Training neural networks for informal road extraction

By

Abraham Johannes Wannenburg

17099481

Supervisoir: Gaonyalelwe Maribe

Co-supervisor: Inger Fabris-Rotelli

Submitted in partial fulfillment of the requirements for the degree MSc in Advanced Data Analytics
in the Faculty of Natural and Agricultural Sciences
University of Pretoria

November 2022

Declaration

I, *Abraham Johannes Wannenburg,* declare that this mini-dissertation, which I hereby submit for the degree MSc in Advanced Data Analytics at the University of Pretoria, is my own work and has not been previously submitted at this or any other tertiary institution.

Signature: [illegible]

Date: 11-02-2023

Summary

Roads found in informal settlements arise out of convenience are often not recorded or maintained by authorities. This may cause issues with service delivery, sustainable development and crisis mitigation, including COVID-19. Therefore, the aim of extracting informal roads from remote sensing images is of importance. Existing techniques aimed at the extraction of formal roads are not completely suitable for the problem due to the complex physical and spectral properties that informal roads pose. The only existing approaches for informal roads, namely [62, 82], do not consider neural networks as a solution. Neural networks show promise in overcoming these complexities due to the way they learn through training. They require a large amount of data to learn, which is currently not available due to the expensive and time-consuming nature of collecting such data sets. A problem that has been shown to come up when working with computer vision data sets is data set bias. Data set bias adds to the already existing problem of machine learning algorithms called overfitting. This paper implements a neural network developed for formal roads to extract informal roads from three data sets digitised by this research group to investigate the presence of data set bias. Three different geological areas from South Africa are digitised. We implement the GAN-UNet model that obtained the highest F1-score in a 2020 review paper [1] on the state-of-the-art deep learning models used to extract formal roads. We present quantitative and qualitative results that concludes the presence of data set bias. We then present further work that can be done to create a robust training data set for the development of an automatic informal road extraction model.

Acknowledgements

I am grateful to be a tutor in the Department of Statistics, an intern at the HUB at the Department of Statistics as well as the partial financial support of the Data Science Africa 2021 Project (PI: Inger Fabris-Rotelli), the Centre for Artificial Intelligence Research and the CoE-MaSS grant (2022 grant: ref #2022-018-MAC-Road), which made it possible for me to fund my masters studies. I also acknowledge the support, guidance and help from both of my supervisors as well as the different digitiser students who digitised the data that the model uses for training and testing. This research was approved under ethics application NAS200/2021.

Contents

List of Figures

Chapter 1

Introduction

The South African census[1] defines an informal settlement as a settlement on land that was not planned, has not been proclaimed to be residential, and mostly has informal dwellings on it and is residential in nature. Most residents have to travel out of informal settlements for school, work or health services [33][2].

When we consider the opposite of informal settlement, urban settlements, we notice a need that arises, namely to expand and develop road networks when there is an increase in population [98, 71]. Therefore, governments invest heavily in the development and expansion of the transportation infrastructures in urban settlements to prevent traffic congestion, overcrowding and the ease of transporting goods and services [98]. Similarly, informal settlements face this same problem of congestion and overcrowding when their population increases. The difference between them lies in the development of transportation infrastructures, therefore, the inhabitants are forced to do this development themselves as the settlement grows in population.

How the transportation infrastructures are created such as the road networks is what poses the problems to extracting these roads. The road networks in informal settlements are informal roads or a combination of formal and informal roads. Informal roads are roads that are created without government approval or knowledge and are neither maintained, nor appear in any official databases or maps. This means that governments are not regularly investing in road networks for informal settlements. Residents create their own road networks to accommodate their need for efficient transport by walking, cycling or driving to points of interest. This presents a problem for service delivery, medical services, criminal activity prevention and environmental emergencies, since the relevant teams will not know the most efficient route to the destination. Information on road networks in informal settlements will also make it easier to plan the best placement for emergency support facilities in informal settlements. In the South African context this

[1]South African census, http://www.statssa.gov.za/census/census_2001/concepts_definitions/concepts_definitions.pdf
[2]Note that parts of this section are covered in the published article [19] that I was a co-author on.

is a great concern since the population in informal settlements is growing at a faster pace than elsewhere [73]. The pace of population growth also causes the informal roads to change more rapidly, therefore real-time road data is critical for service delivery, emergency responses and crisis mitigation measures. An automatic road extraction model from remote sensing imagery stands as a possible solution to finding these roads. In literature, two terms can be found with regard to the finding of roads, namely extraction and detection. Mirriam-Webster[3] defines detection as "the action of detecting", where detecting is defined as "to discover or determine the existence, presence or fact of". They also define extraction as "the act or process of extracting something", where extract is defined as "to draw forth" or "to withdraw by a physical or chemical process". In literature, we have found that the keyword "road extraction" results in articles where deep learning models are used to find roads from high-resolution imagery [1]. Whereas the keyword "road detection" mostly results in articles where models are implemented in self-driving cars [40] and driver assistance systems [20] where a camera of a certain design is used to find the centerline and boundary lines of roads from the perspective of a vehicle. There are exceptions such as [59] where roads are also found from high-resolution imagery.

This mini-dissertation focuses on training a deep learning model, a neural network, to automatically extract informal road networks and to investigate the presence of bias in the data sets used to train the model. The inputs this model uses to train are large data sets of digitised road networks from remote sensing images. Previous methods on the extraction of informal roads [62, 82] did not use deep learning models. Nobrega et al.'s technique uses propriety software, Definiens eCognition, to segment the images and to do the object classification [62]. Therefore, the technique is inaccessible for applications where funding is very limited since it requires a paid license. Their technique also requires many user inputs and tuning parameters to use object information to extract the informal roads from the high-resolution satellite imagery, making it a non-automatic technique. Thiede et al. [82] used Binary Partition Trees (BPT) proposed by [44] to extract informal roads as the model has adaptable parameters that make the model flexible. The model was able to identify the different types of roads, but the accuracy varied depending on the spectral and physical attributes of informal roads.

Figure 1.1 is a snapshot of an area near the Khayelitsha Township in the Western Cape in South Africa. Differences can be seen between the urban roads, on the left hand side, and informal roads, on the right hand side. The challenges informal roads pose to road extraction methods are listed and highlighted on the figure from A to D. A showcases how informal roads vary in length and width, and B how informal roads are not planned so they do not appear in regular patterns. C shows how informal roads' colouring are the same as the area around them and D shows how informal roads' colouring are not constant.

Deep learning models such as neural networks were suggested as a suitable automatic image segmentation method by the authors in [17] where an environment changes frequently due to their relative ease of

[3]Mirriam-Webster, https://www.merriam-webster.com/,accessed: 20/10/2022

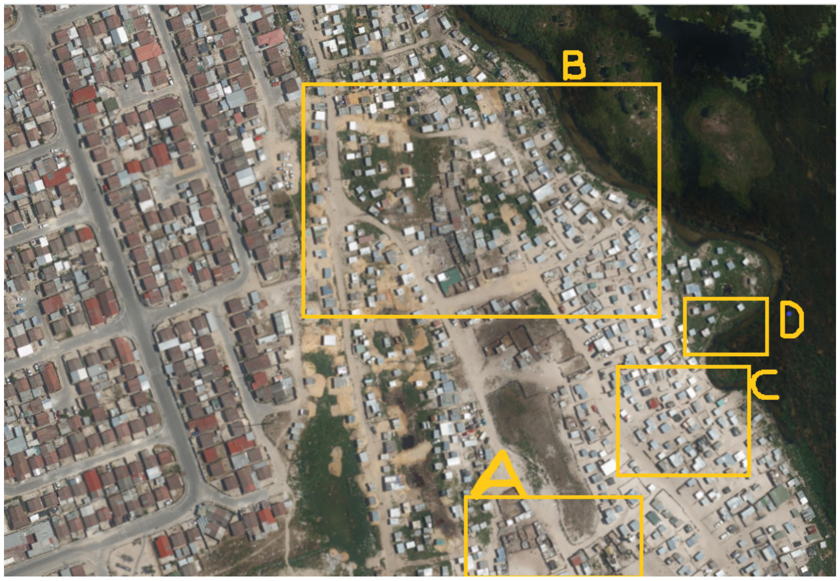


Figure 1.1: Aerial imagery of Khayelitsha Township in Western Cape, South Africa, obtained through a WMS service published by the City of Cape Town[4].

usage and applicability to a wide variety of problems. The literature on automatic road extraction using deep learning like neural networks (NN) has come a long way and developed into strong implementations. The base concept and vision these methods aim to achieve, is to simulate the visual capabilities humans have and teach an algorithm to have these same capabilities. The authors in [63] state that NN models are examples of methods that can emulate aspects of the human information processing system and get automatic or real-time output. Output in this case is an image mask that shows where the roads are on the given image. David Marr was a neuroscientist that joined the MIT Artificial Intelligence Laboratory in the mid-1970s that worked towards this vision. He published a few revolutionary articles on how to describe vision in [51, 52, 53, 54, 55], where he put forward a computational model that can be used to study our visual system [79].

The model implemented in this mini-dissertation attempts to simulate the ability of the human visual system to distinguish between and classify objects. This ability refers to segmentation in a formal mathematical way. Pal & Pal [63] mention that the first important and essential step in low-level vision is segmentation and define segmentation as the process of partitioning an image into some non-intersecting regions such that each region is homogeneous and the union of no two adjacent regions is homogeneous. This concept of segmentation reflects what could also be defined as classification. Since road extraction can be seen as the classification task between road and non-road areas at pixel-level [87], classification is the bridge between the concept of image segmentation and road extraction.

Pal & Pal [63] presented an early review paper in 1993 that critically reviews techniques for the segmen-

[4]Imagery was obtained from the following website, https://citymaps.capetown.gov.za/agsext1/rest/services/Aerial_Photography_Cached/AP_2018_Feb/MapServer

tation of grey scale and colour images. They mention different techniques such as grey level histograms, spatial details, fuzzy set theories, Markov Random Field models and NNs. This was the first such review to include techniques using NN models. Blanz & Gish [6] segmented images using a three-layer feed-forward NN. The number of features in each pixel determined the number of neurons in the input layer and the number of classes determined the number of neurons in the output layer. Pal & Pal [63] mention two further papers, namely [4], that segmented images using a multi-layer neural network trained using backpropagation, and [24], that extracted objects out of a noisy environment using a massively connected neural network.

Ten years later the application of image segmentation as road extraction is so popular that [56] writes a comprehensive review paper on the different methods in literature at the time. Therein 250 references were collected and categorised with regards to their primary objective, the extraction technique and the specific sensor used. We see NNs used to automatically extract roads in [12, 13, 5]. Deep learning methods represented by NN were so popular at the time that [56] reviewed [17], a review paper on 200 NN applications on image processing. Egmont-Petersen et al. [17] categorised the artificial NN applications with regards to the task performed and by the complexity level of the input data. The authors conclude that the application of image segmentation and object detection used one of two approaches, namely pixel-based or feature-based, where both have their own constraints.

Wang et al. [92], in 2016, reviewed 30 year's worth of road extraction methods. These methods were classified into mathematical morphology, active contour models, dynamic programming, classification-based methods and knowledge-based methods. The classification-based methods are further sub-divided into supervised and unsupervised learning, where NNs form part of the former. Again deep learning methods are represented by NN. Some of the Artificial NN (ANN) methods pursued by the authors were trained on spectral and edge information for road centerline delineation [86]. Mokhtarzade et al. [60] used a backpropagation (BP) method by using the best inputs to test a few network structures to see their iteration time. The same BP method was later used for road detection in [37]. This approach has a few weaknesses namely, slow convergence, the need for a lot of training samples, a tendency to get stuck in the local minima, decreasing performance as categories increase and a tendency to overfit. Other NN models were developed that work around these flaws such as the hybrid NN, fuzzy NN, radial basis function NN, spiking NN [22, 43]. Wang et al. [92] conclude that the road extraction was mainly applied on main roads and feeder roads in imagery.

Previous works such as [90, 97, 95] show that NNs aren't the only popular deep learning approaches for solving road extraction problems. The most popular deep learning methods in 2018 for road extraction were the convolutional NNs (CNNs) [11, 93] and generative adversarial networks (GANs) [96]. In 2020 the authors in [1] wrote a robust article about the state-of-the-art deep learning methods used for solving road extraction problems, wherein they systematically reviewed the four main groups of deep learning methods that are currently being used most often for road extraction. These groups are GANs models, fully connected convolutional NNs (FCN), deconvolutional NNs like DenseNets and patch-based convolutional NNs (patch-based CNN). The authors conclude that deep learning methods are the most effective in road extraction compared to regular approaches. Also, due to robust pre- and post-processing techniques, these methods can be used on all types of roads, not just main and feeder roads.

The GANs network was introduced by [25] as a novel method to estimate generative models by training two models simultaneously. Namely, a generative model and a discriminative model are trained, where the former captures the data distribution and the latter estimates how likely it is that a sample did not come from the generative model, but from the training data [25]. The generative model then aims to make the discriminative model make a mistake, causing the network's output to be as close to the training data as possible. The model has since then been used in road segmentation and image classification [34]. Varia et al.[87] used the GANs model proposed by [34] on UAV images to extract formal roads where the generator part of the model followed a U-Net architecture. The model achieved an F1 score of 96% showing its efficiency [87]. Shi et al. [77] applied the GANs model on Google Earth images to extract smooth and accurate road boundaries. The generative part in this method used an encoder-decoder SegNet model that ensures the resolution of the input image and the segmentation is the same [77]. The method achieved an F1 score of 89.63%.

The CNN model that both the patch-based CNN and FCN models are based on, was first implemented by [42] to recognise handwritten characters. The CNN model can only allow images of a fixed size. The FCN model improves on this by being able to accept input images of all sizes [1]. Varia et al. [87] extracted road parts from UAV images using a FCN model named FCN-32 and concluded that the model had satisfactory results, but that the model should be trained on a large variety of images so as to improve its accuracy [87]. Kestur et al. [36] implemented a U-shaped FCN (UFCN) on UAV images to extract roads and compare it to a support vector machine (SVM), a one dimensional CNN and a two dimensional CNN. The UFCN outperformed all of the other models in F1 score, precision, recall and accuracy [36]. Henry et al. [30] used another FCN model, the FCN-8, to extract images from SAR images and found the model to accurately extract the road parts. Alshehhi [2] implemented a patch-based CNN model to simultaneously extract buildings and roads from high-resolution Remote Sensing(RS) images from two data sets. To evaluate the performance of the model, a measure of correctness was calculated to be 91.7% on the first data set and 80.9% on the second data set. To calculate the measure of correctness the fraction between the number of correctly detected target pixels (TP) and the sum of TP and the number of detected target pixels that are actually non-target pixels (FP) is taken, given in equation 2.4 in Section 2.1.3.

The DenseNets model uses an encoder part to map the input image to the latent features and the decoder part maps the latent features to the input image [1]. Panboonyuen et al. [65] implemented a model that used a deep convolutional encoder-decoder network (DCED) that was improved to segment roads from aerial images better. The DCED was compared to the SegNet model in which the proposed model outperformed the SegNet model in F1 score, recall and precision [65]. Later that year the authors of [64] proposed another improved DCED model that was based on the SegNet model to segment high-resolution RS images into road classes. This proposed model was applied to the Massachusetts data set and satellite imagery from the Thailand Earth observation System satellite provided by GISTDA. The model achieved an F1 score of 87.6% for the first data set and 64.9% for the second data set. The model only works on Very High-Resolution(VHR) RS images, since in low- and medium-resolution RS imagery it is challenging to distinguish the different road sections. Xin et al. [94] proposed a Dense UNet model that has robust characteristics and only a few parameters. The model was used to extract roads from the Massachusetts and Conghua data sets and achieved an F1 score of 74.07% for the first data set and 76.25% for the second data set. The results showed that the model has high precision and low noise [1]. Xu et al. [95] introduced a model that uses local and global attention units in a DenseNet model so that road networks could efficiently be extracted from RS imagery. The model achieved an F1 score of 95.72% which outperformed other models such as the FCN and U-Net models. Table 1.1 shows a summary of the strengths and limitations experienced by the four groups of deep learning methods.

This mini-dissertation uses the model that performed the best in [1] and is one of the models that the authors conclude to be the most promising model for formal/urban road extraction. They considered the F1 scores from all of the models since it is a measure that shows the trade-off that exists between recall and precision. Therefore, making it possible to compare the results of all the different deep learning models with each other [1]. The GANs-UNet mode had the highest F1 score of 96.08%. Abdollahi et al. [1] points out that a factor that causes the GANs model to perform so well comes from the multi-scale integration of the extracted road features from all of the convolutional layers. The multi-scale merging causes more precise road boundaries even in the presence of the challenges faced when extracting roads. It also preserves the detail about road features. As previously mentioned the GANs model was tested and trained in [87] on a large, freely available Unmanned Arial Vehicle(UAV) RS dataset that consists out of RGB (Red Green Blue) images of different sizes created by [100]. The test dataset included 4 different resolutions of 2760 images, namely 1280 x 720, 1244 x 748, 1024 x 576 and 848 x 480. Only 189 images were used to train the models and 23 images were used to test it [87].

The literature presented thus far only applies deep learning road extraction methods to urban roads. The biggest arguments against deep learning methods have been the large data sets of labeled data required for training. High-resolution satellite images are very expensive, potentially making deep learning methods impractical for developing countries that do not have their own satellites. UAVs have become more popular in image classification like forest mapping [74] and tomato detection [76] since they are cheaper alternatives to high-resolution satellite imagery. Drone images are cheaper over long-term use and work very well in humid climates [74]. Satellites are dependent on external software when accumulating images, whereas

Approaches	Complexity	Output	Smoothness
`Models based on GANs`	- Model breakdown and lack of convergence for complex and large data - Complex training	- Efficient and robust - Provide constant output	- Capable of achieving boundary information and smooth segmentation map
`Models based on CNNs`	- Require few parameters - Require extensive samples - Low computing process	- Not highly efficient in providing constant output - Do not perform well in highly complex positions - Ignore the correlation among neighbouring pixels - Attain pixel-to-pixel reasoning	- Require high processing to identify boundaries and create a smooth segmentation map
`Models based on FCNs`	- Low adaptability with complex data and depend on images and masks for training	- Issue with road connectivity - Low position accuracy, lack of spatial consistency	- Cannot successfully achieve smoothness estimation for curved lines
`Models based on Deconvolutional Net`	- Require large amounts of memory and storage - Require additional time for training and high computing process	- High spatial accuracy - Efficient and robust for achieving consistent output	- Able to obtain a smooth segmentation map

Table 1.1: The strengths and limitations of various deep learning methods for road extraction [1].

UAVs use an embedded camera and a map of the environment making it a better solution when GPS fails [50]. The existing deep learning models listed in [1] are applied to formal roads and face challenges when it comes to the extraction of informal roads. For example, where formal roads' widths are consistent or change gradually over a piece of road, informal roads' widths vary drastically in over a short distance [62]. We also see that informal roads do not repeat in regular patterns, but formal roads are often planned in regular patterns [62]. The colouring and reflectance of informal roads are variable [45]. The colouring of informal roads is at times indistinguishable from the areas surrounding buildings and other non-road areas, making it easy to misclassify a road as a lawn or building using spectral information [62].

