\min}(x^{(j)}) := \min_i(|x_i^{(j)} - v_i|)$. The ABEL-Splines have the same partition number $z = 2$. One can see that the distributions drop to zero before z^{-1}, which corresponds to 0.5 on the horizontal axis. ABEL-Splines are still min-distal orthogonal models if trained with Adam.

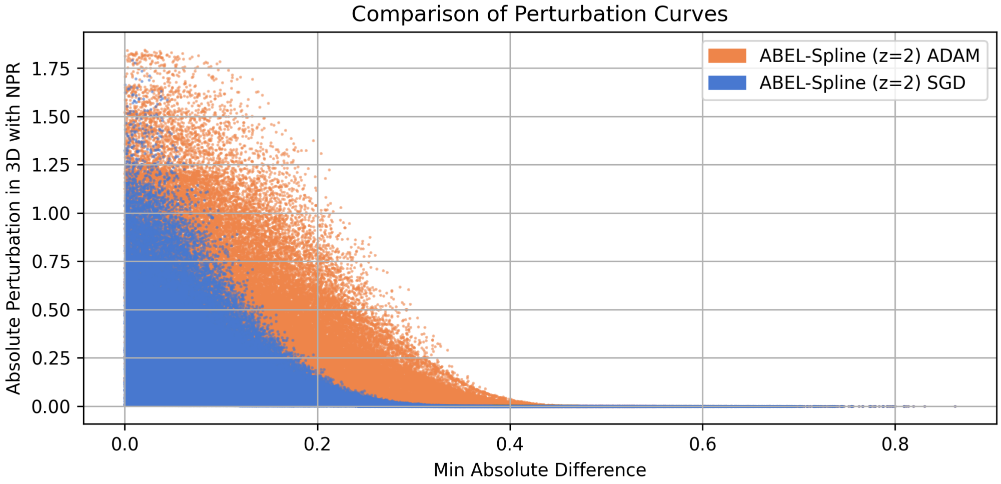


Figure 7.11: Perturbation distributions of ABEL-Splines with Adam and SGD optimisers.

The distribution generated with Adam is more heavy-tailed than SGD, but still reaches zero at z^{-1}. A potential explanation follows: Consider the basis functions shown in Figure 4.5 and Proposition 10. Some of the active and non-zero basis functions further away from an input point have small values close to zero. Thus, the gradient w.r.t. the basis functions are small and close to zero. The rules for the Adam optimiser (see Section 2.3.1) are complicated. One aspect of the Adam optimiser is the tendency to increase the magnitude of extremely small gradients to avoid slow convergence. Adam may scale up small gradients of basis functions with a small amount of overlap, leading to larger absolute changes in ABEL-Splines outputs at points slightly further away. This could explain the heavier tail.

It is necessary to analyse the distribution shift between SGD and Adam. Figure 7.12 shows a whisker plot comparing the distribution of model perturbation $\int_D |f'(x) - f(x)|\,dx \approx \frac{1}{5000}\sum_j AP(x^j)$ of all the models. The data includes all models and input dimensions. The models are trained without pseudo-rehearsal.

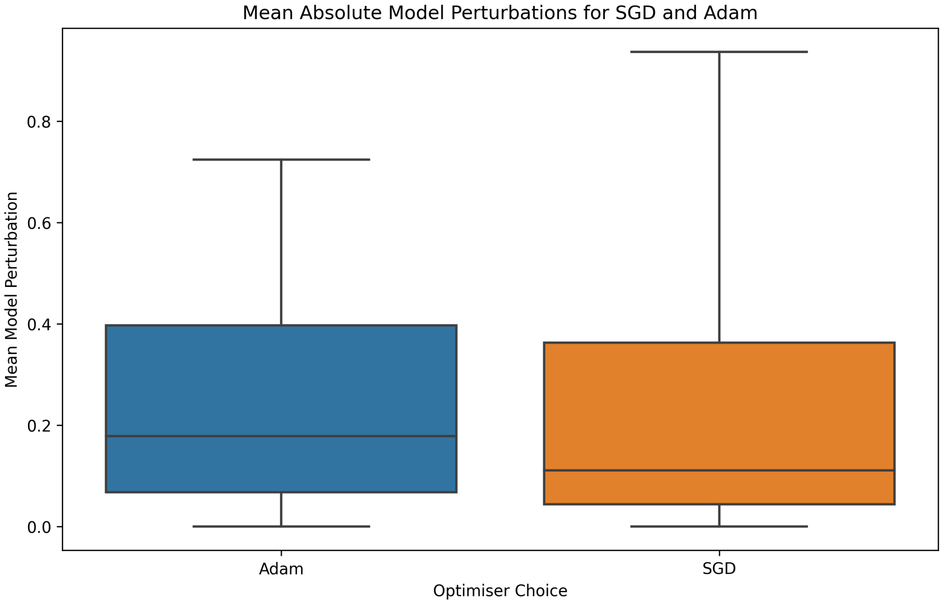


Figure 7.12: Mean model perturbation distributions of all models for different dimensions trained with Adam compared to SGD.

Figure 7.12 shows that the distribution generated with Adam is more evenly balanced than SGD. I.e. the range of values in each quartile is closer to each other in size. The second observation is that SGD produces many mean model perturbation values smaller than Adam and yields larger mean model perturbation values with a larger spread of values. The null hypothesis that the data are normally distributed is rejected with the Shapiro-Wilk test, so the distributions are not normally distributed ($p < 0.01$). It has been shown that specific models exhibit markedly different perturbation distributions, as shown in Figure 7.11.

7.1.8 Pseudo-Rehearsal Effect on Model Perturbation

Pseudo-rehearsal has been used to counteract catastrophic interference [8]. Figure 7.13 and Figure 7.14 show the absolute point-wise perturbation $AP(x^j) := |f(x^j) - f'(x^j)|$ on the vertical axis for wide and deep ReLU ANN models trained with and without pseudo-rehearsal using SGD. The minimum absolute difference $D_{\text{min}}(x^{(j)}) := \min_i(|x_i^{(j)} - v_i|)$ is shown on the horisontal axis.

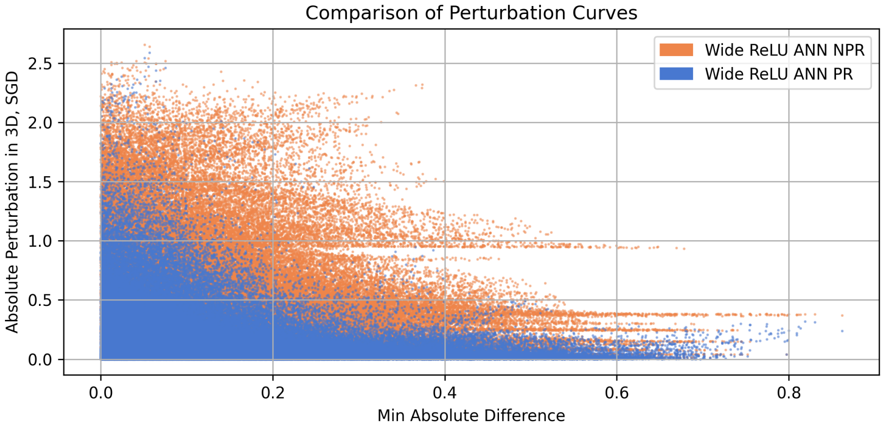


Figure 7.13: Perturbation distributions of wide ReLU ANNs with pseudo-rehearsal and with no pseudo-rehearsal.

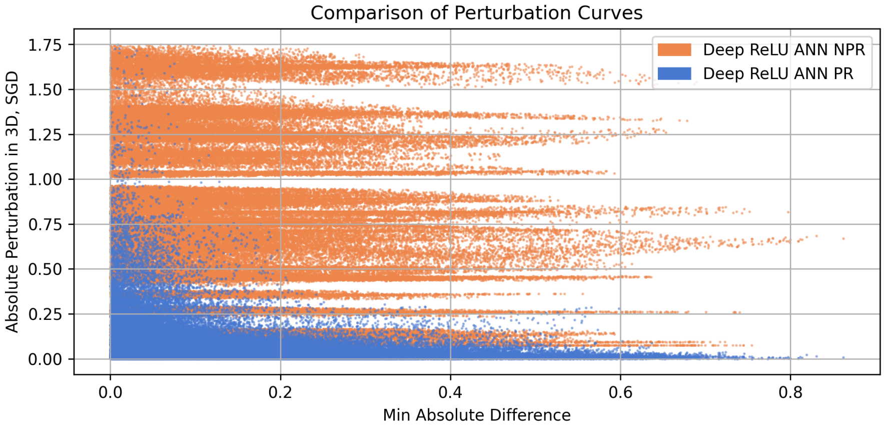


Figure 7.14: Perturbation distributions of deep ReLU ANNs with pseudo-rehearsal and with no pseudo-rehearsal.

Figure 7.13 and Figure 7.14 shows that pseudo-rehearsal decreases the absolute point-wise perturbation of ReLU ANN models, which is expected from the discussion in Section 3.3 and Section 3.7 in Chapter 3. ABEL-Splines trained with SGD exhibit more complicated perturbation distributions if pseudo-rehearsal is applied, as shown in Figure 7.15. The oscillating perturbation distribution is unexpected. Pseudo-rehearsal can perturb models in unforeseen ways that are difficult to predict. The distribution does not drop to zero if pseudo-rehearsal is applied since min-distal orthogonality does not apply to training data sampled from a uniform distribution over the domain.

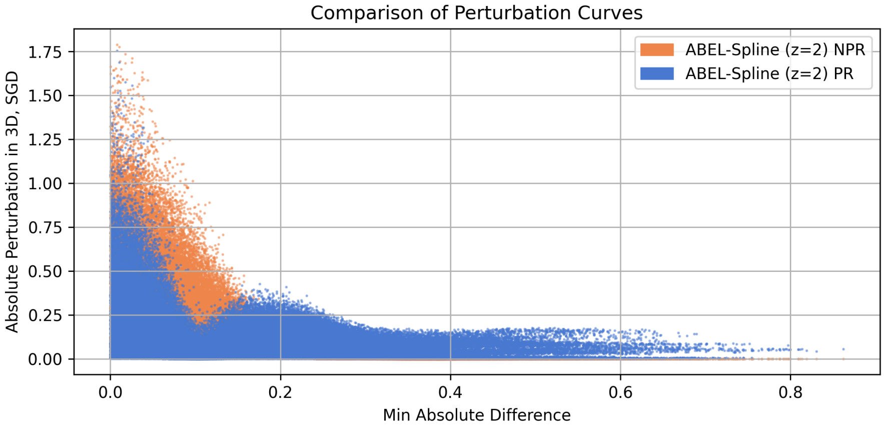


Figure 7.15: Perturbation distributions of an ABEL-Splines with pseudo-rehearsal and with no pseudo-rehearsal.

7.1.9 Model Perturbations with Pseudo-Rehearsal and Adam Combined

The perturbation distributions in Figure 7.16 show three-dimensional ($z = 2$) ABEL-Spline models trained with Adam and compare the application of pseudo-rehearsal with no pseudo-rehearsal. Figure 7.17 shows the same ABEL-Spline models trained with pseudo-rehearsal and compares SGD and Adam. The differences between SGD and Adam diminish if pseudo-rehearsal is used. The unanticipated oscillations seen in Figure 7.15 are present for SGD and Adam optimisers.

In contrast, in the cases where no pseudo-rehearsal is used, as seen in Figure 7.11, Adam and SGD yield strikingly different perturbation distributions. Figure 7.16 and

Figure 7.17 suggest that (larger) datasets, training procedures and regularisation have a more substantial influence on absolute perturbation than the choice of optimiser. Pseudo-rehearsal augments the training set and regularises the model *not* to change a lot over its domain.

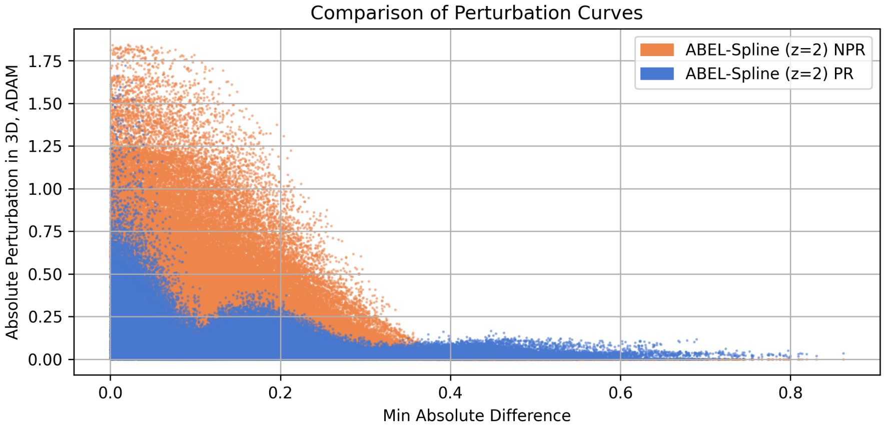


Figure 7.16: Perturbation distributions of ABEL-Splines trained using Adam with pseudo-rehearsal and with no pseudo-rehearsal.

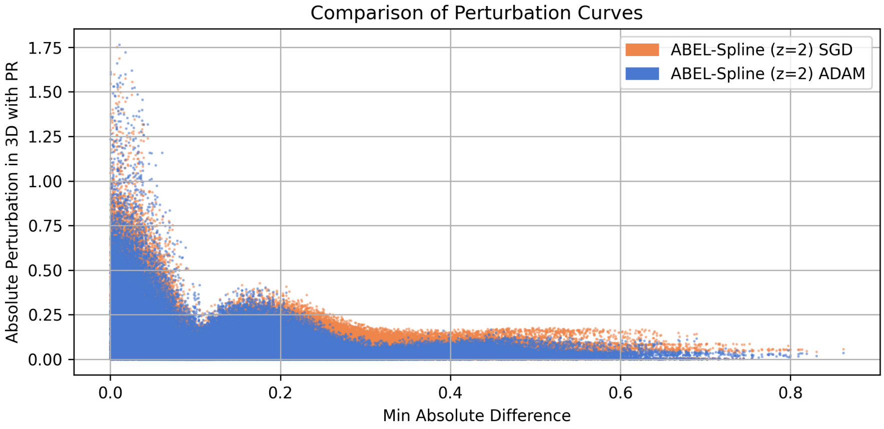


Figure 7.17: Perturbation distributions of ABEL-Splines trained with pseudo-rehearsal using Adam and SGD.

Figure 7.18 shows a whisker plot comparing the distribution of model perturbation $\int_D |f'(x) - f(x)|\, dx \approx \frac{1}{5000} \sum_j AP(x^j)$ of all the models. The data includes all models and input dimensions. The two box plots on the left correspond to the mean absolute model perturbation distribution using Adam and SGD without applying pseudo-rehearsal. The two leftmost box plots in Figure 7.18 correspond to the same data also shown in Figure 7.12, to make the effect of pseudo-rehearsal compared to no pseudo-rehearsal easier to visualise. The two box plots on the right half of Figure 7.18 correspond to the distributions with pseudo-rehearsal being applied.

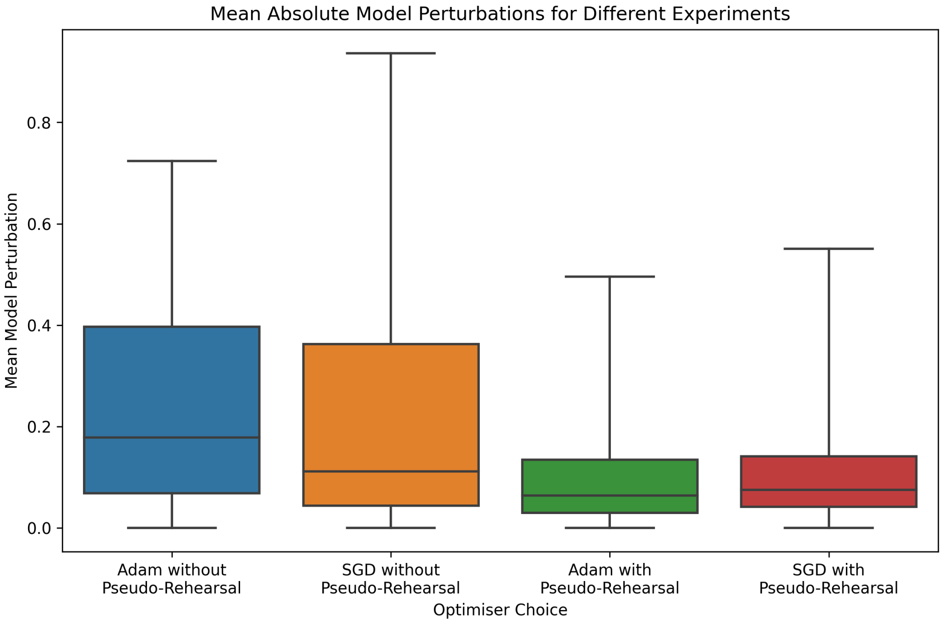


Figure 7.18: Mean model perturbation distributions of all models for different dimensions trained with Adam compared to SGD using pseudo-rehearsal or no pseudo-rehearsal.

The effect of pseudo-rehearsal in Figure 7.18 is potent. If pseudo-rehearsal is applied, then the distribution of model perturbations is shifted towards zero, which is an expected result since pseudo-rehearsal regularises a model not to change too much over its domain (see Section 3.3). The difference between Adam and SGD diminishes when pseudo-rehearsal is applied. This may be due to the distributions shifting towards zero. The Kolmogorov-Smirnov test revealed a significant ($p < 0.01$) difference between using and not using pseudo-rehearsal. However, no significant difference is found between SGD and Adam with applied pseudo-rehearsal.[2]

[2] The data are not normally distributed, so the Kolmogorov-Smirnov test is appropriate.

7.1.10 Summary of Perturbation Experiments

One can calculate the mean absolute perturbation of each model for each experiment for different dimensions, optimisers, and use of pseudo-rehearsal. The mean model perturbation values and standard deviations can be seen in the appended Table C.1. A visual summary of the data using a colour scale to represent different values is shown in Figure 7.19.

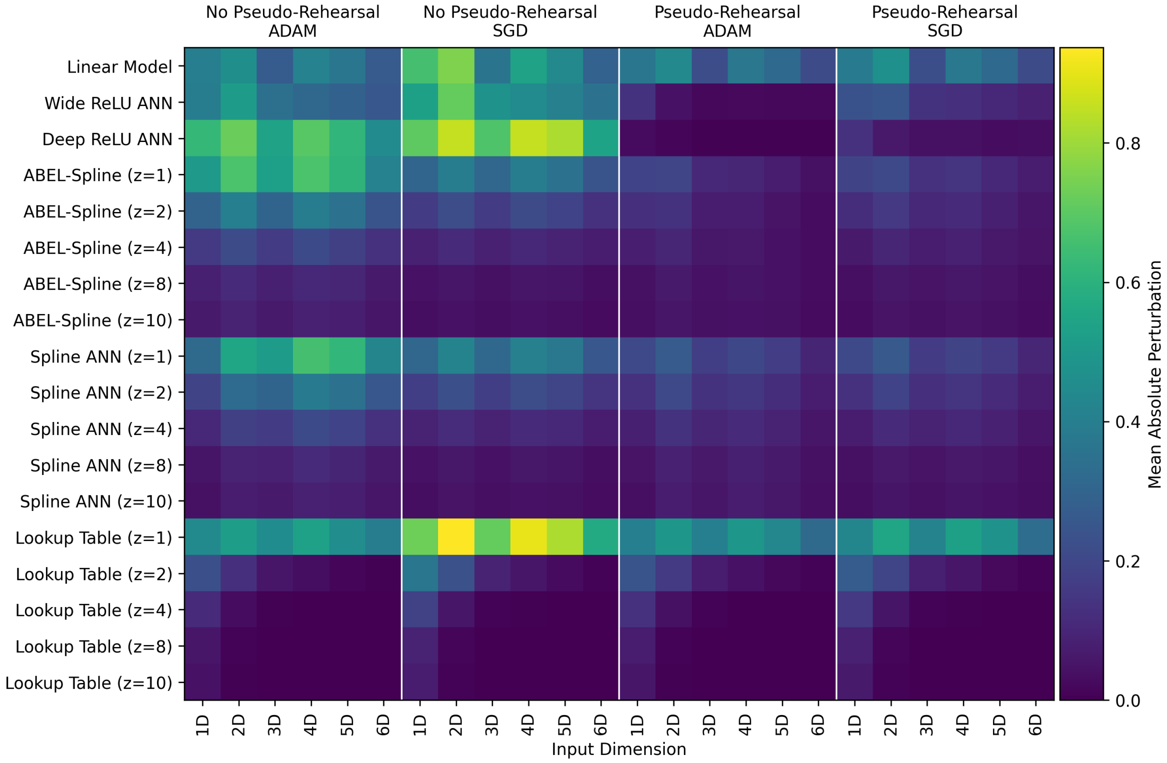


Figure 7.19: Mean absolute perturbation of models over different input dimensions.

The colour bar in Figure 7.19 indicates the mean absolute model perturbation or change. Each row corresponds to the different (labelled) models. Each column corresponds to a different experimental configuration. From left to right, the columns correspond to input dimensions one, two, three, etc. The columns are grouped according to the optimiser (Adam or SGD) and whether pseudo-rehearsal was used.

Some of the trends mentioned in prior sections are visible in Figure 7.19. Increased partition numbers for the ABEL-Spline, z-Spline ANN, and lookup table models evidently lead to smaller mean absolute perturbation values. The effect of partition number was also discussed in Section 7.1.5. The trend for mean absolute perturbation as a function of input dimension was discussed in Section 7.1.6. For lookup table models with partition numbers $z > 1$, the mean perturbation rapidly drops to zero for increasing dimension or partition number. Lookup table models with $z = 1$ are just one-parameter models, hence the consistently large mean perturbation value. There is no consistent trend for changing the input dimension for ABEL-Spline and z-Spline ANN models.

Less pertinent trends can also be seen in Figure 7.19. The choice of optimiser can influence model perturbation, as discussed in Section 7.1.7. The difference between Adam and SGD is most evident in the cases with no pseudo-rehearsal: Adam yields smaller mean perturbation values for ReLU ANNs, linear models and $z = 1$ lookup tables compared with SGD. Adam produces larger mean perturbation values for ABEL-Splines and z-Spline ANNs in the cases without pseudo-rehearsal. The overall effect is that Adam yields less *variation* in mean model perturbation, decreasing it for specific models and increasing it for others. It may be related to the observation that Adam converges quickly during training of ReLU ANNs, but further investigation is needed. On the other hand, SGD yields lower mean perturbation values for Spline models and larger mean perturbation values for ReLU ANNs. This may explain why SGD converges more slowly than Adam for ReLU ANNs during training [12, 58, 59]. However, further study is warranted.

The last notable trend is the cases where pseudo-rehearsal is applied. As discussed in Section 7.1.8, pseudo rehearsal can reduce mean perturbation values. The reduction is striking for the deep ReLU ANN models. Pseudo-rehearsal does alter the mean perturbation values for Spline models, but the effect is complex and difficult to describe succinctly. It has been noted in Section 7.1.9 that the differences between Adam and SGD are diminished if pseudo-rehearsal is applied. This trend can also be seen in Figure 7.19 while comparing the left and right halves of the visualisation.

Many theoretical predictions based on analysis and ideas presented in prior chapters hold up to initial experimentation and scrutiny. It remains to be seen if the theoretical concepts that have been explored translate to better continual learning and a reduction in catastrophic interference. The experiment in Section 7.2 tests the ABEL-Spline model on a two-dimensional continual learning problem.

7.2 Two-Dimensional Demonstration

This section discusses the experiments on a two-dimensional regression task, designed to illustrate universal function approximation and continual learning capabilities of the ABEL-Spline architecture. The models are defined in Section 7.2.1. Section 7.2.2 outlines the 2D regression problem. A synthetic 2D continual learning problem is discussed in Section 7.2.3. Section 7.2.4 demonstrates the effect of pseudo-rehearsal in a 2D setting.

7.2.1 Considered Models

The models considered for this experiment are the same as the models discussed in Section 7.1.1 for input dimension $n = 2$, but $z = 20$. The choice of a larger partition number $z = 20$ is for visual inspection of the lookup table model. Smaller partition numbers make it difficult to discern visually if the lookup table can learn the target function. The spline-based ANN models have the same partition number $z = 20$ for consistency in this experiment. The goal is to inspect the best-case scenario for continual learning. Hence, smaller partition numbers are not considered.

1. Wide ReLU ANN
2. Deep ReLU ANN
3. ABEL-Spline ($z = 20$)
4. z-Spline ANN ($z = 20$)
5. Lookup Table ($z = 20$)

7.2.2 Regression Task

A two-dimensional regression problem on $[0,1]^2 \subset \mathbb{R}^2$ where the target function is a product of two oscillating single-variable functions:

$$y(x) = \sin(4\pi x_1) \cdot \sin(4\pi x_2) \tag{7.1}$$

In this experiment, the target function is specified and known to demonstrate and visualise essential qualities. The target function is shown in Figure 7.20 as an image with a colour bar ranging from -1 to $+1$. The target function resembles a smoothed checkered pattern similar to the XOR problem.

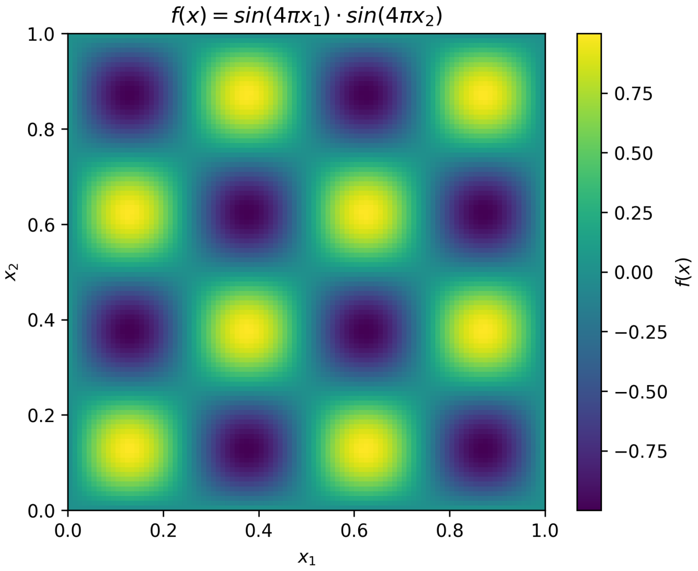


Figure 7.20: The 2D target function.

The 16000 training points are sampled from a uniform distribution over $[0,1]^2 \subset \mathbb{R}^2$, and the target values are calculated from the target function as defined in Equation (7.1). Each model is trained with Adam for 100 epochs with a batch size of 100 using MAE as a loss function.

The models are trained on data uniformly sampled over the domain. The model predictions after training are visualised in Figure 7.21. All the models except the z-Spline ANN model could learn the target function. The piece-wise defined lookup table model has discontinuities between 'flat' regions associated with one parameter, leading to a pixel-like effect. The deep and wide ReLU ANNs and the ABEL-Spline models learn the target function without aberration. This result demonstrates the limited expressive power of z-Spline ANNs and the benefits of developing ABEL-Splines, as discussed in Chapter 4, Chapter 5 and Chapter 6. ABEL-Splines have similar performance to ReLU ANNs in this specific regression problem.

