2022

Ensemble Approach to the Semantic Segmentation of Satellite Images

Brendan Kent

Ensemble Approach to the Semantic Segmentation of Satellite Images

Brendan Kent

A dissertation submitted in partial fulfilment of the requirements of
Technological University Dublin for the degree of
M.Sc. in Computing (Data Science)

June 16, 2022

Declaration

I certify that this dissertation which I now submit for examination for the award of M.Sc. in Computing (Data Science), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Technological University Dublin and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: 

Date: June 16, 2022

Abstract

Automatic classification and segmentation of land use land cover(LULC) is extremely important for understanding the relationship between humans and nature. Human pressures on the environment have drastically accelerated in the last decades, risking biodiversity and ecosystem services. Remote sensing via satellite imagery is an excellent tool to study LULC. Research has shown that deep learning encoder-decoder architectures have achieved worthy results in the area of LULC, however the application of an ensemble approach has not been well quantified. Studies have shown it to be useful in the area of medical imaging. Ensembling by pooling together predictions to produce better predictions is a well known technique in machine learning. This study aims to quantify the statistical improvement that a deep learning ensemble approach can give to solving a semantic segmentation problem on satellite imagery.

Building on existing state-of-the-art approaches to semantic segmentation, such as decoder-encoder architectures, data augmentation, transfer learning, this study asks: to what extent can an average or weighted average ensemble improve the intersection over union metric for a satellite image segmentation problem in comparison to a single base model? The intersection over union measures the similarity between the ground truth labelled images and those predicted by a deep learning model. Based on a review of the literature on semantic segmentation, a U-Net architecture with a ResNet-34 decoder was used to build an average and a weighted average ensemble. A land cover classification dataset presented by the DeepGlobe Challenge was then used for training and evaluating the models.

The results have shown that an average ensemble gave a statistically greater intersection over union than a single average U-Net model, a statistical test found the effect

to be very strong, however a weighted ensemble did not improve on the results of a simple average. The results provide evidence that an ensemble approach can achieve better segmentation classifications of LULC in satellite images by using an averaging technique, however the inference time of an ensemble needs to be taken into consideration, as this study showed a well selected single model can give results as good as an ensemble and has an advantage of a much faster inference time. Further research is needed to identify methods for model selection in ensemble deep learning in order to reduce the inference time.

Keywords: Semantic segmentation, LULC classification, ensemble approach, Deep learning, U-Net

Acknowledgments

I am extremely grateful to my Supervisor Dr. Robert Ross who was so generous and provided me with lots of expertise.

I am also thankful to Dr Luca Longo, Dr Emma Murphy and Dr. John Gilligan for helping me get this far.

I am also grateful to all the classmates and lecturers I've been lucky to meet and work with, It's been a great experience.

Lastly, I could not have done this at all without my friends, a dog called Scoby, my family and Mariana.

Contents

List of Figures

List of Tables

List of Acronyms

LULC	Land Use Land Cover
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Networks
DCNN	Deep Convolutional Neural Networks
IoU	Intersection over Union
FCN	Magnetic resonance imaging
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
VM	Virtual Machine

Chapter 1

Introduction

1.1 Background

The transformation of the Earth's surface by humans is causing profound harm to the functioning of global systems. The detection, prediction and classification of land use land cover (LULC) change is critical for guiding land resource management, planning, and sustainable development. The term "land use" here refers to the purpose the land serves, for example, recreation, wildlife habitat, or agriculture and the term "land cover" is the physical material at the surface of the earth. Land use can describe how people utilize the land for the socio-economic activities.

According to Díaz et al. (2019), the affects of human actions are causing the fabric of life to unravel, posing serious risks for quality of life of people. This review goes on to report that "Since 1970, global population has doubled, per capita consumption has increased by 45%, the value of global economic activity as measured in gross domestic product (GDP) has increased by >300%, global trade has increased by ∼900%, and the extraction of living materials from nature has increased by >200%." The atmospheric scientist Paul Crutzen introduced the term "Anthropocene" in the mid 70s to describe this most recent period of earth's history (Carruthers, 2019). The results of this period are becoming more clearly visible as time passes due to the availability of LULC data and LULC tools for observation and analysis.