UAV images have made it easier and more accessible to create the large annotated data sets required to train NN models. This has led to an increase in more data sets being created. Another problem seen in other computer vision tests puts its head forward, namely, data set bias. The authors in [85] take a deeper look at the bias present in many of the popular image recognition data sets used in computer vision tests. They mention how even with the data set creators' best efforts their data set had a definite built-in bias due to the specific goals the data set were created for. The reason for this is plainly due to how a finite set of sample images can rarely encapsulate the entire diversity seen in our visual world [84]. The different reasons for these biases being present are given in [85]. They mention three specific reasons, namely capture, category (or label), and finally negative bias. Capture bias refers to the bias that can appear in a data set due to how the images were captured with a device or the person that did the collecting. Category (or label) bias refers to how certain labels could be poorly defined causing similar images to be annotated with different names. Negative bias originates since when capturing images for computer vision, since not only are the objects of interest (positive occurrences) captured, but also the objects that are not of interest (negative occurrences) and these negative occurrences may not be an appropriate representation or sufficient sample of all of the possible negative occurrences [85, 84]. To investigate the presence of bias in data sets we can measure how two data sets are related. The authors in [46] use the Fréchet Inception Distance (FID) score to quantitatively compare the models they use to generate faces using neural networks. Previous works have also used data augmentation to combat the presence of data set bias in a NN model [91, 46].

This mini-dissertation considers the work done in [19] and expands on it. In this mini-dissertation we investigate the following:

- The investigation of the theory behind data set bias in the application of computer vision data sets.
- The investigation of the theory that goes into the development of a GANs UNet neural network model.
- The investigation of the challenges and importance informal roads pose to the task of road extraction.

- The implementation and understanding of the different standard evaluation measures implemented when comparing road extraction neural networks.
- Three data sets consisting of manually digitised informal roads combined with existing formal road data sets of three different geological areas in South Africa. We specifically investigate and use the combination of these data sets as the training sets.
- We investigate the implementation of a GANs-UNet model on different combination data sets.
- The investigation of the quantitative results of the model.
- The investigation of the quantitative results of to conclude the presence of data set bias through the CD (Cross-Dataset) measure.
- The investigation of the qualitative results.

Chapter 2 expands on the theory behind data set bias, the different evaluation measures used, the GANs-UNet model and the processes followed to create both data sets. Chapter 3 explains the process taken to prepare the data for the model, the packages and software versions that were implemented to run the model and the details of the different permutations that were run. Chapter 4 provides the quantitative results of the different permutations and the prediction images used in the qualitative investigation. Chapter 5 discusses these results and Chapter 6 concludes everything done in this mini-dissertation.

Chapter 2

Methodology

This chapter presents the theory of the different topics mastered in this mini-dissertation. Section 2.1 presents the theory behind data set bias and how it will be measured in this mini-dissertation. Section 2.2 then showcases the model used in this mini-dissertation to extract the informal roads present in the different data sets. Section 2.3 shows how the data sets were created, and the evaluation metrics used.

2.1 Data set bias

This section is split into different sections to showcase what is meant by the term data set bias. We first define the term bias within the context of traditional statistics and we then build on what was introduced in Chapter 1 with regards to data set bias.

2.1.1 Bias in statistics

The term bias is defined by Merriam-Webster as an inclination of temperament or outlook. But in a statistical sense, it can be defined as either the "deviation of the expected value of a statistical estimate from the quantity estimates" or the systematic error introduced into sampling or testing by selecting or encouraging one outcome or answer over others[1]. Therefore, any bias present in a statistical analysis can cause the results to be changed significantly if no bias is present [35]. This then can lead the researchers involved in the analysis to make incorrect conclusions that could lead to harmful or dangerous suggestions, practices or guidelines being put in place. To prevent bias from entering into a statistical analysis we have to be able to know where bias can enter an analysis.

The definitions that were given by Merriam-Webster give us two routes to consider bias from a statistical perspective. By considering the first definition we see that it is referring to the area of traditional statistics

[1]Bias, https://www.merriam-webster.com/dictionary/bias,Accessed:19/10/2022

where the bias of a parameter estimate is calculated in a statistical analysis. Note that traditional statistics is used here as a term to refer to the descriptive and inferential statistics done through distributional theory from collected sample data. This first type of bias is a measurement of the distance between a parameter's estimate and the true value of the parameter in a statistical analysis [35]. The true value of the parameter is unknown the majority of the time so it has to be estimated using a sample of the population of interest. The true value can be found by doing experiments in a laboratory or by running certain simulations [35].

The authors in [15] define the bias of a parameter estimate in a regression context in the following way. Let A be the estimate of the parameter θ, then the bias of A is defined as the difference between the expected value of A, $\mathbb{E}[A]$, and θ itself, i.e. the distance between the estimate and the true value. The expected value can be estimated using maximum likelihood estimation (MLE), Bayesian estimation or through bootstrapping procedures, where MLE is the most common [89]. In MLE a point estimate is normally calculated by finding the value that maximises the likelihood function [89]. The likelihood function is just a mathematical way to represent the probabilities of observing the data for a given set of parameters. MLE is the most common due to the desirable properties it offers a researcher, namely consistency, efficiency, simplicity, versatility and interpretability [26, 29, 28, 70]. The property of consistency refers to the fact that the estimate will approach the true value as the size of the sample increases. The property of efficiency refers to the fact that an MLE estimate has the smallest variance when working with large sample sizes. Simplicity and versatility then refer to the simple implementation of MLE and the wide range of statistical models it can be used on respectively. Finally, interpretability refers to the interpretation of the estimates being intuitive due to estimate values corresponding with the most likely observed data. The other methods of estimating the parameter values can also be used to get a better understanding of the true value's distribution [83].

When we consider the second definition of bias, we see the bias that occurs within a data set used for statistical analysis like a clinical study. This type of bias has many different sources, but the three main sources are classification, confounding and selection bias [38].

Classification bias happens when improper or vague recording of the variables that affect the study's outcome at the beginning of a study occurs [38]. For example, consider a statistical analysis where researchers were examining if a high-sugar diet over ten years was associated with obesity. The researchers gathered participants that are and are not obese of equal number and question them on their diets as their data set. Participants may not accurately recall if they had a high-sugar diet since they may not be mindful of the presence of sugar in their diets, whereas others may be very conscious of the sugar content of their diets. This may lead to a participant's data values for certain variables being incorrectly recorded.

Confounding bias occurs when the results of a certain variable cause the semblance of an association between an outcome and another variable(the variable of interest normally), but they are not causally related to the outcome itself and can cause incorrect conclusions [41]. This variable is called a confounder variable since it changes or hides the true relationship between the variable of interest and the outcome observed. Nonsensical examples of this include a statement like, "Ice cream consumption increases the risk of shark attacks", but also conclusions on studies like, "coffee drinkers have a greater risk of lung cancer" [21].

Selection bias enters a study when the sample taken does not represent the population and this impacts the conclusions that can be drawn from the study [38]. This normally occurs when a researcher is inadvertently using a subset of their data set instead of the whole data set. This occurs the majority of the time due to the ease there is for the researcher to access the data from the subset rather than the whole data set. The best examples of this lie in survey data sets, for example, consider a company that created an app to assist every person with doing their taxes. They then ask people to test their app to find any flaws and improve the user interface of the application. The majority of the participants liked the app and not a lot of changes were reported or requested. This seems like exactly what the company wanted to launch the app, but note that the majority of people they gathered to test their app have an accounting qualification or background. Naturally, these people would find the app easy to use, barring any huge problems, but they would not report the same experience as participants with no accounting qualifications or background.

When we consider again the two definitions of bias given in Merriam-Webster[2], then we can ask if this can happen in other non-traditional statistical areas such as machine learning. Thomas Mitchel introduced the term bias in a machine learning context in an article written in 1980 [58]. He defines it as "any basis for choosing one generalisation [hypothesis] over another, other than strict consistency with the observed training instances." The authors in [15] mention that there are two examples of machine learning biases, namely, absolute and relative biases. Where, absolute bias is the assumption that a machine learning algorithm will eliminate certain hypotheses completely, and relative bias is where an algorithm chooses to prefer certain hypotheses over others. The authors in [15] extended the first definition of bias given in Mirriam-Webster into the field of classification algorithm applications. To illustrate this extension we define a variable $\hat{p}(x)$ which gives the probability that the algorithm misclassified a test point x. The equation is given in Equation (2.1) shows what $\hat{p}(x)$ is. From this, we can define the bias in a classification context with Equation (2.2), where $\mathbb{E}\left(\hat{p}(x)\right)$ is the average of the variable indicating misclassifications.

[2]Bias, https://www.merriam-webster.com/dictionary/bias,Accessed:19/10/2022

$$\hat{p}(x) = \begin{cases} 1 & \text{if data point was correctly classified} \\ 0 & \text{if data point was incorrectly classified} \end{cases} \tag{2.1}$$

$$\text{Bias} = \begin{cases} 1 & \text{if } \mathbb{E}[\hat{p}(x)] \leq 0.5 \\ 0 & \text{if } \mathbb{E}[\hat{p}(x)] > 0.5 \end{cases} \tag{2.2}$$

When an estimate of $\mathbb{E}[\hat{p}(x)]$ is greater than 0.5, it implies that we expect the model to misclassify the specific test point x. Therefore, we observe that x is a systematic error [15]. The authors calculate the bias in their machine learning algorithm by simulating directly from these definitions.

2.1.2 Data set bias

From the definitions given for bias at the beginning of this section, we can see that data set bias is a combination of them. The three reasons data set bias exists are very similar to some of the types of bias present in data sets from a statistical analysis since the data set collected might not be a good representation of the population data. Recall the three types of data set bias given in Chapter 1, namely, capture, category, and negative bias. All three of these types are a concern since all three may be present in the data sets this mini-dissertation presents. This is due to the data sets being manually digitised, the unique challenges informal roads pose, and the drastic difference in geological area between the data sets. Section 2.3 goes into more detail on the process of creating the data sets and which areas in South Africa were digitised. We, therefore, investigate specifically if data set bias is present in these data sets due to the possibility of the model learning features that are geological region-specific that may confuse the model to classify roads from another completely different geological region incorrectly. The goal is to have a data set of formal and informal roads that can be used in general applications and not be geological region-specific. It is important to point out that the concept of data set bias and the common machine-learning concept of overfitting are similar but are still different. The presence of data set bias has the effect of a model overfitting to its training data set. Overfitting is a common machine-learning concept that helps to explain a model's characteristics and behaviour as it indicates that a model has learnt/captured the noise that is present in the data set and does not perform well on the training data set [90]. In the field of machine learning, fitting a model is straightforward, but the reason why a model might overfit a data set is not deeply understood. There are multiple reasons why a model might overfit, namely, the size of the training set being too small, the data set containing a large amount of irrelevant data, and the model's complexity being too high [90, 18]. By finding that a model has overfitted, it is uncertain why it has overfitted, therefore this mini-dissertation points out that a reason may be from how the data set was created.

To test for the presence of data set bias we can use in-data and cross-data evaluation metrics [84]. By in-data, we refer to where the training and testing samples were done from the same data set, and cross-data to where the training and testing are from different data sets. Standard evaluation measures normally fall into the in-data category and is not sufficient on its own to test for the presence of data set bias. since it may be show overfitting, but not what may have caused the overfitting. Cross-data evaluation measures can then help to clarify this, therefore we use the cross-data measure that was created by the authors of [84] for testing the presence of data set bias. The measure the authors in [84] created is called the Cross-Dataset (CD) measure which is based on the percentage drop an algorithm experiences from being tested on its own test set and other data sets' test sets. The CD measure is defined by Equation (2.3), where O refers to the performance of the model on its own test set and M to the mean of the model's performance on the other data sets.

$$CD = \frac{1}{1 + e^{-O-M}} \tag{2.3}$$

The authors specify that a CD value greater than 0.5 indicates the presence of bias in the training data set.

We investigate the presence of data set bias quantitatively and qualitatively, where the different measures and ways are discussed in-depth in Section 2.1.3. We calculate standard evaluation neural network metrics such as Mean Pixel Accuracy (MPA), Intersection Over Union (IOU), and Fréchet Inception Distance (FID) score to evaluate the models. We then use the CD measure to quantitatively investigate the presence of data set bias by using the MPA and IOU metrics since they are bounded between 0 and 1. We calculate these measures on the first data set and the combinations of it with the other two data sets. These trained models are then tested on their own test samples as well as the data sets that they were not trained on. This approach of training a model on certain data as a starting point for another model's development or data set improvement touches on the concept of transfer learning that is widely used in the machine learning field.

Transfer learning was introduced by Bozinovski et al. in 1972 in unpublished reports and in a published report in 1976 [9, 8]. Transfer learning can be defined as a method used in machine learning model development where a pre-trained model is used as the starting point [81]. The underlying assumption is that the model learned some knowledge, like features, in the training for some tasks and that knowledge may help as the starting point for another task [81]. For example, a model trained on classifying different types of apples could be used as the starting point in developing a model that can classify different types of pears [81]. Transfer learning is a very useful machine learning method that can bridge the gap that the development of neural network models faces, namely the unavailability of large labelled or even unlabelled data sets that are easily available.

Therefore, by training a model on a data set to test whether what the model learned in training is causing it to have a bias with regards to another data set of a different geological area touches on the idea of transfer learning, but not in its entirety. We use a model that performed the best on formal road data as the method to evaluate the presence of data set bias in the informal road training set we contribute. In short, standard transfer learning focuses on the development of new machine learning models by using existing models on similar data as the starting point. We use this framework to evaluate what the desired training set should look like before an optimal and robust automatic informal road extraction model can be developed.

2.1.3 Evaluation metrics

2.1.3.1 Mean pixel accuracy

A very common measure to evaluate a model is to calculate its accuracy as given in Equation (2.4), where TP refers to the true positives and FP to the false positives. The authors in [48] used a measure called pixel accuracy that uses the same parts as the original accuracy measure but specifically applied pixels. The authors use pixel accuracy to report on their findings as it applies the measure to a semantic segmentation task. The change in formula is shown in Equation (2.5), where i and j refers to the different classes in an image with i being the class of interest. Therefore, n_{ii} is the total number of pixels that were classified and labelled as the class of interest and n_{ij} as the number of pixels that is of the class of interest but were classified as another class, j [48]. We then calculate the mean pixel accuracy (MPA) since there are different classes in each image as given in Equation (2.6), where n_{cl} refers to the total number of different classes. Note that Equation (2.7) refers to the total number of pixels of class i [48].

$$\begin{aligned}
\text{Accuracy} &= \frac{TP}{TP+FP} && (2.4)\\
\text{Pixel accuracy} &= \frac{\sum_i n_{ii}}{\sum_i t_i} && (2.5)\\
\text{Mean pixel accuracy} &= \left(\frac{1}{n_{cl}}\right)\sum_i \frac{n_{ii}}{t_i} && (2.6)\\
t_i &= \sum_j n_{ij} && (2.7)
\end{aligned}$$

2.1.3.2 Intersection over union

The IOU (Intersection Over Union) or Jaccard index [11] is another common evaluation metric used in semantic segmentation. It measures the similarity between images by first encoding the two images' shape properties (widths and heights) into the region property after which it calculates a score by using the areas of the two images [67]. The calculation divides the area of the two images that overlap by the area of the union of the two images. For example, if we consider two sets A and B, then the IOU formula can be defined as in Equation (2.8).

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.8}$$

In Equation (2.9) we show how Equation (2.8) can be adapted to an image as a discrete object consisting of pixels [11].

$$J = \frac{1}{n_{cl}} \sum_{i=1}^{n_{cl}} \left(\frac{n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}} \right) \tag{2.9}$$

Note that n_{ii} is the total number of pixels that were classified and labelled as class i, n_{ij} is the number of pixels of class i that were classified as class j, n_{cl} is the total number of different classes and t_i is the total number of pixels of class i [48]. Due to the encoding of the scale parameters, the metric is invariant of the scale of the task it has to measure. Therefore, it is widely used in performance measures to evaluate segmentation, object detection and tracking tasks [67].

2.1.3.3 The Fréchet Inception Distance

The Fréchet Inception Distance (FID) was introduced by Heusel et al. [32] as a GAN-specific evaluation measure. It measures how similar the generated images are to real images. The FID score is an improvement on the Inception score (IS) that was introduced in [75].

The IS uses an Inception-v3 model [80] that was trained on ImageNet on every generated image with the goal of finding the distribution of the conditional label $p(y|x)$ (the probability of image x having label y) [75]. The Inception-v3 model is a convolutional neural network that was created to recognise images. The ImageNet data set is a very large data set created by academics to help them in developing computer vision algorithms since each image was annotated by people through crowdsourcing platforms[3]. From 2010 to 2017 there was an annual visual recognition challenge where participants used subsets of the ImageNet data set to develop the best computer vision model. The Inception model (the Incepttion-v3 model) won the 2014 challenge for the exceptional results it obtained.

IS calculates the similarity by calculating the distance between the probability distribution of the generating model data using the Kullback-Leibler divergence function or relative entropy [75, 32, 49]. This function is shown in Equation (2.10) where $P(x)$ and $Q(x)$ are any two probability distributions.

$$D_{KL}(P||Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)} \tag{2.10}$$

The IS formula is then given in Equation (2.11), where $p(y|x)$ is the conditional label distribution and $p(y)$ refers to the marginal label distribution [75] for the generated data.

[3]A Gentle Introduction to the ImageNet Challenge, https://machinelearningmastery.com/introduction-to-the-imagenet-large-scale-visual-recognition-challenge-ilsvrc/, date accessed: 20-09-2022

$$IS = e^{\mathbb{E}_{x \sim p_g}[D_{KL}(p(y|x||p(y)))]} \tag{2.11}$$

The authors in [75] expected two requirements to be met when implementing the IS. Firstly, images should belong to few labels and secondly, the model should generate different images, meaning the variance between them should be large, which implies that the entropy over images should be high. Heusel et al. [32] points out the limitation that the IS suffers from is that it does not compare or use the statistics of real images to the statistics of the generated images. Another drawback is that it is biased towards the ImageNet dataset [7].

The FID improves on the IS by taking the Fréchet distance (Wasserstein-2) distance between multivariate Gaussian distributions after normality is assumed over the whole feature distribution [32, 7]. It then also uses the Inception-v3 model to encode features. Therefore, the two distributions are fitted by using the deepest layer of the Inception-v3 model that is closer to the output of the model, i.e. label [32]. Therefore, the Gaussians represent the distributions of the generated and real data [32]. The FID is not sensitive to small changes in the distribution of the real images [7]. Equation (2.12) shows the formula that is used to calculate the distance between $X_r \sim N(\mu_r, \Sigma_r)$ and $X_g \sim N(\mu_g, \Sigma_g)$ that is the Gaussian distribution of the real images and generated images respectively.

$$FID = ||\mu_r - \mu_g||^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{\frac{1}{2}}) \tag{2.12}$$

A lower FID score is preferred since it means the distance between the real and generated images is small, implying the images are similar.

2.2 GAN U-Net model

We consider the GAN-UNet model as it was the best performing model and showed the most promise to the authors in [1]. Before we can expand on the theory of the GAN-UNet model we first need to lay the foundational concepts and algorithms it relies on. Firstly, we consider basic neural networks (NN) or also known as artificial neural networks' (ANN) theory. Secondly, we showcase the characteristics and theory of standard GAN models until finally, we expand on this theory to show how the GAN-UNet model is different.