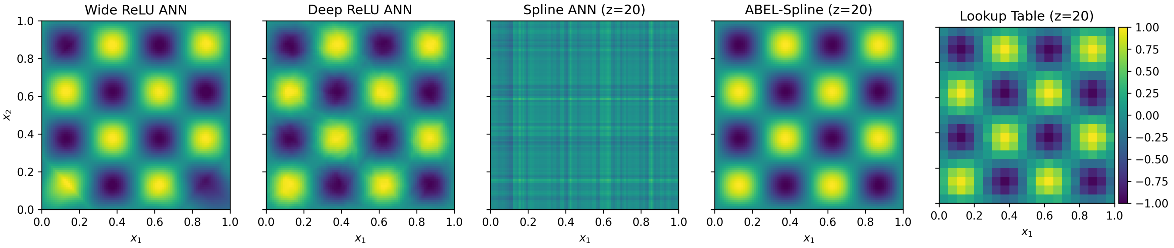


Figure 7.21: Model outputs after training on a 2D regression problem.

7.2.3 Sequential Learning and Catastrophic Interference

This experiment simulates a sequential or continual learning problem that allows visualising the model outputs. The target function is the same as in Equation (7.1) and Figure 7.20. The difference is that the data are not sampled uniformly over the entire domain. Instead, the domain is partitioned into 16 equal-sized regions, shown in Figure 7.22.

The models are trained on 1000 data points sampled inside each partition. After training for 100 epochs on one partition using Adam and batch size of 100, all the models are trained on the second partition and so forth. There are 16000 data points sampled from the target function over all partitions combined.

Figure 7.22 shows the different partitions and the order in which the partitions are sequentially learned. Figure 7.23 shows the model outputs after training on all partitions in the shown (randomly sampled) order. The deleterious effect of distal interference and catastrophic interference is evident.

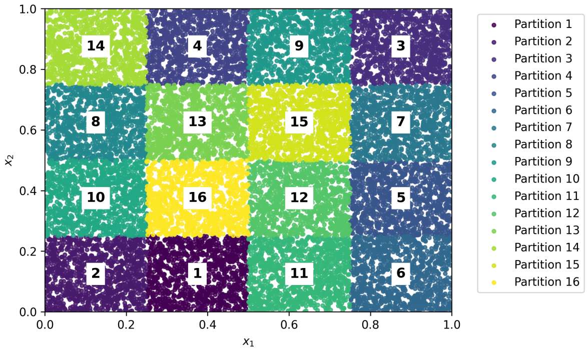


Figure 7.22: Partitions and order for sequential continual learning problem.

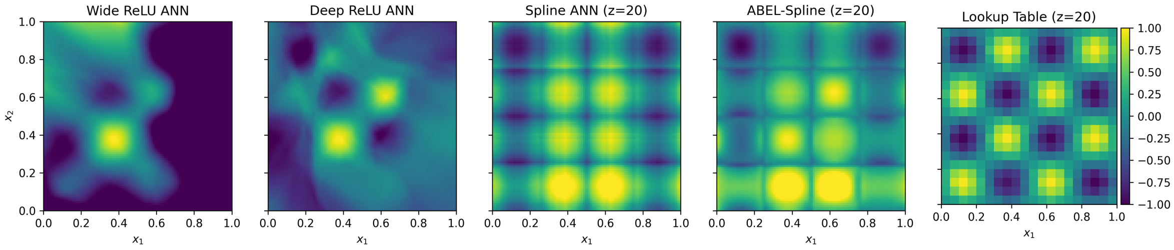


Figure 7.23: Model outputs after training on a 2D continual learning problem.

Only the lookup table model has a nearly identical output, comparing Figure 7.23 and Figure 7.21 against one another, since the lookup table is a max-distal orthogonal model. All other models exhibit catastrophic interference. The ReLU and ABEL-Spline models captured the target values of the last two or so partitions. Surprisingly, the wide and deep ReLU ANN models seem to exhibit slightly better performance regarding memory retention than the ABEL-Spline model. This is due to the order of this specific sequence of partitions since partitions 15 and 16 do not overlap but do overlap with partitions 13 and 12 if one uses the $\min_i(|x_i - v_i|)$ difference between the data.

7.2.4 Sequential Learning and Pseudo-Rehearsal

This experiment extends the experiment discussed in Section 7.2.3. The critical difference is that the training data for each model is augmented with pseudo-rehearsal.

The models f^0 are initialised randomly before training on the first task. The 1000 training data points $(x, y(x))$ for each subsequent partition P from the true target function $y(x)$ is combined with 1000 pseudo-rehearsal input points u that are sampled from a uniform distribution over $[0,1]^2 \subset \mathbb{R}^2$ and target values $f^{P-1}(u)$ calculated from the model after it was trained on the previous task $P-1$. The augmented training dataset consists of 2000 points in total from the target function $(x, y(x))$ and the model itself $(u, f^{P-1}(u))$. This training augmentation allows the (capable) models to learn the target function values inside each partition, while retaining past values on other regions.

The outputs of the models after training with pseudo-rehearsal are shown in Figure 7.24. The ReLU ANNs and ABEL-Spline models can learn the target function sequentially with pseudo-rehearsal. The z-Spline ANN model is still incapable of learning the target function.

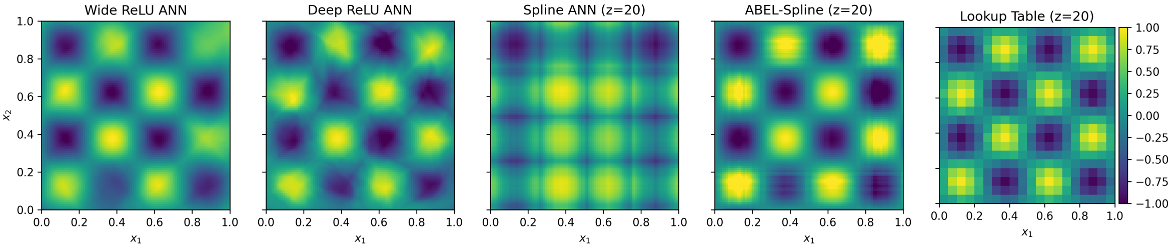


Figure 7.24: Outputs after training on 2D continual learning problem with pseudo-rehearsal.

A potential future improvement to rehearsal techniques might be to remove or filter out rehearsal data that are too close to the training data of interest. Pseudo-rehearsal is adequate for continual learning in a low-dimensional setting.

ABEL-Spline models are insufficient for model-only continual learning on a simple 2D problem without augmentation. Thus, ABEL-Splines cannot learn new tasks continually for arbitrary sequences of tasks without augmentation, such as pseudo-rehearsal. Showing how ABEL-Splines fail in higher dimensions would be redundant. In cases where min-distal orthogonality fails, one must use augmentation techniques.

7.3 PMLB Regression Benchmarks

This section presents the results of a 10-fold cross-validation benchmarking on a set of real-world regression problems. Penn Machine Learning Benchmarks (PMLB) datasets were used [107, 108].

The selection, development, or comparison of machine learning algorithms is complicated since the formats and standards for data storage and organisation are often based on a specific study. PMLB is a collection of curated benchmark datasets for evaluating and comparing supervised machine learning algorithms on many different problems [107, 108]. The datasets cover various applications, including binary/multi-class classification and regression problems, as well as combinations of categorical, ordinal, and continuous features. This study explicitly focuses on all the low-dimensional regression tasks with $n = 6$ or fewer continuous input features due to the exponential z^n size of lookup table models.

The detailed results for mean test and training errors with standard deviations for the mean absolute error (MAE), mean squared error (MSE), and R-squared (R^2) score can be found in Table C.2 in Appendix C.

7.3.1 Considered Models

The models used in this experiment are the same as the models discussed in Section 7.1.1 and shown in Table 7.1. The models are listed again for convenience:

1. Linear model
2. Wide ReLU ANN
3. Deep ReLU ANN
4. ABEL-Splines with partition numbers $z \in [1, 2, 4, 8, 10]$
5. z-Spline ANNs with partition numbers $z \in [1, 2, 4, 8, 10]$
6. Lookup Tables with partition numbers $z \in [1, 2, 4, 8, 10]$

7.3.2 Methodology

The datasets were partitioned using 10-fold cross-validation, creating ten training and testing sets. The input data was preprocessed by standardising it with zero-centering and scaling to unit variance based on the training data. The ABEL-Spline, z-Spline ANN and lookup table models are defined on $[0, 1]^n$, and some of the zero-centred data are outside the domains for which the model implementations are defined. A sigmoid transformation was applied to map all inputs to $[0, 1]^n$, ensuring that the lookup table and ABEL-Spline models were well-posed. For the sake of consistency, all the training data for all the models were transformed with the same procedure. However, this sigmoid procedure is technically unnecessary for ReLU and linear ANNs. The target values were also zero-centred and scaled to unit variance using the training data. The transformations computed using the training data were applied to both the training and test data.

No hyperparameter optimisation or tuning is performed. This experiment is designed to gauge the performance of ABEL-Splines compared to z-Spline ANNs and more conventional ReLU ANNs. The default TensorFlow parameters are used for all training procedures, and the Adam optimiser is used to gauge out-of-the-box performance. The MAE loss function is used for training due to its robustness to outlier target values and non-Gaussian noise compared to the MSE loss function.

All the models were trained for 100 epochs with a default batch size of 32 using the MAE loss function and Adam as an optimiser across all 38 datasets for ten cross-validation folds. After training each model, their respective training MAE, test MAE, test MSE, and test R^2 scores were computed.

7.3.3 Evaluation Results

This section presents visual summaries of the mean test and training errors. The detailed results are in Table C.2 in Appendix C.

Mean Absolute Error

Figure 7.25 shows summary plots of mean training MAE and test MAE from top to bottom. All the models and datasets are shown. Each row corresponds to a model, and columns correspond to datasets.

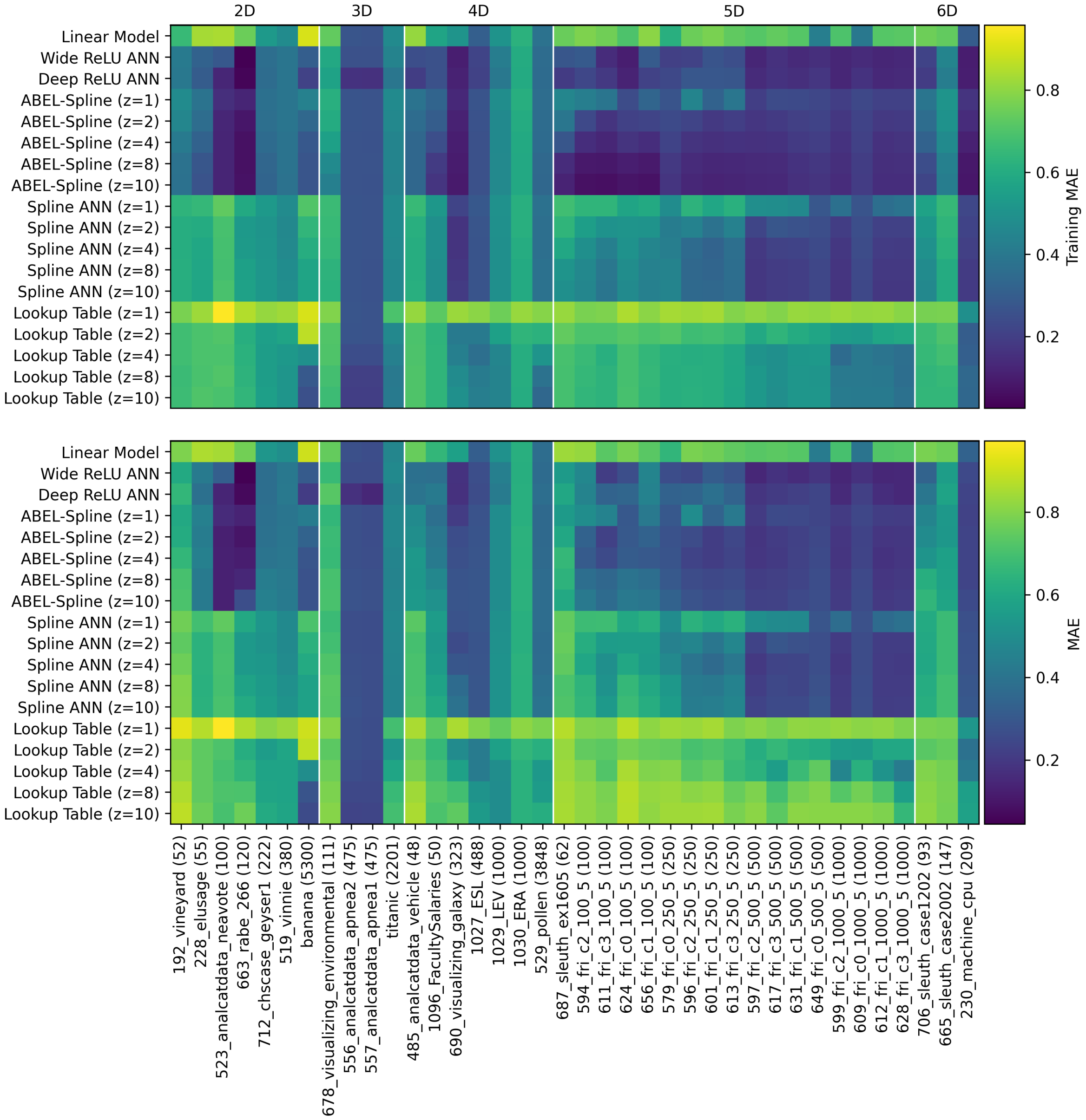


Figure 7.25: Visualisation of averaged training and test MAE on PMLB.

Mean Squared Error and R-Squared

Figure 7.25 shows summary plots of mean MSE and R^2 score on test data from top to bottom. All the models and datasets are shown. Each row corresponds to a model, and columns correspond to datasets.

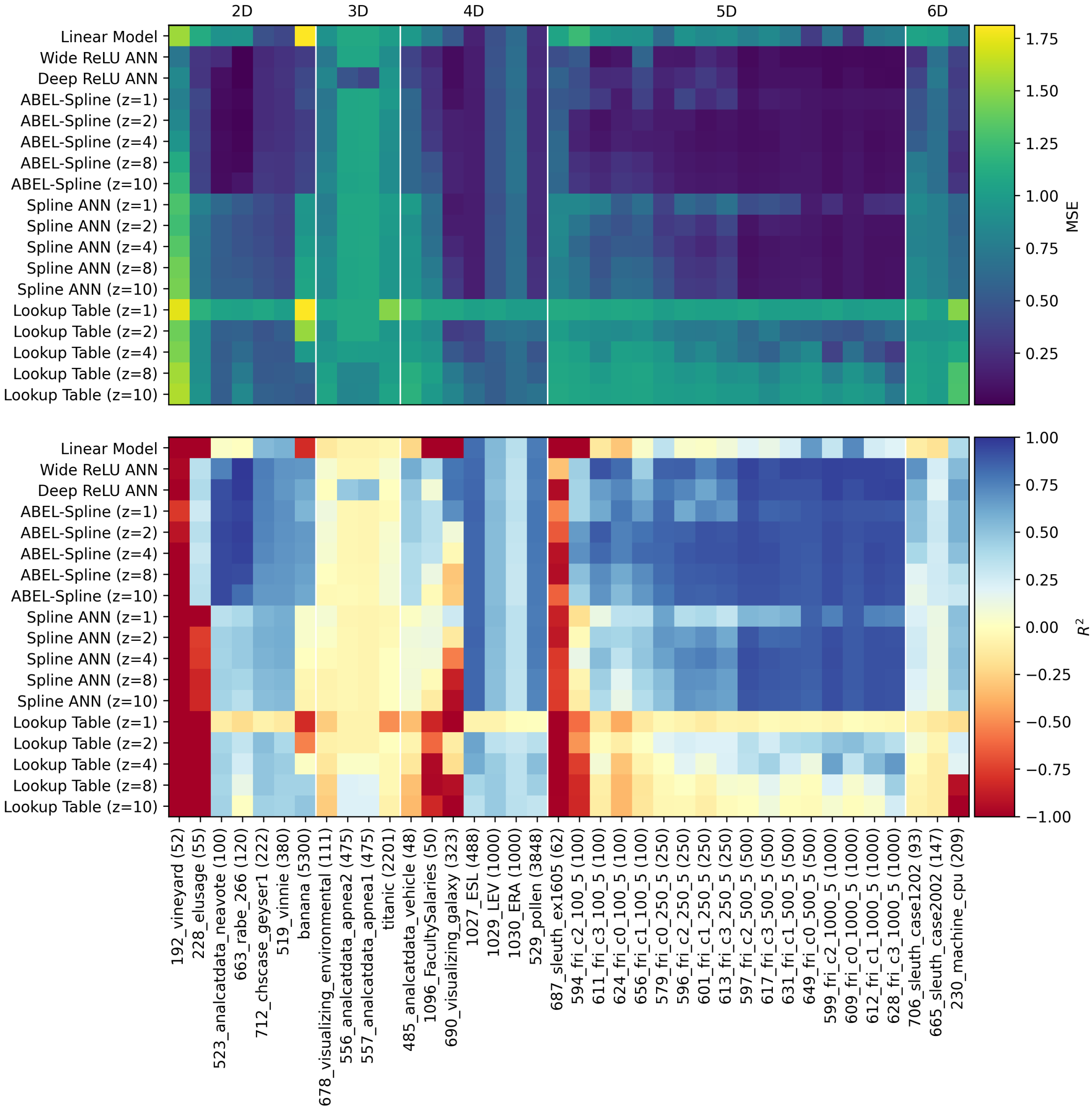


Figure 7.26: Visualisation of averaged test MSE and R^2 score on PMLB.

Figure 7.25 shows summary plots of mean training MAE and test MAE. There is a notable change between ABEL-Splines and z-Spline ANNs, with lower training and test MAE if one uses ABEL-Splines instead of z-Spline ANNs. This highlights the difference in expressive power between z-Spline ANNs and ABEL-Splines, verifying the need to develop ABEL-Splines. Comparing the training error with the test MAE in Figure 7.25 does suggest that the Generalisation gap (difference between training and test error) is relatively small for most models and datasets with some exceptions. ABEL-Splines with large partition numbers $z \geq 8$ tend to have very low training error but larger test MAE on the 5D datasets. This may be due to overfitting or stratification as discussed in Section 4.3.5 and mentioned by Lane et al. [90].

The test MAE in Figure 7.25 show z-Spline ANN, linear and lookup table models tend to have worse test MAE than ABEL-Splines and ReLU ANNs (wide or deep). ABEL-Splines have comparable performance with deep and wide ReLU ANNs for most datasets. The ABEL-Splines performed reasonably well without hyperparameter optimisation and data augmentation techniques to avoid overfitting or stratification.

Figure 7.26 shows summary plots of mean MSE and R^2 score. The results are similar to the test MAE results in Figure 7.25, with small MAE values tending towards smaller MSE and visa versa. Figure 7.26 shows that the MSE and R^2 yield similar results, which is expected since R^2 and MSE are related (see Section 2.3.2). ABEL-Splines in Figure 7.26 have comparable performance with deep and wide ReLU ANNs.

The R^2 score in Figure 7.26 estimates the percentage of the variance in the test sets that are accounted for by a model. Values close to one indicate better performance. A zero (or negative) R^2 score indicates poor performance. In the case of noisy data, it is possible to have an appropriate model that captures the underlying behaviour with a poor R^2 score. The noise in the test set could lead to unexplained variance that the model (rightly) does not embody. R^2 scores cannot be used in isolation, requiring context and analysis of the specific problem to draw valid conclusions. All the models struggled with the ‘192_vineyard (52)’ dataset, with negative R^2 scores for all models. This is hardly surprising since the dataset has 52 data points in total, with 10-fold cross-validation and no data augmentation. None of the models have enough information to learn the underlying problem accurately.

7.4 Summary of All Experiments

Three different types of experiments were performed. The first experiment tested the sensitivity of model outputs to perturbations and verified the mathematically proven properties of ABEL-Splines and z-Spline ANNs. The experiments also showed that ReLU ANNs are not min-distal orthogonal models.

The second experiment demonstrated the importance of universal function approximation and the limits of z-Spline ANNs compared to ABEL-Splines. ABEL-Splines still suffer from catastrophic interference, even though pains were taken to develop a reasonable trade-off between model complexity and designs that prevent distal interference. It is concluded that min-distal orthogonality is insufficient to avoid catastrophic interference. Data augmentation and modified training techniques are probably critical for continual learning with polynomial complexity models.

The final experiment evaluated ABEL-Splines, z-Spline ANNs, lookup table models and ReLU ANNs on regression benchmarks. The benchmarks revealed that ABEL-Splines performed similarly to ReLU ANNs, and ABEL-Splines may perform slightly better on specific problems, depending on the amount of data and underlying problem. The conclusions of the study are presented in Chapter 8.

Chapter 8

Conclusions

This chapter summarises the overarching conclusions of this study. The theoretical and experimental findings are discussed in the context of the research objectives. Potential avenues for future research are also presented. Section 8.1 summarises the most pertinent findings. Future research is discussed in Section 8.2.

8.1 Summary of Conclusions

The first objective was to analyse continual learning and catastrophic interference. Continual learning was explained, and the emergence of catastrophic interference was recounted. The known mechanisms that cause catastrophic interference, such as overlapping representations, were described. This study contributes with an additional refinement: Distant (and unrelated) training inputs with overlapping representations lead to distal interference and catastrophic interference. The importance of distance was not thoroughly considered in the existing literature on catastrophic interference. Techniques to mitigate catastrophic interference from existing literature were reviewed in Chapter 3.

Localised representations introduced in this study overlap if and only if the input points are close enough to each other, as measured by some distance or dissimilarity function. Local representations are orthogonal for sufficiently different inputs, in contrast to distributed or *delocalised* representations in ANNs. However, overlapping (non-orthogonal) representations are theorised to promote generalisation [14].

The second objective was to study model susceptibility to catastrophic interference. The first stage was to investigate specific modes of catastrophic interference in linear regression models and ANN models. The second stage generalised the insights to any differentiable model. A thorough explanation of the properties that make ANNs susceptible to catastrophic interference was given in Chapter 3. A detailed analysis showed why lookup tables are robust to catastrophic interference: Lookup tables have localised representations that are immune to distal interference. This immunity to distal interference is termed distal orthogonality. The first significant contribution of this research showed that max-distal orthogonality and uniform trainability over a differentiable model's domain lead to exponentially large parameter spaces. The contra-positive is that non-exponentially large models do not possess max-distal orthogonality or are not trainable over their entire domain. This finding undermines the potential for polynomial complexity models to learn sequentially without catastrophic interference.

This work provided insight into the mechanics of gradient descent and continual learning under the third objective. It was shown that learning on one part of the domain can affect the model in a distant or non-local region. Training on a task like regression allows the training process to balance changes at distant points in the domain because all the training data is used simultaneously. In continual learning, the training data are not present simultaneously for all tasks. Thus, learning on one task can modify the model in other parts of the domain, degrading performance on previous tasks. Whether distal interference slows down training with SGD is still an open experimental question.

The fourth objective was to develop an efficient architecture capable of continual learning. The ABEL-Spline architecture was designed from first principles for desirable properties like bounded gradients, uniform trainability, and min-distal orthogonality, as discussed in Chapter 6. Antisymmetric exponentials were used to prove the universal function approximation ability of the ABEL-Spline architecture in Chapter 5. A TensorFlow implementation of the architecture was developed and evaluated experimentally. The results showed mixed performance when compared with ReLU models with similar numbers of parameters. The results suggest that min-distal orthogonality is too weak to substantially improve continual learning, especially on uniformly sampled data that do not satisfy the conditions for min-distal orthogonality.

The generalisation and performance of ABEL-Splines on the PMLB regression experiments from Section 7.3 highlighted the significance of choosing the partition number. Generalisation or overfitting depends on the target function and how localised or fine the input space needs to be partitioned. The number of sub-intervals or partition number for a spline corresponds to the spatial resolution of the model. Using small partition numbers constitutes an inductive bias in the model for slow-varying target functions useful for most regression tasks. Large partition numbers constitute an inductive bias for fast-varying target functions and can be useful for anomaly detection, as mentioned in Section 4.3.5.

Generalisation and bridging the gap between training error and test error on unseen data is integral to modern machine learning. The bias-variance decomposition or trade-off for small (under-parameterised) models governed most discussions of generalisation in classical machine learning [109]. The generalisation of over-parameterised ANNs in modern machine learning research was unexpected, with neural tangent kernels constituting a post facto theoretical basis for generalisation [76, 110]. It is not clear which methods of analysis are appropriate for characterising ABEL-Spline generalisation or overfitting.

In summary, min-distal orthogonality is too weak, and max-distal orthogonality is too computationally expensive for practical continual learning. Developing and investigating models with properties between the two extremes might be worthwhile, and careful consideration should be given to generalisation. The method for constructing such models is left for future work. Training augmentation or data augmentation seems necessary for practical and effective continual learning.

8.2 Future Work

This study highlighted a few potential avenues for future research. Some ideas pertain to continual learning, while others are tangential or unrelated to continual learning and catastrophic interference. Some of the recommended future work includes:

1. Developing polynomial complexity models with properties between max-distal and min-distal orthogonality. It is expected that such models would have better continual learning ability than ABEL-Splines while still being more tractable than a lookup table. One could potentially use small or low-dimensional ($n > 1$) max-distal orthogonal models as components in more elaborate multi-variable models over many variables. One need only replace z-Spline ANNs with sums of low-dimensional max-distal orthogonal models, for example:

   $$F(x) = g(x_1, x_2) + h(x_3, x_4, x_5)$$

   where g and h are two- or three-variable max-distal orthogonal models. One can easily extend such models to universal function approximators using antisymmetric bounded exponential layers similar to the ABEL-Spline architecture constructed for this study. It is unclear how effective such architectures could be.

2. Local cluster gradient descent can be developed as a potentially faster training algorithm that keeps distal interference in mind. A summary of the process is:

   (a) Cluster training data with a distance function into neighbourhoods.

   (b) Calculate the average gradient of the loss function on each cluster to lessen the effect of noise.

   (c) Sort the loss gradients from the largest to the smallest error to prioritise clusters with large errors.

   (d) Orthogonalise the gradient vectors associated with each cluster using the Gram-Schmidt procedure [111].

   (e) Apply the computed parameter updates to learn on clusters without distal interference with each update step.

3. The ABEL-Spline architecture created for this study is a shallow neural network with two (effective) hidden layers between input and output. It would be interesting to see if one can extend the architecture to develop deep neural networks with many layers while preserving the original properties. Comparing the performance of such a deep architecture with the shallow ABEL-Spline architecture presented in this work could be an interesting future study.

4. Partial differential equations (PDEs) are challenging to solve. One could evaluate ABEL-Splines' effectiveness in representing solutions to PDEs, where training corresponds to satisfying the boundary conditions and PDEs. The smoothness of ABEL-Splines could be very useful for such applications. A good baseline for comparison would be sinusoidal representation networks called SIRENs [112].

5. Adapting the analysis for antisymmetric exponential for function approximation to other domains like continuous signals is a promising research direction. As an example: The non-linear integral function of the form:

$$A[f] = \exp\left(\int_{t_0}^{t_1} G(t, f(t))\, dt\right) - \exp\left(\int_{t_0}^{t_1} H(t, f(t))\, dt\right),$$

   where G, and H are trainable ANNs. The overall model $A[f]$ maps the function $f(t)$ like an audio signal over some time interval to a scalar number in a non-linear fashion. This could be useful for non-linear signal processing or speech recognition.

Bibliography

[1] Michael McCloskey and Neal J. Cohen. *Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem*, pages 109–165. Psychology of Learning and Motivation. Elsevier Science & Technology, 1989. ISBN 978-0-12-543324-2. doi: 10.1016/S0079-7421(08)60536-8. URL https://doi.org/10.1016/S0079-7421(08)60536-8.