One of the main methods for capturing LULC data is remotely sensed imagery,

the term "remotely sensed" in general refers to using satellite or aircraft-based sensor technologies. These technologies have revolutionized how natural and human resources on the earth's surface are monitored, and they make it possible to cover very large areas.

There are two example areas which in the last year have highlighted the importance and wide variety of satellite image analysis, the first example is the pledge made by world leaders at the 26th Conference of the Parties (COP26) to end and reverse deforestation by 2030 (Barnes, 2022) and the second is the War in Ukraine.

The first example, the pledge made at COP26 by more than 100 countries with around 85% of the world's forest to end deforestation by 2030 relies heavily on the ability of climate scientists to measure that commitment. Land change science is the name given to this interdisciplinary study of changes in land cover. Using earth observation satellites, terabits of data is acquired daily which needs to be processed to help COP track future progress or deterioration of the planet's forest using clear metrics such as deforestation rate and total area deforested. Ireland is ranked the second lowest among EU member states in terms of proportion of land area that is under forest cover ("Land Use - CSO - Central Statistics Office of Ireland", 2022).

The second example for the importance of using remotely sensed imagery can be seen in the current events taking place in Ukraine. Before and after the start of the invasion, satellite data has proved to be essential for the international community in order to establish facts of what is happening on the ground which include the Russian build up of equipment pre-invasion ("Russia planning massive military offensive against Ukraine involving 175,000 troops, U.S. intelligence warns - The Washington Post", 2022) and possible evidence of war crimes after invasion had begun ("Satellite companies join the hunt for Russian war crimes - POLITICO", 2022; "Ukraine war: Satellite images appear to contradict Russian denials over Bucha atrocities — Euronews", 2022). A vital computer vision task is being able to identify and count objects on the ground in this context, a task known as 'object detection'. Deep Learning (DL) has shown considerable success in solving such a problem. This is a difficult task as it requires the knowledge of 'what' is in the image as well as 'where' it is, plus there

is need to monitor the same spot of land over a range of different seasons increasing the difficulty of detection.

Being such a rich source of data, satellite imagery can require intensive effort and time to process and classify, hence adopting an automated technique that enables machines to think without human assistance is a broad research area for LULC. Over the past 20 years, Machine Learning(ML) classification has become a major focus in LULC based problems, for example in 2001, in the USA the National Land-Cover Database (NLCD) land cover classification for a contiguous USA was produced using decision trees ("Development of a 2001 National Land-Cover Database for the United States", 2004). Recently Deep Learning (DL) concepts such as convolutional neural networks (CNN) have become the Gold Standard in the machine learning community for LULC problems (Alzubaidi et al., 2021). DL has achieved outstanding results due thanks not only to large qualities of training data but also the advances in the computational processing whereby repetitive tasks can be done simultaneously. This research will focus in on this concept of Deep Learning(DL) as applied to the problem of LULC from satellite images.

The LULC problem is solved by using a computer vision approach known as semantic segmentation. This approach tries to assign a label to every pixel in an image. In the case of satellite images, the labels will be for example forest land, agricultural land or urban land.

Since the first dedicated satellite(Landsat 1) designed to monitor the planets surface was launched in 1972, there has been many more both commercial and non-commercial launches over the years. According to the Union of Concerned Scientists Satellite Database, more than 4,852 satellites were operational on 1st January 2022 and approximately 1,052 or 22% of them were dedicated to the observation of the Earth ("Satellite Database — Union of Concerned Scientists", 2022). These remote sensing images are often available on a free access policy such as the dataset being used in this research (Demir et al., 2018), allowing researchers to evaluate them in order to improve their algorithms.

1.2 Research problem

This research uses a dataset from a paper entitled "DeepGlobe 2018, a Satellite Image Understanding Challenge" (Demir et al., 2018). In this paper, DeepGlobe aims to raise awareness of remote sensing in the computer vision community and this is the reason the large dataset is provided freely online.