2.2.1 Artificial neural network

Frank Rosenblatt invented the first ANN, the perceptron, in 1958 as an illustration of a model on how the brain perceives visual data[69]. ANNs have stolen the attention of researchers since they can make decisions, learn to solve tasks and make conclusions from confusing, overwhelming, incomplete and complex information [16]. These capabilities make them ideal to provide solutions to problems such as natural language processing (NLP) tasks, recognising images, handwriting, voices and speech analysis [27]. An ANN is inspired by the brain and its interconnected neurons but can also be viewed in a simplified way. In general, it can be viewed as a function that takes in certain inputs x, and if the neuron fires, the function produces certain outputs, y [78, 47]. In short, we can implement the equation $f(x) = w * x + b$ using neural networks. We can show this simple linear model visually to further build the base mentality around neural networks so that more complex examples are easily understood. Figure 2.1 illustrates a linear model in the framework of a neural network. This is a simple case, where x is a single input, w is a single weight and b is the bias, where both w and b are constants. The inputs and outputs are shown using circles called nodes and the weights and biases using arrows called edges.

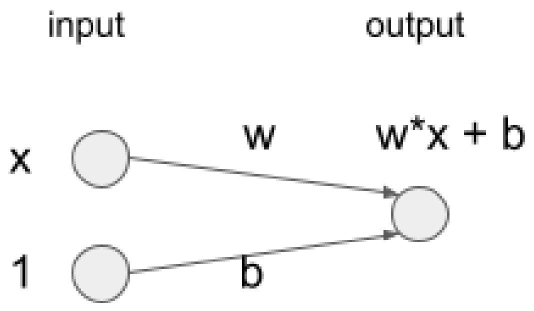


Figure 2.1: Visual illustration of the linear function $f(x) = w * x + b$

Figure 2.2 shows a typical ANN architecture with multiple inputs x_js, weights $w_{ij}^{(k)}$ and outputs $y_i^{(k)}$s, where i refers to the neuron in the next layer, j the neuron from the previous layer and k the layer they are associated with. These $w_{ij}^{(k)}$s are then collected as vectors $\underline{W}_i^{(k)}$ for each neuron i. We can then use $W^{(k)}$ to denote a matrix that consists of all the weights for layer k when working with even bigger ANNs. Note that b_k denotes the bias for layer k. This gives the equation of each neuron as $\underline{y}_k = \underline{W}_i^{(k)T}\underline{x} + b_k$ for layer k. Here $\underline{x}$ and $\underline{y}_i$ are the vectors consisting of all the x_js and y_is. The weight terms and the single bias term are used to modify the influence of certain inputs for the desired outputs and produce the inputs into the hidden neurons within the hidden layer. Hidden layers sound more mysterious than they actually are, they are neither inputs nor outputs but feeder nodes between the input and output layers [61]. Their outputs cannot be directly influenced by changing their input values [78]. These models learns when the weights and biases terms are changed since it learns the correct weights and biases values necessary for the model to classify correctly. The algorithm responsible for this process is called the backpropagation algorithm and more depth will be given further in this section.

With more complex ANNs that are adapted to real-world problems, more complex operations are required to be applied to the inputs. Therefore, a linear approach will not be suitable and non-linear functions are applied to the outputs of each neuron. Here a linear approach refers to where each node's output is the

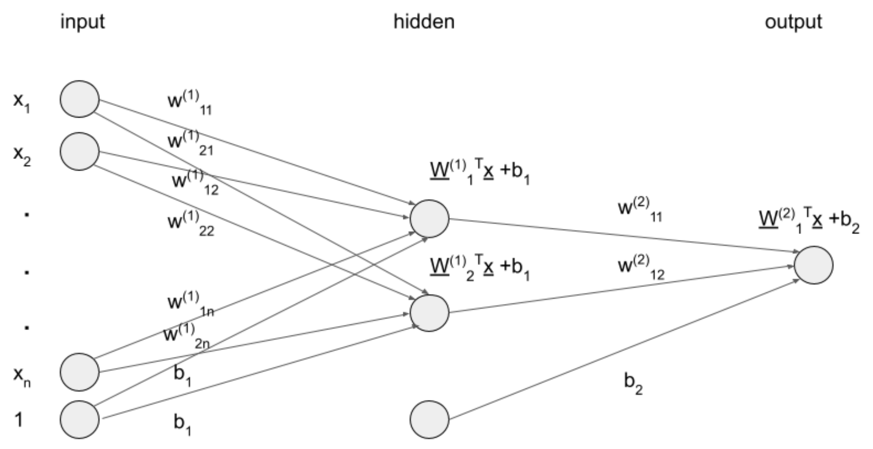


Figure 2.2: An ANN with a single hidden layer.

sum of the inputs multiplied by the weights and the bias. These non-linear functions, which take in the output of each node as its input as in Figure 2.3, are called activation functions. The most commonly used are the hyperbolic tangent (tanh), ramp (ReLu) and sigmoid (σ) functions [78]. They are useful as they take input values and produce an output between [-1, 1] or [0, 1] depending on the activation function chosen [78]. The sigmoid function's curve can also be called the logistic curve referring to the statistical logistic distribution. Therefore, the output between [0, 1] of a sigmoid activation function can be used as probabilities. Figure 2.3 shows how the output we calculate is the input of an activation function.

If the inputs are matrices or vectors, then matrix multiplication, also known as taking the dot product, becomes the main operation between these different layers. An output can only be produced when the activation potential of the activation function is exceeded, in essence implying a neuron fires [78]. This is quantified by defining the bias term, b, highlighting the importance of the term. Essentially, the bias term determines the ease at which a neuron fires. In the above three activation functions the neuron will fire when the dot product of the vector $\underline{x} = (x_1, ..., x_n)$ and the matrix $W_i^{(k)}$ plus the bias is not equal to -1 or 0 depending on the activation function [61, 78]. Equation (2.13) illustrates this by using the sigmoid function as an example.

$$\begin{aligned} \underline{y}_i^{(k)} &= \sigma\Big(\textstyle\sum_{i=1}^{n}(w_{ii}^{(k)} * x_j) + b_k\Big) \\ &= \sigma(\underline{W}_i^{(k)\intercal}\underline{x} + b_k) \end{aligned} \tag{2.13}$$

It shows how the input to the sigmoid function is calculated by taking the dot product of the vector $\underline{x}$ and the matrix W and adding the bias term. Taking the sigmoid function of this produces the output vector

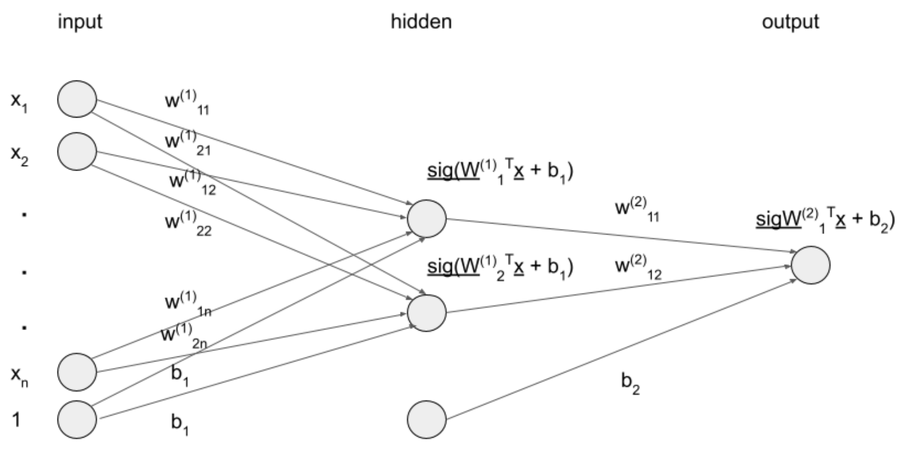


Figure 2.3: A simple ANN with a activation function

$\underline{y}$. The neuron will fire when the output is not less than or equal to 0. Note that the sigmoid function is calculated as $\sigma(x) = \frac{1}{1+e^{-x}}$.

An ANN learns (or trains) through a loss function and by using the backpropagation algorithm. This algorithm also efficiently optimises the loss function [27]. The quadratic loss function is a single example of the form a loss function can take. The quadratic loss function is defined as taking the square of the difference between the actual y_i values and the predicted y_i values, $\sum_i (y_i - y_i^{predicted})^2$ [27]. In image classification y_i would be the actual label or class the image belongs to and $y_i^{predicted}$ would be the predicted label from the model. When the predicted outputs are wrong, then the loss function is high and if it is low the outputs are correct [78]. Therefore, minimising the loss function is the way for an ANN to learn. This can be done by manipulating the weights and biases incrementally since they cause the outputs to also be adjusted incrementally [27]. Equation (2.14) shows how the weights are updated if we simplify it as in Figure 2.1.

$$w_{\text{updated}} = w_{\text{initial}} + \nu\Big(\frac{dJ(w)}{dw}\Big). \tag{2.14}$$

They are adjusted by taking the sum of the initial weights and the partial derivatives of the loss as a function of the weights and the weights multiplied by the learning rate ν. Note that $J(w)$ refers to any loss function of the weights.

The learning rate controls by how much the weights are updated at each step and is chosen manually [27]. The learning rate can be set higher if the ANN should learn faster, but less optimally, or lower if more time is allowed [78]. The partial derivatives then give the direction the weights should move to the nearest local minimum until they are at that minimum [27]. This method of minimising the loss function

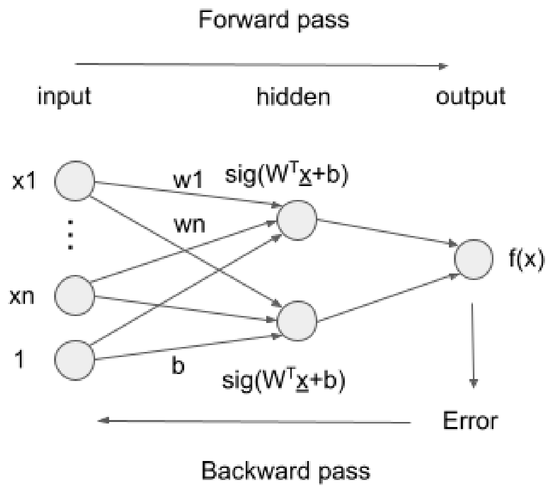


Figure 2.4: A simple visual illustration of the backpropagation algorithm[3]

as a function of weights is called gradient descent [27]. Therefore, a loss function is also a way to measure how well the weights and biases fit the given data to produce the outputs [78].

In 1986 Geoffrey Hinton and his colleagues wrote a paper, [72], where they introduced the backpropagation algorithm that solved the problem gradient descent was facing. When a large number of weights are used to calculate the loss function the process takes a very long time and is quite complex since neural networks have thousands if not millions of parameters (weights and biases). This is the problem backpropagation solves and implements gradient descent a lot faster and more efficiently than other gradient computing algorithms [27] such as the conjugate gradient algorithm [31].

First, two assumptions about the loss function need to be made. Firstly, the loss function, C, can be calculated as the average over loss functions, C_x, for training sets x. This can be written as $C = \frac{1}{n}\sum_x C_x$. Secondly, the loss function can be written as a function of the outputs from the neural network ($C = C(outputs)$) [61]. The algorithm then works in two stages, namely the forward and backward passes. Figure 2.4 shows the two-phase method explicitly. Note that the circles in between the input and output represent the hidden layer.

The forward pass is where the weights are initialised and passed through the network after which the total error can be calculated [27]. The forward pass uses the equations given above to calculate $f(x)$ with the use of the sigmoid activation function can be replaced with any other activation function. Let y represent the actual state of an output node, $j = 1, ..., m$ the index over the different output nodes, c the number of input-output pairs, and d the desired state of the output node, then the total error E is given by Equation

[3]CNN vs RNN vs ANN Analyzing 3 Types of Neural Networks in Deep Learning, https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/,dateaccessed=20-09-2022

(2.15) [72].

$$E = \frac{1}{2}\sum_{c}\sum_{j}(y_{j,c} - d_{j,c})^2. \tag{2.15}$$

The backward pass then uses Equations (2.16) - (2.20) to propagate derivatives backwards to the first layer [72]. Equations (2.16) - (2.20) make it possible to calculate $\frac{\partial E}{\partial y}$ for all units in the last layer of the network and is repeated for all previous layers. Note that when determining these equations the authors of [72] defines x_j as the total input to a node j that is calculated in a linear fashion of the outputs of the nodes that are connected to node j, y_i, and the weights w_{ji}. Let nodes k, l, m be connected to node j, then the output of the node k can be calculated using a sigmoid activation function, resulting in $y_k = \frac{1}{1+e^{-x_k}}$, where x_k refers to the input to the current node's activation function. The outputs of nodes k and l can be calculated in a similar manner. We can then define x_j with the equation $x_j = \sum_i y_i * w_{ji}$, where w_{ji} denotes the weights of going from node i to node j and i the different nodes that are fed into node j.

$$\begin{aligned}
\frac{\partial E}{\partial y_j} &= y_j - d_j && (2.16)\\
\frac{\partial E}{\partial x_j} &= \frac{\partial E}{\partial y_j} * \frac{dy_j}{dx_j} = \frac{\partial E}{\partial y_j} * y_j(1 - y_j) && (2.17)\\
\frac{\partial E}{\partial w_{ji}} &= \frac{\partial E}{\partial x_j} * \frac{\partial x_j}{\partial w_{ji}} = \frac{\partial E}{\partial x_j} * y_i && (2.18)\\
\frac{\partial E}{\partial x_j} * \frac{\partial x_j}{\partial y_i} &= \frac{\partial E}{\partial x_j} * w_{ji} && (2.19)\\
\frac{\partial E}{\partial y_i} &= \sum_j \left(\frac{\partial E}{\partial x_j} * w_{ji}\right) && (2.20)
\end{aligned}$$

Equation (2.16) calculates the partial derivative of the total error for every output unit, notated with $\frac{\partial E}{\partial y_j}$. Note that the derivative of Equation (2.15) is taken for a specific input-output pair c, but the index given by c is then suppressed to give Equation (2.16). By applying the chain rule we can compute the partial derivative of the total error for every input by using Equation(2.16) to get Equation (2.17). We can then determine the effect on total error E when changing the nodes and weights with Equation (2.18). The derivation of $\frac{\partial E}{\partial W}$ can be used to update the weights after every input-output pair. Equation (2.19) is the contribution on $\frac{\partial E}{\partial y_j}$ of node i's output due to moving from node i to node j. When we consider the number of connections from node i we get Equation (2.20). Implementing this algorithm has the added benefit of not requiring any separate memory for the derivatives causing the network to train faster than algorithms where separate memory is required[72].

2.2.2 GAN models

We first define the two concepts used often when working with generative adversarial networks(GAN) models, namely generative and discriminative. Merriam-Webster[4] defines generative as the characteristic of producing or originating something, while discriminative refers to the characteristic of making distinctions. These definitions gives us some context as we start delving into the theory behind GAN models.

Neural networks can be used as discriminative or generative models, where discriminative models map some input data to certain classes and generative models learn the classes' distributions [88]. For example, consider a discriminative model that classifies pictures as either cars, planes or boats. The model calculates the probabilities of each class (z) for each picture (x) and classifies each picture as the class that has the highest probability. Therefore, discriminative models calculate $P(Z|X = x)$. In contrast, generative models predict the input features (x) for a given class (z) [88]. Therefore, a generative model calculates $P(X|Z = z)$ and would generate a picture of a car, plane or boat depending on the class given.

A GAN model is a generative model and as was mentioned in Chapter 1, GAN are known for the two models that work against each other during the training process. These two models are the generator and discriminator networks, with each one working towards its own goal during training. We denote the generator network by $G(z, \theta_g)$, and the discriminator network with $D(x, \theta_d)$, where θ_g and θ_d denote each networks' weights [25]. The generator is a generative multi-layer perception model, that uses a probability distribution as its input (z) to generate realistic output images (x) [88], with z and x having probability distributions of $p_z(z)$ and $p_g(x)$ respectively. The discriminator is a discriminative multi-layer perception model, that takes in two alternating inputs. The one input x is taken from the real training data set that we denote as $x \sim p_{data}(x)$, while the other input is taken from the fake images generated by the generator that we denote with $x \sim p_g(x)$ [88]. During training, the discriminator alternates its input between these two sets of inputs. The goal of a GAN model is to predict if the image taken as input is fake or real [25]. These two networks have contrasting goals during training, therefore they compete against each other. Network G learns how to generate images so that network D will not be able to distinguish between the fake images and the actual data. Network D on the other hand learns how to identify data generated by network G and the actual data [25, 87, 88]. This competition between the two networks is where the adversarial aspect comes from in the name of generative adversarial networks. Note that G and D can be any neural network architecture. We use Figure 2.5 to illustrate this flow of the process from network G to D in a detailed diagram.

These two networks train in a sequential minimax game between the two networks [25, 88]. Let $J^{(G)}$ and $J^{(D)}$ denote the generator's and discriminator's loss functions respectively. Training sequentially refers to

[4]Dictionary by Mirriam-Webster, https://www.merriam-webster.com/,dateaccessed=04-02-2023

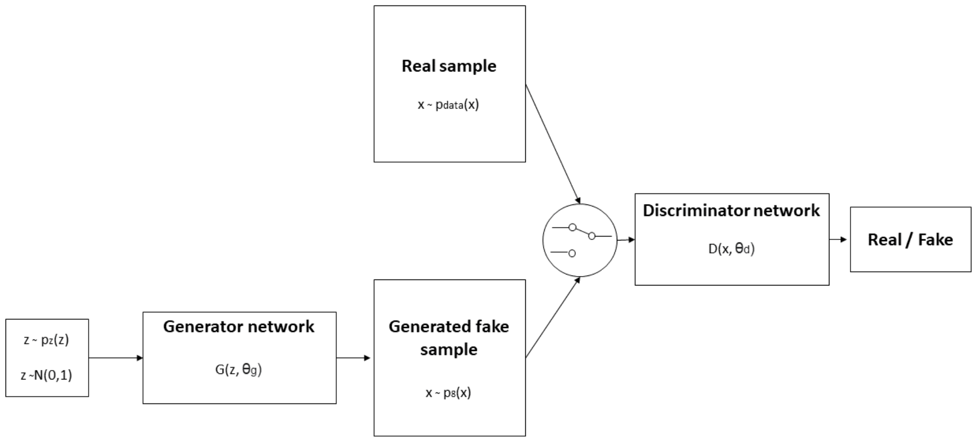


Figure 2.5: A detailed diagram of the process a standard GAN model follows [88].

each player of this game taking a turn after one another to minimise their respective loss functions, $J^{(G)}$ or $J^{(D)}$. The process would follow the following sequence of events. First network D minimises its loss function, $J^{(D)}$, by changing its weights, θ_d with θ_g staying constant. The following step is where network G minimises its loss function, $J^{(G)}$, by changing its weights, θ_g, with θ_d remaining constant. This process is repeated multiple times to train the model. We use the term minimax as done in literature to explain the strategy of the two players. The strategy of G is to minimise the maximum score D can get. For example, as we train, D improves in determining which is fake and which is real, minimising $J^{(D)}$. In response to this, G attempts to reach the level of D, minimising $J^{(G)}$ which is the same as maximising $J^{(D)}$ [25, 88]. We, therefore, denote this game by Equation (2.21), where V is the loss function and G and D are the generator and discriminator respectively.

$$\min_{G} \max_{D} V(G, D) \tag{2.21}$$

Training continues until G is able to successfully generate images that network D is unable to determine what is real or fake [88]

The discriminator trains using gradient descent and backpropagation that we explored and defined in Section 2.2.1, but with the input taken from a training set comprising of equal real and fake data. This can split the training into three steps. First, the data is taken from either the real data set or the fake data generated by the generator. If the input is taken from the real data set it is used to produce $D(x)$, but if the input is taken from the fake data then G and D work together to form a single network [88] (G uses its input z to generate a sample $G(z)$ that is used by D as its input to produce the classification $D(G(z))$). Secondly, the loss is calculated and finally, the error gradient is backpropagated and the weights are updated [88]. Note that at this step of the training process θ_g is locked and only θ_d is updated. Therefore, we are improving D rather than worsening G.