[2] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.01.012. URL https://www.sciencedirect.com/science/article/pii/S0893608019300231.

[3] Z. Chen, B. Liu, R. Brachman, P. Stone, and F. Rossi. *Lifelong Machine Learning: Second Edition*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2018. ISBN 9781681733036. URL https://books.google.ca/books?id=JQ5pDwAAQBAJ.

[4] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. doi: 10.1109/tpami.2021.3057446.

[5] Gido M. van de Ven, Tinne Tuytelaars, and Andreas S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, Dec 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00568-3. URL https://doi.org/10.1038/s42256-022-00568-3.

[6] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[7] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[8] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[9] Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020.

[10] Prakhar Kaushik, Adam Kortylewski, Alex Gain, and Alan Yuille. Understanding catastrophic forgetting and remembering in continual learning with optimal relevance mapping. In *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Pvqe_hQEXTJ.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[12] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2015. URL https://arxiv.org/pdf/1412.6980.pdf.

[13] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/glorot10a.html.

[14] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. *Distributed Representations*, page 77–109. MIT Press, Cambridge, MA, USA, 1986. ISBN 026268053X.

[15] Robert M. French. Pseudo-recurrent connectionist networks: An approach to the 'sensitivity-stability' dilemma. *Connection Science*, 9(4):353–379, 12 1997. URL https://www-proquest-com.uplib.idm.oclc.org/scholarly-journals/pseudo-recurrent-connectionist-networks-approach/docview/206816072/se-2?accountid=14717.

[16] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3762–3773. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/farajtabar20a.html.

[17] Heinrich van Deventer, Pieter Janse van Rensburg, and Anna Bosman. KASAM: Spline additive models for function approximation. *arXiv preprint arXiv:2205.06376*, 2022. URL https://arxiv.org/abs/2205.06376.

[18] Heinrich van Deventer and Anna Bosman. ATLAS: Universal function approximator for memory retention. *arXiv preprint arXiv:2208.05388*, 2022. URL https://arxiv.org/abs/2208.05388.

[19] Heinrich van Deventer and Anna Sergeevna Bosman. Distal interference: Exploring the limits of model-based continual learning. *arXiv preprint arXiv:2402.08255*, 2024. URL https://arxiv.org/abs/2402.08255.

[20] M.F. Bear, B.W. Connors, and M.A. Paradiso. *Neuroscience: Exploring the Brain*. Wolters Kluwer, 2016. ISBN 9780781778176. URL https://books.google.co.za/books?id=vVz4oAEACAAJ.

[21] L. Squire, D. Berg, F.E. Bloom, S. du Lac, A. Ghosh, N.C. Spitzer, and L.R. Squire. *Fundamental Neuroscience*. Elsevier Science, 2008. ISBN 9780080561028. URL https://books.google.co.za/books?id=G0xrtYzmixcC.

[22] Encyclopædia Britannica. myelin sheath. Image on Encyclopædia Britannica, 2023. URL https://www.britannica.com/science/myelin#/media/1/400351/

124171. Caption: The insulating myelin sheath that covers the axons of many neurons is produced by Schwann cells in the peripheral nervous system and by oligodendrocytes in the central nervous system. Accessed: September 12, 2023.

[23] S.A. Siegelbaum, E.R. Kandel, J.D. Koester, and S.H. Mack. *Principles of Neural Science, Sixth Edition*. McGraw-Hill Education, 2021. ISBN 9781259642234. URL https://books.google.co.za/books?id=WPv2zQEACAAJ.

[24] J.G. Nicholls. *From Neuron to Brain*. Sinauer Associates, 2001. ISBN 9780878934393. URL https://books.google.co.za/books?id=OQ3lngEACAAJ.

[25] Casey Paquola, Jakob Seidlitz, Oualid Benkarim, Jessica Royer, Petr Klimes, Richard A. I. Bethlehem, Sara Larivière, Reinder Vos de Wael, Raul Rodríguez-Cruces, Jeffery A. Hall, Birgit Frauscher, Jonathan Smallwood, and Boris C. Bernhardt. A multi-scale cortical wiring space links cellular architecture and functional dynamics in the human brain. *PLOS Biology*, 18(11):1–31, 11 2020. doi: 10.1371/journal.pbio.3000979. URL https://doi.org/10.1371/journal.pbio.3000979.

[26] Robert S. Zucker and Wade G. Regehr. Short-term synaptic plasticity. *Annual Review of Physiology*, 64(1):355–405, March 2002. doi: 10.1146/annurev.physiol.64.092501.114547. URL https://doi.org/10.1146/annurev.physiol.64.092501.114547.

[27] Alvaro Pascual-Leone, Amir Amedi, Felipe Fregni, and Lotfi B. Merabet. The plastic human brain cortex. *Annual Review of Neuroscience*, 28(1):377–401, July 2005. doi: 10.1146/annurev.neuro.27.070203.144216. URL https://doi.org/10.1146/annurev.neuro.27.070203.144216.

[28] Lars Buesing, Johannes Bill, Bernhard Nessler, and Wolfgang Maass. Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLoS Computational Biology*, 7(11):e1002211, November 2011. doi: 10.1371/journal.pcbi.1002211. URL https://doi.org/10.1371/journal.pcbi.1002211.

[29] Gustavo Deco, Viktor K. Jirsa, Peter A. Robinson, Michael Breakspear, and Karl Friston. The dynamic brain: From spiking neurons to neural masses and cortical fields. *PLoS Computational Biology*, 4(8):e1000092, August 2008. doi: 10.1371/journal.pcbi.1000092. URL https://doi.org/10.1371/journal.pcbi.1000092.

[30] T. V. P. Bliss and G. L. Collingridge. A synaptic model of memory: long-term potentiation in the hippocampus. *Nature*, 361(6407):31–39, Jan 1993. ISSN 1476-4687. doi: 10.1038/361031a0. URL https://doi.org/10.1038/361031a0.

[31] Pierre Maquet, Jean-Marie Péters, Joël Aerts, Guy Delfiore, Christian Degueldre, André Luxen, and Georges Franck. Functional neuroanatomy of human rapid-eye-movement sleep and dreaming. *Nature*, 383(6596):163–166, Sep 1996. doi: 10.1038/383163a0. URL https://doi.org/10.1038%2F383163a0.

[32] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[33] Terrence J. Sejnowski. The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences*, 117(48):30033–30038, jan 2020. doi: 10.1073/pnas.1907373117. URL https://doi.org/10.1073%2Fpnas.1907373117.

[34] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, nov 2019. doi: 10.1109/tnnls.2018.2876865. URL https://doi.org/10.1109%2Ftnnls.2018.2876865.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[36] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[37] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. ISSN 0007-4985. doi: 10.1007/BF02478259. URL https://doi.org/10.1007/BF02478259. 115.

[38] Donald O. Hebb. *The organization of behavior : a neuropsychological theory*. Wiley's books in clinical psychology. Wiley, New York, 1949.

[39] P. Werbos. Beyond regression : New tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*, 1974. URL https://cir.nii.ac.jp/crid/1572824499094264320.

[40] Carla J Shatz. The developing brain. *Scientific American*, 267(3):60–67, 1992.

[41] Simon S. 1931 Haykin. *Neural networks and learning machines*. Prentice Hall/Pearson, New York, 3rd ed. edition, 2009. URL http://catdir.loc.gov/catdir/toc/ecip0824/2008034079.html.

[42] M. Minsky, S.A. Papert, and L. Bottou. *Perceptrons, Reissue of the 1988 Expanded Edition with a new foreword by Léon Bottou: An Introduction to Computational Geometry*. The MIT Press. MIT Press, 2017. ISBN 9780262534772. URL https://books.google.co.za/books?id=PLQ5DwAAQBAJ.

[43] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, Berlin, Heidelberg, 2006. ISBN 0387310738.

[44] David E. Rumelhart, James L. McClelland, and PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. The MIT Press, 07 1986. ISBN 9780262291408. doi: 10.7551/mitpress/5236.001.0001. URL https://doi.org/10.7551/mitpress/5236.001.0001.

[45] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 10 1986. doi: 10.1038/323533a0.

[46] Brendan A. Bicknell and Michael Häusser. A synaptic learning rule for exploiting nonlinear dendritic computation. *Neuron*, 109(24):4001–4017.e10, December 2021. doi: 10.1016/j.neuron.2021.09.044. URL https://doi.org/10.1016/j.neuron.2021.09.044.

[47] Matthew E. Larkum. Are dendrites conceptually useful? *Neuroscience*, 489:4–14, May 2022. doi: 10.1016/j.neuroscience.2022.03.008. URL https://doi.org/10.1016/j.neuroscience.2022.03.008.

[48] Terence D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6):459–473, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90044-0. URL https://www.sciencedirect.com/science/article/pii/0893608089900440.

[49] Daniel Svozil, Vladimír Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39(1):43–62, 1997. ISSN 0169-7439. doi: https://doi.org/10.1016/S0169-7439(97)00061-0. URL https://www.sciencedirect.com/science/article/pii/S0169743997000610.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[51] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, Mar 2003. ISSN 1532-4435.

[52] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

[53] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013. URL https://arxiv.org/abs/1310.4546.

[54] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[55] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[56] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2014.09.003. URL https://www.sciencedirect.com/science/article/pii/S0893608014002135.

[57] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Physica-Verlag HD, 2010. doi: 10.1007/978-3-7908-2604-3_16. URL https://doi.org/10.1007%2F978-3-7908-2604-3_16.

[58] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.,

2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/81b3833e2504647f9d794f7d7b9bf341-Paper.pdf.

[59] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from Adam to SGD. *ArXiv*, abs/1712.07628, 2017. URL https://arxiv.org/abs/2102.11343.

[60] P.J. Bickel and K.A. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics*. Number v. 2 in A Chapman & Hall book. CRC Press, 2015. ISBN 9781498722681. URL https://books.google.co.za/books?id=x0e_tQEACAAJ.

[61] M.H. DeGroot. *Optimal Statistical Decisions*. Wiley Classics Library. Wiley, 2004. ISBN 9780471680291. URL https://books.google.co.za/books?id=1pXFDwAAQBAJ.

[62] Jun Qi, Jun Du, Sabato Marco Siniscalchi, Xiaoli Ma, and Chin-Hui Lee. On mean absolute error for deep neural network based vector-to-vector regression. *IEEE Signal Processing Letters*, 27:1485–1489, 2020. doi: 10.1109/lsp.2020.3016837. URL https://doi.org/10.1109%2Flsp.2020.3016837.

[63] Davide Chicco, Matthijs J. Warrens, and Giuseppe Jurman. The coefficient of determination r-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *PeerJ Computer Science*, 7:e623, jul 2021. doi: 10.7717/peerj-cs.623. URL https://doi.org/10.7717%2Fpeerj-cs.623.

[64] S.A. Glantz, B.K. Slinker, and T.B. Neilands. *Primer of Applied Regression & Analysis of Variance 3E*. McGraw Hill LLC, 2016. ISBN 9780071822442. URL https://books.google.co.za/books?id=E73NCwAAQBAJ.

[65] N.R. Draper and H. Smith. *Applied Regression Analysis*. Wiley Series in Probability and Statistics. Wiley, 2014. ISBN 9781118625682. URL https://books.google.co.za/books?id=uSReBAAAQBAJ.

[66] J.L. Devore. *Probability and Statistics for Engineering and the Sciences*. Probability and Statistics for Engineering and the Sciences. Thomson/Brooks/Cole,

2008. ISBN 9781337762021. URL https://books.google.co.za/books?id=HUNuswEACAAJ.

[67] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, Aaron Courville, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states, 2018. URL https://arxiv.org/abs/1806.05236.

[68] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization, 2017. URL https://arxiv.org/abs/1710.09412.

[69] Hongyu Guo, Yongyi Mao, and Richong Zhang. Mixup as locally linear out-of-manifold regularization, 2018. URL https://arxiv.org/abs/1809.02499.

[70] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/0efbe98067c6c73dba1250d2beaa81f9-Paper.pdf.

[71] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[72] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/ioffe15.html.

[73] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

[74] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013. URL https://arxiv.org/abs/1312.6211.

[75] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/content/114/13/3521.

[76] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf.

[77] Daoyun Ji and Matthew A. Wilson. Coordinated memory replay in the visual cortex and hippocampus during sleep. *Nature Neuroscience*, 10(1):100–107, Jan 2007. ISSN 1546-1726. doi: 10.1038/nn1825. URL https://doi.org/10.1038/nn1825.

[78] Larry R. Squire. Memory and the hippocampus: A synthesis from findings with rats, monkeys, and humans. *Psychological Review*, 99(2):195–231, 1991. ISSN 0033-295X. 195.

[79] J. Hawkins and R. Dawkins. *A Thousand Brains: A New Theory of Intelligence*. Basic Books, 2021. ISBN 9781541675803. URL https://books.google.co.za/books?id=hYrvDwAAQBAJ.

[80] Yuwei Cui, Subutai Ahmad, and Jeff Hawkins. Continuous online sequence learning with an unsupervised neural network model. *Neural Computation*, 28(11):2474–2504, nov 2016. doi: 10.1162/neco_a_00893. URL https://doi.org/10.1162%2Fneco_a_00893.

[81] Dileep George and Jeff Hawkins. Towards a mathematical theory of cortical micro-circuits. *PLoS Computational Biology*, 5(10):e1000532, oct 2009. doi: 10.1371/journal.pcbi.1000532. URL https://doi.org/10.1371%2Fjournal.pcbi.1000532.

[82] P. Kanerva. *Sparse Distributed Memory*. Bradford Bks. Cambridge, Mass., 1988. ISBN 9780262111324. URL https://books.google.co.za/books?id=I9tCr21-s-AC.

[83] C. De Boor. *A Practical Guide to Splines*. Applied Mathematical Sciences. Springer New York, 1978. ISBN 9783540903567. URL https://books.google.co.za/books?id=mZMQAQAAIAAJ.

[84] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-Spline Techniques*. Mathematics and Visualization. Springer Berlin Heidelberg, 2013. ISBN 9783662049198. URL https://books.google.co.za/books?id=fEiqCAAAQBAJ.

[85] D. Salomon. *Curves and Surfaces for Computer Graphics*. Springer New York, 2007. ISBN 9780387284521. URL https://books.google.co.za/books?id=r5o5biZPDKEC.

[86] I.J. Schoenberg. *Cardinal Spline Interpolation*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1973. ISBN 9780898710090. URL https://books.google.co.za/books?id=O3ipetjS_64C.

[87] K. Hollig. *Finite Element Methods with B-Splines*. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics, 2012. ISBN 9780898716993. URL https://books.google.co.za/books?id=V9foSFYQnNgC.

[88] K. Hollig and J. Horner. *Approximation and Modeling with B-Splines*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 2015. ISBN 9781611972948. URL https://books.google.co.za/books?id=ynYXAgAAQBAJ.

[89] R.N. Goldman and T. Lyche. *Knot Insertion and Deletion Algorithms for B-Spline Curves and Surfaces*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 1993. ISBN 9780898713060. URL https://books.google.co.za/books?id=eiLmAgAAQBAJ.

[90] Stephen Lane, Marshall Flax, David Handelman, and Jack Gelfand. Multi-layer perceptrons with b-spline receptive field functions. In R.P. Lippmann, J. Moody, and D. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3. Morgan-Kaufmann, 1990. URL https://proceedings.neurips.cc/paper_files/paper/1990/file/94f6d7e04a4d452035300f18b984988c-Paper.pdf.

[91] Andre Sevaldsen Douzette. B-splines in machine learning. Master's thesis, Department of Mathematics, University of Oslo, 2017.

[92] Simone Scardapane, Michele Scarpiniti, Danilo Comminiello, and Aurelio Uncini. Learning activation functions from data using cubic spline interpolation. In *Neural Advances in Processing Nonlinear Dynamic Signals*, pages 73–83. Springer International Publishing, jul 2018. doi: 10.1007/978-3-319-95098-3_7. URL https://doi.org/10.1007%2F978-3-319-95098-3_7.

[93] John P. Boyd. Defeating the Runge phenomenon for equispaced polynomial interpolation via Tikhonov regularization. *Applied Mathematics Letters*, 5(6):57–59, 1992. ISSN 0893-9659. doi: https://doi.org/10.1016/0893-9659(92)90014-Z. URL https://www.sciencedirect.com/science/article/pii/089396599290014Z.

[94] Bengt Fornberg and Julia Zuev. The runge phenomenon and spatially variable shape parameters in rbf interpolation. *Computers And Mathematics with Applications*, 54(3):379–398, 2007. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2007.01.028. URL https://www.sciencedirect.com/science/article/pii/S0898122107002210.

[95] S. De Marchi, F. Marchetti, E. Perracchione, and D. Poggiali. Polynomial interpolation via mapped bases without resampling. *Journal of Computational and Applied Mathematics*, 364:112347, 2020. ISSN 0377-0427. doi: https://doi.org/10.1016/j.cam.2019.112347. URL https://www.sciencedirect.com/science/article/pii/S0377042719303449.

[96] Simon N Wood. Stable and efficient multiple smoothing parameter estimation for generalized additive models. *Journal of the American Statistical Association*, 99 (467):673–686, sep 2004. doi: 10.1198/016214504000000980. URL https://doi.org/10.1198%2F016214504000000980.

[97] T.J. Hastie. *Generalized Additive Models*. CRC Press, 2017. ISBN 9781351445962. URL https://books.google.co.za/books?id=ZLs6DwAAQBAJ.

[98] C. Molnar. *Interpretable Machine Learning*. Leanpub, 2020. ISBN 9780244768522. URL https://books.google.co.za/books?id=jBm3DwAAQBAJ.

[99] Simone Scardapane, Michele Scarpiniti, Danilo Comminiello, and Aurelio Uncini. Learning activation functions from data using cubic spline interpolation. In *Italian Workshop on Neural Nets*, pages 73–83. Springer, 2017.

[100] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4602–4609, Jul 2019. doi: 10.1609/aaai.v33i01.33014602. URL https://doi.org/10.1609%2Faaai.v33i01.33014602.

[101] Yang You, Yujing Lou, Qi Liu, Yu-Wing Tai, Lizhuang Ma, Cewu Lu, and Weiming Wang. Pointwise rotation-invariant network with adaptive sampling and 3D

spherical voxel convolution. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):12717–12724, apr 2020. doi: 10.1609/aaai.v34i07.6965. URL https://doi.org/10.1609%2Faaai.v34i07.6965.

[102] Edward Wagstaff, Fabian B. Fuchs, Martin Engelcke, Michael A. Osborne, and Ingmar Posner. Universal approximation of functions on sets. *Journal of Machine Learning Research*, 23(151):1–56, 2022. URL http://jmlr.org/papers/v23/21-0730.html.

[103] Robert S Strichartz. *The way of analysis*. Jones & Bartlett Learning, 2000.

[104] W. Rudin. *Principles of Mathematical Analysis*. International series in pure and applied mathematics. McGraw-Hill, 1976. ISBN 9780070856134. URL https://books.google.co.za/books?id=kwqzPAAACAAJ.

[105] M.H. Kalos and P.A. Whitlock. *Monte Carlo Methods*. Monte Carlo Methods. Wiley, 2008. ISBN 9783527617401. URL https://books.google.co.za/books?id=IEBjLmmbzDMC.

[106] I. Dimov. *Monte Carlo Methods for Applied Scientists*. World Scientific, 2008. ISBN 9789812779892. URL https://books.google.co.za/books?id=pV1hDQAAQBAJ.

[107] Joseph D Romano, Trang T Le, William La Cava, John T Gregg, Daniel J Goldberg, Praneel Chakraborty, Natasha L Ray, Daniel Himmelstein, Weixuan Fu, and Jason H Moore. PMLB v1.0: an open-source dataset collection for benchmarking machine learning methods. *Bioinformatics*, 38(3):878–880, October 2021. doi: 10.1093/bioinformatics/btab727. URL https://doi.org/10.1093/bioinformatics/btab727.

[108] Randal S. Olson, William La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, 10(36):1–13, Dec 2017. ISSN 1756-0381. doi: 10.1186/s13040-017-0154-4. URL https://doi.org/10.1186/s13040-017-0154-4.

[109] Jerome H. Friedman. On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1(1):55–77, Mar 1997. ISSN 1573-756X. doi: 10.1023/A:1009778005914. URL https://doi.org/10.1023/A:1009778005914.

[110] Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt, and Yi Ma. Rethinking bias-variance trade-off for generalization of neural networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10767–10777. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/yang20j.html.

[111] 1929 Cheney, E. W. (Elliott Ward) and David (David Ronald) Kincaid. *Linear algebra : theory and applications*. Jones & Bartlett Learning, Sudbury, MA, 2nd edition, 2012.

[112] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7462–7473. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf.

Appendix A

Acronyms

This appendix lists and defines acronyms and abbreviations used in this work.

ABEL	Antisymmetric bounded exponential layer
ABEL-Spline	Antisymmetric bounded exponential layer spline
ADAM	Adaptive moment estimation
AI	Artificial intelligence
ANN	Artificial neural network
CNN	convolutional neural network
FFNN	Feedforward neural network
LTD	Long-term (neural) depression
LTP	Long-term (neural) potentiation
n	Number of perturbations or dataset instances
N	Number of data points in a dataset
R^2	Coefficient of determination, goodness of fit also called R-squared
MAE	Mean absolute error
MLP	Multi-layer perceptron
MSE	Mean squared error
NPR	No pseudo-rehearsal
OGD	Orthogonal gradient descent

PDE	Partial differential equation
PMLB	Penn machine learning benchmarks
PR	Pseudo-rehearsal
ReLU	Rectified linear unit
REM	Rapid eye movement (sleep phase)
RNN	Recurrent neural network
Spline ANN	Spline artificial neural network
SIREN	Sinusoidal representation network
SDR	Sparse distributed representation
SGD	Stochastic gradient descent
T. MSE	Test mean squared error
T. MAE	Test mean absolute error
Tr. MAE	Final (training) mean absolute error

Appendix B

Symbols

This appendix lists and defines the numerous mathematical symbols and operations used throughout this work. The definitions are grouped according to the relevant chapters, with their first occurrence in most cases.

B.1 Chapter 1: Introduction

f	A differentiable model trainable with gradient descent optimisation
f'	A modified model after training with updated or perturbed parameters
$D(f)$	The domain of a model f, same as $D(f')$
v	A training data point $v \in D(f)$
x	Any input point $x \in D(f)$
$d(x, v)$	Some general non-negative distance or difference measure $d(x, v) > 0$
δ	Some positive number $\delta > 0$
$\|f(x) - f'(x)\|$	The absolute difference between function values
ε	Some positive number $\varepsilon > 0$
$\max_i(\|x_i - v_i\|)$	Maximum norm distance $\\|x - v\\|_\infty$ between points $x, v \in D(f)$
$\min_i(\|x_i - v_i\|)$	Minimum absolute difference between points, not a proper metric

B.2 Chapter 2: Artificial Neural Networks

σ	Activation function
x	Input or vector
x_i	Scalar input to the neuron
w_i	Weight associated with input x_i
b	Bias term
n	Input dimension or number of inputs to neuron
y_i	Actual value in the loss functions (MAE, MSE)
$\hat{y}_i$	Predicted value in the loss functions (MAE, MSE)
$L(\cdot)$	Loss function
R^2	Coefficient of determination (evaluation metric)
V	Vocabulary size in embedding layers
D	Desired embedding dimension in embedding layers
θ	Trainable model parameters
∇_θ	Gradient operator w.r.t. model parameters θ
$\nabla_\theta L(\theta^{(t)}, x_i, y_i)$	Point estimate of loss function gradient
t	Time step or iteration number in optimisation algorithm
$\hat{g}^{(t)}$	Averaged estimate of gradient at iteration t
η	Learning rate or step size for a gradient descent optimiser
$m^{(t)}$, $v^{(t)}$, $\hat{m}$, $\hat{v}$	Variables used in Adam optimisation algorithm
MAE	Mean Absolute Error
MSE	Mean Squared Error
R^2	Coefficient of determination, goodness of fit also called R-Squared

B.3 Chapter 3: Catastrophic Interference and Continual Learning

$v \cdot u$	The inner, scalar, or dot-product of vectors v and u
$f(x)$	Function f evaluated at x
$f(\theta, x)$	Function f with trainable parameters θ evaluated at x
$f(\theta_0, x)$	Function f with initial parameters θ_0 evaluated at x
c	Constant term in a function
k	Index variable in a sum of terms added together
$a_k,\ b_k$	Real coefficients
x^k	Monomial term in a polynomial function, $k \in \mathbb{N}^0$
$\hat{g}$	Update of the parameters θ based on gradient of loss function
y	Previously learned input
v	Input data for pseudo-rehearsal or generative replay
$P(x, y)$	Data distribution of the current task
$P(v)$, $U_{D(f)}$	Distribution of input data for generative replay
ρ	Mixing coefficient for pseudo-rehearsal
$u^{(i)} \in [0,1]^n$	Grid-point in the domain of f
θ	Trainable parameters $\theta \in \mathbb{R}^p$ of a model f
$\nabla_\theta f(x)$	Parameter gradient of a model f
$z \in \mathbb{N}$	Partition number of a model f
$x, v \in [0,1]^n$	Points in the domain of a model f
$\nabla_\theta f(x) \cdot \nabla_\theta f(v)$	Overlap between v and x in a model f
$\Theta(z^n)$	Asymptotically tight (upper and lower) bound on number of vectors
$\Omega(z^n)$	Asymptotic lower bound on the number of model parameters
$R(f)$	Expected risk of model f
$\ell(f(x), y)$	Loss between model's output $f(x)$ and target value y
$D(f)$, $D(f(\theta))$	Domain of function f or function with parameters θ

B.4 Chapter 4: Spline Artificial Neural Networks

z	The partition number or density of cardinal B-spline intervals
$f(x)$	Spline Artificial Neural Network
$f_j(x_j)$	Single-variable B-spline function in z-Spline ANN
x, y	Input vectors in domain
$D(f)$	Domain of function $f(x)$ and $f(x)$
$S_i(x)$	Cardinal cubic B-spline basis functions
$S_{i,j}(x_j)$	Basis functions in z-Spline ANN
$\|x - v\|$	Absolute difference between two real numbers x, v
$\|\|x - v\|\|_{-\infty}$	Minimum absolute difference between points: $\min_i(\|x_i - v_i\|)$
$\|\|x - v\|\|_{\infty}$	Maximum norm distance between points $\max_i(\|x_i - v_i\|)$
$\nabla_\theta f(x)$	Gradient w.r.t parameters θ of $f(x)$, evaluated at x

B.5 Chapter 5: Universal Function Approximation

exp	Exponential function
S	Set
X	A set of points or a space
d	Metric function
(X, d)	A metric space with distance metric d
$C_b(X)$	Normed subspace of bounded continuous functions on X
(X, d)	Compact metric space
$(x_1, .., x_n)$	Components of the vector x
$g_{k,j}, h_{k,j}$	Continuous single-variable functions
G_k, H_k	Sum-decomposable functions: Sums of single-variable functions
α	Scalar constant
Ψ	Set of antisymmetric exponentials
M, M', M''	Denote the number of positive or negative exponentials

B.6 Chapter 6: ABEL-Splines: Antisymmetric Bounded Exponential Layer Splines

K	The number of (positive and negative) exponentials
z	The partition number or density of cardinal B-spline intervals
n	The input dimension of the model
m	The output dimension of the model
$A(x)$	ABEL-Spline function
$F(x)$	z-Spline ANN function for $A(x)$
$G_k(x)$	z-Spline ANN function for positive exponential term in $A(x)$
$H_k(x)$	z-Spline ANN function for negative exponential term in $A(x)$
$f_j(x_j)$	Single-variable density B-spline function in $A(x)$
$g_{k,j}(x_j)$	Single-variable density B-spline function in $A(x)$
$h_{k,j}(x_j)$	Single-variable density B-spline function in $A(x)$

B.7 Chapter 7: Experimentation

$f(x)$	Model function before perturbation or training with an optimiser
$f'(x)$	Model function after weights have been changed by some process
z	Partition number for ABEL-Splines, Spline ANNs, and lookup tables
K	Number of positive and negative exponentials in ABEL-Splines models
$v \in D(f)$	Random input point for training
$\tau \in \mathbb{R}$	Random target value from normal distribution for training
$\|f' - f\|_1$	Model perturbation equal to $\int_D \|f'(x) - f(x)\| \, dx$
$x^j \in D(f)$	Randomly sampled points from a uniform distribution over $D(f)$
$AP(x)$	Absolute point-wise perturbation equal to $\|f(x) - f'(x)\|$
$\frac{1}{5000} \sum_j AP(x^j)$	Approximates $\|f' - f\|_1$ when the domain is $D(f) = [0,1]^n$
$D_{\max}(x^{(j)})$	Equal to $\max_i(\|x_i^{(j)} - v_i\|)$ max-distance from training data
$D_{\min}(x^{(j)})$	Equal to $\min_i(\|x_i^{(j)} - v_i\|)$ min-distance from training data

B.8 Chapter 8: Conclusions

$A[f]$	The non-linear integral function
$f(t)$	A function such as an audio signal over some time interval
$\exp$	Exponential function
$G,\ H$	Trainable ANNs
t_0 **and** t_1	Time interval limits for the integral operation
dt	Differential of time, used in the integral operation
$\int dt$	Integral operation over time

Appendix C

Experimentation Data

Tables of raw experimental data with means and standard deviations are presented in this appendix.