In this challenge, there exists 7 classes of LULC. The problem is how can we best classify these LULC classes, which algorithm and technique works best? The problem is essentially a supervised machine learning problem, the dataset contains a mask for the images which is the ground truth upon which any machine learning approach can be measured. In order to compare results, a pixel-wise Intersection over Union (IoU) score is used as the evaluation metric. The baseline CNN model outlined in the DeepGlobe paper achieved an IoU score of 0.433. The higher this score the more accurate the model is at the classification the LULC satellite images.

This research will focus on achieving the highest possible IoU score for the DeepGlobe dataset because of it's importance in accuracy identifying LULC labels. There has been significant progress in this area of semantic segmentation in recent years thanks to the advances that have been made in deep convolutional neural networks(DCNN). Recent papers have developed DCNN models which have been able to increase accuracy by fiddling with the building blocks of these networks.

There are now a large array of different CNN architectures, this area has seen an evolution of different networks ever since "AlexNet" in 2012 won the ImageNet Large Scale Visual Recognition Challenge (Krizhevsky et al., 2012). Many of these networks are more suitable for the task of image classification, where is the goal is to assign one or more labels to the image as a whole. Semantic segmentation is a more difficult task because pixelwise precision is needed. This was due to the vanishing gradient problem which will be discussed in the Literature Review, "ResNet" came along in 2015 with a solution to this problem by adding skip connections to the networks, this proved valuable to the semantic segmentation task, winning the Common Objects in Context (COCO) segmentation competition in 2015 (He et al., 2015). Furthermore,

in 2015 a popular image segmentation model called "U-Net" was developed and outperformed the best available methods at the time when applied to biomedical images (Ronneberger et al., 2015). A U-Net architecture contains two main parts, an encoder(backbone) downsampling part and a decoder upsampling part and uses skip connections also to solve the vanishing gradient problem.

A popular approach to solving segmentation in LULC problems is to apply an encoder-decoder architecture such as U-Net with an encoder such as ResNet or AlexNet (Wu et al., 2019; X. X. Zhu et al., 2017). To be effective, these encoders are pre-trained on very large datasets of images such as ImageNet (Deng et al., 2010). This design can give decent results however using model ensembling by pooling together predictions from a set of different models should be able to produce better predictions. By applying augmentation and random crop sampling, a set of different models is produced, with each model looking at a slightly different aspect of the problem. It is important to highlight the advantage that ensembling can provide.

The goal of this research is to apply ensembling to these different models in order to better understand to what extent can it improve the IoU score. There are two specific types of ensemble approach which are of interest. The first is to average the different models predictions at inference time and the second is to do a weighted average by training a weighted average model.

1.2.1 Research aim

To quantify the statistical improvement that an ensemble approach can provide in terms of an IoU score to the semantic segmentation of the satellite images provided by the DeepGlobe Land Cover challenge (Demir et al., 2018). Two methods of ensembling using the U-Net architecture will be analysed and compared against a single U-Net result. The first method of ensembling is model averaging and the second is a trained weighted average model.

The aim will be to address these research questions:

1. Which ensemble technique receives greater results, a simple model averaging approach or a weighted model averaging approach on a withheld test set of satellite images?

2. Is there an advantage to using an ensemble approach in LULC classification when compared to a single base model which was determined to be the average of a group of models?

3. If there is an advantage to an ensemble compared to a single model, to what extent can an ensemble approach boost the IoU results overall?

4. Is there an advantage to using an ensemble approach in LULC classification when compared to a single base model which was determined to be the best from a group of models?

5. If there is an advantage compared to a selected base model, to what extent can an ensemble approach boost the IoU results overall?

1.2.2 Research hypothesis

If data augmentation and an ensembling technique are used during training then the testing IoU score of a LULC semantic segmentation model is statistically significantly higher than the testing IoU score of a model with data augmentation and without an ensembling technique.

1.3 Research Methodologies

To conduct this research I acquired quantitative data from dataset that is provided in Demir et al. (2018). This is secondary data that was collected by a satellite belonging to the American company DigitalGlobe (Now Maxar Technologies).