The loss function of D looks very similar to a cross-entropy loss function for a binary classification. The cross-entropy loss function is given in Equation (2.22), where $q_i(x)$ is the estimated probability that the output is from class i of the total n classes and $p_i(x)$ is the actual probability [88].

$$H(p,q) = -\sum_{i=1}^{n}(p_i(x)\log(q_i(x))) \tag{2.22}$$

$$H(p,q) = -(p(x)\log(q(x)) + (1-p(x))\log(1-q(x))) \tag{2.23}$$

$$H(p,q) = -\frac{1}{m}\sum_{j=1}^{m}(p(x_j)\log(q(x_j)) + (1-p(x_j))\log(1-q(x_j))) \tag{2.24}$$

$$J^{(D)} = -\frac{1}{2}\,\mathbb{E}_{x\sim p_{data}}[\log(D(x))] - \frac{1}{2}\,\mathbb{E}_z[\log(1-D(G(z)))] \tag{2.25}$$

$$\tag{2.26}$$

Equation (2.22) simplifies to Equation (2.23) for a binary classification model, where $p(x) \to 0, 1$. Equation (2.24) then expands Equation (2.23) for m mini-batch samples. We then define $J^{(D)}$ in Equation (2.25), where $\frac{1}{2}\,\mathbb{E}_{x\sim p_{data}}[\log(D(x))]$ refers to the loss when real data is taken as input, where we want to have $D(x) = 1$, and $\frac{1}{2}\,\mathbb{E}_z[\log(1-D(G(z)))]$ refers to the loss when fake data is taken as input, where we want $D(x) = 0$ with $x \sim p_g$ or $x = G(z)$ [88]. The $\frac{1}{2}$ refers to the cumulative class probability of each of the real and fake data since exactly half of the training data set is real and the other is fake [88].

As mentioned earlier, the goal of network G during training involves improving its ability to deceive the network D. We can also split this into three steps. Firstly, we use the input z and have it go through network G and the output of that goes through network D to give the output $D(G(z))$. Secondly, we calculate the loss for network D, but with the goal of maximising it so that network G deceives network D. Note that the part in the loss function given in Equation (2.25) that refers to real data will always be 0 so the function changes to Equation (2.27).

$$J^{(G)} = \mathbb{E}_z[\log(1-D(G(z)))] \tag{2.27}$$

$$J^{(G)} = -\,\mathbb{E}_z[\log(D(G(z)))] \tag{2.28}$$

$$\tag{2.29}$$

The gradient, $-\frac{1}{1-D(G(z))}$, of this equation causes a limitation to the training. When training starts, network D can easily determine when the input data is real or fake images which causes the gradient to be close to zero at this time, meaning that little learning of the weights happen [88]. Therefore, a solution is to use Equation (2.28). During the backward pass of backpropagation, θ_d is locked and only θ_g can be changed. Therefore, we are improving network G by maximising D's loss rather than worsening network D's performance.

We can then define the full minimax objective function in Equation (2.30) [25, 88], where G wants to minimise the objective function and D wants to maximise it. Note that the objective function is the negative of D's loss, meaning that for D to minimise its loss it will maximise the objective function.

$$\min_G \max_D V_{GAN}(G, D) = \frac{1}{2}\,\mathbb{E}_{x\sim p_{data}}[\log(D(x))] + \frac{1}{2}\,\mathbb{E}_z[\log(1 - D(G(z)))] \tag{2.30}$$

2.2.3 GAN U-Net model

One problem GAN models face is that during training, as more layers with specific activation functions are used, the loss function's gradients approach zero. This is a common problem neural networks face. Certain activation functions force large input values it receives into a small output space (between 0 and 1), therefore even a large change in the input will have a small impact on the output. Therefore, the gradient becomes smaller and smaller as layers are added to a network. Fortunately, GAN models are very popular [88] and many improvements have been suggested[5]. In a repository on GitHub[6] Avinash Hindupur started listing all of the different types of GAN models, there are over 400 types of GAN models in the repository with links to the research papers where each was introduced.

One of these improvement models is called the pix2pix model [34] which is also known as a GAN U-Net model. This model trains a generative model like a normal GAN, but in a conditional way [34]. To understand this we first define what conditional GAN or cGAN models are. They were introduced by Mirza et al. [57] as a natural extension of the original GAN model, where both G and D receive extra conditional information as their inputs, that we will denote as y, where y could be the image's class or any other property. Figure 2.5 would then only change with $G(z, \theta_g)$ and $D(x, \theta_d)$ becoming $G((z|y), \theta_g)$ and $D((x|y), \theta_d)$ [57]. This gives more control over the properties of the images that are generated, wherein a GAN model all of the information is in z [88]. This property makes cGAN models the appropriate model when doing tasks that involve image-to-image translation [34].

Therefore, a cGAN model's G learns a mapping from an observed image x and a random noise vector z to y, G: $x, z \rightarrow y$. The expression of the objective function of a cGAN model is given by Equation (2.31) [57].

$$\begin{aligned}
\min_G \max_D V_{cGAN}(G, D) &= \mathbb{E}_{x\sim p_{data}}[\log(D(x|y))] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y)))] && (2.31)\\
O &= \arg\min_G \max_D V_{cGAN}(G, D) && (2.32)\\
O &= \arg\min_G \max_D V_{cGAN}(G, D) + \lambda V_{L1}(G) && (2.33)\\
V_{L1}(G) &= \mathbb{E}_{x,y,z}[(||y - G(z)||)_1] && (2.34)
\end{aligned}$$

The goal of G is to minimise the objective function against D, while D tries to maximise the objective,

[5]Note that parts of this section are covered in the published article [19] that I was a co-author on.
[6]The GAN Zoo, https://github.com/hindupuravinash/the-gan-zoo,dateaccessed=18-09-2022

which is shown in Equation (2.32) [34]. The authors of [34] note that prior cGAN models have all been tailored to specific applications such as text, discreet labels and images, whereas theirs is not application-specific. Therefore, the final objective function used by the authors of [34] is given in Equation (2.33), where the equation for V_{L1} is given in Equation (2.34 which refers to the L_1 distance between y and $G(z)$. Note that λ refers to the regularisation rate that nudges the weights and the average of the weights to 0, causing it to have a bell-shaped distribution.[7]

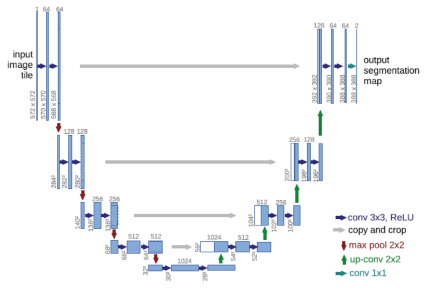
This model's main differences to prior works also lie in their architectural choices in their G and D networks. The generator network follows a U-Net-based architecture [68] that consists of a contracting and a symmetric expanding path. The contracting path captures the context and the expanding path makes it possible to achieve precise localisation [68]. A U-Net type of architecture network has that the input is passed through layers that down-samples layer by layer until a bottleneck layer is encountered and the following layers up-samples layer by layer [68, 34]. The authors in [34] then point out that in most image translation problems a lot of low-level information that is shared between input and output is desired to be directly fed across the network. The example they pose is that of an image colourisation task where the input and output share the positions of the most notable edges.

Skip connections provide the solution to this problem as they connect the last layer with any preceding layers, meaning that information is retained between layers [48, 99]. In this instance, these connections feed low-level information directly to the decoder part of the network causing the model to have a more refined spatial precision of the output [48]. Therefore skip connections were included between each layer i and layer $n-i$ with n being the total number of layers in the model. Every skip connection concatenates the channels that are at layer i with the channels that are at layer $n-i$. When the authors in [87] implement the model they repeat a down-sampling stack of two convolutional layers that uses filters of size 3×3, a ReLU layer and a maximum pooling layer that uses a stride of 2 before the bottleneck layer. The function for the ReLU function is given by $\text{ReLU} : f(x) = \max\{0, x\}$.

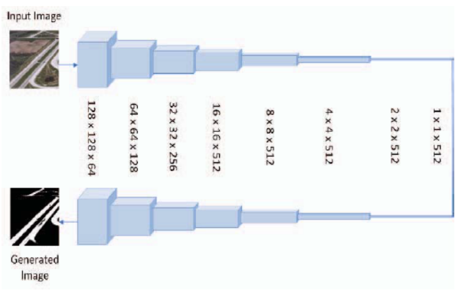
Figure 2.6(a) shows what the original U-Net architecture that was proposed by [68] and Figure 2.6(b) shows the architecture of the generator function proposed by [34], implemented in [87] on formal road data and we apply to informal road data.

The discriminator network follows a PatchGAN network to distinguish between a real and a fake image that was generated by the generator network. It was designed by the research team of [34] to be used in different image generation tasks [14]. This model focuses on the patches (a small area of an image) of an image when discriminating. You can think of it as instead of the discriminator grading the whole image and deciding if it is real or fake, a window is slid across the image that grades each patch as real or fake. Figure 2.7(a) illustrates this concept by also showing how each value in the output matrix is the

[7]Regularization for Simplicity: Lambda, RegularizationforSimplicity:Lambda,Accessed:03/02/2023

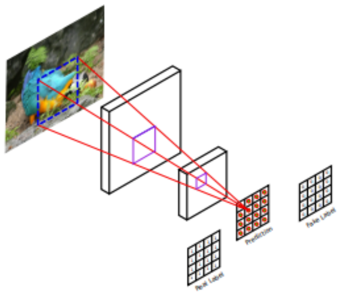


(a) U-Net architecture proposed in [68].

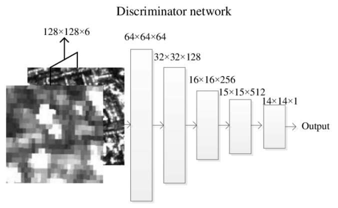


(b) GAN-UNet generator function's architecture [34].

Figure 2.6: U-Networks from original design to implementation in cGan model.



(a) PatchGAN discriminator output matrix and window illustration [14].



(b) GAN-UNet discriminator function's architecture [3].

Figure 2.7: PatchGAN network from illustration to implementation in GAN-UNet model.

probability of the corresponding image patch being real or fake [14]. The output of the discriminator is the average of all of the responses after the discriminator has run a convolution across the image [34]. The term convolution comes from the mathematical operation where one function slides over another and calculates the integral of their point-wise multiplication [23]. The benefits of the PatchGAN network are that it runs faster, has less parameters and can be used on large images [34]. Figure 2.7(b) shows the discriminator function's architecture that [34] proposed and implemented by [87] to extract formal roads.

2.3 Data sets

We use three data sets from different provinces in South Africa that have distinctly different geological areas and were digitised by two groups of honours students to test whether there is data set bias present. Section 3.3 lists the different permutations that were run to investigate this. In this section, we discuss where and how the data sets were created.

The different data sets consist of aerial images (containing the formal and informal roads) of the informal settlements of Khayelitsha[8], Melusi[9], and those near the coastal region of Coffee Bay[10] in South Africa respectively. In the first data set the high-resolution imagery was demarcated into 23 regions of 30cm resolution at a 1:500m scale representing informal roads. While in the second and third data sets, the high-resolution imagery was demarcated into 9 and 7 regions with both at a 50cm resolution respectively with both at a 1:500m scale. The areas that the polygons are taken from are shown in Figures 2.9 - 2.8 with 2.9, 2.10 and 2.8 as the first, second and third data sets respectively. The polygon demarcated in red in Figure 2.9 indicates the polygon used in [19].

For the first data set each digitiser worked separately on all 23 regions and marked each road as a polygon. The first data set was then made by merging the four digitisers' digitised data sets, considering human error by excluding any digitised regions without significant consensus. This data intersection is then combined with the City of Cape Town's official road centre lines data set buffered according to the width attribute when available, otherwise a 5m buffering was used[11]. All 23 regions make up 18.6km^2 coverage at a resolution of 30cm.

For the second and third data sets, each digitiser received an allocated subset of both the 9 and 7 regions to digitise to ensure there were no overlapping areas between digitisers. Both data sets were then made by merging the digitisers' digitised data after each subset was manually inspected to correct any errors and to add anything that may have been missing. This data is then combined with the City of Tshwane's and Eastern Cape official road centre lines data set buffered according to the width attribute when available, otherwise a 2.5m buffering was used.

For all three datasets, digitisers used a set of logical rules and definitions that were defined to reduce the variation between each digitisers data set. Three types of informal roads were defined, namely footpaths, vehicle roads, and throughways. Footpaths are those that are used by pedestrians that are not broad enough for vehicles but have the possibility to be broadened. Vehicle roads are broad enough for a vehicle to use and a throughway is those routes in-between houses that are too narrow to be used by a vehicle. The logical rules can be summarised into three categories, namely the conceptual, format, and topological rules. The first category specifies how to treat vegetation and shadows as well as how to determine road widths. The format rule category specifies the format the captured data should be in and the process it should be captured in. The last category helps to classify the spatial relationships between the road polygons.

In this chapter we explored, defined and presented the theoretical concepts used in the data set bias,

[8]Imagery was obtained from the following website, https://citymaps.capetown.gov.za/agsext1/rest/services/Aerial_Photography_Cached/AP_2018_Feb/MapServer

[9]Resolution information obtained, https://e-gis001.tshwane.gov.za/server/rest/services/Imagery/AERIAL_2018/ImageServer

[10]Resolution information obtained at, https://learn.microsoft.com/en-us/bingmaps/articles/understanding-scale-and-resolution

[11]https://odp/capetown.gov.za/datasets/tct-road-centerlines/explore

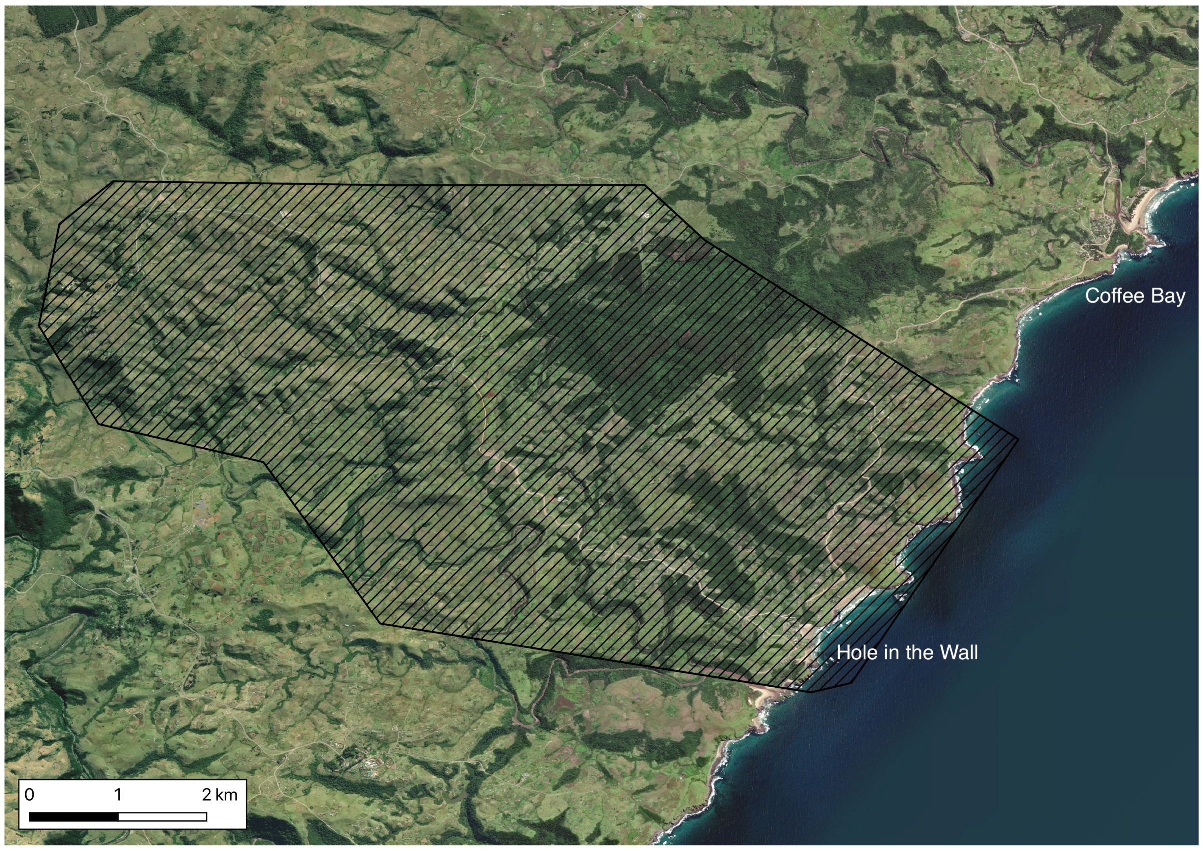


Figure 2.8: Aerial imagery indicating the region areas digitised near Coffee Bay, South Africa.

standard road extraction evaluation metrics, and the GAN-UNet model. We also described how and what each of the three data sets is that we combine. In the next chapter, we will explore and present the steps taken to implement and evaluate the model as well as explaining what the different permutations were that were used to evaluate the presence of data set bias.