C.1 Perturbation Experiment Data

The means and standard deviations (indicated with a ±-symbol) averaged over 30 trials for the perturbation experiments discussed in Section 7.1 are presented in Table C.1. The table columns correspond to dimension, model names and experiment configurations (Adam or SGD, pseudo-rehearsal or no pseudo-rehearsal). The columns from left to right are: The input dimension n; the model name; no pseudo-rehearsal (NPR) with Adam; NPR with SGD; pseudo-rehearsal (PR) with Adam; and PR with SGD.

Table C.1: Mean and standard deviation of absolute model perturbation.

n	**Model**	**NPR with ADAM**	**NPR with SGD**	**PR with ADAM**	**PR with SGD**
1	Linear Model	0.402 ± 0.340	0.659 ± 0.597	0.369 ± 0.377	0.391 ± 0.424
1	Wide ReLU ANN	0.394 ± 0.414	0.532 ± 0.552	0.136 ± 0.309	0.236 ± 0.341
1	Deep ReLU ANN	0.623 ± 0.528	0.703 ± 0.531	0.027 ± 0.145	0.135 ± 0.230
1	ABEL-Spline (z=1)	0.508 ± 0.493	0.305 ± 0.398	0.190 ± 0.300	0.191 ± 0.297
1	ABEL-Spline (z=2)	0.297 ± 0.440	0.168 ± 0.329	0.131 ± 0.264	0.123 ± 0.262
1	ABEL-Spline (z=4)	0.161 ± 0.355	0.087 ± 0.251	0.079 ± 0.222	0.068 ± 0.214
1	ABEL-Spline (z=8)	0.083 ± 0.270	0.044 ± 0.186	0.044 ± 0.173	0.038 ± 0.169
1	ABEL-Spline (z=10)	0.067 ± 0.244	0.035 ± 0.165	0.039 ± 0.165	0.030 ± 0.150
1	Spline ANN (z=1)	0.324 ± 0.244	0.311 ± 0.409	0.208 ± 0.265	0.208 ± 0.316
1	Spline ANN (z=2)	0.192 ± 0.241	0.173 ± 0.338	0.134 ± 0.233	0.134 ± 0.285
1	Spline ANN (z=4)	0.103 ± 0.198	0.088 ± 0.254	0.084 ± 0.198	0.075 ± 0.230
1	Spline ANN (z=8)	0.053 ± 0.152	0.046 ± 0.190	0.048 ± 0.159	0.042 ± 0.180
1	Spline ANN (z=10)	0.043 ± 0.137	0.036 ± 0.168	0.038 ± 0.143	0.033 ± 0.161
1	Lookup Table (z=1)	0.449 ± 0.231	0.729 ± 0.565	0.403 ± 0.288	0.429 ± 0.333

Continued on next page

Table C.1 – Continued from previous page

n	Model	NPR with ADAM	NPR with SGD	PR with ADAM	PR with SGD
1	Lookup Table (z=2)	0.227 ± 0.279	0.367 ± 0.539	0.242 ± 0.333	0.272 ± 0.400
1	Lookup Table (z=4)	0.113 ± 0.227	0.182 ± 0.421	0.132 ± 0.286	0.155 ± 0.360
1	Lookup Table (z=8)	0.057 ± 0.171	0.092 ± 0.315	0.070 ± 0.223	0.085 ± 0.290
1	Lookup Table (z=10)	0.045 ± 0.153	0.072 ± 0.278	0.055 ± 0.201	0.067 ± 0.260
2	Linear Model	0.467 ± 0.393	0.756 ± 0.692	0.440 ± 0.443	0.470 ± 0.523
2	Wide ReLU ANN	0.519 ± 0.617	0.714 ± 0.545	0.046 ± 0.148	0.250 ± 0.353
2	Deep ReLU ANN	0.724 ± 0.546	0.856 ± 0.564	0.014 ± 0.086	0.065 ± 0.153
2	ABEL-Spline (z=1)	0.672 ± 0.576	0.398 ± 0.413	0.194 ± 0.274	0.206 ± 0.252
2	ABEL-Spline (z=2)	0.405 ± 0.460	0.220 ± 0.308	0.139 ± 0.203	0.154 ± 0.192
2	ABEL-Spline (z=4)	0.217 ± 0.342	0.113 ± 0.222	0.100 ± 0.168	0.101 ± 0.157
2	ABEL-Spline (z=8)	0.112 ± 0.248	0.058 ± 0.160	0.067 ± 0.138	0.062 ± 0.130
2	ABEL-Spline (z=10)	0.089 ± 0.221	0.046 ± 0.143	0.056 ± 0.127	0.050 ± 0.121
2	Spline ANN (z=1)	0.555 ± 0.353	0.418 ± 0.424	0.268 ± 0.305	0.260 ± 0.302
2	Spline ANN (z=2)	0.328 ± 0.313	0.228 ± 0.325	0.207 ± 0.247	0.188 ± 0.244
2	Spline ANN (z=4)	0.179 ± 0.248	0.120 ± 0.238	0.138 ± 0.200	0.117 ± 0.199
2	Spline ANN (z=8)	0.091 ± 0.184	0.060 ± 0.172	0.085 ± 0.160	0.063 ± 0.158
2	Spline ANN (z=10)	0.074 ± 0.167	0.049 ± 0.156	0.074 ± 0.150	0.051 ± 0.146
2	Lookup Table (z=1)	0.525 ± 0.189	0.937 ± 0.664	0.496 ± 0.288	0.551 ± 0.391
2	Lookup Table (z=2)	0.130 ± 0.245	0.231 ± 0.519	0.158 ± 0.318	0.197 ± 0.442
2	Lookup Table (z=4)	0.032 ± 0.133	0.056 ± 0.274	0.041 ± 0.181	0.054 ± 0.263
2	Lookup Table (z=8)	0.008 ± 0.069	0.014 ± 0.140	0.011 ± 0.095	0.014 ± 0.139
2	Lookup Table (z=10)	0.005 ± 0.053	0.009 ± 0.111	0.006 ± 0.073	0.009 ± 0.110
3	Linear Model	0.273 ± 0.324	0.361 ± 0.418	0.223 ± 0.285	0.227 ± 0.292
3	Wide ReLU ANN	0.344 ± 0.475	0.476 ± 0.431	0.022 ± 0.060	0.137 ± 0.219
3	Deep ReLU ANN	0.539 ± 0.436	0.679 ± 0.478	0.005 ± 0.029	0.043 ± 0.093
3	ABEL-Spline (z=1)	0.530 ± 0.425	0.308 ± 0.285	0.104 ± 0.148	0.134 ± 0.160
3	ABEL-Spline (z=2)	0.301 ± 0.312	0.164 ± 0.206	0.076 ± 0.106	0.103 ± 0.122
3	ABEL-Spline (z=4)	0.162 ± 0.222	0.086 ± 0.145	0.062 ± 0.088	0.075 ± 0.096
3	ABEL-Spline (z=8)	0.083 ± 0.159	0.043 ± 0.103	0.045 ± 0.077	0.048 ± 0.079
3	ABEL-Spline (z=10)	0.068 ± 0.143	0.036 ± 0.093	0.041 ± 0.074	0.041 ± 0.074
3	Spline ANN (z=1)	0.516 ± 0.393	0.317 ± 0.291	0.174 ± 0.210	0.166 ± 0.195
3	Spline ANN (z=2)	0.303 ± 0.292	0.173 ± 0.211	0.136 ± 0.164	0.128 ± 0.147
3	Spline ANN (z=4)	0.165 ± 0.215	0.092 ± 0.152	0.100 ± 0.128	0.088 ± 0.117
3	Spline ANN (z=8)	0.084 ± 0.155	0.046 ± 0.109	0.066 ± 0.104	0.051 ± 0.094
3	Spline ANN (z=10)	0.067 ± 0.138	0.037 ± 0.097	0.056 ± 0.096	0.041 ± 0.087
3	Lookup Table (z=1)	0.454 ± 0.237	0.714 ± 0.508	0.403 ± 0.273	0.420 ± 0.299
3	Lookup Table (z=2)	0.058 ± 0.173	0.090 ± 0.295	0.071 ± 0.223	0.083 ± 0.272
3	Lookup Table (z=4)	0.007 ± 0.063	0.011 ± 0.107	0.009 ± 0.084	0.011 ± 0.106
3	Lookup Table (z=8)	0.001 ± 0.023	0.001 ± 0.039	0.001 ± 0.031	0.001 ± 0.039
3	Lookup Table (z=10)	0.000 ± 0.016	0.001 ± 0.028	0.001 ± 0.022	0.001 ± 0.028
4	Linear Model	0.414 ± 0.426	0.539 ± 0.535	0.372 ± 0.414	0.373 ± 0.421
4	Wide ReLU ANN	0.316 ± 0.433	0.451 ± 0.443	0.023 ± 0.066	0.125 ± 0.230
4	Deep ReLU ANN	0.694 ± 0.577	0.853 ± 0.562	0.006 ± 0.025	0.042 ± 0.095
4	ABEL-Spline (z=1)	0.671 ± 0.452	0.397 ± 0.315	0.102 ± 0.140	0.140 ± 0.153
4	ABEL-Spline (z=2)	0.394 ± 0.352	0.216 ± 0.227	0.073 ± 0.097	0.111 ± 0.117
4	ABEL-Spline (z=4)	0.209 ± 0.237	0.110 ± 0.151	0.065 ± 0.076	0.086 ± 0.088
4	ABEL-Spline (z=8)	0.109 ± 0.170	0.056 ± 0.109	0.052 ± 0.072	0.059 ± 0.075
4	ABEL-Spline (z=10)	0.086 ± 0.150	0.044 ± 0.096	0.047 ± 0.068	0.050 ± 0.070
4	Spline ANN (z=1)	0.659 ± 0.428	0.408 ± 0.326	0.203 ± 0.235	0.193 ± 0.215
4	Spline ANN (z=2)	0.388 ± 0.332	0.222 ± 0.233	0.153 ± 0.179	0.143 ± 0.156
4	Spline ANN (z=4)	0.210 ± 0.237	0.116 ± 0.163	0.114 ± 0.130	0.104 ± 0.115
4	Spline ANN (z=8)	0.111 ± 0.169	0.060 ± 0.116	0.082 ± 0.104	0.066 ± 0.094
4	Spline ANN (z=10)	0.087 ± 0.150	0.047 ± 0.103	0.070 ± 0.097	0.053 ± 0.087
4	Lookup Table (z=1)	0.537 ± 0.183	0.907 ± 0.552	0.495 ± 0.267	0.534 ± 0.325
4	Lookup Table (z=2)	0.034 ± 0.139	0.058 ± 0.264	0.044 ± 0.187	0.056 ± 0.253
4	Lookup Table (z=4)	0.002 ± 0.036	0.004 ± 0.068	0.003 ± 0.049	0.004 ± 0.068
4	Lookup Table (z=8)	0.000 ± 0.009	0.000 ± 0.016	0.000 ± 0.012	0.000 ± 0.016
4	Lookup Table (z=10)	0.000 ± 0.007	0.000 ± 0.014	0.000 ± 0.010	0.000 ± 0.014
5	Linear Model	0.369 ± 0.399	0.441 ± 0.474	0.320 ± 0.379	0.320 ± 0.383
5	Wide ReLU ANN	0.292 ± 0.360	0.408 ± 0.365	0.022 ± 0.046	0.106 ± 0.185
5	Deep ReLU ANN	0.618 ± 0.560	0.821 ± 0.675	0.006 ± 0.015	0.033 ± 0.067
5	ABEL-Spline (z=1)	0.611 ± 0.540	0.352 ± 0.334	0.072 ± 0.112	0.106 ± 0.130

Continued on next page

Table C.1 – Continued from previous page

n	Model	NPR with ADAM	NPR with SGD	PR with ADAM	PR with SGD
5	ABEL-Spline (z=2)	0.349 ± 0.353	0.189 ± 0.219	0.050 ± 0.075	0.084 ± 0.103
5	ABEL-Spline (z=4)	0.178 ± 0.224	0.093 ± 0.139	0.045 ± 0.060	0.064 ± 0.076
5	ABEL-Spline (z=8)	0.099 ± 0.161	0.051 ± 0.101	0.045 ± 0.062	0.051 ± 0.066
5	ABEL-Spline (z=10)	0.078 ± 0.141	0.040 ± 0.088	0.040 ± 0.059	0.044 ± 0.062
5	Spline ANN (z=1)	0.619 ± 0.522	0.373 ± 0.352	0.169 ± 0.229	0.158 ± 0.202
5	Spline ANN (z=2)	0.348 ± 0.344	0.197 ± 0.229	0.121 ± 0.169	0.113 ± 0.149
5	Spline ANN (z=4)	0.189 ± 0.238	0.104 ± 0.159	0.094 ± 0.129	0.086 ± 0.111
5	Spline ANN (z=8)	0.098 ± 0.160	0.053 ± 0.107	0.068 ± 0.097	0.056 ± 0.083
5	Spline ANN (z=10)	0.079 ± 0.143	0.042 ± 0.095	0.060 ± 0.089	0.047 ± 0.077
5	Lookup Table (z=1)	0.459 ± 0.243	0.820 ± 0.682	0.434 ± 0.322	0.483 ± 0.401
5	Lookup Table (z=2)	0.015 ± 0.092	0.026 ± 0.188	0.019 ± 0.126	0.026 ± 0.183
5	Lookup Table (z=4)	0.000 ± 0.015	0.001 ± 0.030	0.001 ± 0.020	0.001 ± 0.030
5	Lookup Table (z=8)	0.000 ± 0.001	0.000 ± 0.001	0.000 ± 0.001	0.000 ± 0.001
5	Lookup Table (z=10)	0.000 ± 0.002	0.000 ± 0.004	0.000 ± 0.003	0.000 ± 0.004
6	Linear Model	0.267 ± 0.317	0.294 ± 0.332	0.218 ± 0.272	0.217 ± 0.269
6	Wide ReLU ANN	0.253 ± 0.276	0.353 ± 0.270	0.018 ± 0.035	0.084 ± 0.132
6	Deep ReLU ANN	0.452 ± 0.366	0.542 ± 0.416	0.006 ± 0.012	0.036 ± 0.059
6	ABEL-Spline (z=1)	0.416 ± 0.353	0.243 ± 0.220	0.042 ± 0.061	0.076 ± 0.092
6	ABEL-Spline (z=2)	0.242 ± 0.237	0.133 ± 0.147	0.030 ± 0.044	0.058 ± 0.073
6	ABEL-Spline (z=4)	0.133 ± 0.151	0.072 ± 0.097	0.033 ± 0.040	0.048 ± 0.055
6	ABEL-Spline (z=8)	0.069 ± 0.103	0.036 ± 0.065	0.030 ± 0.038	0.035 ± 0.042
6	ABEL-Spline (z=10)	0.056 ± 0.091	0.029 ± 0.058	0.028 ± 0.037	0.031 ± 0.040
6	Spline ANN (z=1)	0.428 ± 0.351	0.253 ± 0.226	0.098 ± 0.131	0.099 ± 0.124
6	Spline ANN (z=2)	0.254 ± 0.246	0.141 ± 0.157	0.075 ± 0.102	0.075 ± 0.099
6	Spline ANN (z=4)	0.135 ± 0.156	0.074 ± 0.101	0.058 ± 0.076	0.056 ± 0.068
6	Spline ANN (z=8)	0.068 ± 0.106	0.036 ± 0.068	0.042 ± 0.059	0.037 ± 0.051
6	Spline ANN (z=10)	0.057 ± 0.093	0.030 ± 0.061	0.038 ± 0.053	0.033 ± 0.047
6	Lookup Table (z=1)	0.395 ± 0.234	0.572 ± 0.466	0.326 ± 0.252	0.336 ± 0.273
6	Lookup Table (z=2)	0.006 ± 0.058	0.009 ± 0.094	0.008 ± 0.076	0.009 ± 0.093
6	Lookup Table (z=4)	0.000 ± 0.007	0.000 ± 0.012	0.000 ± 0.009	0.000 ± 0.012
6	Lookup Table (z=8)	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
6	Lookup Table (z=10)	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000

C.2 PMLB Regression Benchmark Data

The means and standard deviations (indicated with a ±-symbol) for the regression benchmarks averaged over the 10 folds of cross-validation discussed in Section 7.3 are presented in Table C.2. The table columns correspond to the dataset names, input dimension n, data sample size N, model name, The test R^2 or R-squared score, test mean squared error (T. MSE), test mean absolute error (T. MAE), and final training mean absolute error (Tr. MAE).

Table C.2: PMLB: Mean and standard deviation of test and training error.

Dataset	n	**N**	**Model**	R^2	**T. MSE**	**T. MAE**	**Tr. MAE**
192_vineyard	2	52	Linear Model	−2.028 ± 2.907	1.546 ± 3.441	0.791 ± 0.850	0.652 ± 0.167
192_vineyard	2	52	Wide ReLU ANN	−0.967 ± 0.996	0.716 ± 1.093	0.608 ± 0.402	0.405 ± 0.040
192_vineyard	2	52	Deep ReLU ANN	−1.203 ± 1.059	0.850 ± 1.347	0.657 ± 0.452	0.397 ± 0.051
192_vineyard	2	52	ABEL-Spline (z=1)	−0.768 ± 1.093	0.786 ± 1.504	0.604 ± 0.491	0.475 ± 0.035
192_vineyard	2	52	ABEL-Spline (z=2)	−0.913 ± 1.217	0.838 ± 1.595	0.616 ± 0.512	0.452 ± 0.028
192_vineyard	2	52	ABEL-Spline (z=4)	−1.257 ± 1.372	0.937 ± 1.742	0.660 ± 0.527	0.411 ± 0.029
192_vineyard	2	52	ABEL-Spline (z=8)	−1.703 ± 1.915	1.112 ± 2.139	0.704 ± 0.597	0.379 ± 0.029
192_vineyard	2	52	ABEL-Spline (z=10)	−1.702 ± 1.969	1.197 ± 2.406	0.710 ± 0.622	0.375 ± 0.029
192_vineyard	2	52	Spline ANN (z=1)	−1.905 ± 2.388	1.298 ± 2.643	0.772 ± 0.697	0.631 ± 0.025
192_vineyard	2	52	Spline ANN (z=2)	−1.732 ± 2.296	1.254 ± 2.598	0.743 ± 0.694	0.607 ± 0.030
192_vineyard	2	52	Spline ANN (z=4)	−1.902 ± 2.524	1.335 ± 2.782	0.769 ± 0.720	0.605 ± 0.029
192_vineyard	2	52	Spline ANN (z=8)	−2.204 ± 2.796	1.423 ± 2.925	0.797 ± 0.737	0.609 ± 0.024
192_vineyard	2	52	Spline ANN (z=10)	−2.182 ± 2.791	1.439 ± 2.992	0.801 ± 0.742	0.608 ± 0.023
192_vineyard	2	52	Lookup Table (z=1)	−3.298 ± 3.477	1.735 ± 3.344	0.933 ± 0.783	0.771 ± 0.019
192_vineyard	2	52	Lookup Table (z=2)	−2.302 ± 2.949	1.415 ± 2.860	0.810 ± 0.730	0.674 ± 0.017
192_vineyard	2	52	Lookup Table (z=4)	−2.454 ± 3.142	1.457 ± 2.922	0.829 ± 0.740	0.672 ± 0.023
192_vineyard	2	52	Lookup Table (z=8)	−2.894 ± 3.730	1.564 ± 3.099	0.860 ± 0.762	0.660 ± 0.025
192_vineyard	2	52	Lookup Table (z=10)	−2.971 ± 3.471	1.599 ± 3.118	0.880 ± 0.762	0.664 ± 0.020
228_elusage	2	55	Linear Model	−3.922 ± 9.989	1.116 ± 0.786	0.858 ± 0.297	0.833 ± 0.252
228_elusage	2	55	Wide ReLU ANN	0.339 ± 0.433	0.294 ± 0.190	0.424 ± 0.160	0.318 ± 0.017
228_elusage	2	55	Deep ReLU ANN	0.378 ± 0.398	0.282 ± 0.173	0.389 ± 0.129	0.299 ± 0.015
228_elusage	2	55	ABEL-Spline (z=1)	0.271 ± 0.499	0.387 ± 0.390	0.448 ± 0.184	0.380 ± 0.012
228_elusage	2	55	ABEL-Spline (z=2)	0.345 ± 0.379	0.357 ± 0.323	0.434 ± 0.179	0.357 ± 0.012
228_elusage	2	55	ABEL-Spline (z=4)	0.296 ± 0.425	0.378 ± 0.329	0.449 ± 0.189	0.319 ± 0.014
228_elusage	2	55	ABEL-Spline (z=8)	0.332 ± 0.389	0.356 ± 0.304	0.424 ± 0.153	0.275 ± 0.013
228_elusage	2	55	ABEL-Spline (z=10)	0.348 ± 0.358	0.367 ± 0.323	0.423 ± 0.156	0.266 ± 0.014
228_elusage	2	55	Spline ANN (z=1)	−1.199 ± 3.774	0.790 ± 0.745	0.691 ± 0.273	0.648 ± 0.019
228_elusage	2	55	Spline ANN (z=2)	−0.752 ± 2.800	0.712 ± 0.717	0.641 ± 0.261	0.598 ± 0.014
228_elusage	2	55	Spline ANN (z=4)	−0.770 ± 2.804	0.702 ± 0.681	0.641 ± 0.252	0.581 ± 0.014
228_elusage	2	55	Spline ANN (z=8)	−0.838 ± 3.087	0.695 ± 0.680	0.633 ± 0.255	0.576 ± 0.010
228_elusage	2	55	Spline ANN (z=10)	−0.836 ± 3.012	0.702 ± 0.684	0.637 ± 0.248	0.574 ± 0.015
228_elusage	2	55	Lookup Table (z=1)	−2.300 ± 5.478	1.165 ± 0.999	0.869 ± 0.304	0.822 ± 0.011
228_elusage	2	55	Lookup Table (z=2)	−1.661 ± 4.626	0.894 ± 0.789	0.740 ± 0.267	0.693 ± 0.017
228_elusage	2	55	Lookup Table (z=4)	−1.583 ± 4.388	0.900 ± 0.791	0.745 ± 0.267	0.693 ± 0.014
228_elusage	2	55	Lookup Table (z=8)	−1.675 ± 4.668	0.907 ± 0.837	0.744 ± 0.276	0.691 ± 0.013
228_elusage	2	55	Lookup Table (z=10)	−1.787 ± 4.747	0.932 ± 0.776	0.764 ± 0.263	0.701 ± 0.009
523_analcatdata_neavote	2	100	Linear Model	0.027 ± 0.651	0.927 ± 0.628	0.849 ± 0.375	0.839 ± 0.337
523_analcatdata_neavote	2	100	Wide ReLU ANN	0.757 ± 0.229	0.232 ± 0.215	0.324 ± 0.225	0.282 ± 0.173
523_analcatdata_neavote	2	100	Deep ReLU ANN	0.920 ± 0.078	0.077 ± 0.085	0.141 ± 0.059	0.126 ± 0.006
523_analcatdata_neavote	2	100	ABEL-Spline (z=1)	0.929 ± 0.037	0.067 ± 0.037	0.160 ± 0.041	0.151 ± 0.018
523_analcatdata_neavote	2	100	ABEL-Spline (z=2)	0.941 ± 0.038	0.054 ± 0.033	0.128 ± 0.048	0.128 ± 0.006
523_analcatdata_neavote	2	100	ABEL-Spline (z=4)	0.934 ± 0.048	0.062 ± 0.048	0.131 ± 0.051	0.124 ± 0.007
523_analcatdata_neavote	2	100	ABEL-Spline (z=8)	0.937 ± 0.043	0.059 ± 0.042	0.129 ± 0.050	0.123 ± 0.007
523_analcatdata_neavote	2	100	ABEL-Spline (z=10)	0.934 ± 0.048	0.062 ± 0.048	0.131 ± 0.051	0.124 ± 0.006