The data contains 7 LULC classifications; urban, agriculture, rangeland, forest, water, barren, and unknown. The data was made freely available for the purpose of releasing three challenges namely; road extraction, building detection and land cover

classification, the latter being the one of interest in this study. The dataset contains satellite images captured over Thailand, Indonesia, and India. The dataset has high resolution sub meter images and mainly focuses on rural locations. The images follow the RGB(Red Green Blue) model, with each pixel containing a value from 0 to 255 for the RGB triplet.

This study utilizes deductive reasoning starting off with the research problem of labelling each pixel with an LULC class, creates a hypothesis to falsify, collects data to measure and tests the hypothesis, essentially the theory already exists.

The research methodology used is quantitative using secondary data in the form of the satellite images and corresponding masks. The data will be in the form of unsigned integers number ranging from 0 to 255. The study will follow an empirical experiment based approach which will use the hypothesis outlined above.

1.4 Scope and Limitations

A fundamental assumption of this problem is that the ground truth masks are accurate. The training classification masks must correspond accurately to the satellite images, these have been done by professional annotators and furthermore the fact that the dataset is widely used supports the view, that this masks can be trusted.

This study is limiting its analysis to only the images contained in this DeepGlobe set, the results of the models may not be as accurate for other remote sensing locations, different image resolutions and other class types. There are also many datasets available which for each pixel may have thermal infrared readings and other bands, such as Yuan et al. (2021) which uses image from the satellite Sentinel-2 containing 12 bands with only 3 being RGB, this study is focused on just using the RGB images and so this study is only comparable to others using only RGB images.

The scope of this study will be delimited to focusing solely on only two ensemble methods of classification, the reasons for this will be outlined in the following section by following a literature review of the problem, in summary these two ensemble methods are commonly used in other domains such as biomedical imagining and they

easily implementable which is important if they are to be taken up in practise. This delimitation was also done to narrow the scope so that it was more manageable and feasibly possible to complete it within the 20 week period.

The figures reported in this research regarding any computational time taken to complete tasks are hardware specific and also the number of samples which could be processed are limited by the available resources.

1.5 Document Outline

The rest of the document is organized as follows.

- Chapter 2 - Review of existing literature.
  This chapter is dedicated to the literature survey of the previous research papers and their proposals, in the general area of semantic segmentation and the more specific application on satellite imagery. This chapter concludes by discussing the key ensemble methods used in the area of semantic segmentation.

- Chapter 3 - Experiment design and methodology
  This chapter discusses the proposed methods for LULC prediction and provides the necessary background to the model's design and required resources. Furthermore, the chapter contains a detailed explanation of the dataset, model, how it will be evaluated and how the hypothesis will be tested.

- Chapter 4 - Results, evaluation and discussion
  This chapter contains a detailed analysis of the results, describes how the research hypothesis was tested and is followed by a discussion of the key findings.

- Chapter 5 - Conclusion
  This chapter summarizes the overall research and results obtained from the experiments conducted. Furthermore, it suggests possible future work which could be performed as an extension to this research.

Chapter 2

Literature Review

2.1 Introduction

Efforts to improve the labelling of satellite images have been widely associated with machine learning and deep learning topics, however as this topic improves in small and sometimes large increments through the years, there is still scope for refining and fine-tuning learned techniques and architectures to focus on faster inference, better accuracies and more efficient models. This review will explain the topic, the solutions we currently have to solve it and the techniques we can use to improve current solutions.

In the general case, image segmentation is about using a model to assign a label to each pixel in an image, thus segmenting the image into different categories. In the case of satellite LULC classification, these different classes range from grassland to urban land and can have different total number of classes depending on which organization is followed, 18 classes if following the International Geosphere Biosphere Programme (IGBP) ("International Geosphere Biosphere Programme (IGBP) - Surface types", 2022) to 8 when using the Food and Agriculture Organization of the United Nations(FAO)'s highest level of classification (Camara, 2022). This study uses the Anderson classification which segments the image into maximum 7 classes (Anderson et al., 1976).