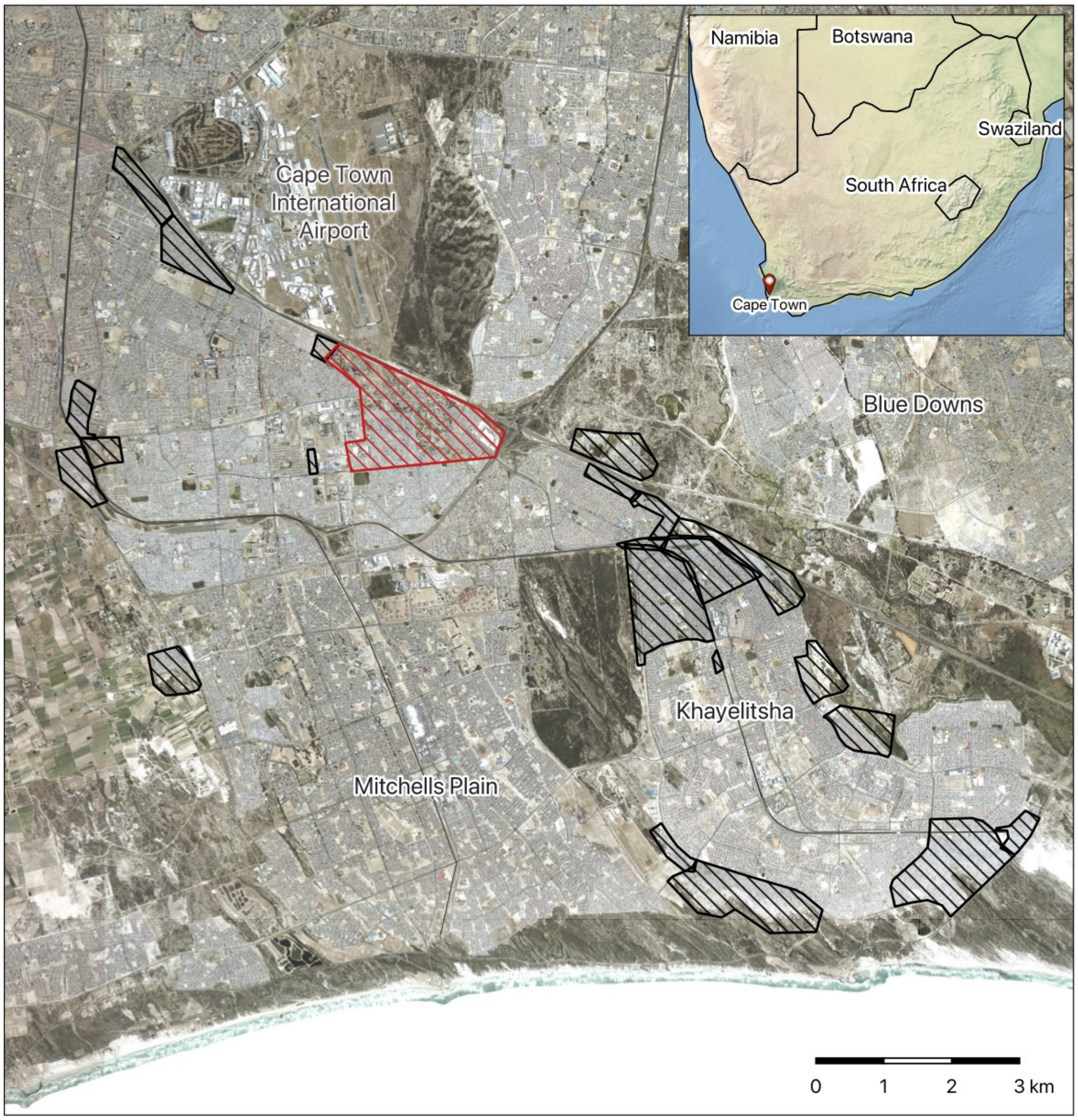


Figure 2.9: Aerial imagery indicating the region areas digitised in Khayelitsha, South Africa.

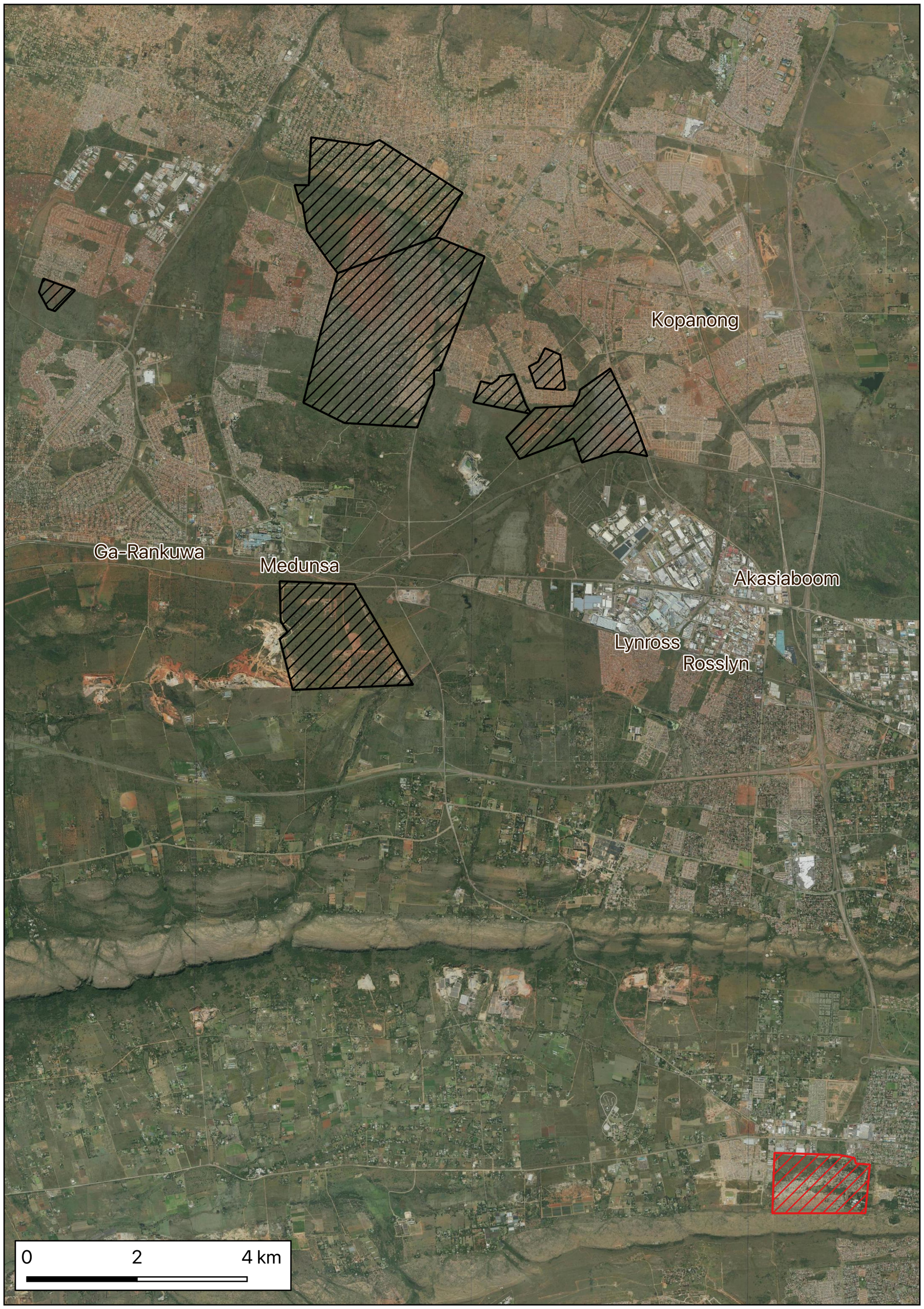


Figure 2.10: Aerial imagery indicating the region areas digitised in Melusi, South Africa.

Chapter 3

Implementation and training

In this section, we present the steps taken to implement the GANs UNet model and present the permutations that were used to evaluate the presence of data set bias. We first explain the steps taken to prepare the data for training and testing in Section 3.1. Section 3.2 then specifies the packages, software and code used to train and test the model. Sections 3.3 shows the permutations that were used to investigate the presence of data set bias and Section 3.4 explains the Python script that was used to evaluate the models with the evaluation measure presented in Section2.1.3.

3.1 Preprocessing steps

For training the model, we used high-resolution aerial imagery and split all images into tiles of 600×600 pixels and their corresponding digitised segmentation masks. We further split the tiles by randomly assigning the tiles and their digitised maps into the following sets: 70% training, 15% validation and 15% testing. As mentioned earlier we want to reduce overfitting to minimise data set bias. To this end, we artificially enlarge the data set by using label-preserving transformations to generate extra training data from the available tiles, otherwise known as data augmentation techniques. This is done by flipping, cropping, rotating, and translating each image. Therefore, the train, test, validation split for the first data set is $665, 142, 143$, the second data set is $59, 19, 19$ and the third data set's split is $89, 19, 19$ respectively.

A Jupyter Notebook [39] script was written to implement this using the newest versions of the `OpenCV-python`

Package	Version
`Python`	3.7.12
`PyTorch`	1.11.0
`torchvision`[10]	0.12
`NumPy`	1.21.6
`Pillow`	9.2.0
`Scipy`	1.7.3
`visdom`[11]	0.1.8.9
`Pip`[12]	21.2.4

Table 3.1: Package versions used to implement the pix2pix model [34]

[10], `os`[1], `NumPy`[2], `shutil`[3], `ffmpy`[4], `PIL`[5], `scikit-image`[6], `SciPy`[7] and `tifffile`[8] packages.

3.2 Code implementation

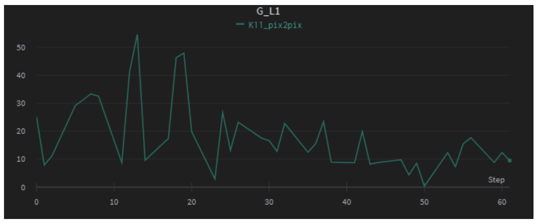
3.2.1 Package and software versions

The GANs-UNet model code can be found on GitHub under the pix2pix repository[9]. The model uses the popular machine learning framework `PyTorch` [66]. The model was implemented from the terminal on a Ubuntu 20.04 system with an NVIDIA GeForce RTX 3060 GPU and CUDA version of 11.4. The authors of the [34]'s repository read.me file explains extremely well all of the steps to implement the pix2pix model. The requirements file indicates the package versions the authors used to implement the model, but we found that some of these aren't compatible with the newer CUDA versions. Therefore, we created a Conda environment in the terminal to simplify testing what versions of the different packages are compatible. A Conda environment acts as a separate system where packages and softwares can be installed without posing a threat to the computer that is used. This allows us to have multiple package versions on the same system simultaneously by having multiple environments enabled which simplifies testing which combination of packages works the best. The versions used are listed in Table 3.1.

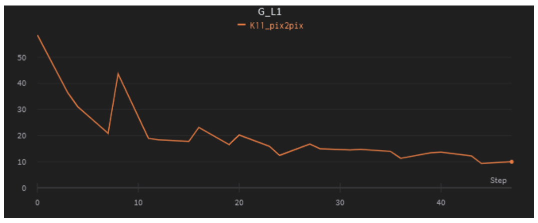
3.2.2 Hyperparameter optimisation

There are a few hyperparameters that have to be read into the model, namely the batch size, learning rate, and the momentum parameter for the ADAM optimiser. To find the values for these hyperparameters

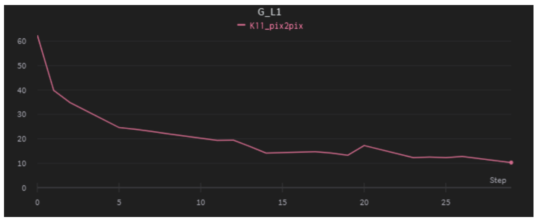
[1]os - Miscellaneous operating system interfaces, https://docs.python.org/3/library/os.html
[2]NumPy, https://numpy.org/doc/stable/
[3]shutil– High-level file operations - Python documentation, https://getdocs.org/Python/docs/3.10/library/shutil
[4]ffmpeg Documentation, https://ffmpeg.org/ffmpeg.html
[5]Pillow, https://pillow.readthedocs.io/en/stable/
[6]scikit-image image processing in python, https://scikit-image.org/docs/stable/
[7]SciPy documentation, https://docs.scipy.org/doc//scipy/index.html
[8]tifffile package,https://iridescent.ink/tifffile/tifffile.html
[9]pytorch-CycleGAN-and-pix2pix, https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
[10]TORCHVISION, https://pytorch.org/vision/0.8/index.html
[11]Visdom, https://openbase.com/python/visdom/documentation
[12]pip - package installer for Python, https://pip.pypa.io/en/stable/index.html



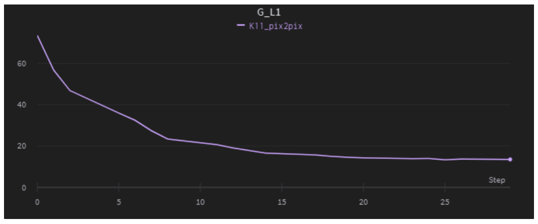
(a) Batch size of 1.



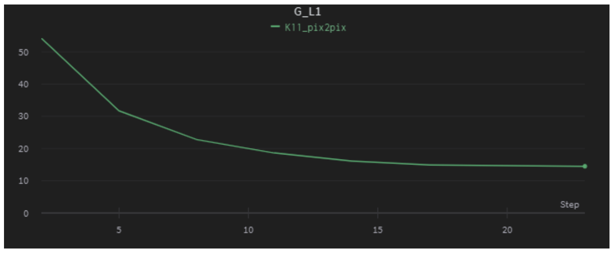
(b) Batch size of 8.



(c) Batch size of 15.



(d) Batch size of 30.



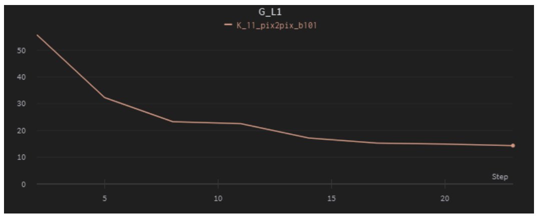
(e) Batch size of 64.

Figure 3.1: Loss functions of different batch size values

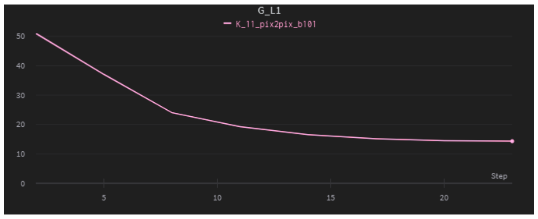
that would work best for the model trained on our data, we ran a few scenarios for each parameter by keeping the others constant. For each we considered which value for the parameters gave the smoothest loss function and reached the lowest loss value at the end. This was tested on a small sample set from the Khayelitsha data set and only for 100 epochs.

The first parameter tested was the batch size parameter. We tested the model on five values ranging from 1 (the default for the model) to 64 (the value used in [19]). The loss functions produced during training for all five values are given in Figure (3.1). We see that a batch size of 64 gives the smoothest and most gradual loss function during training. It reached the point of 13.618 at the end of training, which is only 0.131 lower than for a batch size of 30 that had the second smoothest loss function. From these tests we saw that the batch size value that reached the lowest point after training was the value of 1 at 9.521, but the loss function of this batch size was very inconsistent. Therefore, we decided to use the value of 64 due to the smoothness of its loss function.

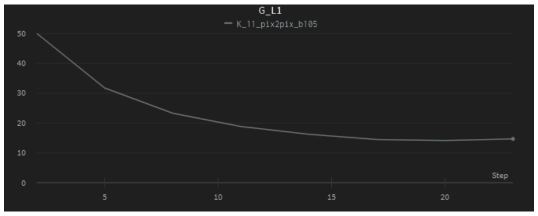
The second parameter investigated was the momentum term of the Adam optimiser. During training and testing, we specify this parameter by using the beta1 option that is defined in the training option file. We tested values in a range of 0 to 1, with 0.5 being the default option for the model. The loss functions produced during training for all five values are given in Figure (3.2). From the results, we see that the



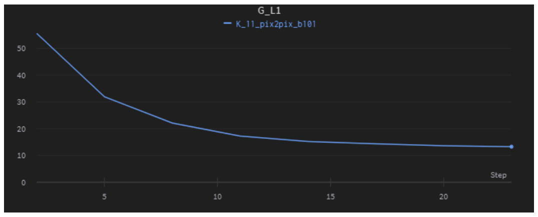
(a) Beta1 of 0.1.



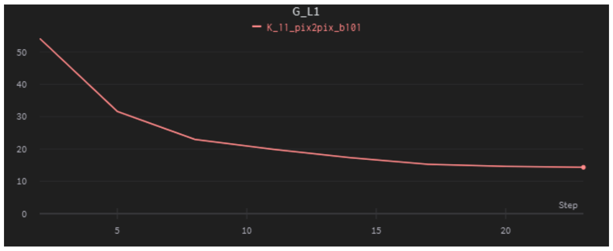
(b) Beta1 of 0.3.



(c) Beta1 of 0.5.



(d) Beta1 of 0.7.



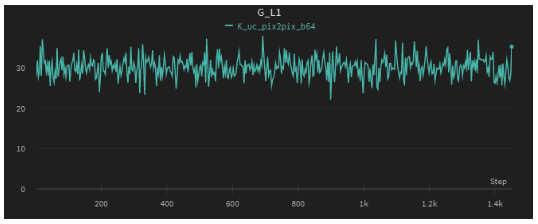
(e) Beta1 of 0.9.

Figure 3.2: Loss functions of different momentum terms of the Adam optimiser

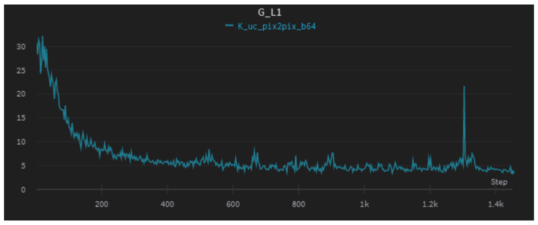
value of 0.5 has the smoothest loss curve, but reaches the second lowest point of 13.618, with the value of 0.7 reaching the point of 13.289. We chose the value of 0.5 due to the smoothness of the graph in favour of a 0.4 difference in the lowest point of the loss function.

Lastly, we looked at the learning rate parameter. We tested the six different values starting from 0.02 that was used in [19] and multiplying by $10^{-}1$ up to a value of 0.0000002 and by changing the number of epochs to 1000. This was done after training the model on the default value (0.0002) we saw that at the end of training, the learning rate had been minimised to a value of 0.0000002. The loss functions produced during training for all five values are given in Figure (3.3). From these results, we see that values of 0.0002 and 0.00002 have the smoothest and most gradual loss functions. The difference lies in the minimum each reached, where 0.0002 reached the point of 6.474 at the end of training, and 0.0002 reached the point of 3.863 at the end of training. The only value that reached a lower point was the 0.002 at 3.582, a 0.281 difference from the 0.0002. We therefore chose the value of 0.0002 for the smoothness of the loss curve.

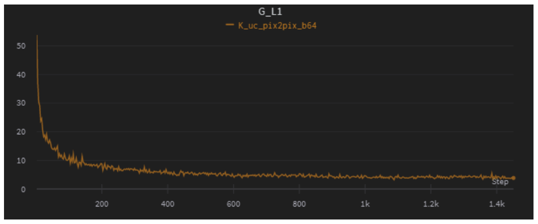
Other variables were also investigated such as the number of epochs to decay and the number of epochs parameter. The former refers to by how many epochs the model linearly reduces the learning rate to zero and the latter to the number of epochs the model will run for. In the first case we tested multiple values



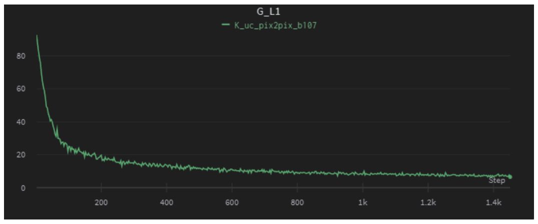
(a) Learning rate of 0.02.



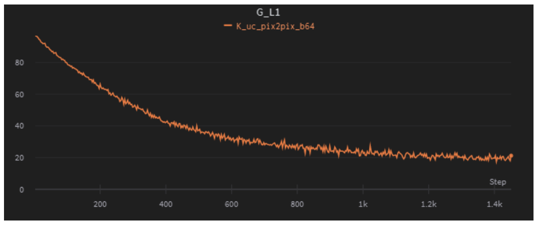
(b) Learning rate of 0.002.



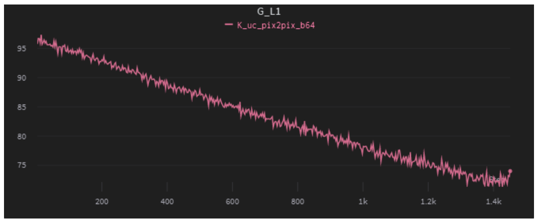
(c) Learning rate of 0.0002.