Continued on next page

Table C.2 – Continued from previous page

Dataset	n	**N**	**Model**	R^2	**T. MSE**	**T. MAE**	**Tr. MAE**
523_analcatdata_neavote	2	100	Spline ANN (z=1)	0.343 ± 0.086	0.619 ± 0.084	0.744 ± 0.067	0.729 ± 0.013
523_analcatdata_neavote	2	100	Spline ANN (z=2)	0.423 ± 0.069	0.545 ± 0.080	0.700 ± 0.068	0.687 ± 0.013
523_analcatdata_neavote	2	100	Spline ANN (z=4)	0.421 ± 0.063	0.547 ± 0.082	0.701 ± 0.069	0.684 ± 0.008
523_analcatdata_neavote	2	100	Spline ANN (z=8)	0.431 ± 0.049	0.537 ± 0.064	0.697 ± 0.059	0.681 ± 0.011
523_analcatdata_neavote	2	100	Spline ANN (z=10)	0.414 ± 0.071	0.555 ± 0.092	0.707 ± 0.075	0.688 ± 0.016
523_analcatdata_neavote	2	100	Lookup Table (z=1)	−0.114 ± 0.114	1.051 ± 0.124	0.972 ± 0.079	0.958 ± 0.008
523_analcatdata_neavote	2	100	Lookup Table (z=2)	0.409 ± 0.062	0.559 ± 0.078	0.714 ± 0.060	0.706 ± 0.020
523_analcatdata_neavote	2	100	Lookup Table (z=4)	0.433 ± 0.046	0.535 ± 0.060	0.700 ± 0.055	0.693 ± 0.015
523_analcatdata_neavote	2	100	Lookup Table (z=8)	0.413 ± 0.042	0.555 ± 0.070	0.712 ± 0.060	0.705 ± 0.018
523_analcatdata_neavote	2	100	Lookup Table (z=10)	0.426 ± 0.048	0.541 ± 0.054	0.703 ± 0.054	0.692 ± 0.013
663_rabe_266	2	120	Linear Model	0.001 ± 0.664	0.934 ± 0.709	0.774 ± 0.288	0.748 ± 0.311
663_rabe_266	2	120	Wide ReLU ANN	0.991 ± 0.019	0.005 ± 0.009	0.045 ± 0.035	0.026 ± 0.003
663_rabe_266	2	120	Deep ReLU ANN	0.979 ± 0.050	0.012 ± 0.023	0.071 ± 0.070	0.039 ± 0.012
663_rabe_266	2	120	ABEL-Spline (z=1)	0.945 ± 0.078	0.038 ± 0.033	0.138 ± 0.052	0.123 ± 0.004
663_rabe_266	2	120	ABEL-Spline (z=2)	0.964 ± 0.067	0.023 ± 0.030	0.098 ± 0.057	0.081 ± 0.006
663_rabe_266	2	120	ABEL-Spline (z=4)	0.955 ± 0.085	0.028 ± 0.038	0.120 ± 0.067	0.068 ± 0.009
663_rabe_266	2	120	ABEL-Spline (z=8)	0.926 ± 0.151	0.044 ± 0.068	0.150 ± 0.099	0.068 ± 0.007
663_rabe_266	2	120	ABEL-Spline (z=10)	0.819 ± 0.338	0.114 ± 0.152	0.266 ± 0.165	0.071 ± 0.007
663_rabe_266	2	120	Spline ANN (z=1)	0.393 ± 0.180	0.524 ± 0.108	0.604 ± 0.069	0.592 ± 0.013
663_rabe_266	2	120	Spline ANN (z=2)	0.464 ± 0.150	0.466 ± 0.108	0.548 ± 0.071	0.531 ± 0.010
663_rabe_266	2	120	Spline ANN (z=4)	0.462 ± 0.133	0.472 ± 0.120	0.548 ± 0.077	0.533 ± 0.008
663_rabe_266	2	120	Spline ANN (z=8)	0.413 ± 0.156	0.512 ± 0.121	0.580 ± 0.072	0.554 ± 0.003
663_rabe_266	2	120	Spline ANN (z=10)	0.346 ± 0.262	0.554 ± 0.124	0.611 ± 0.076	0.558 ± 0.004
663_rabe_266	2	120	Lookup Table (z=1)	−0.206 ± 0.341	1.035 ± 0.191	0.862 ± 0.078	0.851 ± 0.003
663_rabe_266	2	120	Lookup Table (z=2)	0.311 ± 0.299	0.575 ± 0.113	0.622 ± 0.069	0.609 ± 0.008
663_rabe_266	2	120	Lookup Table (z=4)	0.250 ± 0.277	0.634 ± 0.124	0.652 ± 0.067	0.620 ± 0.006
663_rabe_266	2	120	Lookup Table (z=8)	0.136 ± 0.255	0.744 ± 0.168	0.711 ± 0.090	0.629 ± 0.004
663_rabe_266	2	120	Lookup Table (z=10)	0.014 ± 0.297	0.845 ± 0.173	0.768 ± 0.090	0.641 ± 0.006
712_chscase_geyser1	2	222	Linear Model	0.532 ± 0.230	0.453 ± 0.207	0.531 ± 0.132	0.531 ± 0.116
712_chscase_geyser1	2	222	Wide ReLU ANN	0.759 ± 0.114	0.223 ± 0.070	0.379 ± 0.071	0.362 ± 0.009
712_chscase_geyser1	2	222	Deep ReLU ANN	0.751 ± 0.113	0.233 ± 0.083	0.380 ± 0.075	0.357 ± 0.010
712_chscase_geyser1	2	222	ABEL-Spline (z=1)	0.745 ± 0.133	0.234 ± 0.081	0.387 ± 0.076	0.370 ± 0.009
712_chscase_geyser1	2	222	ABEL-Spline (z=2)	0.742 ± 0.130	0.237 ± 0.073	0.392 ± 0.074	0.365 ± 0.009
712_chscase_geyser1	2	222	ABEL-Spline (z=4)	0.724 ± 0.145	0.252 ± 0.082	0.401 ± 0.075	0.352 ± 0.008
712_chscase_geyser1	2	222	ABEL-Spline (z=8)	0.686 ± 0.145	0.289 ± 0.074	0.437 ± 0.069	0.334 ± 0.010
712_chscase_geyser1	2	222	ABEL-Spline (z=10)	0.665 ± 0.173	0.306 ± 0.095	0.449 ± 0.072	0.327 ± 0.010
712_chscase_geyser1	2	222	Spline ANN (z=1)	0.562 ± 0.080	0.434 ± 0.125	0.533 ± 0.088	0.530 ± 0.004
712_chscase_geyser1	2	222	Spline ANN (z=2)	0.586 ± 0.085	0.409 ± 0.120	0.521 ± 0.086	0.510 ± 0.006
712_chscase_geyser1	2	222	Spline ANN (z=4)	0.563 ± 0.108	0.430 ± 0.128	0.534 ± 0.086	0.512 ± 0.006
712_chscase_geyser1	2	222	Spline ANN (z=8)	0.538 ± 0.115	0.454 ± 0.132	0.551 ± 0.087	0.515 ± 0.007
712_chscase_geyser1	2	222	Spline ANN (z=10)	0.520 ± 0.129	0.469 ± 0.129	0.561 ± 0.085	0.519 ± 0.007
712_chscase_geyser1	2	222	Lookup Table (z=1)	−0.081 ± 0.039	1.080 ± 0.282	0.813 ± 0.132	0.809 ± 0.005
712_chscase_geyser1	2	222	Lookup Table (z=2)	0.518 ± 0.127	0.478 ± 0.174	0.558 ± 0.118	0.553 ± 0.009
712_chscase_geyser1	2	222	Lookup Table (z=4)	0.489 ± 0.071	0.507 ± 0.142	0.583 ± 0.094	0.545 ± 0.009
712_chscase_geyser1	2	222	Lookup Table (z=8)	0.487 ± 0.058	0.513 ± 0.150	0.585 ± 0.092	0.543 ± 0.006
712_chscase_geyser1	2	222	Lookup Table (z=10)	0.433 ± 0.077	0.565 ± 0.166	0.613 ± 0.113	0.552 ± 0.008
519_vinnie	2	380	Linear Model	0.578 ± 0.123	0.387 ± 0.157	0.504 ± 0.108	0.477 ± 0.084
519_vinnie	2	380	Wide ReLU ANN	0.684 ± 0.106	0.271 ± 0.057	0.416 ± 0.046	0.387 ± 0.007
519_vinnie	2	380	Deep ReLU ANN	0.675 ± 0.110	0.280 ± 0.066	0.420 ± 0.050	0.388 ± 0.009
519_vinnie	2	380	ABEL-Spline (z=1)	0.696 ± 0.094	0.261 ± 0.052	0.406 ± 0.043	0.390 ± 0.006
519_vinnie	2	380	ABEL-Spline (z=2)	0.694 ± 0.091	0.265 ± 0.054	0.408 ± 0.043	0.388 ± 0.006
519_vinnie	2	380	ABEL-Spline (z=4)	0.686 ± 0.088	0.273 ± 0.054	0.413 ± 0.046	0.382 ± 0.006
519_vinnie	2	380	ABEL-Spline (z=8)	0.674 ± 0.094	0.283 ± 0.062	0.421 ± 0.050	0.379 ± 0.006
519_vinnie	2	380	ABEL-Spline (z=10)	0.671 ± 0.095	0.286 ± 0.065	0.423 ± 0.051	0.378 ± 0.006
519_vinnie	2	380	Spline ANN (z=1)	0.592 ± 0.070	0.367 ± 0.093	0.485 ± 0.060	0.474 ± 0.007
519_vinnie	2	380	Spline ANN (z=2)	0.594 ± 0.076	0.360 ± 0.072	0.486 ± 0.051	0.463 ± 0.006
519_vinnie	2	380	Spline ANN (z=4)	0.586 ± 0.073	0.370 ± 0.087	0.492 ± 0.060	0.465 ± 0.005
519_vinnie	2	380	Spline ANN (z=8)	0.552 ± 0.070	0.403 ± 0.096	0.512 ± 0.063	0.480 ± 0.006
519_vinnie	2	380	Spline ANN (z=10)	0.537 ± 0.065	0.417 ± 0.098	0.521 ± 0.063	0.487 ± 0.006
519_vinnie	2	380	Lookup Table (z=1)	−0.130 ± 0.137	1.017 ± 0.214	0.831 ± 0.116	0.824 ± 0.006
519_vinnie	2	380	Lookup Table (z=2)	0.392 ± 0.090	0.543 ± 0.110	0.588 ± 0.049	0.571 ± 0.007
519_vinnie	2	380	Lookup Table (z=4)	0.449 ± 0.065	0.503 ± 0.135	0.583 ± 0.078	0.524 ± 0.009

Continued on next page

Table C.2 – Continued from previous page

Dataset	n	N	Model	R^2	T. MSE	T. MAE	Tr. MAE
519_vinnie	2	380	Lookup Table (z=8)	0.428 ± 0.049	0.528 ± 0.157	0.583 ± 0.089	0.512 ± 0.009
519_vinnie	2	380	Lookup Table (z=10)	0.415 ± 0.081	0.546 ± 0.197	0.587 ± 0.102	0.513 ± 0.005
banana	2	5300	Linear Model	−0.815 ± 0.106	1.809 ± 0.126	0.902 ± 0.062	0.902 ± 0.006
banana	2	5300	Wide ReLU ANN	0.673 ± 0.031	0.326 ± 0.033	0.312 ± 0.025	0.305 ± 0.004
banana	2	5300	Deep ReLU ANN	0.603 ± 0.051	0.395 ± 0.050	0.208 ± 0.025	0.211 ± 0.007
banana	2	5300	ABEL-Spline (z=1)	0.561 ± 0.025	0.438 ± 0.027	0.434 ± 0.017	0.436 ± 0.012
banana	2	5300	ABEL-Spline (z=2)	0.665 ± 0.011	0.334 ± 0.014	0.342 ± 0.012	0.340 ± 0.002
banana	2	5300	ABEL-Spline (z=4)	0.659 ± 0.024	0.340 ± 0.026	0.289 ± 0.017	0.284 ± 0.003
banana	2	5300	ABEL-Spline (z=8)	0.634 ± 0.027	0.365 ± 0.029	0.277 ± 0.018	0.270 ± 0.002
banana	2	5300	ABEL-Spline (z=10)	0.617 ± 0.029	0.382 ± 0.031	0.282 ± 0.019	0.273 ± 0.003
banana	2	5300	Spline ANN (z=1)	0.041 ± 0.026	0.955 ± 0.033	0.710 ± 0.016	0.706 ± 0.001
banana	2	5300	Spline ANN (z=2)	0.047 ± 0.047	0.950 ± 0.050	0.642 ± 0.022	0.641 ± 0.002
banana	2	5300	Spline ANN (z=4)	0.002 ± 0.055	0.995 ± 0.055	0.612 ± 0.024	0.606 ± 0.003
banana	2	5300	Spline ANN (z=8)	−0.053 ± 0.068	1.049 ± 0.068	0.576 ± 0.027	0.569 ± 0.004
banana	2	5300	Spline ANN (z=10)	−0.068 ± 0.069	1.064 ± 0.069	0.574 ± 0.027	0.567 ± 0.003
banana	2	5300	Lookup Table (z=1)	−0.817 ± 0.106	1.811 ± 0.127	0.902 ± 0.062	0.902 ± 0.006
banana	2	5300	Lookup Table (z=2)	−0.538 ± 0.149	1.533 ± 0.159	0.884 ± 0.047	0.867 ± 0.005
banana	2	5300	Lookup Table (z=4)	0.022 ± 0.090	0.974 ± 0.087	0.498 ± 0.030	0.493 ± 0.008
banana	2	5300	Lookup Table (z=8)	0.428 ± 0.057	0.570 ± 0.062	0.288 ± 0.031	0.286 ± 0.003
banana	2	5300	Lookup Table (z=10)	0.492 ± 0.047	0.506 ± 0.047	0.257 ± 0.023	0.255 ± 0.004
678_visualizing_environmental	3	111	Linear Model	−0.136 ± 0.313	0.924 ± 0.387	0.755 ± 0.146	0.734 ± 0.091
678_visualizing_environmental	3	111	Wide ReLU ANN	0.061 ± 0.387	0.729 ± 0.320	0.671 ± 0.134	0.589 ± 0.015
678_visualizing_environmental	3	111	Deep ReLU ANN	−0.007 ± 0.267	0.827 ± 0.362	0.731 ± 0.160	0.563 ± 0.037
678_visualizing_environmental	3	111	ABEL-Spline (z=1)	0.109 ± 0.239	0.713 ± 0.300	0.658 ± 0.114	0.614 ± 0.013
678_visualizing_environmental	3	111	ABEL-Spline (z=2)	0.071 ± 0.241	0.748 ± 0.329	0.679 ± 0.133	0.598 ± 0.012
678_visualizing_environmental	3	111	ABEL-Spline (z=4)	0.060 ± 0.261	0.746 ± 0.321	0.671 ± 0.110	0.556 ± 0.016
678_visualizing_environmental	3	111	ABEL-Spline (z=8)	0.003 ± 0.272	0.794 ± 0.350	0.702 ± 0.126	0.471 ± 0.018
678_visualizing_environmental	3	111	ABEL-Spline (z=10)	−0.038 ± 0.346	0.833 ± 0.455	0.705 ± 0.142	0.430 ± 0.017
678_visualizing_environmental	3	111	Spline ANN (z=1)	0.065 ± 0.169	0.767 ± 0.309	0.677 ± 0.127	0.659 ± 0.008
678_visualizing_environmental	3	111	Spline ANN (z=2)	0.029 ± 0.181	0.795 ± 0.324	0.691 ± 0.135	0.653 ± 0.011
678_visualizing_environmental	3	111	Spline ANN (z=4)	−0.027 ± 0.169	0.848 ± 0.355	0.715 ± 0.146	0.646 ± 0.012
678_visualizing_environmental	3	111	Spline ANN (z=8)	−0.071 ± 0.179	0.882 ± 0.367	0.734 ± 0.141	0.629 ± 0.009
678_visualizing_environmental	3	111	Spline ANN (z=10)	−0.064 ± 0.166	0.879 ± 0.363	0.731 ± 0.142	0.620 ± 0.008
678_visualizing_environmental	3	111	Lookup Table (z=1)	−0.275 ± 0.246	1.065 ± 0.474	0.803 ± 0.191	0.781 ± 0.006
678_visualizing_environmental	3	111	Lookup Table (z=2)	−0.088 ± 0.198	0.907 ± 0.392	0.738 ± 0.163	0.705 ± 0.010
678_visualizing_environmental	3	111	Lookup Table (z=4)	−0.152 ± 0.237	0.954 ± 0.411	0.767 ± 0.174	0.701 ± 0.008
678_visualizing_environmental	3	111	Lookup Table (z=8)	−0.254 ± 0.325	1.029 ± 0.455	0.789 ± 0.184	0.679 ± 0.009
678_visualizing_environmental	3	111	Lookup Table (z=10)	−0.280 ± 0.248	1.060 ± 0.450	0.802 ± 0.192	0.666 ± 0.005
556_analcatdata_apnea2	3	475	Linear Model	−0.077 ± 0.015	1.094 ± 0.503	0.271 ± 0.065	0.270 ± 0.003
556_analcatdata_apnea2	3	475	Wide ReLU ANN	−0.051 ± 0.014	1.072 ± 0.498	0.261 ± 0.062	0.258 ± 0.002
556_analcatdata_apnea2	3	475	Deep ReLU ANN	0.474 ± 0.418	0.474 ± 0.413	0.177 ± 0.074	0.163 ± 0.070
556_analcatdata_apnea2	3	475	ABEL-Spline (z=1)	−0.052 ± 0.012	1.072 ± 0.495	0.270 ± 0.064	0.268 ± 0.003
556_analcatdata_apnea2	3	475	ABEL-Spline (z=2)	−0.051 ± 0.009	1.071 ± 0.496	0.270 ± 0.063	0.267 ± 0.002
556_analcatdata_apnea2	3	475	ABEL-Spline (z=4)	−0.050 ± 0.012	1.068 ± 0.491	0.272 ± 0.058	0.266 ± 0.003
556_analcatdata_apnea2	3	475	ABEL-Spline (z=8)	−0.045 ± 0.020	1.069 ± 0.502	0.277 ± 0.063	0.266 ± 0.003
556_analcatdata_apnea2	3	475	ABEL-Spline (z=10)	−0.030 ± 0.026	1.052 ± 0.490	0.282 ± 0.057	0.264 ± 0.002
556_analcatdata_apnea2	3	475	Spline ANN (z=1)	−0.059 ± 0.012	1.078 ± 0.498	0.269 ± 0.064	0.267 ± 0.003
556_analcatdata_apnea2	3	475	Spline ANN (z=2)	−0.052 ± 0.010	1.072 ± 0.497	0.269 ± 0.064	0.267 ± 0.003
556_analcatdata_apnea2	3	475	Spline ANN (z=4)	−0.053 ± 0.007	1.072 ± 0.496	0.270 ± 0.062	0.267 ± 0.003
556_analcatdata_apnea2	3	475	Spline ANN (z=8)	−0.049 ± 0.011	1.069 ± 0.496	0.275 ± 0.059	0.267 ± 0.003
556_analcatdata_apnea2	3	475	Spline ANN (z=10)	−0.048 ± 0.011	1.069 ± 0.495	0.278 ± 0.057	0.267 ± 0.003
556_analcatdata_apnea2	3	475	Lookup Table (z=1)	−0.078 ± 0.015	1.095 ± 0.503	0.271 ± 0.065	0.270 ± 0.003
556_analcatdata_apnea2	3	475	Lookup Table (z=2)	−0.076 ± 0.017	1.093 ± 0.503	0.271 ± 0.064	0.270 ± 0.003
556_analcatdata_apnea2	3	475	Lookup Table (z=4)	0.028 ± 0.102	1.009 ± 0.496	0.264 ± 0.060	0.248 ± 0.015
556_analcatdata_apnea2	3	475	Lookup Table (z=8)	0.205 ± 0.098	0.832 ± 0.424	0.247 ± 0.048	0.204 ± 0.003
556_analcatdata_apnea2	3	475	Lookup Table (z=10)	0.210 ± 0.059	0.828 ± 0.427	0.235 ± 0.054	0.200 ± 0.004
557_analcatdata_apnea1	3	475	Linear Model	−0.075 ± 0.015	1.094 ± 0.519	0.265 ± 0.057	0.264 ± 0.003
557_analcatdata_apnea1	3	475	Wide ReLU ANN	−0.053 ± 0.006	1.074 ± 0.514	0.261 ± 0.056	0.257 ± 0.002
557_analcatdata_apnea1	3	475	Deep ReLU ANN	0.529 ± 0.462	0.381 ± 0.400	0.148 ± 0.075	0.158 ± 0.070
557_analcatdata_apnea1	3	475	ABEL-Spline (z=1)	−0.060 ± 0.014	1.080 ± 0.513	0.265 ± 0.057	0.263 ± 0.003
557_analcatdata_apnea1	3	475	ABEL-Spline (z=2)	−0.060 ± 0.015	1.080 ± 0.514	0.265 ± 0.057	0.263 ± 0.003
557_analcatdata_apnea1	3	475	ABEL-Spline (z=4)	−0.060 ± 0.018	1.078 ± 0.511	0.269 ± 0.055	0.262 ± 0.003

Continued on next page

Table C.2 – Continued from previous page

Dataset	n	N	Model	R^2	T. MSE	T. MAE	Tr. MAE
557_analcatdata_apnea1	3	475	ABEL-Spline (z=8)	−0.054 ± 0.016	1.077 ± 0.520	0.271 ± 0.063	0.263 ± 0.003
557_analcatdata_apnea1	3	475	ABEL-Spline (z=10)	−0.046 ± 0.016	1.067 ± 0.509	0.275 ± 0.056	0.261 ± 0.003
557_analcatdata_apnea1	3	475	Spline ANN (z=1)	−0.064 ± 0.015	1.083 ± 0.515	0.265 ± 0.057	0.263 ± 0.003
557_analcatdata_apnea1	3	475	Spline ANN (z=2)	−0.063 ± 0.014	1.082 ± 0.514	0.265 ± 0.057	0.263 ± 0.003
557_analcatdata_apnea1	3	475	Spline ANN (z=4)	−0.066 ± 0.014	1.085 ± 0.514	0.267 ± 0.056	0.263 ± 0.003
557_analcatdata_apnea1	3	475	Spline ANN (z=8)	−0.063 ± 0.011	1.084 ± 0.516	0.267 ± 0.055	0.263 ± 0.003
557_analcatdata_apnea1	3	475	Spline ANN (z=10)	−0.063 ± 0.015	1.082 ± 0.515	0.269 ± 0.059	0.263 ± 0.003
557_analcatdata_apnea1	3	475	Lookup Table (z=1)	−0.075 ± 0.014	1.094 ± 0.519	0.265 ± 0.057	0.264 ± 0.003
557_analcatdata_apnea1	3	475	Lookup Table (z=2)	−0.074 ± 0.015	1.093 ± 0.518	0.265 ± 0.057	0.264 ± 0.003
557_analcatdata_apnea1	3	475	Lookup Table (z=4)	0.026 ± 0.103	1.004 ± 0.500	0.255 ± 0.058	0.245 ± 0.015
557_analcatdata_apnea1	3	475	Lookup Table (z=8)	0.193 ± 0.068	0.836 ± 0.430	0.247 ± 0.057	0.202 ± 0.001
557_analcatdata_apnea1	3	475	Lookup Table (z=10)	0.206 ± 0.063	0.838 ± 0.456	0.239 ± 0.062	0.200 ± 0.002
titanic	3	2201	Linear Model	−0.052 ± 0.168	1.007 ± 0.120	0.472 ± 0.056	0.472 ± 0.007
titanic	3	2201	Wide ReLU ANN	0.061 ± 0.204	0.891 ± 0.118	0.484 ± 0.050	0.443 ± 0.008
titanic	3	2201	Deep ReLU ANN	0.007 ± 0.173	0.948 ± 0.100	0.456 ± 0.050	0.393 ± 0.014
titanic	3	2201	ABEL-Spline (z=1)	−0.027 ± 0.173	0.982 ± 0.118	0.476 ± 0.054	0.473 ± 0.007
titanic	3	2201	ABEL-Spline (z=2)	0.048 ± 0.161	0.912 ± 0.120	0.473 ± 0.056	0.460 ± 0.008
titanic	3	2201	ABEL-Spline (z=4)	0.062 ± 0.194	0.896 ± 0.139	0.462 ± 0.065	0.448 ± 0.007
titanic	3	2201	ABEL-Spline (z=8)	0.076 ± 0.190	0.881 ± 0.127	0.461 ± 0.064	0.445 ± 0.007
titanic	3	2201	ABEL-Spline (z=10)	0.074 ± 0.194	0.882 ± 0.129	0.467 ± 0.062	0.444 ± 0.006
titanic	3	2201	Spline ANN (z=1)	−0.049 ± 0.167	1.005 ± 0.120	0.472 ± 0.056	0.472 ± 0.007
titanic	3	2201	Spline ANN (z=2)	−0.032 ± 0.154	0.990 ± 0.120	0.471 ± 0.056	0.470 ± 0.007
titanic	3	2201	Spline ANN (z=4)	0.001 ± 0.138	0.962 ± 0.131	0.468 ± 0.058	0.463 ± 0.008
titanic	3	2201	Spline ANN (z=8)	0.004 ± 0.156	0.959 ± 0.142	0.463 ± 0.062	0.457 ± 0.008
titanic	3	2201	Spline ANN (z=10)	0.007 ± 0.156	0.956 ± 0.145	0.464 ± 0.062	0.457 ± 0.009
titanic	3	2201	Lookup Table (z=1)	−0.500 ± 0.173	1.485 ± 0.375	0.694 ± 0.171	0.693 ± 0.014
titanic	3	2201	Lookup Table (z=2)	−0.008 ± 0.180	0.960 ± 0.091	0.525 ± 0.058	0.461 ± 0.007
titanic	3	2201	Lookup Table (z=4)	−0.043 ± 0.172	0.998 ± 0.103	0.591 ± 0.093	0.431 ± 0.007
titanic	3	2201	Lookup Table (z=8)	−0.044 ± 0.079	1.014 ± 0.143	0.672 ± 0.163	0.401 ± 0.006
titanic	3	2201	Lookup Table (z=10)	−0.083 ± 0.074	1.053 ± 0.153	0.696 ± 0.167	0.411 ± 0.011
485_analcatdata_vehicle	4	48	Linear Model	−0.232 ± 0.658	0.982 ± 0.953	0.756 ± 0.312	0.811 ± 0.179
485_analcatdata_vehicle	4	48	Wide ReLU ANN	0.596 ± 0.429	0.366 ± 0.537	0.388 ± 0.323	0.251 ± 0.018
485_analcatdata_vehicle	4	48	Deep ReLU ANN	0.483 ± 0.684	0.404 ± 0.641	0.401 ± 0.374	0.194 ± 0.040
485_analcatdata_vehicle	4	48	ABEL-Spline (z=1)	0.450 ± 0.438	0.608 ± 0.856	0.489 ± 0.380	0.379 ± 0.016
485_analcatdata_vehicle	4	48	ABEL-Spline (z=2)	0.446 ± 0.392	0.591 ± 0.795	0.503 ± 0.363	0.356 ± 0.017
485_analcatdata_vehicle	4	48	ABEL-Spline (z=4)	0.382 ± 0.383	0.607 ± 0.718	0.538 ± 0.323	0.338 ± 0.013
485_analcatdata_vehicle	4	48	ABEL-Spline (z=8)	0.379 ± 0.406	0.610 ± 0.720	0.536 ± 0.330	0.337 ± 0.012
485_analcatdata_vehicle	4	48	ABEL-Spline (z=10)	0.374 ± 0.384	0.608 ± 0.704	0.541 ± 0.320	0.338 ± 0.014
485_analcatdata_vehicle	4	48	Spline ANN (z=1)	0.016 ± 0.568	0.995 ± 1.280	0.735 ± 0.377	0.653 ± 0.020
485_analcatdata_vehicle	4	48	Spline ANN (z=2)	0.097 ± 0.510	0.911 ± 1.157	0.700 ± 0.368	0.613 ± 0.015
485_analcatdata_vehicle	4	48	Spline ANN (z=4)	0.081 ± 0.545	0.939 ± 1.215	0.706 ± 0.381	0.608 ± 0.014
485_analcatdata_vehicle	4	48	Spline ANN (z=8)	0.088 ± 0.549	0.933 ± 1.227	0.705 ± 0.384	0.607 ± 0.018
485_analcatdata_vehicle	4	48	Spline ANN (z=10)	0.065 ± 0.531	0.940 ± 1.201	0.713 ± 0.371	0.616 ± 0.017
485_analcatdata_vehicle	4	48	Lookup Table (z=1)	−0.330 ± 0.436	1.192 ± 1.211	0.854 ± 0.312	0.810 ± 0.010
485_analcatdata_vehicle	4	48	Lookup Table (z=2)	−0.069 ± 0.421	0.966 ± 1.052	0.756 ± 0.307	0.692 ± 0.007
485_analcatdata_vehicle	4	48	Lookup Table (z=4)	−0.143 ± 0.401	1.012 ± 1.048	0.782 ± 0.307	0.697 ± 0.010
485_analcatdata_vehicle	4	48	Lookup Table (z=8)	−0.324 ± 0.416	1.139 ± 1.097	0.841 ± 0.283	0.702 ± 0.009
485_analcatdata_vehicle	4	48	Lookup Table (z=10)	−0.351 ± 0.382	1.168 ± 1.103	0.855 ± 0.276	0.699 ± 0.008
1096_FacultySalaries	4	50	Linear Model	−1.404 ± 3.343	0.759 ± 0.479	0.660 ± 0.170	0.570 ± 0.156
1096_FacultySalaries	4	50	Wide ReLU ANN	0.395 ± 0.735	0.207 ± 0.074	0.386 ± 0.089	0.257 ± 0.014
1096_FacultySalaries	4	50	Deep ReLU ANN	0.079 ± 1.249	0.270 ± 0.105	0.424 ± 0.097	0.256 ± 0.019
1096_FacultySalaries	4	50	ABEL-Spline (z=1)	0.347 ± 0.844	0.261 ± 0.164	0.385 ± 0.091	0.326 ± 0.012
1096_FacultySalaries	4	50	ABEL-Spline (z=2)	0.350 ± 0.698	0.286 ± 0.168	0.413 ± 0.091	0.300 ± 0.013
1096_FacultySalaries	4	50	ABEL-Spline (z=4)	0.324 ± 0.652	0.313 ± 0.150	0.425 ± 0.088	0.249 ± 0.014
1096_FacultySalaries	4	50	ABEL-Spline (z=8)	0.125 ± 0.769	0.464 ± 0.359	0.473 ± 0.146	0.187 ± 0.008
1096_FacultySalaries	4	50	ABEL-Spline (z=10)	−0.025 ± 0.921	0.526 ± 0.367	0.516 ± 0.144	0.171 ± 0.010
1096_FacultySalaries	4	50	Spline ANN (z=1)	0.101 ± 0.675	0.659 ± 0.560	0.556 ± 0.186	0.519 ± 0.014
1096_FacultySalaries	4	50	Spline ANN (z=2)	0.122 ± 0.695	0.645 ± 0.581	0.539 ± 0.189	0.491 ± 0.010
1096_FacultySalaries	4	50	Spline ANN (z=4)	0.093 ± 0.709	0.674 ± 0.613	0.553 ± 0.198	0.489 ± 0.009
1096_FacultySalaries	4	50	Spline ANN (z=8)	−0.028 ± 0.823	0.750 ± 0.665	0.578 ± 0.201	0.484 ± 0.011
1096_FacultySalaries	4	50	Spline ANN (z=10)	−0.118 ± 0.874	0.806 ± 0.682	0.609 ± 0.209	0.483 ± 0.007
1096_FacultySalaries	4	50	Lookup Table (z=1)	−0.840 ± 2.022	1.089 ± 0.752	0.740 ± 0.214	0.725 ± 0.010