The literature for image segmentation is broken into two similar topics: semantic segmentation and instance segmentation. Semantic segmentation is the type of seg-

mentation used for LULC based problems, where each pixel is independently labeled. Instance segmentation goes further and tries to label each pixel as well as trying to separate different object instances, not a problem LULC is trying to solve, therefore this literature will focus on semantic segmentation in the board sense and more specifically semantic segmentation when applied to the problem of classifying LULC maps using satellite images.

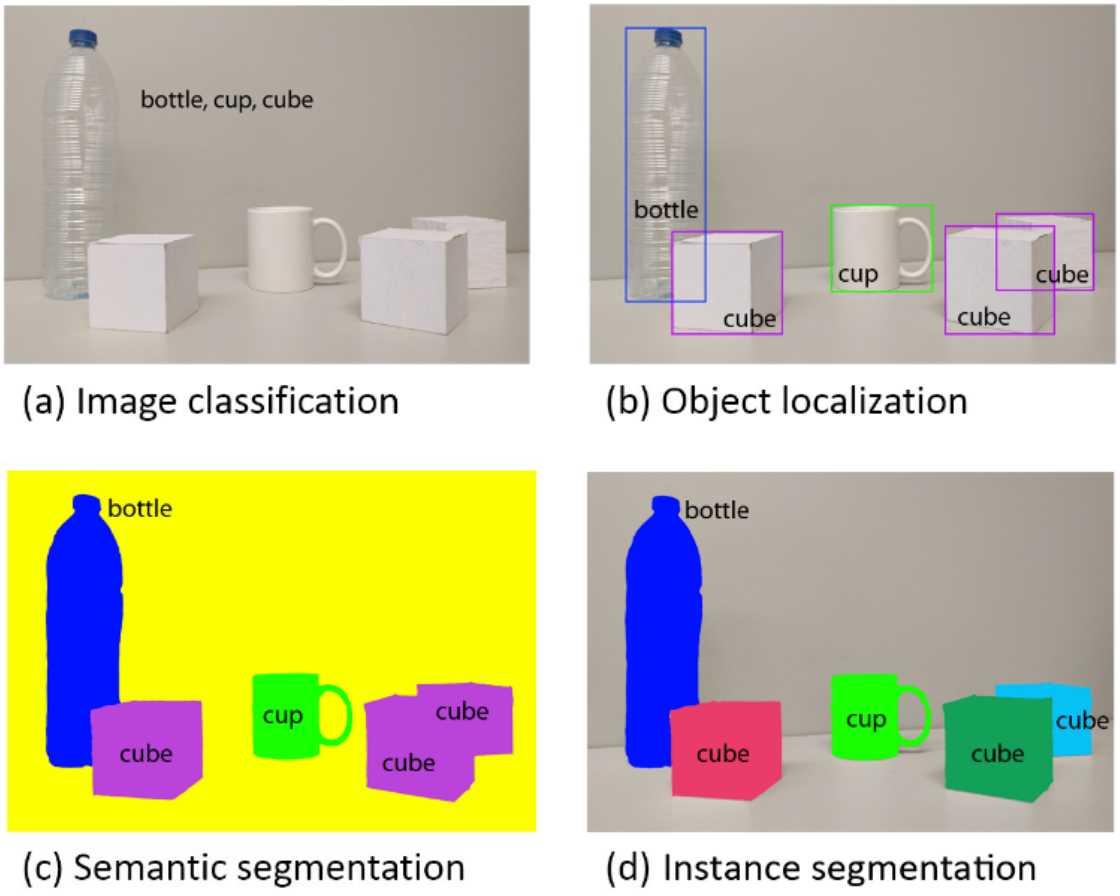


Figure 2.1: Four of the most essential computer vision tasks, arranged by level of difficulty from a) to d). Taken from Garcia-Garcia et al. (2017)

2.2 Semantic segmentation

This topic could be again split into two methods: unsupervised methods and supervised methods. Unsupervised methods which by definition try to learn without tagged

data, try to group pixels which are homogeneous in low level features (H. Zhu et al., 2016). These low level features could be for example, colour, texture or curvature. Unsupervised methods tend to be faster and can be applied to many tasks(where labelled data is not available), but performance is poorer in comparison to supervised methods especially for tasks such as scene parsing due in part to the emergence of very large scale image databases such as ImageNet (Deng et al., 2010). Supervised learning methods can take advantage of fully annotated data in very fine-grained form, and it is the supervised methods which will be discussed in depth next as this is a method which is becoming more popular over time in the literature.