(d) Learning rate of 0.00002.



(e) Learning rate of 0.000002.



(f) Learning rate of 0.0000002.

Figure 3.3: Loss functions of different learning rate values

but did not find any valuable insights that would change the default value the model is already using. In the case of the number of epochs, we saw a dramatic increase in run time of the model when increase from 1000 to 8000, but not a significant improvement in the loss function or lowest point reached. The model trained for 1000 epochs reached the lowest point of 3.863 after 2h and 57min, while the model trained for 8000 epochs reached a point of 3.745 after 1 day 2h and 10min. Since there was no significant increase in the model's performance or convergence, we used the value of 1000 for the number of epochs parameter.

3.3 Permutations

To investigate the bias of the data set we run a few different permutations. Data set 1 has the most images and is used as our baseline data set. Each permutation is then the combination of the other two data sets to data set 1. Adding these data sets together will showcase if the model has learned biased features due to the geological areas. These are listed in Table 3.2, where the text in bold indicates the data the model is trained on and the text below what it was tested on. We use certain labels to label each of the four different training sets, namely data set 1 (K), combined data set 1, 2 (KT), combined data set 1, 3 (KE), and combined data set 1, 2, 3 (KTE).

K	**KT**	**KE**	**KTE**
Own test sample T imagery E imagery	Own test sample E imagery	Own test sample T imagery	Own test sample

Table 3.2: A list of the different permutations run by specifying the data set the model was trained on and listing the different data sets it was tested on.

3.4 Evaluation

To implement the different evaluation metrics proposed in Section 2.1.3 we created a Jupyter Notebook file that uses the newest versions of `Tensorflow`'s[13] `keras`[14] `OpenCV-python`, `PyTorch`, `os`, `NumPy`, `PIL`, `scipy`, `torchvision` and `scikit-image` packages. When implementing machine learning models the mathematical structure concept of a tensor comes up quite quickly. We, therefore, define a tensor as being the generalisation of the mathematical terms scalar, vector and matrix. A scalar is a tensor of rank zero, a vector a tensor of rank one, and a matrix a tensor or rank two, therefore we see that a tensor of rank three is a list of matrices. This generalisation makes tensors very popular in state-in-of-the-art machine learning models and algorithms with the rank normally being three for more complex inputs such as images.

The script is split into three different parts, namely the import and preparation of the test images, the predictions made by the model, the evaluation metrics' functions and the running of the functions on the test images and predictions. The first part reads all of the test prediction and ground truth images into Jupyter Notebook and each image is transformed to a size of 256×256 and then into a tensor.

We then define the different functions that are used to calculate the evaluation metrics. The function to calculate the pixel accuracy of the images takes in two arguments, namely the ground truth and prediction images tensors, and use `PyTorch`'s sum function to calculate the number of pixels that are labelled in total and those that were correctly labelled. This is done by summing together all of the ground truth tensor's values that are greater than 0 and summing all those that are equal to the prediction tensor's values respectively. A separate function is then created that uses the previous function to calculate the mean pixel accuracy over all of the test images. This is done by having the previous function's output stored in three separate empty `NumPy` lists that are the length of the number of images in the test set. The values of all the pixels of all the images that were correctly labelled are summed and divided by the sum of all the pixels of all the images that were labelled to get accuracy.

The function created for the IOU metric takes in three arguments, namely the ground truth and the prediction image tensors as well as the number of classes, i.e. the number of different labels. The areas

[13]TensorFlow: Large-scale machine learning on heterogeneous systems, MartÃn Abadi et al., Software available from tensorflow.org, 2015
[14]Keras, Chollet, François et al., https://keras.io, 2015

of intersection and union are then calculated. To calculate the area of intersection, we calculate the intersection by testing which tensor values between the ground truth and predicted are the same. We then add the *.long*() argument to the logical test, since it converts the True and False values to 1's and 0's. This is then multiplied by the prediction image tensor to give the intersection. To get calculate the areas we compute `NumPy`'s histogram function by specifying the number of bins and the range of the bins as the number of classes. The output returns an array of the values of the histogram and an array of the bin edge. The histogram values outputted when reading in the intersection and the two image tensors into the function separately and set the number of classes to two outputs an array of histogram values of length 2 with the second one always being 0, therefore we can safely assign the first value as the area of each variable separately. To calculate the area of the union we sum the areas of the two image tensors minus the area of the intersection. The area of the intersection is then divided by the area of the union to get our IOU metric. This function is then applied in a loop over all of the images of the test set and the final average IOU value over all of the test images is reported.

To implement the FID score we import the pre-trained InceptionV3 model from `keras`. Before defining the function that will calculate the FID score itself we have to load in the InceptionV3 model's weights and biases, resize the images to the size the model takes in, and pre-process them in the same way as in the InceptionV3 model's training. The InceptionV3 model is prepared, but the final output is removed since it is not what is required in this instance. The removal also removes the average pooling layer that we do require, but can easily be included as a parameter when loading the model. The function calculating the FID score takes in three arguments, namely the InceptionV3 model and the two sets of images. The model is used to predict the feature vectors for the predicted and ground truth images. From these, we can calculate the mean and covariance of each to calculate the sum of squared differences between the means and the covariance mean. These are then used in equation (2.12). The code for this can be found in an article written by Jason Brownlee on the Machine Learning Mastery website[15]. In the article, the author walks step by step through how to make sense of and calculate the FID score.

In this chapter, we presented the steps taken to implement the GANs-UNet model and how the different models were evaluated. We also presented the different permutations that the model is tested on. In the next chapter we showcase the quantitative and qualitative results obtained by testing the different models on different seen and unseen imagery. All of the training data sets and python files used that is not in the pix2pix repository[10] can be found at the DOI: https://doi.org/10.25403/UPresearchdata.21522360

[15]How to Calculate the Frechet Inception Distance for Real Images, https://machinelearningmastery.com/how-to-implement-the-frechet-inception-distance-fid-from-scratch/, Accessed:28/08/2022

[10]pytorch-CycleGAN-and-pix2pix, https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

Chapter 4

Results

This chapter gives the quantitative and qualitative results obtained by testing the model for the presence of data set bias by testing it on different permutations. We split these into two different sections that will report on different types of results. Section 4.1 reports the quantitative results, which includes the evaluation metrics' results of each data set permutation and the CD values to show if data set bias is present. Section 4.2 then reports the qualitative results through a visual investigation of unseen and incomplete data. Chapter 5 will discuss what is meant by incomplete data and what was found qualitatively as a whole.

We also make a note of the size of the data before and after training. The aerial imagery and the digitisations were 40GB in size for the first data set. Due to the fewer number of images of data sets 2 and 3, the size of their data is a lot less. If the same scale of digitisation is done on data sets 2 and three, then because the aerial imagery would be of similar size it can easily be inferred that they will have approximately the same data set sizes as data set 1. In such a situation, all three data sets would have a total of about 240GB in memory to just store the data before preprocessing, training, or testing. The tiles we generated reduced the size of the data considerably. We used each training set's tiles to create the combination data sets' training sets. A simple Jupyter Notebook was written that would rename the files to complete each of the train, test, and validation sets. This helped significantly since all of the tiled training sets had a size of up to 350MB together. Another memory usage consideration lies in the training of the model. During training, the checkpoints (the weights and biases) are saved to the computer each time after a certain number of epochs are passed. Each permutation's training set had a checkpoints data size of 50.6GB. It is important to consider memory storage space when working with so such high-resolution imagery and thorough digitisations. To solve this issue in our case we used a 4TB hard drive to store the data and checkpoints on a separate internal hard drive to ensure the speed of the computer was not jeopardised for normal tasks on it.

Training set	**Test set**	**MPA**	**IOU**	**FID**	**CD**$_{MPA}$	**CD**$_{IOU}$
K	Own test sample	0.662	0.718	436.479	0.777	0.789
	T imagery	0.569	0.589	501.198		
	E imagery	0.600	0.608	518.650		
KT	Own test sample	0.538	0.595	391.625	0.728	0.740
	E imagery	0.447	0.453	471.549		
KE	Own test sample	0.526	0.577	449.050	0.728	0.741
	T imagery	0.458	0.475	462.837		
KTE	Own test sample	0.569	0.623	461.424		

Table 4.1: Results of the different permutations.

4.1 Quantitative results

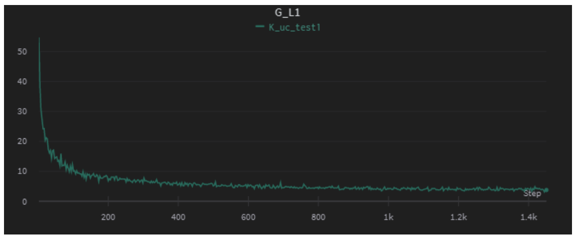
This section showcases the results of the different evaluation metrics, the CD values and the graphs of the loss functions of the model trained on the four different permutations. We use the abbreviations given in Section 2.1.3 for the different evaluation metrics. We use MPA to refer to Mean Pixel Accuracy, IOU for Intersection Over Union, FID for Fréchet Inception Distance and CD for Cross Dataset measure. We also use the data set abbreviations defined in Section 3.3, where K is used to refer to data set 1, KT for the combined data set consisting of data set 1 and 2, KE for the combination of data set 1 and 3, and KTE for the combination of all three datasets.

The evaluation metric's results are given in Table 4.1, where we show the different training data sets being tested on the different permutations. When we consider these results we can see the drop in the metrics from their own test sets to unseen imagery of another geological area, which shows us that the model has overfitted to the data. The CD values are then given by using the MPA and IOU metrics, and we compare them with the value of 0.5 to investigate the presence of data set bias. By considering the CD values in the table we can clearly see that all of the values are greater than 0.5, meaning that data set bias is present. We discuss these results further in Section 5.

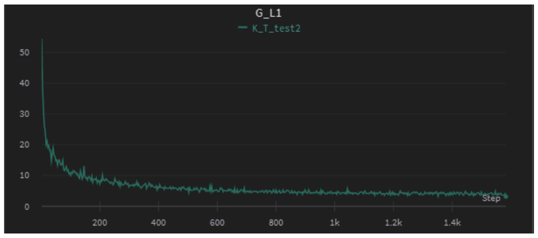
Figure 4.1 showcases the loss function's graphs during training. We note how the graphs look very similar for all four permutations. We see how each graph converges to its respective minimums.

4.2 Qualitative results

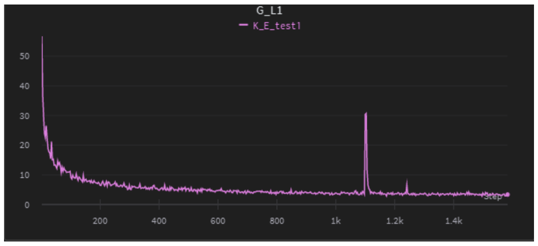
The figures in the following sections showcase the predictions from the model trained on four different training sets. Figures 4.2 and 4.3 are the different models' predictions on four unseen test images. Figure 4.2 shows the results when the trained models were tested on imagery from T, while Figure 4.3 shows the results when they were tested on imagery from E. Through this visual investigation we consider the differences between the models' predictions by comparing the base data set's model with the combination data sets' models for different test images.



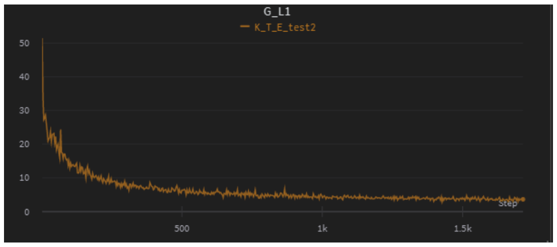
(a) Loss function of K.



(b) Loss function of combination data KT.



(c) Loss function of combination data set KE.



(d) Loss function of combination data set KTE.

Figure 4.1: Loss functions of the model trained on the different data sets.

Figures 4.4 and 4.5 gives the model's predictions for data where some digitisations have been done, but not of the whole area. Here we compare the differences between the incomplete digitisation and the models' predictions to see which struggled with capturing the features digitised and those that have not been digitised. Figure 4.4 shows the results when the trained models were tested on imagery from T, while Figure 4.5 is when they were tested on imagery from E.

This chapter showcased the quantitative and qualitative results obtained after running the different permutations. We illustrated the quantitative results through standard road extracion metrics and CD values to investigate the presence of data set bias. We showcased the qualitative results by showing the models' predictions on unseen and incomplete data The next chapter will discuss these results in more depth.

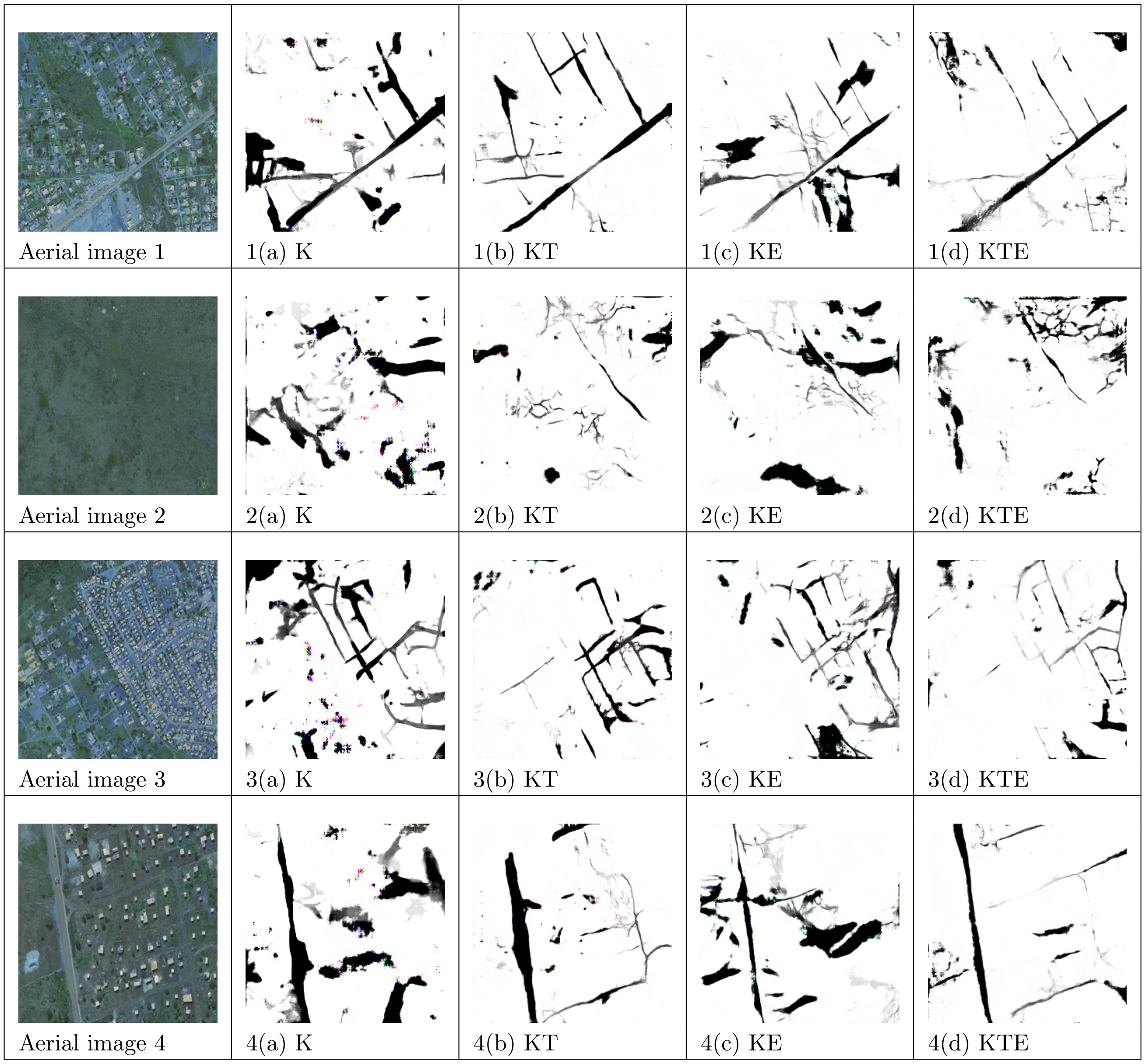


Figure 4.2: Visual investigation of data set bias when tested on unseen data from data set 2.

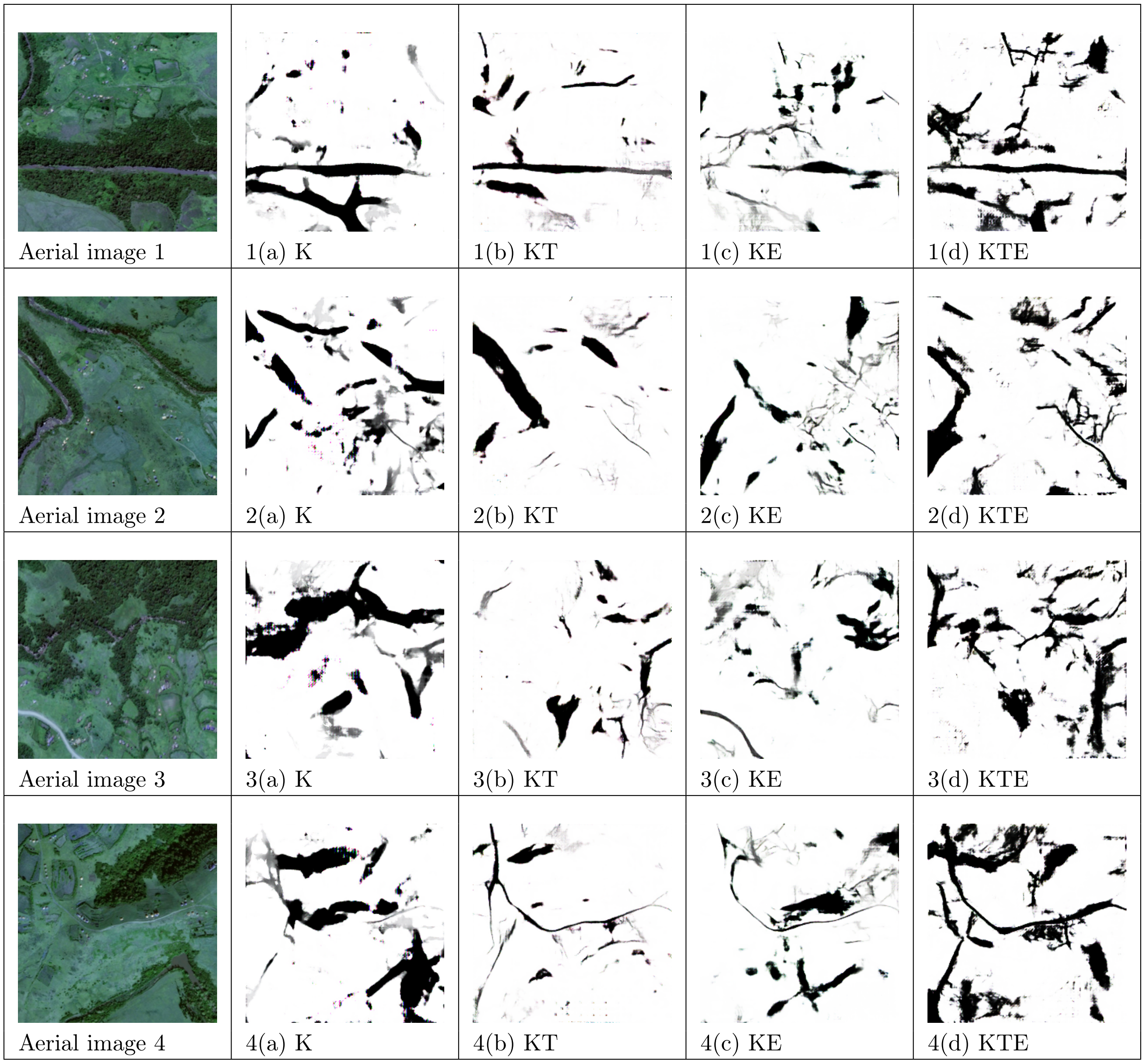


Figure 4.3: Visual investigation of data set bias when tested on unseen data from data set 3.