Continued on next page

Table C.2 – Continued from previous page

Dataset	n	N	Model	R^2	T. MSE	T. MAE	Tr. MAE
1096_FacultySalaries	4	50	Lookup Table (z=2)	−0.614 ± 2.094	0.882 ± 0.616	0.664 ± 0.195	0.631 ± 0.010
1096_FacultySalaries	4	50	Lookup Table (z=4)	−0.955 ± 2.918	0.944 ± 0.608	0.694 ± 0.202	0.634 ± 0.012
1096_FacultySalaries	4	50	Lookup Table (z=8)	−0.962 ± 2.568	1.029 ± 0.696	0.728 ± 0.200	0.637 ± 0.013
1096_FacultySalaries	4	50	Lookup Table (z=10)	−0.833 ± 2.455	0.981 ± 0.670	0.709 ± 0.207	0.635 ± 0.012
690_visualizing_galaxy	4	323	Linear Model	−1.162 ± 4.374	0.507 ± 0.313	0.572 ± 0.197	0.507 ± 0.106
690_visualizing_galaxy	4	323	Wide ReLU ANN	0.794 ± 0.269	0.067 ± 0.071	0.180 ± 0.085	0.106 ± 0.004
690_visualizing_galaxy	4	323	Deep ReLU ANN	0.803 ± 0.306	0.065 ± 0.066	0.176 ± 0.081	0.114 ± 0.007
690_visualizing_galaxy	4	323	ABEL-Spline (z=1)	0.705 ± 0.404	0.086 ± 0.095	0.210 ± 0.116	0.133 ± 0.009
690_visualizing_galaxy	4	323	ABEL-Spline (z=2)	0.087 ± 1.650	0.161 ± 0.202	0.266 ± 0.198	0.116 ± 0.009
690_visualizing_galaxy	4	323	ABEL-Spline (z=4)	−0.033 ± 1.878	0.162 ± 0.200	0.273 ± 0.186	0.104 ± 0.008
690_visualizing_galaxy	4	323	ABEL-Spline (z=8)	−0.305 ± 2.442	0.185 ± 0.254	0.286 ± 0.210	0.096 ± 0.008
690_visualizing_galaxy	4	323	ABEL-Spline (z=10)	−0.276 ± 2.372	0.185 ± 0.247	0.288 ± 0.205	0.094 ± 0.007
690_visualizing_galaxy	4	323	Spline ANN (z=1)	0.272 ± 1.301	0.127 ± 0.107	0.283 ± 0.108	0.217 ± 0.019
690_visualizing_galaxy	4	323	Spline ANN (z=2)	−0.129 ± 2.133	0.150 ± 0.207	0.257 ± 0.186	0.166 ± 0.017
690_visualizing_galaxy	4	323	Spline ANN (z=4)	−0.539 ± 2.918	0.205 ± 0.273	0.292 ± 0.217	0.166 ± 0.020
690_visualizing_galaxy	4	323	Spline ANN (z=8)	−0.866 ± 3.467	0.283 ± 0.309	0.368 ± 0.229	0.178 ± 0.026
690_visualizing_galaxy	4	323	Spline ANN (z=10)	−0.938 ± 3.542	0.313 ± 0.316	0.394 ± 0.229	0.188 ± 0.028
690_visualizing_galaxy	4	323	Lookup Table (z=1)	−1.300 ± 2.413	1.078 ± 0.465	0.852 ± 0.254	0.819 ± 0.011
690_visualizing_galaxy	4	323	Lookup Table (z=2)	−0.112 ± 1.584	0.335 ± 0.112	0.492 ± 0.104	0.408 ± 0.027
690_visualizing_galaxy	4	323	Lookup Table (z=4)	−0.199 ± 1.231	0.505 ± 0.212	0.575 ± 0.160	0.428 ± 0.013
690_visualizing_galaxy	4	323	Lookup Table (z=8)	−0.931 ± 2.704	0.747 ± 0.291	0.692 ± 0.201	0.474 ± 0.015
690_visualizing_galaxy	4	323	Lookup Table (z=10)	−1.072 ± 2.818	0.830 ± 0.390	0.741 ± 0.241	0.492 ± 0.009
1027_ESL	4	488	Linear Model	0.819 ± 0.031	0.175 ± 0.060	0.322 ± 0.051	0.310 ± 0.015
1027_ESL	4	488	Wide ReLU ANN	0.834 ± 0.063	0.152 ± 0.041	0.268 ± 0.042	0.218 ± 0.004
1027_ESL	4	488	Deep ReLU ANN	0.840 ± 0.049	0.148 ± 0.031	0.259 ± 0.033	0.209 ± 0.008
1027_ESL	4	488	ABEL-Spline (z=1)	0.845 ± 0.039	0.144 ± 0.030	0.283 ± 0.031	0.267 ± 0.003
1027_ESL	4	488	ABEL-Spline (z=2)	0.843 ± 0.036	0.148 ± 0.033	0.284 ± 0.034	0.260 ± 0.003
1027_ESL	4	488	ABEL-Spline (z=4)	0.837 ± 0.035	0.154 ± 0.037	0.290 ± 0.035	0.258 ± 0.003
1027_ESL	4	488	ABEL-Spline (z=8)	0.836 ± 0.034	0.155 ± 0.037	0.294 ± 0.036	0.256 ± 0.004
1027_ESL	4	488	ABEL-Spline (z=10)	0.837 ± 0.033	0.154 ± 0.036	0.295 ± 0.037	0.256 ± 0.003
1027_ESL	4	488	Spline ANN (z=1)	0.842 ± 0.034	0.150 ± 0.039	0.293 ± 0.035	0.283 ± 0.004
1027_ESL	4	488	Spline ANN (z=2)	0.844 ± 0.035	0.147 ± 0.036	0.288 ± 0.033	0.271 ± 0.004
1027_ESL	4	488	Spline ANN (z=4)	0.840 ± 0.031	0.153 ± 0.043	0.289 ± 0.035	0.268 ± 0.003
1027_ESL	4	488	Spline ANN (z=8)	0.836 ± 0.029	0.157 ± 0.048	0.292 ± 0.037	0.268 ± 0.003
1027_ESL	4	488	Spline ANN (z=10)	0.836 ± 0.029	0.158 ± 0.049	0.292 ± 0.037	0.269 ± 0.003
1027_ESL	4	488	Lookup Table (z=1)	−0.070 ± 0.110	1.034 ± 0.282	0.798 ± 0.093	0.797 ± 0.004
1027_ESL	4	488	Lookup Table (z=2)	0.628 ± 0.059	0.371 ± 0.156	0.430 ± 0.089	0.408 ± 0.004
1027_ESL	4	488	Lookup Table (z=4)	0.597 ± 0.057	0.405 ± 0.168	0.430 ± 0.088	0.367 ± 0.004
1027_ESL	4	488	Lookup Table (z=8)	0.409 ± 0.051	0.590 ± 0.228	0.523 ± 0.094	0.413 ± 0.005
1027_ESL	4	488	Lookup Table (z=10)	0.344 ± 0.062	0.657 ± 0.251	0.551 ± 0.101	0.425 ± 0.003
1029_LEV	4	1000	Linear Model	0.546 ± 0.100	0.452 ± 0.133	0.514 ± 0.055	0.505 ± 0.004
1029_LEV	4	1000	Wide ReLU ANN	0.532 ± 0.118	0.466 ± 0.151	0.446 ± 0.070	0.422 ± 0.008
1029_LEV	4	1000	Deep ReLU ANN	0.511 ± 0.124	0.487 ± 0.159	0.429 ± 0.079	0.400 ± 0.013
1029_LEV	4	1000	ABEL-Spline (z=1)	0.537 ± 0.105	0.460 ± 0.136	0.497 ± 0.062	0.481 ± 0.005
1029_LEV	4	1000	ABEL-Spline (z=2)	0.528 ± 0.101	0.469 ± 0.136	0.495 ± 0.060	0.469 ± 0.006
1029_LEV	4	1000	ABEL-Spline (z=4)	0.527 ± 0.102	0.471 ± 0.139	0.494 ± 0.064	0.467 ± 0.006
1029_LEV	4	1000	ABEL-Spline (z=8)	0.526 ± 0.102	0.471 ± 0.139	0.494 ± 0.064	0.466 ± 0.006
1029_LEV	4	1000	ABEL-Spline (z=10)	0.525 ± 0.104	0.473 ± 0.141	0.494 ± 0.065	0.466 ± 0.006
1029_LEV	4	1000	Spline ANN (z=1)	0.543 ± 0.106	0.455 ± 0.137	0.509 ± 0.062	0.498 ± 0.005
1029_LEV	4	1000	Spline ANN (z=2)	0.525 ± 0.093	0.471 ± 0.126	0.516 ± 0.059	0.493 ± 0.005
1029_LEV	4	1000	Spline ANN (z=4)	0.516 ± 0.095	0.480 ± 0.128	0.517 ± 0.058	0.492 ± 0.005
1029_LEV	4	1000	Spline ANN (z=8)	0.515 ± 0.097	0.481 ± 0.129	0.518 ± 0.058	0.492 ± 0.005
1029_LEV	4	1000	Spline ANN (z=10)	0.514 ± 0.095	0.482 ± 0.128	0.519 ± 0.057	0.492 ± 0.005
1029_LEV	4	1000	Lookup Table (z=1)	−0.071 ± 0.058	1.052 ± 0.099	0.752 ± 0.053	0.751 ± 0.003
1029_LEV	4	1000	Lookup Table (z=2)	0.333 ± 0.096	0.664 ± 0.152	0.574 ± 0.084	0.554 ± 0.016
1029_LEV	4	1000	Lookup Table (z=4)	0.454 ± 0.095	0.544 ± 0.142	0.499 ± 0.062	0.450 ± 0.006
1029_LEV	4	1000	Lookup Table (z=8)	0.463 ± 0.102	0.535 ± 0.148	0.497 ± 0.068	0.443 ± 0.007
1029_LEV	4	1000	Lookup Table (z=10)	0.463 ± 0.103	0.536 ± 0.151	0.497 ± 0.066	0.442 ± 0.007
1030_ERA	4	1000	Linear Model	0.347 ± 0.157	0.639 ± 0.134	0.629 ± 0.060	0.624 ± 0.006
1030_ERA	4	1000	Wide ReLU ANN	0.325 ± 0.170	0.660 ± 0.142	0.627 ± 0.066	0.599 ± 0.007
1030_ERA	4	1000	Deep ReLU ANN	0.316 ± 0.167	0.669 ± 0.145	0.631 ± 0.068	0.596 ± 0.006
1030_ERA	4	1000	ABEL-Spline (z=1)	0.331 ± 0.174	0.653 ± 0.143	0.628 ± 0.065	0.610 ± 0.007

Continued on next page

Table C.2 – Continued from previous page

Dataset	n	N	Model	R^2	T. MSE	T. MAE	Tr. MAE
1030_ERA	4	1000	ABEL-Spline (z=2)	0.306 ± 0.178	0.679 ± 0.150	0.640 ± 0.066	0.600 ± 0.007
1030_ERA	4	1000	ABEL-Spline (z=4)	0.299 ± 0.165	0.687 ± 0.145	0.639 ± 0.070	0.591 ± 0.007
1030_ERA	4	1000	ABEL-Spline (z=8)	0.291 ± 0.171	0.694 ± 0.151	0.640 ± 0.072	0.589 ± 0.007
1030_ERA	4	1000	ABEL-Spline (z=10)	0.295 ± 0.165	0.691 ± 0.148	0.639 ± 0.071	0.589 ± 0.007
1030_ERA	4	1000	Spline ANN (z=1)	0.335 ± 0.174	0.649 ± 0.145	0.629 ± 0.065	0.616 ± 0.007
1030_ERA	4	1000	Spline ANN (z=2)	0.334 ± 0.168	0.651 ± 0.142	0.632 ± 0.065	0.613 ± 0.007
1030_ERA	4	1000	Spline ANN (z=4)	0.329 ± 0.151	0.658 ± 0.136	0.641 ± 0.066	0.604 ± 0.006
1030_ERA	4	1000	Spline ANN (z=8)	0.323 ± 0.147	0.666 ± 0.143	0.638 ± 0.070	0.595 ± 0.007
1030_ERA	4	1000	Spline ANN (z=10)	0.323 ± 0.147	0.665 ± 0.143	0.635 ± 0.071	0.595 ± 0.007
1030_ERA	4	1000	Lookup Table (z=1)	−0.015 ± 0.012	1.007 ± 0.147	0.815 ± 0.072	0.814 ± 0.002
1030_ERA	4	1000	Lookup Table (z=2)	0.294 ± 0.144	0.696 ± 0.150	0.658 ± 0.066	0.637 ± 0.007
1030_ERA	4	1000	Lookup Table (z=4)	0.334 ± 0.144	0.655 ± 0.136	0.635 ± 0.067	0.608 ± 0.007
1030_ERA	4	1000	Lookup Table (z=8)	0.335 ± 0.135	0.654 ± 0.132	0.637 ± 0.069	0.606 ± 0.007
1030_ERA	4	1000	Lookup Table (z=10)	0.338 ± 0.141	0.651 ± 0.135	0.637 ± 0.071	0.603 ± 0.008
529_pollen	4	3848	Linear Model	0.768 ± 0.020	0.232 ± 0.023	0.377 ± 0.020	0.377 ± 0.002
529_pollen	4	3848	Wide ReLU ANN	0.782 ± 0.024	0.218 ± 0.028	0.371 ± 0.025	0.355 ± 0.003
529_pollen	4	3848	Deep ReLU ANN	0.769 ± 0.020	0.231 ± 0.022	0.379 ± 0.021	0.353 ± 0.006
529_pollen	4	3848	ABEL-Spline (z=1)	0.790 ± 0.021	0.210 ± 0.023	0.362 ± 0.021	0.359 ± 0.002
529_pollen	4	3848	ABEL-Spline (z=2)	0.790 ± 0.022	0.210 ± 0.024	0.364 ± 0.022	0.356 ± 0.002
529_pollen	4	3848	ABEL-Spline (z=4)	0.785 ± 0.025	0.215 ± 0.028	0.367 ± 0.024	0.350 ± 0.003
529_pollen	4	3848	ABEL-Spline (z=8)	0.776 ± 0.026	0.224 ± 0.028	0.373 ± 0.024	0.342 ± 0.003
529_pollen	4	3848	ABEL-Spline (z=10)	0.773 ± 0.027	0.226 ± 0.030	0.375 ± 0.025	0.338 ± 0.003
529_pollen	4	3848	Spline ANN (z=1)	0.769 ± 0.020	0.230 ± 0.023	0.376 ± 0.019	0.375 ± 0.002
529_pollen	4	3848	Spline ANN (z=2)	0.779 ± 0.020	0.221 ± 0.023	0.369 ± 0.019	0.366 ± 0.002
529_pollen	4	3848	Spline ANN (z=4)	0.772 ± 0.024	0.227 ± 0.026	0.373 ± 0.021	0.365 ± 0.002
529_pollen	4	3848	Spline ANN (z=8)	0.747 ± 0.025	0.252 ± 0.027	0.389 ± 0.020	0.372 ± 0.002
529_pollen	4	3848	Spline ANN (z=10)	0.735 ± 0.026	0.264 ± 0.028	0.397 ± 0.021	0.375 ± 0.002
529_pollen	4	3848	Lookup Table (z=1)	−0.002 ± 0.002	1.001 ± 0.043	0.788 ± 0.018	0.788 ± 0.001
529_pollen	4	3848	Lookup Table (z=2)	0.350 ± 0.032	0.649 ± 0.047	0.628 ± 0.026	0.623 ± 0.002
529_pollen	4	3848	Lookup Table (z=4)	0.520 ± 0.032	0.479 ± 0.035	0.540 ± 0.022	0.509 ± 0.002
529_pollen	4	3848	Lookup Table (z=8)	0.441 ± 0.058	0.559 ± 0.066	0.554 ± 0.030	0.377 ± 0.002
529_pollen	4	3848	Lookup Table (z=10)	0.308 ± 0.045	0.692 ± 0.059	0.622 ± 0.026	0.338 ± 0.001
687_sleuth_ex1605	5	62	Linear Model	−1.472 ± 3.349	1.062 ± 0.603	0.836 ± 0.296	0.743 ± 0.136
687_sleuth_ex1605	5	62	Wide ReLU ANN	−0.315 ± 1.663	0.438 ± 0.192	0.552 ± 0.138	0.274 ± 0.015
687_sleuth_ex1605	5	62	Deep ReLU ANN	−0.950 ± 3.144	0.571 ± 0.414	0.599 ± 0.223	0.189 ± 0.046
687_sleuth_ex1605	5	62	ABEL-Spline (z=1)	−0.528 ± 2.369	0.499 ± 0.285	0.552 ± 0.174	0.454 ± 0.025
687_sleuth_ex1605	5	62	ABEL-Spline (z=2)	−0.663 ± 2.470	0.553 ± 0.309	0.602 ± 0.185	0.392 ± 0.018
687_sleuth_ex1605	5	62	ABEL-Spline (z=4)	−0.915 ± 2.421	0.621 ± 0.313	0.662 ± 0.154	0.273 ± 0.020
687_sleuth_ex1605	5	62	ABEL-Spline (z=8)	−0.941 ± 2.648	0.629 ± 0.317	0.644 ± 0.178	0.143 ± 0.019
687_sleuth_ex1605	5	62	ABEL-Spline (z=10)	−0.640 ± 1.876	0.608 ± 0.320	0.603 ± 0.168	0.119 ± 0.017
687_sleuth_ex1605	5	62	Spline ANN (z=1)	−0.862 ± 2.116	0.843 ± 0.360	0.751 ± 0.206	0.673 ± 0.017
687_sleuth_ex1605	5	62	Spline ANN (z=2)	−0.891 ± 2.111	0.857 ± 0.373	0.758 ± 0.197	0.637 ± 0.015
687_sleuth_ex1605	5	62	Spline ANN (z=4)	−0.771 ± 1.713	0.841 ± 0.359	0.746 ± 0.166	0.584 ± 0.016
687_sleuth_ex1605	5	62	Spline ANN (z=8)	−0.751 ± 1.808	0.788 ± 0.359	0.709 ± 0.153	0.508 ± 0.014
687_sleuth_ex1605	5	62	Spline ANN (z=10)	−0.764 ± 1.863	0.805 ± 0.373	0.715 ± 0.155	0.490 ± 0.016
687_sleuth_ex1605	5	62	Lookup Table (z=1)	−1.133 ± 2.069	1.110 ± 0.508	0.868 ± 0.221	0.813 ± 0.009
687_sleuth_ex1605	5	62	Lookup Table (z=2)	−1.048 ± 2.039	1.010 ± 0.442	0.828 ± 0.203	0.734 ± 0.012
687_sleuth_ex1605	5	62	Lookup Table (z=4)	−1.036 ± 2.049	1.069 ± 0.501	0.835 ± 0.225	0.686 ± 0.010
687_sleuth_ex1605	5	62	Lookup Table (z=8)	−1.006 ± 1.999	1.084 ± 0.521	0.848 ± 0.227	0.682 ± 0.009
687_sleuth_ex1605	5	62	Lookup Table (z=10)	−0.990 ± 1.938	1.092 ± 0.531	0.850 ± 0.233	0.682 ± 0.009
594_fri_c2_100_5	5	100	Linear Model	−1.108 ± 2.431	1.233 ± 0.693	0.824 ± 0.257	0.776 ± 0.045
594_fri_c2_100_5	5	100	Wide ReLU ANN	0.438 ± 0.263	0.508 ± 0.365	0.475 ± 0.231	0.291 ± 0.015
594_fri_c2_100_5	5	100	Deep ReLU ANN	0.416 ± 0.267	0.436 ± 0.284	0.459 ± 0.179	0.224 ± 0.059
594_fri_c2_100_5	5	100	ABEL-Spline (z=1)	0.421 ± 0.333	0.459 ± 0.323	0.501 ± 0.196	0.425 ± 0.017
594_fri_c2_100_5	5	100	ABEL-Spline (z=2)	0.663 ± 0.363	0.192 ± 0.143	0.336 ± 0.129	0.231 ± 0.014
594_fri_c2_100_5	5	100	ABEL-Spline (z=4)	0.698 ± 0.286	0.175 ± 0.150	0.311 ± 0.123	0.146 ± 0.017
594_fri_c2_100_5	5	100	ABEL-Spline (z=8)	0.511 ± 0.618	0.252 ± 0.171	0.382 ± 0.135	0.090 ± 0.011
594_fri_c2_100_5	5	100	ABEL-Spline (z=10)	0.432 ± 0.646	0.288 ± 0.173	0.419 ± 0.137	0.067 ± 0.009
594_fri_c2_100_5	5	100	Spline ANN (z=1)	−0.192 ± 0.994	0.844 ± 0.551	0.682 ± 0.239	0.634 ± 0.010
594_fri_c2_100_5	5	100	Spline ANN (z=2)	−0.026 ± 0.915	0.708 ± 0.471	0.621 ± 0.216	0.548 ± 0.011
594_fri_c2_100_5	5	100	Spline ANN (z=4)	0.135 ± 0.638	0.615 ± 0.415	0.593 ± 0.204	0.492 ± 0.010
594_fri_c2_100_5	5	100	Spline ANN (z=8)	−0.061 ± 0.997	0.684 ± 0.448	0.647 ± 0.216	0.460 ± 0.010