Within supervised methods, we have a clear break in the trajectory of research ever since 2012. Before 2012, image segmentation was a task to be solved by algorithms such as thresholding, k-means clustering, watershed methods and graph theories such as Markov Random Field(MRF) or Conditional Random Field(CRF) (Minaee et al., 2021). There were difficulties with using these algorithms; this type of feature engineering required domain expertise in computer vision, and they could not adjust very well for an incorrect prediction (Sultana et al., 2020), however elements from these traditional algorithms have been utilized in models such as "DeepLab" which will be described later. Over the years, deep learning(DL) models were yielding better performance results and didn't require the same domain understanding. Since LeNet-5 in 1998 (LeCun et al., 1998), there was an understanding in the community that Convolutional Neural Networks(CNN or ConvNets) which extract features using "feature maps" would be effective in the topic of semantic segmentation. CNN's were proving to be very useful at classifying a whole image, they could find and recognize objects within an image, graphic processing units(GPUs) achieved training times which were 70 times faster than a dual-core CPU implementation (Raina et al., 2009).

CNN's are comprised of 3 types of layers; convolutional layers, pooling layers and fully connected layers. These CNN layers are the hidden layers of the neural network, for image data the input layer will hold the pixel values and the output layer will be the different classes which are to be predicted, a brief introduction is given in O'Shea and Nash (2015). In the next few paragraphs a review of the methods that were

first used for image classification that become invaluable for semantic segmentation is conducted, afterwards their performances will be compared.

The breakthrough in 2012 was a paper by Krizhevsky et al. (2012) that spurred the growth of a Deep Convolutional Neural Networks (DCNN) in the area of image classification, this paper achieved a considerably higher image classification accuracy on the ImageNet Large Scale Visual Recognition Challenge(ILSVRC) by using a DCNN called AlexNet. AlexNet was the first to start using Rectified Linear Units (ReLUs) as activation functions. Figure 2.2 below depicts this basic structure of a CNN.

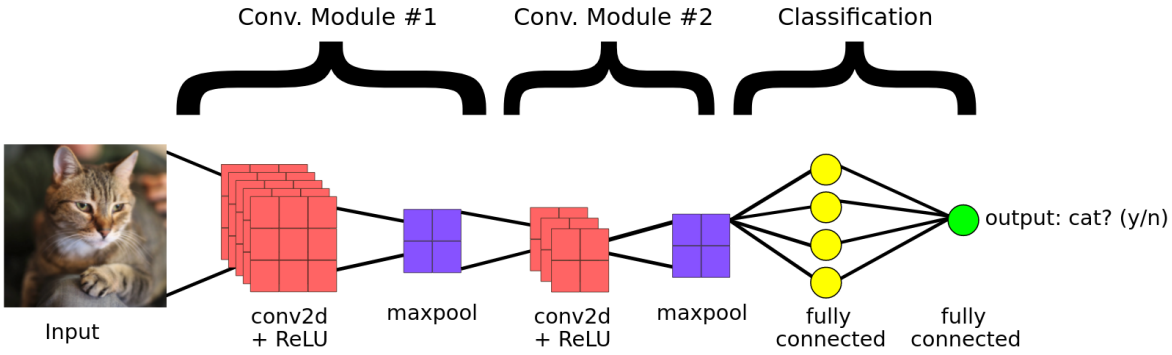


Figure 2.2: End to end structure of a basic CNN showing two convolutional modules containing a convolution plus a ReLU and a max pool, it is followed by a fully connected neural network that is trained to predict a class for the image, taken from Google (2022b).