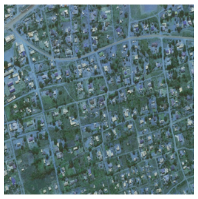
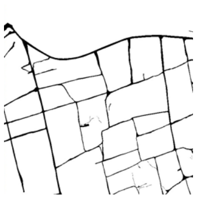
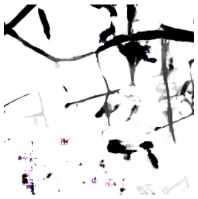
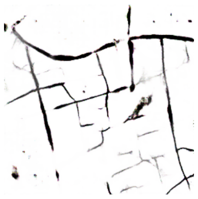
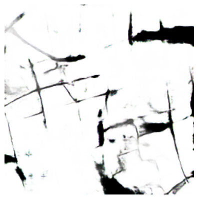
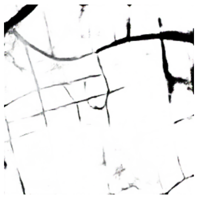
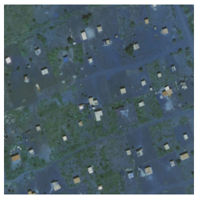
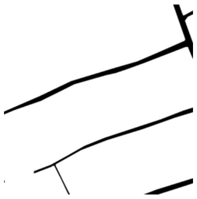
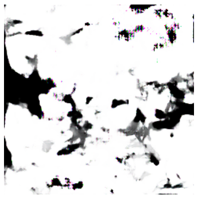
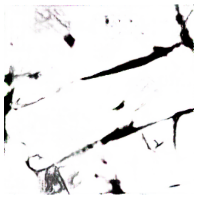
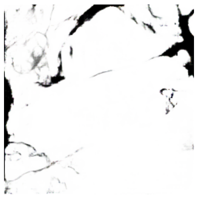
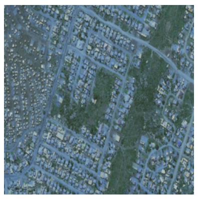
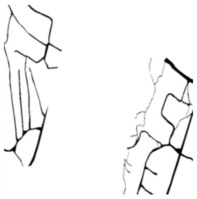
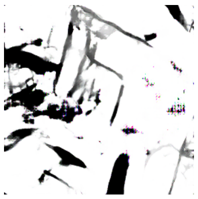
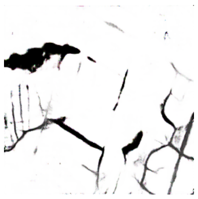
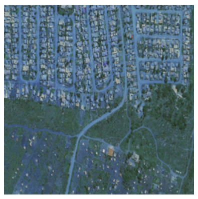
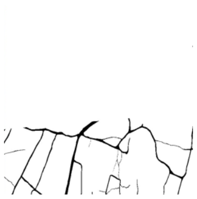
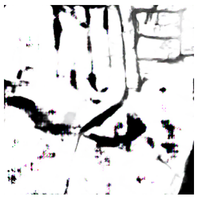
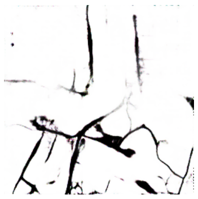
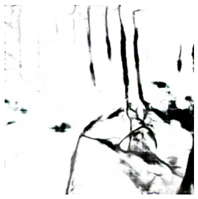
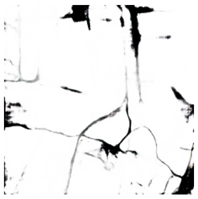
 Aerial image 1	 Incomplete digitisation 1	 1(a) K	 1(b) KT	 1(c) KE	 1(d) KTE
 Aerial image 2	 Incomplete digitisation 2	 2(a) K	 2(b) KT	2(c) KE	 2(d) KTE
 Aerial image 3	 Incomplete digitisation 3	 3(a) K	 3(b) KT	3(c) KE	3(d) KTE
 Aerial image 4	 Incomplete digitisation	 4(a) K	 4(b) KT	 4(c) KE	 4(d) KTE

Figure 4.4: Visual investigation of data set bias when tested on incomplete data from data set 2.

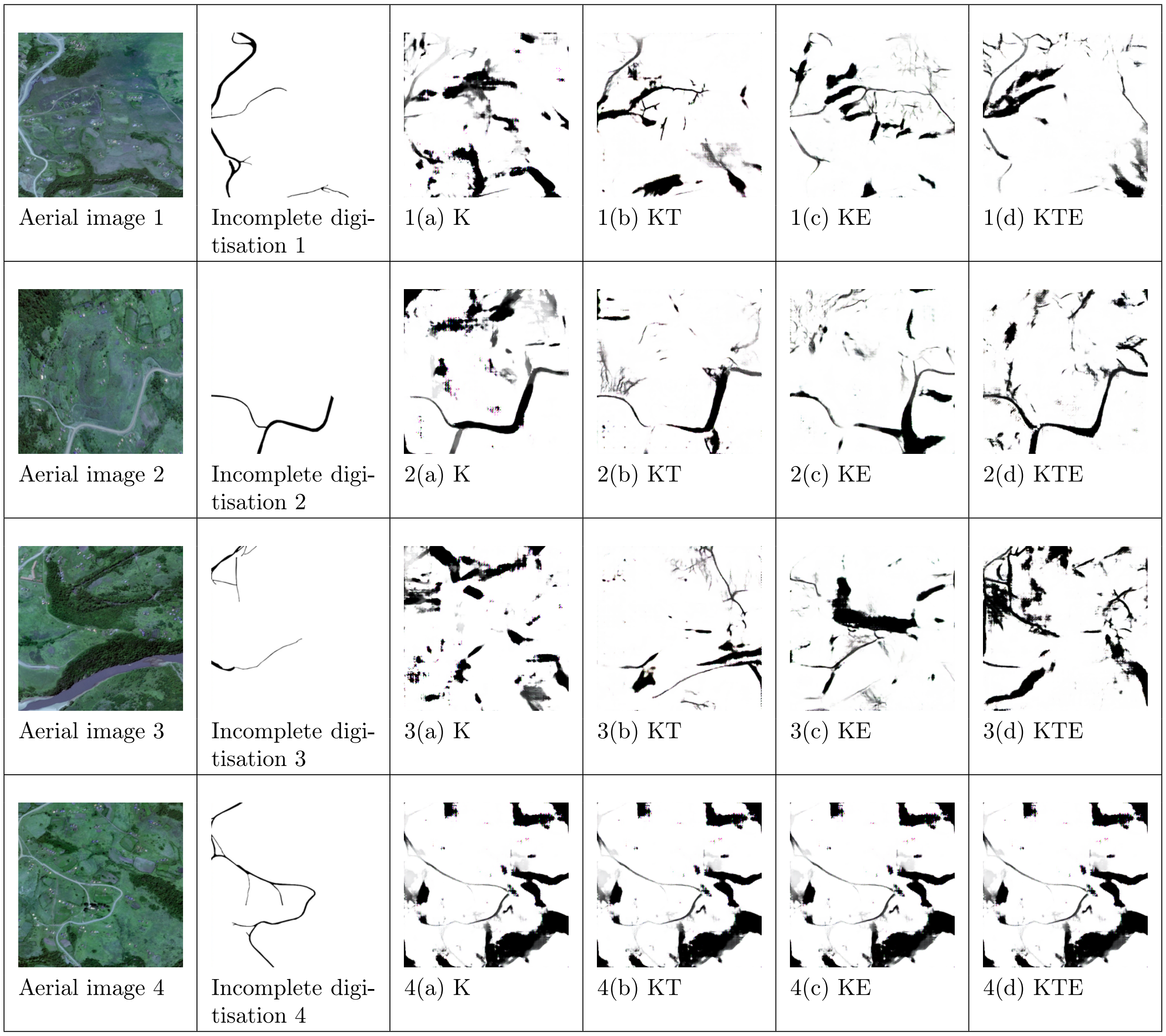


Figure 4.5: Visual investigation of data set bias when tested on incomplete data from data set 3.

Chapter 5

Discussion

This chapter discusses the results presented in Chapter 4. It is structured by following the chronological order that the results were presented in Chapter 4. While discussing the quantitative results, it is important to recall the distinction made between overfitting and data set bias in Section2.1.2. We first evaluate if the model did overfit and might be from data set bias using the standard evaluation metrics, after which we use the cross-data evaluation measure, the CD evaluation measure, to investigate the presence of data set bias.

By considering the MPA, IOU and FID measures in Table 4.1 we see how the model performed for different training and testing sets. We see that each model has overfitted to its data during training and struggled with the unseen imagery in the testing sets. This can be seen by the $6\% - 10\%$ decrease in MPA when tested on completely unseen imagery. This said we note that the model for K achieved the highest MPA on its own data set than any other model. This might be due to the fewer example images from data sets 2 and 3 that are included. We do note that the drop of 10% seen with K decreased to 9% and 6.5% when tested on data sets T and E respectively. We see a definite improvement in the model's ability to digitise unseen geological imagery relative to how it performed on its own training set. We also note that the model trained on the combination of all three models achieved the best accuracy on its own test samples of the combined data sets. From this, we see that combining different data sets can result in better performance of the model on unseen data.

When evaluating models on the basis of their IOU values, the closer to 1, the better. We see that K achieved the best IOU value on its own test set and its values on the unseen data are better than the combined data sets. We see that the combined data sets dropped by 0.142 and 0.102 respectively when tested on unseen geological imagery. Note that KE achieved the second-highest IOU value when comparing all of the combination data sets, the only better-performing model was K on its own test set.

The FID scores, even though large, give us some insight into the performance of the generative ability of the models when comparing them to each other. Comparing the FID scores of each permutation between its own training set and unseen imagery, we can see that there are dramatic jumps for K and combined KT showcasing that there is a substantial amount of data set bias learned by these two data sets causing them to struggle on unseen imagery and misclassifying objects in them. This will further be explored by the qualitative results section. If we compare all of the FID scores together we see that the model trained on KT achieved the best score. We point out that the FID score decreased as more data was added and tested on unseen imagery showing that fewer data set bias was learned by the combined data sets than in the base model.

When considering the CD values in Table 4.1 we note that all are greater than 0.5, meaning that we can conclude that data set bias is present in the data set. We calculated the CD scores using the MPA and IOU values since their values are both bounded within 0 and 1. No CD value was calculated for KTE since there are no test sets it has not seen that can be tested on. When comparing the CD values of the two combination data sets, we see that they achieve the same values when using MPA and IOU, indicating that neither one of these combinations decreased the presence of data set bias. This may be due to only a few images from data sets T and E being added to data set K and it not enough to help reduce the presence of data set bias. When we compare the CD values of the combination data sets with that of data set K, we do see that even for only a small amount of data added reduced the CD values.

We can qualitatively evaluate the performance of the model by considering the loss functions of the different permutations. Considering Figure 4.1 we see that all four data sets' loss functions minimise gradually to their respective minimums of 3.83, 3.141, 3.345, 3.657. These results are given in the order of the permutations listed, namely, K, KT, KE, and KTE. Note that the loss function summarises the losses made when models make errors, the smaller the loss function the better. We see that the loss functions converge to values that are less than 4. We especially want a loss function to converge to a single value showing that it is stable and has reached its minimum. When comparing the minimums reached by the loss functions, we see that the three combinations achieved lower loss values than the original data set. We do note that the difference in minimums is not large enough to conclude any inferences. What we do see is that there is no improvement at about the 1000th epoch or so, which may show that the model has overfitted beyond that point. This overfitting often leads to the unstable initialisation of the weights when the model is tested on unseen data. This can be adjusted by including an early stopping condition to the model that stops the model from training after the model has not improved. By comparing the loss functions and the results of the evaluation metrics, we can see the model overfitted to the test cases. This is seen by how well the model trained by considering their loss functions, yet their predictions of unseen data (own and other data sets) did very poorly. The spike seen in KE's loss function in Figure 4.1 is most likely due to the initialization of the weights causing the model to diverge at that specific epoch which produced an outlier.

One way to cater for this is to use early stopping. This just tells your trainer; if there is no improvement after n-epochs, stop training!

We show a few examples of aerial, digitisation, and predicted digitisation of the models tested on their own training sets in Figure 5.1. From these, we see the models misclassify roads and other objects, for example in Figure 1(c) we see some roads merged together and others missed entirely. We see this problem being even worse in very complex informal road structures as in Figure 4(c).

To investigate the presence of data set bias qualitatively, we can consider a few examples of the model's predictions for unseen data. Figures 4.2 and 4.3 showcase the models' performance on imagery it has not seen at all. When comparing the images next to each other we see that as we add more data, the digitisation becomes more and more refined, but still misses a lot of the informal roads in the data sets. We can therefore see that data set bias is present. We specifically see that by training the model on K, the model has learned to identify certain groupings in the data causing it to digitise vegetation and rivers. This comes from the geological difference in vegetation and rivers in each of the data sets. One solution to this may be to add the rivers into the training sets as a separate label, which will lead to the model learning what to classify as a river and what as a road so that it does not misclassify the roads.

Figures 4.4 and 4.5 then showcases another scenario, namely, what might happen if we have a situation where we have aerial imagery larger than what we have digitisations for and see if the model trained on the digitisations can accurately predict the extra roads in those images. From the figures, we see that the models do see those roads, but also miss the others.

In summary, we see that data set bias is present in our data set. The quantitative results of CD values and qualitative visual comparisons shows that there is a larger amount of data set bias present. We do see the inclusion of even a small amount of data from other geological areas reduces the data set bias in the training set. This is encouraging as the increase of T and E to that of K, may prove useful in creating a robust data set for the training of automatic informal road extraction models.

The main limitation lies in the size of the additional data sets T and E. The last point especially illustrates that the GANs-UNet model can still be improved to perform more optimally on informal road imagery.

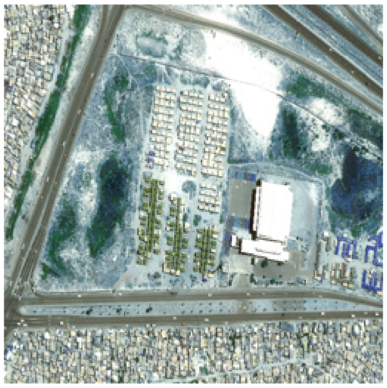
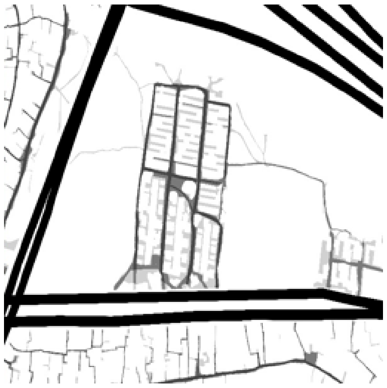
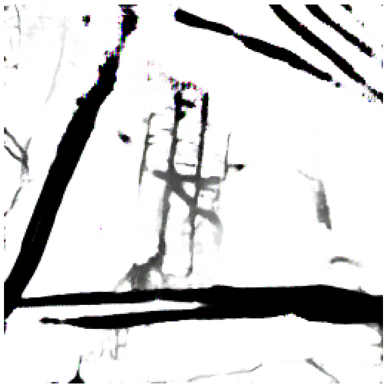
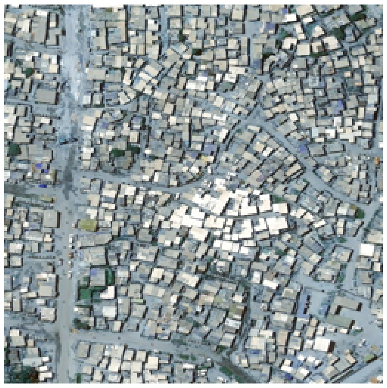
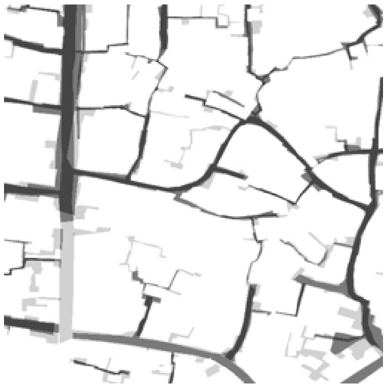
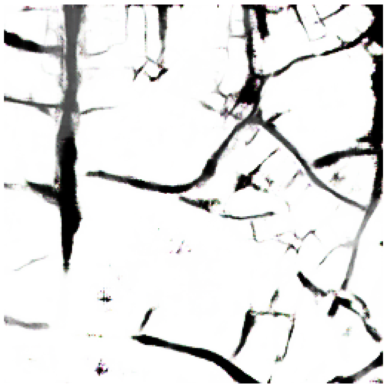
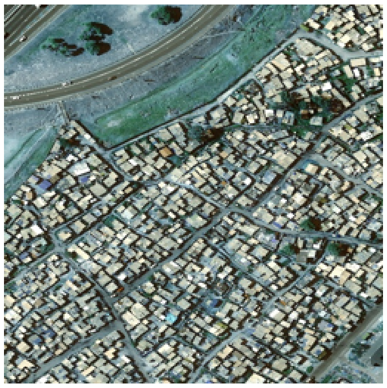
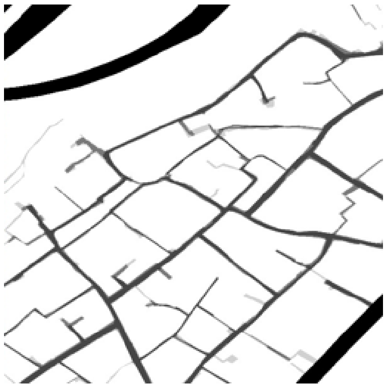
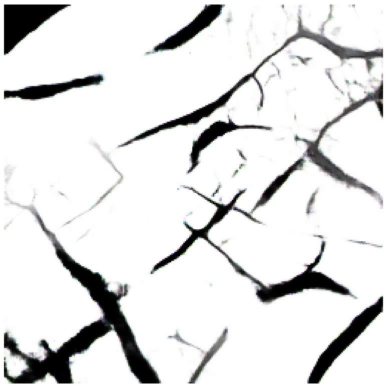
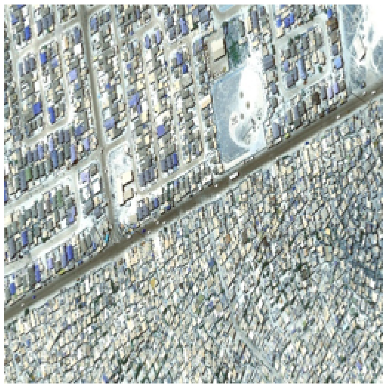
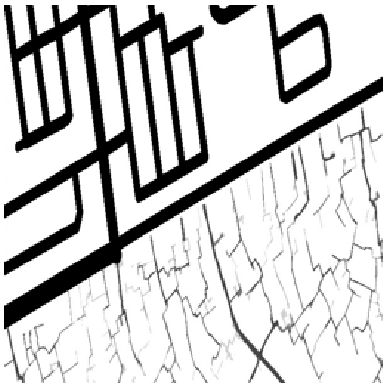
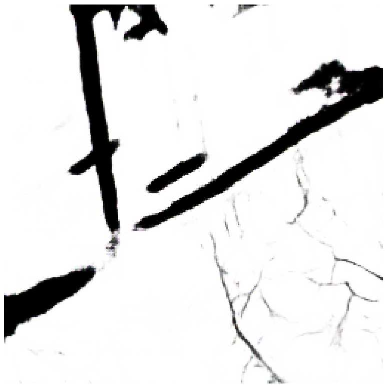
 1(a) Aerial image from K	 1(b) Digitisation from K	 1(c) Prediction from K
 2(a) Aerial image from KT	 2(b) Digitisation from KT	 2(c) Aerial image from KT
 3(a) Aerial image from KE	 3(b) Digitisation from KE	 3(c) Prediction from KE
 4(a) Aerial image from KTE	 4(b) Digitisation from KTE	 4(c) Prediction from KTE

Figure 5.1: Examples of results obtained when the models were tested on their own test samples.