Continued on next page

Table C.2 – Continued from previous page

Dataset	n	N	Model	R^2	T. MSE	T. MAE	Tr. MAE
594_fri_c2_100_5	5	100	Spline ANN (z=10)	−0.113 ± 1.175	0.692 ± 0.444	0.656 ± 0.222	0.453 ± 0.008
594_fri_c2_100_5	5	100	Lookup Table (z=1)	−0.601 ± 1.247	1.102 ± 0.637	0.798 ± 0.242	0.771 ± 0.009
594_fri_c2_100_5	5	100	Lookup Table (z=2)	−0.474 ± 1.435	0.920 ± 0.536	0.748 ± 0.217	0.693 ± 0.013
594_fri_c2_100_5	5	100	Lookup Table (z=4)	−0.753 ± 1.870	1.030 ± 0.520	0.796 ± 0.215	0.659 ± 0.009
594_fri_c2_100_5	5	100	Lookup Table (z=8)	−0.825 ± 1.893	1.059 ± 0.503	0.818 ± 0.209	0.645 ± 0.011
594_fri_c2_100_5	5	100	Lookup Table (z=10)	−0.834 ± 1.951	1.065 ± 0.514	0.817 ± 0.209	0.637 ± 0.009
611_fri_c3_100_5	5	100	Linear Model	−0.135 ± 0.514	0.985 ± 0.448	0.736 ± 0.182	0.757 ± 0.039
611_fri_c3_100_5	5	100	Wide ReLU ANN	0.901 ± 0.059	0.079 ± 0.046	0.215 ± 0.062	0.144 ± 0.009
611_fri_c3_100_5	5	100	Deep ReLU ANN	0.649 ± 0.454	0.221 ± 0.133	0.341 ± 0.103	0.180 ± 0.035
611_fri_c3_100_5	5	100	ABEL-Spline (z=1)	0.608 ± 0.199	0.312 ± 0.154	0.455 ± 0.129	0.387 ± 0.017
611_fri_c3_100_5	5	100	ABEL-Spline (z=2)	0.878 ± 0.080	0.093 ± 0.059	0.224 ± 0.065	0.168 ± 0.007
611_fri_c3_100_5	5	100	ABEL-Spline (z=4)	0.834 ± 0.148	0.113 ± 0.071	0.255 ± 0.047	0.127 ± 0.006
611_fri_c3_100_5	5	100	ABEL-Spline (z=8)	0.712 ± 0.223	0.201 ± 0.088	0.355 ± 0.057	0.088 ± 0.008
611_fri_c3_100_5	5	100	ABEL-Spline (z=10)	0.685 ± 0.241	0.220 ± 0.094	0.371 ± 0.064	0.066 ± 0.010
611_fri_c3_100_5	5	100	Spline ANN (z=1)	0.132 ± 0.172	0.769 ± 0.352	0.685 ± 0.183	0.636 ± 0.009
611_fri_c3_100_5	5	100	Spline ANN (z=2)	0.424 ± 0.108	0.543 ± 0.285	0.559 ± 0.168	0.500 ± 0.013
611_fri_c3_100_5	5	100	Spline ANN (z=4)	0.524 ± 0.105	0.457 ± 0.249	0.508 ± 0.163	0.427 ± 0.012
611_fri_c3_100_5	5	100	Spline ANN (z=8)	0.438 ± 0.176	0.483 ± 0.225	0.518 ± 0.130	0.396 ± 0.011
611_fri_c3_100_5	5	100	Spline ANN (z=10)	0.381 ± 0.184	0.526 ± 0.228	0.543 ± 0.127	0.390 ± 0.011
611_fri_c3_100_5	5	100	Lookup Table (z=1)	−0.153 ± 0.177	1.070 ± 0.492	0.792 ± 0.228	0.778 ± 0.009
611_fri_c3_100_5	5	100	Lookup Table (z=2)	−0.018 ± 0.216	0.892 ± 0.389	0.738 ± 0.188	0.695 ± 0.005
611_fri_c3_100_5	5	100	Lookup Table (z=4)	0.073 ± 0.105	0.857 ± 0.417	0.720 ± 0.199	0.616 ± 0.008
611_fri_c3_100_5	5	100	Lookup Table (z=8)	−0.098 ± 0.122	0.999 ± 0.447	0.784 ± 0.209	0.628 ± 0.009
611_fri_c3_100_5	5	100	Lookup Table (z=10)	−0.116 ± 0.098	1.036 ± 0.493	0.794 ± 0.221	0.630 ± 0.007
624_fri_c0_100_5	5	100	Linear Model	−0.300 ± 0.836	0.929 ± 0.444	0.781 ± 0.209	0.697 ± 0.117
624_fri_c0_100_5	5	100	Wide ReLU ANN	0.825 ± 0.138	0.113 ± 0.054	0.268 ± 0.078	0.104 ± 0.010
624_fri_c0_100_5	5	100	Deep ReLU ANN	0.729 ± 0.111	0.230 ± 0.161	0.351 ± 0.127	0.118 ± 0.028
624_fri_c0_100_5	5	100	ABEL-Spline (z=1)	0.788 ± 0.151	0.139 ± 0.053	0.299 ± 0.079	0.241 ± 0.010
624_fri_c0_100_5	5	100	ABEL-Spline (z=2)	0.750 ± 0.209	0.156 ± 0.068	0.305 ± 0.077	0.207 ± 0.009
624_fri_c0_100_5	5	100	ABEL-Spline (z=4)	0.754 ± 0.156	0.170 ± 0.066	0.316 ± 0.079	0.165 ± 0.006
624_fri_c0_100_5	5	100	ABEL-Spline (z=8)	0.607 ± 0.235	0.273 ± 0.099	0.400 ± 0.096	0.106 ± 0.009
624_fri_c0_100_5	5	100	ABEL-Spline (z=10)	0.539 ± 0.360	0.301 ± 0.102	0.419 ± 0.096	0.079 ± 0.008
624_fri_c0_100_5	5	100	Spline ANN (z=1)	0.329 ± 0.193	0.535 ± 0.197	0.606 ± 0.122	0.569 ± 0.012
624_fri_c0_100_5	5	100	Spline ANN (z=2)	0.409 ± 0.190	0.468 ± 0.182	0.562 ± 0.125	0.508 ± 0.011
624_fri_c0_100_5	5	100	Spline ANN (z=4)	0.328 ± 0.222	0.526 ± 0.197	0.601 ± 0.127	0.500 ± 0.015
624_fri_c0_100_5	5	100	Spline ANN (z=8)	0.177 ± 0.307	0.625 ± 0.213	0.664 ± 0.114	0.483 ± 0.013
624_fri_c0_100_5	5	100	Spline ANN (z=10)	0.135 ± 0.343	0.650 ± 0.215	0.678 ± 0.113	0.474 ± 0.009
624_fri_c0_100_5	5	100	Lookup Table (z=1)	−0.403 ± 0.502	1.078 ± 0.353	0.876 ± 0.145	0.845 ± 0.007
624_fri_c0_100_5	5	100	Lookup Table (z=2)	−0.104 ± 0.313	0.876 ± 0.317	0.779 ± 0.146	0.711 ± 0.007
624_fri_c0_100_5	5	100	Lookup Table (z=4)	−0.311 ± 0.404	1.029 ± 0.361	0.855 ± 0.155	0.683 ± 0.008
624_fri_c0_100_5	5	100	Lookup Table (z=8)	−0.353 ± 0.459	1.053 ± 0.363	0.872 ± 0.153	0.684 ± 0.006
624_fri_c0_100_5	5	100	Lookup Table (z=10)	−0.353 ± 0.457	1.052 ± 0.360	0.870 ± 0.151	0.686 ± 0.007
656_fri_c1_100_5	5	100	Linear Model	0.037 ± 0.276	0.922 ± 0.611	0.735 ± 0.269	0.796 ± 0.047
656_fri_c1_100_5	5	100	Wide ReLU ANN	0.439 ± 0.383	0.594 ± 0.520	0.465 ± 0.238	0.295 ± 0.016
656_fri_c1_100_5	5	100	Deep ReLU ANN	0.563 ± 0.220	0.400 ± 0.269	0.457 ± 0.156	0.208 ± 0.037
656_fri_c1_100_5	5	100	ABEL-Spline (z=1)	0.651 ± 0.170	0.353 ± 0.271	0.420 ± 0.173	0.331 ± 0.014
656_fri_c1_100_5	5	100	ABEL-Spline (z=2)	0.799 ± 0.068	0.192 ± 0.128	0.337 ± 0.129	0.216 ± 0.008
656_fri_c1_100_5	5	100	ABEL-Spline (z=4)	0.833 ± 0.037	0.150 ± 0.078	0.305 ± 0.088	0.148 ± 0.008
656_fri_c1_100_5	5	100	ABEL-Spline (z=8)	0.718 ± 0.121	0.235 ± 0.102	0.382 ± 0.102	0.094 ± 0.007
656_fri_c1_100_5	5	100	ABEL-Spline (z=10)	0.675 ± 0.130	0.270 ± 0.110	0.410 ± 0.107	0.072 ± 0.010
656_fri_c1_100_5	5	100	Spline ANN (z=1)	0.342 ± 0.119	0.658 ± 0.451	0.631 ± 0.238	0.590 ± 0.011
656_fri_c1_100_5	5	100	Spline ANN (z=2)	0.471 ± 0.129	0.543 ± 0.402	0.563 ± 0.228	0.503 ± 0.011
656_fri_c1_100_5	5	100	Spline ANN (z=4)	0.491 ± 0.112	0.517 ± 0.369	0.554 ± 0.215	0.450 ± 0.005
656_fri_c1_100_5	5	100	Spline ANN (z=8)	0.407 ± 0.145	0.603 ± 0.433	0.604 ± 0.236	0.435 ± 0.010
656_fri_c1_100_5	5	100	Spline ANN (z=10)	0.360 ± 0.120	0.633 ± 0.429	0.624 ± 0.226	0.430 ± 0.007
656_fri_c1_100_5	5	100	Lookup Table (z=1)	−0.130 ± 0.119	1.066 ± 0.648	0.815 ± 0.268	0.795 ± 0.008
656_fri_c1_100_5	5	100	Lookup Table (z=2)	0.014 ± 0.129	0.912 ± 0.515	0.756 ± 0.228	0.694 ± 0.007
656_fri_c1_100_5	5	100	Lookup Table (z=4)	−0.078 ± 0.112	1.021 ± 0.625	0.797 ± 0.262	0.643 ± 0.005
656_fri_c1_100_5	5	100	Lookup Table (z=8)	−0.161 ± 0.161	1.063 ± 0.598	0.826 ± 0.245	0.647 ± 0.008
656_fri_c1_100_5	5	100	Lookup Table (z=10)	−0.151 ± 0.134	1.059 ± 0.582	0.818 ± 0.241	0.644 ± 0.005
579_fri_c0_250_5	5	250	Linear Model	0.320 ± 0.253	0.633 ± 0.216	0.654 ± 0.135	0.614 ± 0.082
579_fri_c0_250_5	5	250	Wide ReLU ANN	0.855 ± 0.087	0.125 ± 0.044	0.266 ± 0.044	0.202 ± 0.006

Continued on next page

Table C.2 – Continued from previous page

Dataset	n	N	Model	R^2	T. MSE	T. MAE	Tr. MAE
579_fri_c0_250_5	5	250	Deep ReLU ANN	0.797 ± 0.076	0.191 ± 0.058	0.336 ± 0.057	0.227 ± 0.023
579_fri_c0_250_5	5	250	ABEL-Spline (z=1)	0.829 ± 0.084	0.150 ± 0.041	0.300 ± 0.044	0.268 ± 0.007
579_fri_c0_250_5	5	250	ABEL-Spline (z=2)	0.845 ± 0.078	0.137 ± 0.042	0.281 ± 0.047	0.236 ± 0.008
579_fri_c0_250_5	5	250	ABEL-Spline (z=4)	0.841 ± 0.078	0.141 ± 0.039	0.285 ± 0.054	0.213 ± 0.007
579_fri_c0_250_5	5	250	ABEL-Spline (z=8)	0.823 ± 0.086	0.159 ± 0.054	0.306 ± 0.057	0.182 ± 0.008
579_fri_c0_250_5	5	250	ABEL-Spline (z=10)	0.793 ± 0.090	0.187 ± 0.055	0.337 ± 0.061	0.170 ± 0.007
579_fri_c0_250_5	5	250	Spline ANN (z=1)	0.629 ± 0.040	0.363 ± 0.098	0.495 ± 0.088	0.478 ± 0.010
579_fri_c0_250_5	5	250	Spline ANN (z=2)	0.676 ± 0.050	0.314 ± 0.086	0.452 ± 0.079	0.417 ± 0.011
579_fri_c0_250_5	5	250	Spline ANN (z=4)	0.635 ± 0.048	0.357 ± 0.102	0.483 ± 0.085	0.426 ± 0.012
579_fri_c0_250_5	5	250	Spline ANN (z=8)	0.530 ± 0.102	0.448 ± 0.118	0.540 ± 0.081	0.441 ± 0.013
579_fri_c0_250_5	5	250	Spline ANN (z=10)	0.502 ± 0.097	0.478 ± 0.129	0.558 ± 0.082	0.445 ± 0.013
579_fri_c0_250_5	5	250	Lookup Table (z=1)	−0.044 ± 0.071	1.011 ± 0.235	0.838 ± 0.121	0.834 ± 0.004
579_fri_c0_250_5	5	250	Lookup Table (z=2)	0.180 ± 0.074	0.801 ± 0.211	0.750 ± 0.114	0.707 ± 0.007
579_fri_c0_250_5	5	250	Lookup Table (z=4)	0.013 ± 0.078	0.959 ± 0.237	0.803 ± 0.119	0.616 ± 0.004
579_fri_c0_250_5	5	250	Lookup Table (z=8)	−0.036 ± 0.083	1.003 ± 0.238	0.833 ± 0.121	0.609 ± 0.004
579_fri_c0_250_5	5	250	Lookup Table (z=10)	−0.048 ± 0.073	1.016 ± 0.239	0.840 ± 0.122	0.610 ± 0.004
596_fri_c2_250_5	5	250	Linear Model	0.039 ± 0.174	0.923 ± 0.266	0.784 ± 0.115	0.749 ± 0.018
596_fri_c2_250_5	5	250	Wide ReLU ANN	0.863 ± 0.106	0.125 ± 0.091	0.246 ± 0.068	0.209 ± 0.016
596_fri_c2_250_5	5	250	Deep ReLU ANN	0.731 ± 0.249	0.235 ± 0.173	0.345 ± 0.122	0.277 ± 0.055
596_fri_c2_250_5	5	250	ABEL-Spline (z=1)	0.604 ± 0.107	0.383 ± 0.134	0.495 ± 0.099	0.446 ± 0.011
596_fri_c2_250_5	5	250	ABEL-Spline (z=2)	0.857 ± 0.056	0.134 ± 0.048	0.261 ± 0.033	0.215 ± 0.006
596_fri_c2_250_5	5	250	ABEL-Spline (z=4)	0.890 ± 0.025	0.105 ± 0.029	0.234 ± 0.026	0.167 ± 0.004
596_fri_c2_250_5	5	250	ABEL-Spline (z=8)	0.874 ± 0.033	0.123 ± 0.046	0.261 ± 0.039	0.141 ± 0.006
596_fri_c2_250_5	5	250	ABEL-Spline (z=10)	0.860 ± 0.033	0.134 ± 0.040	0.282 ± 0.046	0.131 ± 0.002
596_fri_c2_250_5	5	250	Spline ANN (z=1)	0.360 ± 0.084	0.621 ± 0.186	0.642 ± 0.103	0.628 ± 0.010
596_fri_c2_250_5	5	250	Spline ANN (z=2)	0.644 ± 0.074	0.349 ± 0.127	0.469 ± 0.092	0.438 ± 0.011
596_fri_c2_250_5	5	250	Spline ANN (z=4)	0.729 ± 0.069	0.267 ± 0.108	0.402 ± 0.088	0.352 ± 0.011
596_fri_c2_250_5	5	250	Spline ANN (z=8)	0.696 ± 0.075	0.301 ± 0.123	0.424 ± 0.090	0.342 ± 0.010
596_fri_c2_250_5	5	250	Spline ANN (z=10)	0.682 ± 0.072	0.315 ± 0.131	0.435 ± 0.089	0.344 ± 0.006
596_fri_c2_250_5	5	250	Lookup Table (z=1)	−0.077 ± 0.076	1.045 ± 0.278	0.833 ± 0.105	0.826 ± 0.004
596_fri_c2_250_5	5	250	Lookup Table (z=2)	0.233 ± 0.093	0.745 ± 0.221	0.702 ± 0.103	0.670 ± 0.009
596_fri_c2_250_5	5	250	Lookup Table (z=4)	0.182 ± 0.076	0.792 ± 0.219	0.731 ± 0.097	0.606 ± 0.005
596_fri_c2_250_5	5	250	Lookup Table (z=8)	0.013 ± 0.061	0.951 ± 0.226	0.812 ± 0.095	0.614 ± 0.004
596_fri_c2_250_5	5	250	Lookup Table (z=10)	−0.045 ± 0.067	1.005 ± 0.236	0.840 ± 0.093	0.617 ± 0.002
601_fri_c1_250_5	5	250	Linear Model	0.028 ± 0.260	0.864 ± 0.171	0.772 ± 0.094	0.769 ± 0.034
601_fri_c1_250_5	5	250	Wide ReLU ANN	0.749 ± 0.112	0.222 ± 0.104	0.313 ± 0.070	0.269 ± 0.021
601_fri_c1_250_5	5	250	Deep ReLU ANN	0.623 ± 0.225	0.316 ± 0.142	0.388 ± 0.068	0.280 ± 0.044
601_fri_c1_250_5	5	250	ABEL-Spline (z=1)	0.733 ± 0.110	0.244 ± 0.130	0.340 ± 0.075	0.319 ± 0.017
601_fri_c1_250_5	5	250	ABEL-Spline (z=2)	0.904 ± 0.045	0.082 ± 0.029	0.212 ± 0.038	0.182 ± 0.005
601_fri_c1_250_5	5	250	ABEL-Spline (z=4)	0.901 ± 0.046	0.084 ± 0.029	0.213 ± 0.037	0.156 ± 0.005
601_fri_c1_250_5	5	250	ABEL-Spline (z=8)	0.877 ± 0.051	0.106 ± 0.036	0.247 ± 0.034	0.128 ± 0.006
601_fri_c1_250_5	5	250	ABEL-Spline (z=10)	0.858 ± 0.071	0.120 ± 0.045	0.265 ± 0.049	0.118 ± 0.006
601_fri_c1_250_5	5	250	Spline ANN (z=1)	0.398 ± 0.108	0.550 ± 0.144	0.605 ± 0.088	0.578 ± 0.013
601_fri_c1_250_5	5	250	Spline ANN (z=2)	0.699 ± 0.076	0.273 ± 0.087	0.409 ± 0.071	0.378 ± 0.010
601_fri_c1_250_5	5	250	Spline ANN (z=4)	0.761 ± 0.057	0.216 ± 0.061	0.364 ± 0.062	0.322 ± 0.010
601_fri_c1_250_5	5	250	Spline ANN (z=8)	0.692 ± 0.075	0.277 ± 0.060	0.418 ± 0.057	0.329 ± 0.010
601_fri_c1_250_5	5	250	Spline ANN (z=10)	0.662 ± 0.076	0.306 ± 0.074	0.441 ± 0.066	0.339 ± 0.012
601_fri_c1_250_5	5	250	Lookup Table (z=1)	−0.122 ± 0.134	1.031 ± 0.239	0.846 ± 0.097	0.834 ± 0.005
601_fri_c1_250_5	5	250	Lookup Table (z=2)	0.205 ± 0.089	0.733 ± 0.175	0.717 ± 0.093	0.675 ± 0.010
601_fri_c1_250_5	5	250	Lookup Table (z=4)	0.081 ± 0.088	0.854 ± 0.231	0.754 ± 0.107	0.609 ± 0.006
601_fri_c1_250_5	5	250	Lookup Table (z=8)	−0.069 ± 0.115	0.988 ± 0.250	0.822 ± 0.102	0.609 ± 0.005
601_fri_c1_250_5	5	250	Lookup Table (z=10)	−0.102 ± 0.117	1.018 ± 0.257	0.842 ± 0.104	0.610 ± 0.005
613_fri_c3_250_5	5	250	Linear Model	0.091 ± 0.163	0.887 ± 0.281	0.747 ± 0.124	0.731 ± 0.021
613_fri_c3_250_5	5	250	Wide ReLU ANN	0.845 ± 0.068	0.155 ± 0.083	0.284 ± 0.075	0.249 ± 0.021
613_fri_c3_250_5	5	250	Deep ReLU ANN	0.754 ± 0.116	0.231 ± 0.101	0.341 ± 0.066	0.277 ± 0.031
613_fri_c3_250_5	5	250	ABEL-Spline (z=1)	0.683 ± 0.097	0.310 ± 0.120	0.421 ± 0.089	0.393 ± 0.017
613_fri_c3_250_5	5	250	ABEL-Spline (z=2)	0.895 ± 0.031	0.098 ± 0.030	0.244 ± 0.036	0.200 ± 0.006
613_fri_c3_250_5	5	250	ABEL-Spline (z=4)	0.903 ± 0.033	0.091 ± 0.028	0.234 ± 0.034	0.162 ± 0.005
613_fri_c3_250_5	5	250	ABEL-Spline (z=8)	0.884 ± 0.030	0.112 ± 0.032	0.258 ± 0.034	0.134 ± 0.007
613_fri_c3_250_5	5	250	ABEL-Spline (z=10)	0.870 ± 0.041	0.126 ± 0.043	0.275 ± 0.051	0.123 ± 0.006
613_fri_c3_250_5	5	250	Spline ANN (z=1)	0.309 ± 0.102	0.675 ± 0.206	0.637 ± 0.110	0.617 ± 0.012
613_fri_c3_250_5	5	250	Spline ANN (z=2)	0.608 ± 0.085	0.388 ± 0.149	0.472 ± 0.100	0.437 ± 0.010

Continued on next page

Table C.2 – Continued from previous page

Dataset	n	N	Model	R^2	T. MSE	T. MAE	Tr. MAE
613_fri_c3_250_5	5	250	Spline ANN (z=4)	0.699 ± 0.050	0.299 ± 0.110	0.414 ± 0.083	0.367 ± 0.011
613_fri_c3_250_5	5	250	Spline ANN (z=8)	0.646 ± 0.065	0.349 ± 0.135	0.456 ± 0.092	0.362 ± 0.008
613_fri_c3_250_5	5	250	Spline ANN (z=10)	0.635 ± 0.046	0.358 ± 0.108	0.466 ± 0.087	0.359 ± 0.006
613_fri_c3_250_5	5	250	Lookup Table (z=1)	−0.063 ± 0.099	1.030 ± 0.247	0.792 ± 0.112	0.785 ± 0.005
613_fri_c3_250_5	5	250	Lookup Table (z=2)	0.210 ± 0.098	0.771 ± 0.225	0.687 ± 0.110	0.657 ± 0.008
613_fri_c3_250_5	5	250	Lookup Table (z=4)	0.214 ± 0.104	0.759 ± 0.185	0.672 ± 0.099	0.586 ± 0.006
613_fri_c3_250_5	5	250	Lookup Table (z=8)	0.057 ± 0.078	0.911 ± 0.213	0.758 ± 0.099	0.581 ± 0.006
613_fri_c3_250_5	5	250	Lookup Table (z=10)	−0.017 ± 0.065	0.984 ± 0.230	0.785 ± 0.105	0.579 ± 0.005
597_fri_c2_500_5	5	500	Linear Model	0.189 ± 0.097	0.785 ± 0.105	0.730 ± 0.053	0.711 ± 0.008
597_fri_c2_500_5	5	500	Wide ReLU ANN	0.952 ± 0.033	0.048 ± 0.037	0.148 ± 0.021	0.126 ± 0.008
597_fri_c2_500_5	5	500	Deep ReLU ANN	0.933 ± 0.020	0.066 ± 0.023	0.187 ± 0.021	0.141 ± 0.016
597_fri_c2_500_5	5	500	ABEL-Spline (z=1)	0.907 ± 0.055	0.092 ± 0.063	0.208 ± 0.038	0.198 ± 0.006
597_fri_c2_500_5	5	500	ABEL-Spline (z=2)	0.925 ± 0.030	0.073 ± 0.033	0.190 ± 0.032	0.172 ± 0.003
597_fri_c2_500_5	5	500	ABEL-Spline (z=4)	0.925 ± 0.025	0.072 ± 0.027	0.192 ± 0.029	0.157 ± 0.003
597_fri_c2_500_5	5	500	ABEL-Spline (z=8)	0.913 ± 0.028	0.084 ± 0.030	0.214 ± 0.028	0.139 ± 0.003
597_fri_c2_500_5	5	500	ABEL-Spline (z=10)	0.906 ± 0.028	0.091 ± 0.031	0.225 ± 0.030	0.130 ± 0.003
597_fri_c2_500_5	5	500	Spline ANN (z=1)	0.554 ± 0.073	0.429 ± 0.056	0.507 ± 0.034	0.486 ± 0.007
597_fri_c2_500_5	5	500	Spline ANN (z=2)	0.896 ± 0.027	0.102 ± 0.034	0.233 ± 0.035	0.219 ± 0.005
597_fri_c2_500_5	5	500	Spline ANN (z=4)	0.920 ± 0.027	0.078 ± 0.029	0.199 ± 0.028	0.180 ± 0.003
597_fri_c2_500_5	5	500	Spline ANN (z=8)	0.910 ± 0.024	0.088 ± 0.028	0.217 ± 0.029	0.176 ± 0.002
597_fri_c2_500_5	5	500	Spline ANN (z=10)	0.898 ± 0.025	0.100 ± 0.029	0.231 ± 0.028	0.175 ± 0.003
597_fri_c2_500_5	5	500	Lookup Table (z=1)	−0.073 ± 0.076	1.061 ± 0.243	0.811 ± 0.111	0.805 ± 0.005
597_fri_c2_500_5	5	500	Lookup Table (z=2)	0.352 ± 0.073	0.628 ± 0.083	0.622 ± 0.050	0.589 ± 0.009
597_fri_c2_500_5	5	500	Lookup Table (z=4)	0.335 ± 0.070	0.650 ± 0.118	0.642 ± 0.070	0.506 ± 0.005
597_fri_c2_500_5	5	500	Lookup Table (z=8)	0.047 ± 0.054	0.936 ± 0.180	0.792 ± 0.086	0.523 ± 0.003
597_fri_c2_500_5	5	500	Lookup Table (z=10)	0.001 ± 0.021	0.978 ± 0.160	0.812 ± 0.076	0.524 ± 0.002
617_fri_c3_500_5	5	500	Linear Model	0.137 ± 0.071	0.862 ± 0.203	0.740 ± 0.092	0.733 ± 0.015
617_fri_c3_500_5	5	500	Wide ReLU ANN	0.902 ± 0.054	0.095 ± 0.056	0.182 ± 0.032	0.160 ± 0.012
617_fri_c3_500_5	5	500	Deep ReLU ANN	0.909 ± 0.034	0.090 ± 0.040	0.205 ± 0.030	0.163 ± 0.013
617_fri_c3_500_5	5	500	ABEL-Spline (z=1)	0.847 ± 0.051	0.148 ± 0.045	0.254 ± 0.033	0.240 ± 0.006
617_fri_c3_500_5	5	500	ABEL-Spline (z=2)	0.906 ± 0.025	0.090 ± 0.023	0.211 ± 0.021	0.186 ± 0.003
617_fri_c3_500_5	5	500	ABEL-Spline (z=4)	0.919 ± 0.024	0.080 ± 0.027	0.202 ± 0.027	0.163 ± 0.003
617_fri_c3_500_5	5	500	ABEL-Spline (z=8)	0.908 ± 0.021	0.091 ± 0.026	0.220 ± 0.026	0.141 ± 0.002
617_fri_c3_500_5	5	500	ABEL-Spline (z=10)	0.899 ± 0.023	0.099 ± 0.028	0.229 ± 0.027	0.133 ± 0.002
617_fri_c3_500_5	5	500	Spline ANN (z=1)	0.535 ± 0.110	0.470 ± 0.168	0.490 ± 0.075	0.481 ± 0.007
617_fri_c3_500_5	5	500	Spline ANN (z=2)	0.834 ± 0.042	0.167 ± 0.060	0.293 ± 0.046	0.275 ± 0.004
617_fri_c3_500_5	5	500	Spline ANN (z=4)	0.892 ± 0.027	0.107 ± 0.034	0.237 ± 0.037	0.210 ± 0.005
617_fri_c3_500_5	5	500	Spline ANN (z=8)	0.884 ± 0.026	0.114 ± 0.034	0.245 ± 0.033	0.191 ± 0.004
617_fri_c3_500_5	5	500	Spline ANN (z=10)	0.877 ± 0.024	0.122 ± 0.034	0.254 ± 0.027	0.191 ± 0.003
617_fri_c3_500_5	5	500	Lookup Table (z=1)	−0.025 ± 0.031	1.016 ± 0.200	0.789 ± 0.080	0.786 ± 0.004
617_fri_c3_500_5	5	500	Lookup Table (z=2)	0.247 ± 0.093	0.752 ± 0.195	0.661 ± 0.077	0.634 ± 0.005
617_fri_c3_500_5	5	500	Lookup Table (z=4)	0.427 ± 0.055	0.571 ± 0.136	0.560 ± 0.061	0.489 ± 0.007
617_fri_c3_500_5	5	500	Lookup Table (z=8)	0.171 ± 0.048	0.822 ± 0.165	0.696 ± 0.081	0.490 ± 0.005
617_fri_c3_500_5	5	500	Lookup Table (z=10)	0.096 ± 0.054	0.898 ± 0.194	0.738 ± 0.083	0.486 ± 0.005
631_fri_c1_500_5	5	500	Linear Model	0.228 ± 0.078	0.748 ± 0.079	0.728 ± 0.049	0.712 ± 0.005
631_fri_c1_500_5	5	500	Wide ReLU ANN	0.941 ± 0.027	0.057 ± 0.025	0.162 ± 0.015	0.136 ± 0.005
631_fri_c1_500_5	5	500	Deep ReLU ANN	0.902 ± 0.059	0.091 ± 0.041	0.221 ± 0.040	0.151 ± 0.018
631_fri_c1_500_5	5	500	ABEL-Spline (z=1)	0.865 ± 0.047	0.133 ± 0.049	0.249 ± 0.034	0.235 ± 0.005
631_fri_c1_500_5	5	500	ABEL-Spline (z=2)	0.884 ± 0.049	0.114 ± 0.049	0.223 ± 0.037	0.201 ± 0.004
631_fri_c1_500_5	5	500	ABEL-Spline (z=4)	0.883 ± 0.041	0.115 ± 0.042	0.228 ± 0.035	0.183 ± 0.003
631_fri_c1_500_5	5	500	ABEL-Spline (z=8)	0.885 ± 0.032	0.113 ± 0.034	0.235 ± 0.031	0.161 ± 0.003
631_fri_c1_500_5	5	500	ABEL-Spline (z=10)	0.876 ± 0.030	0.121 ± 0.031	0.250 ± 0.030	0.154 ± 0.003
631_fri_c1_500_5	5	500	Spline ANN (z=1)	0.570 ± 0.104	0.411 ± 0.072	0.484 ± 0.042	0.468 ± 0.008
631_fri_c1_500_5	5	500	Spline ANN (z=2)	0.843 ± 0.039	0.151 ± 0.035	0.277 ± 0.034	0.259 ± 0.004
631_fri_c1_500_5	5	500	Spline ANN (z=4)	0.878 ± 0.034	0.119 ± 0.036	0.240 ± 0.034	0.212 ± 0.003
631_fri_c1_500_5	5	500	Spline ANN (z=8)	0.879 ± 0.033	0.118 ± 0.034	0.241 ± 0.032	0.199 ± 0.004
631_fri_c1_500_5	5	500	Spline ANN (z=10)	0.870 ± 0.036	0.127 ± 0.035	0.254 ± 0.032	0.200 ± 0.004
631_fri_c1_500_5	5	500	Lookup Table (z=1)	−0.046 ± 0.051	1.023 ± 0.145	0.819 ± 0.053	0.814 ± 0.001
631_fri_c1_500_5	5	500	Lookup Table (z=2)	0.380 ± 0.052	0.603 ± 0.079	0.624 ± 0.055	0.598 ± 0.005
631_fri_c1_500_5	5	500	Lookup Table (z=4)	0.244 ± 0.063	0.743 ± 0.133	0.678 ± 0.061	0.519 ± 0.005
631_fri_c1_500_5	5	500	Lookup Table (z=8)	0.041 ± 0.057	0.940 ± 0.152	0.776 ± 0.069	0.514 ± 0.003
631_fri_c1_500_5	5	500	Lookup Table (z=10)	−0.014 ± 0.043	0.992 ± 0.146	0.812 ± 0.061	0.516 ± 0.002