After 2012, networks to solve segmentation problems become deeper and deeper (Simonyan & Zisserman, 2014; Szegedy et al., 2014), started to contain fully convolutional networks(FCN) (with no fully-connected dense layers) (Shelhamer et al., 2017), added more blocks and batch normalisation (Szegedy et al., 2014) and started using skip connections(He et al., 2015). Skip connections were massively valuable to semantic segmentation because the tasks require information on the "what" as well as the "where". Moreover, a criticism of FCN is that the resolution of the segmentation boundaries was still losing lots of information during the downsampling.

In 2015, U-net (Ronneberger et al., 2015) was first proposed to deal with this criticism of losing information in the resolution and also to address the state-of-the-

art segmentation models in 2015 which were performing a full CNN on each pixel or patch of the image (Cireşan et al., 2012) which were slow and had redundancy. U-net modified "the upsampling part to contain a large number of feature channels, which allow the network to propagate context information to higher resolution layers" (Ronneberger et al., 2015). U-net consists of an encoder and a decoder section; with layers of convolutions and deconvolution. The design increases the resolution of the output by adding the downsampled feature extractions to the location information during upsampling, this is achieved using the same idea of skip connections used in He et al. (2015). U-net advanced semantic segmentation by allowing fast training and good results on a small about of data with the use of image augmentation (Ronneberger et al., 2015).

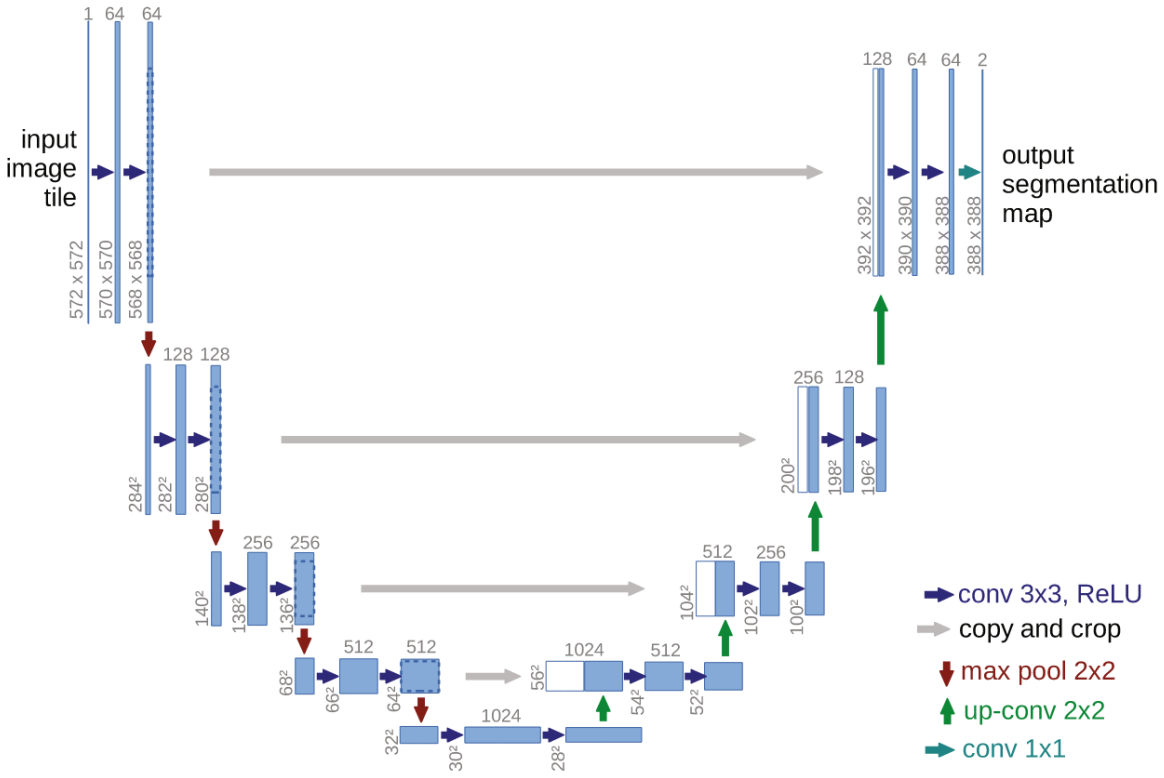


Figure 2.3: U-Net architecture, downsampling via max pooling on the left side, upsampling on the right side, there are also concatenation layers on the right side which add the spatial information, taken from Ronneberger et al. (2015)