Chapter 6

Conclusion

This mini-dissertation presents an investigation into the presence of data set bias in an automatic informal road extraction data set. The GANs-UNet model was used to predict the digitisations of differing geological areas in South Africa. The performance of the model is measured quantitatively and qualitatively by comparing the predictions to the manual digitisations. The model is freely available and can be easily implemented making it accessible with available high-resolution imagery. This mini-dissertation contributes the following:

- The theory behind data set bias in the application of computer vision data sets was mastered in Section 2.1.
- The theory that goes into the development of a GANs UNet neural network model was mastered in Section 2.2.
- The understanding of the challenges and importance informal roads pose to the task of road extraction was mastered in Chapter 1.
- The implementation and understanding of different standard evaluation measures used when comparing road extraction neural networks was mastered in Section 2.1.3.
- Three data sets consisting of manually digitised informal roads combined with existing formal road data sets of three different geological areas in South Africa are given in Section 2.3.
- The implementation of a GANs-UNet model on the different combination data sets was done and mastered in Chapter 3.
- The investigation on the quantitative results of the model's performance was done and reported on in Section 4.1.

- The quantitative investigation of the presence of data set bias by using the CD measure on the combination of the different data sets was done and reported on in Section 4.1.
- The qualitative investigation of the presence of data set bias by comparing different predictions of different images from unseen (and non-digitised) imagery and with incomplete digitisation areas was done and reported on in Section 4.2.

We conclude from the quantitative and qualitative results that data set bias is present in the training data. We conclude that in future work, more digitised imagery of data sets 2 and 3, and cleaning the digitisations of data set 1 can be pursued to improve the model's performance. A more in-depth investigation into the hyperparameters that deliver even better results can also be pursued. The task of automatic informal road extraction is a big need in the world we live in today due to the rapid urban expansion that is happening in the developing world, meaning that the road networks people use every day change rapidly too. When trying to solve this problem, data set bias should be accounted for by creating data sets that are good representations of the population. This will give developing countries the ability to improve the lives of the inhabitants of its informal settlements. This can be achieved when developing countries have the necessary information of the informal and formal roads in any given informal settlement to plan the steps for sustainable growth and proper service delivery in those informal settlements.

Bibliography

[1] A. Abdollahi, B. Pradhan, N. Shukla, S. Chakraborty, and A. Alamri. Deep learning approaches applied to remote sensing datasets for road extraction: A state-of-the-art review. *Remote Sensing*, 12(9):1444, 2020.

[2] R. Alshehhi, P.R. Marpu, W.L. Woon, and M. Dalla Mura. Simultaneous extraction of roads and buildings in remote sensing imagery with convolutional neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 130:139–149, 2017.

[3] D. Ao, C.O. Dumitru, G. Schwarz, and M. Datcu. Dialectical GAN for SAR image translation: From Sentinel-1 to TerraSAR-X. *Remote Sensing*, 10(10):1597, 2018.

[4] N. Babaguchi, K. Yamada, K. Kise, and Y. Tezuka. Connectionist model binarization. In *Neural Networks In Pattern Recognition And Their Applications*, pages 127–142. World Scientific, 1991.

[5] A. Barsi, C. Heipke, and F. Willrich. Junction extraction by artificial neural network system–JEANS. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives 34 (2002)*, 34, 2002.

[6] WE Blanz and S. Gish. A connectionist classifier architecture applied to image segmentation. In *Proceedings of the 10th International Conference on Pattern Recognition*, volume 2, pages 272–277. IEEE, 1990.

[7] A. Borji. Pros and cons of gan evaluation measures: New developments. *Computer Vision and Image Understanding*, 215:103329, 2022.

[8] S. Bozinovski. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica*, 44(3), 2020.

[9] S. Bozinovski and A. Fulgosi. The influence of pattern similarity and transfer learning upon training of a base perceptron b2. In *Proceedings of Symposium Informatica*, volume 3, pages 121–126, 1976.

[10] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[11] A. Buslaev, S. Seferbekov, V.r Iglovikov, and A. Shvets. Fully convolutional network for automatic road extraction from satellite imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 207–210, 2018.

[12] N. Campbell, W. Mackeown, B. Thomas, and T. Troscianko. Automatic interpretation of outdoor scenes. In *British Machine Vision Conference*, pages 1–10. Citeseer, 1995.

[13] N. Campbell, B. Thomas, and T. Troscianko. Neural networks for the segmentation of outdoor images. In *Proceedings of the International Conference on Engineering Applications of Neural Networks*, volume 1, pages 343–6. Citeseer, 1996.

[14] U. Demir and G. Unal. Patch-based image inpainting with generative adversarial networks. *arXiv preprint arXiv:1803.07422*, 2018.

[15] T. Dietterich and E. Kong. Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical report, Technical report, Department of Computer Science, Oregon State University, 1995.

[16] R. Eberhart. *Neural Network PC Tools: A Practical Guide*. Academic Press, 2014.

[17] M. Egmont-Petersen, D. de Ridder, and H. Handels. Image processing with neural networks a review. *Pattern Recognition*, 35(10):2279–2301, 2002.

[18] FI Eyiokur, D. Yaman, and HK Ekenel. Domain adaptation for ear recognition using deep convolutional neural networks. *IET Biometrics*, 7(3):199–206, 2018.

[19] I Fabris-Rotelli, A Wannenburg, G Maribe, R Thiede, M Vogel, M Coetzee, K Sethaelo, E Selahle, P Debba, and V Rautenbach. An informal road detection neural network for societal impact in developing countries. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4:267–274, 2022.

[20] J. Fritsch, T. Kuehnl, and A. Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 1693–1700. IEEE, 2013.

[21] V. Galarraga and P. Boffetta. Coffee drinking and risk of lung cancer: a meta analysis. *Cancer Epidemiology, Biomarkers & Prevention*, 25(6):951–957, 2016.

[22] J. George, L. Mary, and K.S. Riyas. Vehicle detection and classification from acoustic signal using ANN and KNN. In *International Conference on Control Communication and Computing (ICCC)*,

pages 436–439. IEEE, 2013.

[23] Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[24] A. Ghosh, N. Pal, and S. Pal. Neural network, Gibbs distribution and object extraction. In *International Symposium on Intelligent Robots*, pages 95–106, 1991.

[25] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Stat*, 1050:10, 2014.

[26] C. Gourieroux, A. Monfort, and A. Trognon. Pseudo maximum likelihood methods: theory. *Econometrica: Journal of the Econometric Society*, pages 681–700, 1984.

[27] P. Goyal, S. Pandey, and K. Jain. Deep learning for natural language processing. *Deep Learning for Natural Language Processing: Creating Neural Networks with Python [Berkeley, CA]: Apress*, pages 138–143, 2018.

[28] ED Hahn. The tilted beta-binomial distribution in overdispersed data: Maximum likelihood and bayesian estimation. *Journal of Statistical Theory and Practice*, 16(3):43, 2022.

[29] HO Hartley and JNK Rao. Maximum-likelihood estimation for the mixed analysis of variance model. *Biometrika*, 54(1-2):93–108, 1967.

[30] C. Henry, S.M. Azimi, and N. Merkle. Road segmentation in SAR satellite images with deep fully convolutional neural networks. *IEEE Geoscience and Remote Sensing Letters*, 15(12):1867–1871, 2018.

[31] MR Hestenes and E Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[32] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing systems*, 30, 2017.

[33] A. Iimi, F. Ahmed, E.C. Anderson, A.S. Diehl, L. Maiyo, T. Peralta-Quirós, and K. Rao. New rural access index: main determinants and correlation to poverty. *World Bank Policy Research Working Paper*, (7876), 2016.

[34] P. Isola, J. Zhu, T. Zhou, and A.A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[35] GG Judge and ME Bock. Biased estimation. *Handbook of Econometrics*, 1:599–649, 1983.

[36] S. Kestur, R.and Farooq, R. Abdal, E. Mehraj, O.S. Narasipura, and M. Mudigere. UFCN, a fully convolutional neural network for road extraction in RGB imagery acquired by remote sensing from an unmanned aerial vehicle. *Journal of Applied Remote Sensing*, 12(1):016020, 2018.

[37] A. Kirthika and A. Mookambiga. Automated road network extraction using artificial neural network. In *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, pages 1061–1065. IEEE, 2011.

[38] D. Kleinbaum, L. Kupper, and H.l Morgenstern. *Epidemiologic Research: Principles and Quantitative Methods*. John Wiley & Sons, 1991.

[39] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[40] H. Kong, J. Audibert, and J. Ponce. General road detection from a single image. *IEEE Transactions on Image Processing*, 19(8):2211–2220, 2010.

[41] J. Lambert. Statistics in brief: How to assess bias in clinical studies? *Clinical Orthopaedics and Related Research*, 469(6):1794–1796, 2011.

[42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[43] J. Li and M. Chen. On-road multiple obstacles detection in dynamical background. In *Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 1, pages 102–105. IEEE, 2014.

[44] Mengmeng Li, Alfred Stein, Wietske Bijker, and Qingming Zhan. Region-based urban road extraction from VHR satellite images using binary partition tree. *International Journal of Applied Earth Observation and Geoinformation*, 44:217–225, 2016.

[45] W. Liu, Z. Zhang, S. Li, and D. Tao. Road detection by using a generalized Hough transform. *Remote Sensing*, 9(6):590, 2017.

[46] Y. Liu, Q. Sun, X. He, A. Liu, Y. Su, and T. Chua. Generating face images with attributes for free. *IEEE Transactions on Neural Networks and Learning Systems*, 32(6):2733–2743, 2020.

[47] D. Livingstone, D. Manallack, and I. Tetko. Data modelling with neural networks: advantages and limitations. *Journal of Computer-Aided Molecular Design*, 11(2):135–142, 1997.

[48] J.n Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[49] D. MacKay and D. JC Mac Kay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[50] M. Mantelli, D. Pittol, R. Neuland, A. Ribacki, R. Maffei, V. Jorge, E. Prestes, and M. Kolberg. A novel measurement model based on abBRIEF for global localization of a UAV over satellite images. *Robotics and Autonomous Systems*, 112:304–319, 2019.

[51] David Marr. On the purpose of low-level vision. *MIT AI Laboratory Memos*, (324), 1974.

[52] David Marr. Analyzing natural images: A computational theory of texture vision. In *Cold Spring Harbor symposia on Quantitative Biology*, volume 40, pages 647–662. Cold Spring Harbor Laboratory Press, 1976.

[53] David Marr. Early processing of visual information. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 275(942):483–519, 1976.

[54] David Marr. Analysis of occluding contour. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 197(1129):441–475, 1977.

[55] David Marr. Vision: A computational investigation into the human representation and processing of visual information. *Inc., New York, NY*, 2(4.2), 1982.

[56] Juan B Mena. State of the art on automatic road extraction for GIS update: a novel classification. *Pattern Recognition Letters*, 24(16):3037–3058, 2003.

[57] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[58] T. Mitchell. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, 1980.

[59] M. Mokhtarzade and M. Zoej. Road detection from high-resolution satellite images using artificial neural networks. *International Journal of Applied Earth Observation and Geoinformation*, 9(1):32–40, 2007.

[60] M. Mokhtarzade and M.J. Zoej. Road detection from high-resolution satellite images using artificial

neural networks. *International Journal of Applied Earth Observation and Geoinformation*, 9(1):32–40, 2007.

[61] M. Nielsen. *Neural Networks and Deep Learning*, volume 2018. Determination Press San Francisco, CA, USA, 2015.

[62] R. Nobrega, C.G. O'Hara, and J.A. Quintanilha. Detecting roads in informal settlements surrounding Sao Paulo city by using object-based classification. In *Proceedings of the 1st International Conference on Object-based Image Analysis (OBIA 2006), Salzburg, Austria*, pages 4–5, 2006.

[63] N. Pal and S. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277–1294, 1993.

[64] T. et al Panboonyuen. Road segmentation of remotely-sensed images using deep convolutional neural networks with landscape metrics and conditional random fields. *Remote Sensing*, 9(7):680, 2017.

[65] Teerapong Panboonyuen, Peerapon Vateekul, Kulsawasd Jitkajornwanich, and Siam Lawawirojwong. An enhanced deep convolutional encoder-decoder network for road segmentation on aerial imagery. In *International Conference on Computing and Information Technology*, pages 191–201. Springer, 2017.

[66] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[67] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 658–666, 2019.

[68] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[69] F. Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[70] M. Rosenblum and MJ Van Der Laan. Targeted maximum likelihood estimation of the parameter of a marginal structural model. *The International Journal of Biostatistics*, 6(2), 2010.

[71] Y. Rui. *Urban growth modeling based on land-use changes and road network expansion.* PhD thesis, KTH Royal Institute of Technology, 2013.

[72] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[73] S. Runsten, F.F. Nerini, and L. Tait. Energy provision in South African informal urban settlements-a multi-criteria sustainability analysis. *Energy Strategy Reviews*, 19:76–84, 2018.

[74] M. Ruwaimana, B. Satyanarayana, V. Otero, A.M. Muslim, M. Syafiq A., S. Ibrahim, D. Raymaekers, N. Koedam, and F. Dahdouh-Guebas. The advantages of using drones over space-borne imagery in the mapping of mangrove forests. *PlOS ONE*, 13(7):e0200288, 2018.

[75] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *Advances in Neural Information Processing systems*, 29, 2016.

[76] J. Senthilnath, A. Dokania, M. Kandukuri, K.N. Ramesh, Ga. Anand, and S.N. Omkar. Detection of tomatoes using spectral-spatial methods in remotely sensed RGB images captured by UAV. *Biosystems Engineering*, 146:16–32, 2016.

[77] Q. Shi, X. Liu, and X. Li. Road detection from remote sensing images by generative adversarial networks. *IEEE Access*, 6:25486–25494, 2017.

[78] N. Shukla. *Machine Learning with TensorFlow.* Manning Publications Co., 2018.

[79] K.A. Stevens. The vision of David Marr. *Perception*, 41(9):1061–1072, 2012.

[80] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Cision and Pattern Recognition*, pages 2818–2826, 2016.

[81] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. In *International Conference on Artificial Neural Networks*, pages 270–279. Springer, 2018.

[82] R. Thiede, I. Nicolette Fabris-Rotelli, P. Debba, M. Lib, and A. Stein. Uncertainty quantification for the extraction of informal roads from remote sensing images of South Africa. *South African Geographical Journal*, 102(2):249–272, 2020.

[83] SK Tomer and MS Panwar. Estimation procedures for Maxwell distribution under typei progressive hybrid censoring scheme. *Journal of Statistical Computation and Simulation*, 85(2):339–356, 2015.

[84] T. Tommasi, N. Patricia, B. Caputo, and T. Tuytelaars. A deeper look at dataset bias. In *Domain Adaptation in Computer Vision Applications*, pages 37–55. Springer, 2017.

[85] A. Torralba and A. Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011.

[86] K. Tu-Ko. A hybrid road identification system using image processing techniques and backpropagation neural network. *Mississippi State Univ., Mississippi State, MS, Tech. Report MS*, 39:762, 2003.

[87] N. Varia, A. Dokania, and J. Senthilnath. Deepext: A convolution neural network for road extraction using RGB images captured by UAV. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1890–1895. IEEE, 2018.

[88] I. Vasilev, D. Slater, G. Spacagna, P.r Roelants, and V. Zocca. *Python Deep Learning: Exploring Deep Learning Techniques and Neural Network Architectures with Pytorch, Keras, and TensorFlow*. Packt Publishing Ltd, 2019.

[89] FAM Verdonck, J. Jaworska, O. Thas, and PA Vanrolleghem. Determining environmental standards using bootstrapping, Bayesian and maximum likelihood techniques: a comparative study. *Analytica Chimica Acta*, 446(1-2):427–436, 2001.

[90] J. Wang, J. Song, M. Chen, and Z. Yang. Road network extraction: A neural-dynamic framework based on deep learning and a finite state machine. *International Journal of Remote Sensing*, 36(12):3144–3169, 2015.

[91] Q. Wang, J. Gao, and Y. Yuan. A joint convolutional neural networks and context transfer for street scenes labeling. *IEEE Transactions on Intelligent Transportation Systems*, 19(5):1457–1470, 2017.

[92] W. Wang, N. Yang, Y. Zhang, F. Wang, T. Cao, and P. Eklund. A review of road extraction from remote sensing images. *Journal of Traffic and Transportation Engineering (English Edition)*, 3(3):271–282, 2016.

[93] Y. Wei, Z. Wang, and M. Xu. Road structure refined CNN for road extraction in aerial image. *IEEE Geoscience and Remote Sensing Letters*, 14(5):709–713, 2017.

[94] J. Xin, X. Zhang, Z. Zhang, and W. Fang. Road extraction of high-resolution remote sensing images derived from denseunet. *Remote Sensing*, 11(21):2499, 2019.

[95] Y. Xu, Z. Xie, Y. Feng, and Z. Chen. Road extraction from highresolution remote sensing imagery using deep learning. *Remote Sensing*, 10(9):1461, 2018.

[96] X. Zhang, X. Han, C. Li, X. Tang, H. Zhou, and L. Jiao. Aerial image road extraction based on an improved generative adversarial network. *Remote Sensing*, 11(8):930, 2019.

[97] Z. Zhang, Q. Liu, and Y. Wang. Road extraction by deep residual U-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018.

[98] F. Zhao, J. Wu, H. Sun, Z. Gao, and R. Liu. Population driven urban road evolution dynamic model. *Networks and Spatial Economics*, 16(4):997–1018, 2016.

[99] N. Zheng, Y. Shi, W. Rong, and Y. Kang. Effects of skip connections in CNN-based architectures for speech enhancement. *Journal of Signal Processing Systems*, 92:875–884, 2020.

[100] H. Zhou, H. Kong, L. Wei, D. Creighton, and S. Nahavandi. Efficient road detection and tracking for unmanned aerial vehicle. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):297–309, 2014.