Continued on next page

Table C.2 – Continued from previous page

Dataset	n	N	Model	R^2	T. MSE	T. MAE	Tr. MAE
649_fri_c0_500_5	5	500	Linear Model	0.664 ± 0.097	0.323 ± 0.095	0.431 ± 0.069	0.425 ± 0.032
649_fri_c0_500_5	5	500	Wide ReLU ANN	0.934 ± 0.026	0.061 ± 0.015	0.189 ± 0.017	0.152 ± 0.004
649_fri_c0_500_5	5	500	Deep ReLU ANN	0.900 ± 0.026	0.095 ± 0.022	0.240 ± 0.022	0.172 ± 0.012
649_fri_c0_500_5	5	500	ABEL-Spline (z=1)	0.887 ± 0.053	0.104 ± 0.034	0.239 ± 0.024	0.225 ± 0.004
649_fri_c0_500_5	5	500	ABEL-Spline (z=2)	0.886 ± 0.049	0.106 ± 0.035	0.245 ± 0.028	0.214 ± 0.003
649_fri_c0_500_5	5	500	ABEL-Spline (z=4)	0.881 ± 0.043	0.111 ± 0.031	0.253 ± 0.026	0.199 ± 0.004
649_fri_c0_500_5	5	500	ABEL-Spline (z=8)	0.872 ± 0.044	0.119 ± 0.036	0.264 ± 0.032	0.177 ± 0.003
649_fri_c0_500_5	5	500	ABEL-Spline (z=10)	0.867 ± 0.045	0.124 ± 0.038	0.269 ± 0.029	0.165 ± 0.004
649_fri_c0_500_5	5	500	Spline ANN (z=1)	0.852 ± 0.047	0.140 ± 0.036	0.289 ± 0.029	0.279 ± 0.004
649_fri_c0_500_5	5	500	Spline ANN (z=2)	0.873 ± 0.043	0.119 ± 0.028	0.268 ± 0.026	0.250 ± 0.004
649_fri_c0_500_5	5	500	Spline ANN (z=4)	0.874 ± 0.043	0.117 ± 0.028	0.267 ± 0.028	0.235 ± 0.004
649_fri_c0_500_5	5	500	Spline ANN (z=8)	0.863 ± 0.043	0.128 ± 0.032	0.276 ± 0.035	0.224 ± 0.006
649_fri_c0_500_5	5	500	Spline ANN (z=10)	0.849 ± 0.044	0.141 ± 0.034	0.290 ± 0.034	0.223 ± 0.005
649_fri_c0_500_5	5	500	Lookup Table (z=1)	−0.038 ± 0.039	1.016 ± 0.235	0.816 ± 0.114	0.808 ± 0.003
649_fri_c0_500_5	5	500	Lookup Table (z=2)	0.404 ± 0.090	0.583 ± 0.162	0.611 ± 0.085	0.573 ± 0.004
649_fri_c0_500_5	5	500	Lookup Table (z=4)	0.116 ± 0.044	0.866 ± 0.204	0.747 ± 0.108	0.526 ± 0.004
649_fri_c0_500_5	5	500	Lookup Table (z=8)	−0.008 ± 0.039	0.986 ± 0.228	0.800 ± 0.111	0.511 ± 0.003
649_fri_c0_500_5	5	500	Lookup Table (z=10)	−0.028 ± 0.035	1.007 ± 0.236	0.808 ± 0.115	0.514 ± 0.003
599_fri_c2_1000_5	5	1000	Linear Model	0.301 ± 0.052	0.694 ± 0.064	0.702 ± 0.037	0.696 ± 0.006
599_fri_c2_1000_5	5	1000	Wide ReLU ANN	0.957 ± 0.017	0.044 ± 0.019	0.144 ± 0.015	0.126 ± 0.006
599_fri_c2_1000_5	5	1000	Deep ReLU ANN	0.957 ± 0.010	0.043 ± 0.012	0.157 ± 0.014	0.125 ± 0.006
599_fri_c2_1000_5	5	1000	ABEL-Spline (z=1)	0.907 ± 0.027	0.093 ± 0.033	0.206 ± 0.018	0.198 ± 0.002
599_fri_c2_1000_5	5	1000	ABEL-Spline (z=2)	0.928 ± 0.015	0.073 ± 0.020	0.191 ± 0.017	0.178 ± 0.001
599_fri_c2_1000_5	5	1000	ABEL-Spline (z=4)	0.929 ± 0.017	0.071 ± 0.021	0.189 ± 0.017	0.161 ± 0.002
599_fri_c2_1000_5	5	1000	ABEL-Spline (z=8)	0.921 ± 0.015	0.079 ± 0.020	0.201 ± 0.015	0.150 ± 0.002
599_fri_c2_1000_5	5	1000	ABEL-Spline (z=10)	0.918 ± 0.016	0.082 ± 0.021	0.205 ± 0.017	0.143 ± 0.001
599_fri_c2_1000_5	5	1000	Spline ANN (z=1)	0.752 ± 0.025	0.248 ± 0.039	0.376 ± 0.028	0.370 ± 0.002
599_fri_c2_1000_5	5	1000	Spline ANN (z=2)	0.916 ± 0.018	0.085 ± 0.024	0.206 ± 0.018	0.196 ± 0.001
599_fri_c2_1000_5	5	1000	Spline ANN (z=4)	0.922 ± 0.018	0.078 ± 0.023	0.196 ± 0.021	0.179 ± 0.002
599_fri_c2_1000_5	5	1000	Spline ANN (z=8)	0.918 ± 0.016	0.083 ± 0.021	0.204 ± 0.018	0.170 ± 0.002
599_fri_c2_1000_5	5	1000	Spline ANN (z=10)	0.916 ± 0.016	0.085 ± 0.022	0.207 ± 0.018	0.167 ± 0.002
599_fri_c2_1000_5	5	1000	Lookup Table (z=1)	−0.061 ± 0.049	1.054 ± 0.083	0.824 ± 0.045	0.823 ± 0.003
599_fri_c2_1000_5	5	1000	Lookup Table (z=2)	0.484 ± 0.082	0.512 ± 0.080	0.559 ± 0.040	0.537 ± 0.005
599_fri_c2_1000_5	5	1000	Lookup Table (z=4)	0.637 ± 0.037	0.361 ± 0.045	0.467 ± 0.026	0.350 ± 0.004
599_fri_c2_1000_5	5	1000	Lookup Table (z=8)	0.206 ± 0.052	0.790 ± 0.075	0.727 ± 0.042	0.402 ± 0.003
599_fri_c2_1000_5	5	1000	Lookup Table (z=10)	0.062 ± 0.024	0.933 ± 0.070	0.807 ± 0.037	0.412 ± 0.002
609_fri_c0_1000_5	5	1000	Linear Model	0.720 ± 0.041	0.276 ± 0.047	0.404 ± 0.032	0.400 ± 0.003
609_fri_c0_1000_5	5	1000	Wide ReLU ANN	0.950 ± 0.003	0.050 ± 0.006	0.178 ± 0.012	0.140 ± 0.003
609_fri_c0_1000_5	5	1000	Deep ReLU ANN	0.920 ± 0.017	0.078 ± 0.012	0.221 ± 0.019	0.164 ± 0.008
609_fri_c0_1000_5	5	1000	ABEL-Spline (z=1)	0.893 ± 0.031	0.104 ± 0.027	0.241 ± 0.028	0.229 ± 0.003
609_fri_c0_1000_5	5	1000	ABEL-Spline (z=2)	0.901 ± 0.026	0.097 ± 0.023	0.231 ± 0.024	0.214 ± 0.005
609_fri_c0_1000_5	5	1000	ABEL-Spline (z=4)	0.895 ± 0.027	0.102 ± 0.024	0.237 ± 0.026	0.207 ± 0.003
609_fri_c0_1000_5	5	1000	ABEL-Spline (z=8)	0.887 ± 0.027	0.111 ± 0.026	0.251 ± 0.029	0.195 ± 0.003
609_fri_c0_1000_5	5	1000	ABEL-Spline (z=10)	0.881 ± 0.028	0.117 ± 0.027	0.259 ± 0.029	0.191 ± 0.003
609_fri_c0_1000_5	5	1000	Spline ANN (z=1)	0.872 ± 0.026	0.125 ± 0.025	0.274 ± 0.028	0.267 ± 0.004
609_fri_c0_1000_5	5	1000	Spline ANN (z=2)	0.879 ± 0.029	0.118 ± 0.025	0.261 ± 0.028	0.249 ± 0.004
609_fri_c0_1000_5	5	1000	Spline ANN (z=4)	0.877 ± 0.029	0.120 ± 0.026	0.262 ± 0.026	0.241 ± 0.003
609_fri_c0_1000_5	5	1000	Spline ANN (z=8)	0.875 ± 0.027	0.122 ± 0.025	0.264 ± 0.028	0.232 ± 0.003
609_fri_c0_1000_5	5	1000	Spline ANN (z=10)	0.873 ± 0.027	0.124 ± 0.024	0.267 ± 0.028	0.227 ± 0.003
609_fri_c0_1000_5	5	1000	Lookup Table (z=1)	−0.015 ± 0.023	1.004 ± 0.122	0.812 ± 0.052	0.810 ± 0.001
609_fri_c0_1000_5	5	1000	Lookup Table (z=2)	0.517 ± 0.046	0.476 ± 0.050	0.552 ± 0.034	0.529 ± 0.005
609_fri_c0_1000_5	5	1000	Lookup Table (z=4)	0.319 ± 0.065	0.673 ± 0.092	0.635 ± 0.057	0.403 ± 0.005
609_fri_c0_1000_5	5	1000	Lookup Table (z=8)	0.049 ± 0.023	0.941 ± 0.111	0.776 ± 0.053	0.404 ± 0.001
609_fri_c0_1000_5	5	1000	Lookup Table (z=10)	−0.009 ± 0.027	0.999 ± 0.121	0.807 ± 0.052	0.408 ± 0.002
612_fri_c1_1000_5	5	1000	Linear Model	0.247 ± 0.059	0.747 ± 0.090	0.721 ± 0.051	0.714 ± 0.003
612_fri_c1_1000_5	5	1000	Wide ReLU ANN	0.952 ± 0.016	0.047 ± 0.013	0.148 ± 0.012	0.126 ± 0.004
612_fri_c1_1000_5	5	1000	Deep ReLU ANN	0.951 ± 0.007	0.049 ± 0.009	0.167 ± 0.016	0.134 ± 0.009
612_fri_c1_1000_5	5	1000	ABEL-Spline (z=1)	0.897 ± 0.039	0.100 ± 0.034	0.212 ± 0.018	0.205 ± 0.002
612_fri_c1_1000_5	5	1000	ABEL-Spline (z=2)	0.921 ± 0.021	0.077 ± 0.019	0.192 ± 0.013	0.180 ± 0.003
612_fri_c1_1000_5	5	1000	ABEL-Spline (z=4)	0.934 ± 0.014	0.065 ± 0.013	0.182 ± 0.011	0.160 ± 0.004
612_fri_c1_1000_5	5	1000	ABEL-Spline (z=8)	0.927 ± 0.016	0.072 ± 0.015	0.192 ± 0.014	0.151 ± 0.003
612_fri_c1_1000_5	5	1000	ABEL-Spline (z=10)	0.922 ± 0.017	0.077 ± 0.014	0.201 ± 0.014	0.147 ± 0.002

Continued on next page

Table C.2 – Continued from previous page

Dataset	n	N	Model	R^2	T. MSE	T. MAE	Tr. MAE
612_fri_c1_1000_5	5	1000	Spline ANN (z=1)	0.747 ± 0.019	0.250 ± 0.029	0.368 ± 0.028	0.362 ± 0.003
612_fri_c1_1000_5	5	1000	Spline ANN (z=2)	0.907 ± 0.026	0.091 ± 0.022	0.215 ± 0.018	0.207 ± 0.002
612_fri_c1_1000_5	5	1000	Spline ANN (z=4)	0.911 ± 0.019	0.088 ± 0.017	0.209 ± 0.014	0.192 ± 0.002
612_fri_c1_1000_5	5	1000	Spline ANN (z=8)	0.912 ± 0.018	0.086 ± 0.016	0.209 ± 0.014	0.180 ± 0.002
612_fri_c1_1000_5	5	1000	Spline ANN (z=10)	0.912 ± 0.017	0.087 ± 0.015	0.212 ± 0.011	0.177 ± 0.002
612_fri_c1_1000_5	5	1000	Lookup Table (z=1)	−0.028 ± 0.036	1.021 ± 0.117	0.816 ± 0.055	0.814 ± 0.002
612_fri_c1_1000_5	5	1000	Lookup Table (z=2)	0.430 ± 0.076	0.562 ± 0.069	0.601 ± 0.044	0.574 ± 0.005
612_fri_c1_1000_5	5	1000	Lookup Table (z=4)	0.525 ± 0.058	0.469 ± 0.060	0.522 ± 0.040	0.383 ± 0.007
612_fri_c1_1000_5	5	1000	Lookup Table (z=8)	0.169 ± 0.054	0.820 ± 0.064	0.717 ± 0.030	0.397 ± 0.004
612_fri_c1_1000_5	5	1000	Lookup Table (z=10)	0.039 ± 0.017	0.952 ± 0.088	0.791 ± 0.045	0.405 ± 0.001
628_fri_c3_1000_5	5	1000	Linear Model	0.200 ± 0.076	0.788 ± 0.127	0.727 ± 0.053	0.718 ± 0.004
628_fri_c3_1000_5	5	1000	Wide ReLU ANN	0.945 ± 0.037	0.054 ± 0.037	0.151 ± 0.021	0.134 ± 0.004
628_fri_c3_1000_5	5	1000	Deep ReLU ANN	0.957 ± 0.013	0.042 ± 0.011	0.151 ± 0.015	0.136 ± 0.010
628_fri_c3_1000_5	5	1000	ABEL-Spline (z=1)	0.888 ± 0.058	0.110 ± 0.061	0.210 ± 0.028	0.204 ± 0.003
628_fri_c3_1000_5	5	1000	ABEL-Spline (z=2)	0.914 ± 0.022	0.084 ± 0.022	0.195 ± 0.018	0.183 ± 0.003
628_fri_c3_1000_5	5	1000	ABEL-Spline (z=4)	0.916 ± 0.020	0.082 ± 0.019	0.192 ± 0.017	0.168 ± 0.001
628_fri_c3_1000_5	5	1000	ABEL-Spline (z=8)	0.911 ± 0.021	0.086 ± 0.017	0.199 ± 0.016	0.153 ± 0.002
628_fri_c3_1000_5	5	1000	ABEL-Spline (z=10)	0.904 ± 0.023	0.093 ± 0.018	0.209 ± 0.017	0.150 ± 0.002
628_fri_c3_1000_5	5	1000	Spline ANN (z=1)	0.733 ± 0.029	0.263 ± 0.041	0.391 ± 0.033	0.382 ± 0.005
628_fri_c3_1000_5	5	1000	Spline ANN (z=2)	0.901 ± 0.032	0.096 ± 0.032	0.213 ± 0.021	0.205 ± 0.003
628_fri_c3_1000_5	5	1000	Spline ANN (z=4)	0.908 ± 0.025	0.089 ± 0.024	0.204 ± 0.020	0.187 ± 0.003
628_fri_c3_1000_5	5	1000	Spline ANN (z=8)	0.906 ± 0.024	0.091 ± 0.020	0.210 ± 0.019	0.179 ± 0.002
628_fri_c3_1000_5	5	1000	Spline ANN (z=10)	0.903 ± 0.024	0.094 ± 0.020	0.216 ± 0.018	0.177 ± 0.002
628_fri_c3_1000_5	5	1000	Lookup Table (z=1)	−0.042 ± 0.039	1.027 ± 0.146	0.793 ± 0.055	0.792 ± 0.003
628_fri_c3_1000_5	5	1000	Lookup Table (z=2)	0.348 ± 0.139	0.646 ± 0.175	0.603 ± 0.065	0.588 ± 0.006
628_fri_c3_1000_5	5	1000	Lookup Table (z=4)	0.670 ± 0.049	0.324 ± 0.056	0.431 ± 0.036	0.367 ± 0.009
628_fri_c3_1000_5	5	1000	Lookup Table (z=8)	0.440 ± 0.051	0.551 ± 0.083	0.566 ± 0.046	0.363 ± 0.004
628_fri_c3_1000_5	5	1000	Lookup Table (z=10)	0.247 ± 0.056	0.740 ± 0.090	0.661 ± 0.040	0.372 ± 0.005
706_sleuth_case1202	6	93	Linear Model	−0.111 ± 0.503	1.053 ± 0.794	0.787 ± 0.256	0.753 ± 0.156
706_sleuth_case1202	6	93	Wide ReLU ANN	0.688 ± 0.168	0.260 ± 0.182	0.342 ± 0.119	0.214 ± 0.014
706_sleuth_case1202	6	93	Deep ReLU ANN	0.571 ± 0.208	0.375 ± 0.286	0.408 ± 0.145	0.190 ± 0.026
706_sleuth_case1202	6	93	ABEL-Spline (z=1)	0.509 ± 0.138	0.469 ± 0.299	0.464 ± 0.136	0.388 ± 0.013
706_sleuth_case1202	6	93	ABEL-Spline (z=2)	0.502 ± 0.117	0.485 ± 0.313	0.491 ± 0.137	0.360 ± 0.011
706_sleuth_case1202	6	93	ABEL-Spline (z=4)	0.403 ± 0.252	0.544 ± 0.373	0.526 ± 0.128	0.319 ± 0.015
706_sleuth_case1202	6	93	ABEL-Spline (z=8)	0.185 ± 0.278	0.749 ± 0.548	0.626 ± 0.164	0.246 ± 0.016
706_sleuth_case1202	6	93	ABEL-Spline (z=10)	0.145 ± 0.288	0.789 ± 0.571	0.654 ± 0.174	0.221 ± 0.017
706_sleuth_case1202	6	93	Spline ANN (z=1)	0.216 ± 0.210	0.862 ± 0.703	0.613 ± 0.231	0.563 ± 0.010
706_sleuth_case1202	6	93	Spline ANN (z=2)	0.252 ± 0.170	0.812 ± 0.656	0.605 ± 0.211	0.519 ± 0.009
706_sleuth_case1202	6	93	Spline ANN (z=4)	0.259 ± 0.171	0.806 ± 0.679	0.602 ± 0.208	0.499 ± 0.012
706_sleuth_case1202	6	93	Spline ANN (z=8)	0.204 ± 0.160	0.854 ± 0.740	0.618 ± 0.201	0.474 ± 0.009
706_sleuth_case1202	6	93	Spline ANN (z=10)	0.187 ± 0.197	0.869 ± 0.759	0.625 ± 0.210	0.462 ± 0.009
706_sleuth_case1202	6	93	Lookup Table (z=1)	−0.086 ± 0.074	1.104 ± 0.785	0.785 ± 0.247	0.767 ± 0.008
706_sleuth_case1202	6	93	Lookup Table (z=2)	0.077 ± 0.113	0.944 ± 0.689	0.734 ± 0.235	0.662 ± 0.011
706_sleuth_case1202	6	93	Lookup Table (z=4)	−0.020 ± 0.115	1.035 ± 0.726	0.797 ± 0.252	0.650 ± 0.005
706_sleuth_case1202	6	93	Lookup Table (z=8)	−0.067 ± 0.080	1.049 ± 0.700	0.807 ± 0.230	0.641 ± 0.006
706_sleuth_case1202	6	93	Lookup Table (z=10)	−0.080 ± 0.048	1.067 ± 0.716	0.815 ± 0.237	0.642 ± 0.006
665_sleuth_case2002	6	147	Linear Model	−0.174 ± 0.379	1.025 ± 0.351	0.768 ± 0.158	0.734 ± 0.051
665_sleuth_case2002	6	147	Wide ReLU ANN	0.238 ± 0.326	0.678 ± 0.320	0.562 ± 0.139	0.446 ± 0.013
665_sleuth_case2002	6	147	Deep ReLU ANN	0.192 ± 0.325	0.734 ± 0.378	0.584 ± 0.152	0.423 ± 0.020
665_sleuth_case2002	6	147	ABEL-Spline (z=1)	0.278 ± 0.240	0.657 ± 0.308	0.539 ± 0.132	0.508 ± 0.015
665_sleuth_case2002	6	147	ABEL-Spline (z=2)	0.287 ± 0.244	0.648 ± 0.310	0.536 ± 0.142	0.501 ± 0.012
665_sleuth_case2002	6	147	ABEL-Spline (z=4)	0.251 ± 0.249	0.671 ± 0.287	0.573 ± 0.148	0.473 ± 0.013
665_sleuth_case2002	6	147	ABEL-Spline (z=8)	0.261 ± 0.230	0.672 ± 0.319	0.575 ± 0.144	0.431 ± 0.012
665_sleuth_case2002	6	147	ABEL-Spline (z=10)	0.254 ± 0.227	0.668 ± 0.283	0.587 ± 0.122	0.420 ± 0.012
665_sleuth_case2002	6	147	Spline ANN (z=1)	0.125 ± 0.184	0.785 ± 0.259	0.675 ± 0.117	0.649 ± 0.007
665_sleuth_case2002	6	147	Spline ANN (z=2)	0.121 ± 0.191	0.785 ± 0.248	0.676 ± 0.108	0.637 ± 0.009
665_sleuth_case2002	6	147	Spline ANN (z=4)	0.119 ± 0.191	0.784 ± 0.243	0.678 ± 0.111	0.622 ± 0.010
665_sleuth_case2002	6	147	Spline ANN (z=8)	0.113 ± 0.166	0.791 ± 0.233	0.691 ± 0.102	0.600 ± 0.011
665_sleuth_case2002	6	147	Spline ANN (z=10)	0.099 ± 0.184	0.798 ± 0.230	0.692 ± 0.096	0.595 ± 0.011
665_sleuth_case2002	6	147	Lookup Table (z=1)	−0.142 ± 0.183	1.027 ± 0.294	0.777 ± 0.129	0.772 ± 0.006
665_sleuth_case2002	6	147	Lookup Table (z=2)	−0.060 ± 0.205	0.948 ± 0.282	0.754 ± 0.113	0.697 ± 0.007
665_sleuth_case2002	6	147	Lookup Table (z=4)	−0.122 ± 0.213	1.000 ± 0.288	0.777 ± 0.111	0.675 ± 0.012

Continued on next page

Table C.2 – Continued from previous page

Dataset	n	N	Model	R^2	T. MSE	T. MAE	Tr. MAE
665_sleuth_case2002	6	147	Lookup Table (z=8)	-0.123 ± 0.214	1.010 ± 0.314	0.783 ± 0.131	0.649 ± 0.010
665_sleuth_case2002	6	147	Lookup Table (z=10)	-0.121 ± 0.177	1.002 ± 0.275	0.777 ± 0.118	0.644 ± 0.009
230_machine_cpu	6	209	Linear Model	0.380 ± 0.256	0.789 ± 1.342	0.338 ± 0.289	0.299 ± 0.033
230_machine_cpu	6	209	Wide ReLU ANN	0.534 ± 0.350	0.358 ± 0.616	0.251 ± 0.207	0.102 ± 0.010
230_machine_cpu	6	209	Deep ReLU ANN	0.634 ± 0.264	0.328 ± 0.584	0.236 ± 0.184	0.106 ± 0.012
230_machine_cpu	6	209	ABEL-Spline (z=1)	0.573 ± 0.248	0.362 ± 0.678	0.230 ± 0.176	0.166 ± 0.006
230_machine_cpu	6	209	ABEL-Spline (z=2)	0.576 ± 0.268	0.345 ± 0.656	0.234 ± 0.176	0.143 ± 0.008
230_machine_cpu	6	209	ABEL-Spline (z=4)	0.515 ± 0.354	0.271 ± 0.475	0.237 ± 0.168	0.111 ± 0.008
230_machine_cpu	6	209	ABEL-Spline (z=8)	0.349 ± 0.767	0.234 ± 0.426	0.234 ± 0.155	0.086 ± 0.006
230_machine_cpu	6	209	ABEL-Spline (z=10)	0.284 ± 0.857	0.284 ± 0.556	0.255 ± 0.192	0.082 ± 0.005
230_machine_cpu	6	209	Spline ANN (z=1)	0.509 ± 0.299	0.608 ± 1.104	0.273 ± 0.233	0.226 ± 0.008
230_machine_cpu	6	209	Spline ANN (z=2)	0.494 ± 0.250	0.596 ± 1.087	0.277 ± 0.230	0.213 ± 0.007
230_machine_cpu	6	209	Spline ANN (z=4)	0.512 ± 0.233	0.631 ± 1.171	0.283 ± 0.240	0.204 ± 0.007
230_machine_cpu	6	209	Spline ANN (z=8)	0.496 ± 0.212	0.649 ± 1.217	0.292 ± 0.238	0.189 ± 0.005
230_machine_cpu	6	209	Spline ANN (z=10)	0.433 ± 0.245	0.661 ± 1.238	0.299 ± 0.240	0.184 ± 0.005
230_machine_cpu	6	209	Lookup Table (z=1)	-0.200 ± 0.152	1.480 ± 2.309	0.531 ± 0.468	0.490 ± 0.016
230_machine_cpu	6	209	Lookup Table (z=2)	0.239 ± 0.255	0.973 ± 1.654	0.392 ± 0.336	0.301 ± 0.007
230_machine_cpu	6	209	Lookup Table (z=4)	0.180 ± 0.239	1.079 ± 1.816	0.419 ± 0.371	0.307 ± 0.008
230_machine_cpu	6	209	Lookup Table (z=8)	-0.925 ± 1.304	1.293 ± 2.015	0.569 ± 0.374	0.341 ± 0.013
230_machine_cpu	6	209	Lookup Table (z=10)	-1.045 ± 1.416	1.296 ± 1.995	0.574 ± 0.370	0.339 ± 0.013