An encoder-decoder network is always trying to do two things in practise; the en-

coder is trying to gradually reduce the feature map and capture high semantic information while the decoder is then trying to recover the "where" or spatial information of the input image.

After U-net, more decoder-encoder designs were to follow to help improve the semantic segmentation task, Segnet in 2017 (Badrinarayanan et al., 2017), works similar to add location information to the upsampling. Furthermore, PSPNet (Pyramid Scene Parsing) (H. Zhao et al., 2017) tries to solve semantic segmentation by adding a Pyramid Pooling Module between the feature map and the final prediction, this module allowed for distinguishing patterns are different scales by using different pooling scales and adding this information to the feature extraction output.

DeepLab created by Google has seen multiple versions and improvements in the state of semantic segmentation, making use of a dilated convolution, a dilated or "atrous" convolution has the advantage of using the same size of receptive field as a normal convolution but with a lower amount of parameters, which will lower the computational cost. DeepLab version 1 (DeepLabv1) (L.-C. Chen et al., 2014) uses this atrous convolutions in the upsampling to create feature extraction maps combined with Conditional Random Field (CRF) to improve localisation accuracy. Version 2 (DeepLabv2) (L.-C. Chen et al., 2016) uses a new technique called Atrous Spatial Pyramid Pooling (ASPP), which applies this diluted convolution to the feature map using different sampling rates and adds the output together, greatly improving accuracy for objects with different scales in the input images. DeepLabv3 and DeepLabv3+ (L.-C. Chen et al., 2017; L.-C. Chen et al., 2018) tries to match more the encoder-decoder relationship seen in U-net to try to capture sharper object boundaries, using features from v1 and v2 except the CRF combination, and introduces something called dilated separable convolutions, which dilutes the convolutions in two dimensions. The solution proposed in DeepLabv3 is to use a Xception network model as the backbone to construct the features.

The main point of DeepLab in general is to try to circumvent the issue of convolutions being very local and not having global information of the image, dilated convolutions and spatial pyramid pooling are methods used to give the model more

global information to help it label the pixels. This is known as enlarging the receptive field. After DeepLab, there were many flavours of DCNN which tried to solve the same problem.

Between the years 2015 to 2018, there was a spike in these so-called "classical semantic segmentation" models according to Mo et al. (2022) and a table from their review is shown in Table 2.1 that highlights some of the most successful methods.

Method	**Publish**	**Year**	**Method**	**Publish**	**Year**
FCN	CVPR	2015	Dilated Convolutions	ICLR	2016
U-net	MICCAI	2015	RefineNet	CVPR	2017
DeconvNet	ICCV	2015	DUC	WACV	2018
SegNet	TPAMI	2015	ICNet	ECCV	2018
ERFNet	TITS	2018	BiSeNet	ECCV	2018
PSPNet	CVPR	2017	CCNet	ICCV	2019
Deeplab v1	ICLR	2015	AdaptSegNet	CVPR	2018
Deeplab v2	TPAMI	2018	EncNet	CVPR	2018
Deeplab v3	arXiv	2016	Large Kernel Matters	CVPR	2017
Deeplab v3+	ECCV	2018			

Table 2.1: Classical models of semantic segmentation based on deep learning taken from Mo et al. (2022)

This paper studies the usage of weakly-supervised methods applied to this classical models as a way to reduce the economic and time cost of the pixel level annotation, which is a burden to preparing good deep learning training data. The authors say it's the quality of the annotations which is preventing further segmentation performance improvements. The general method used for weak supervising is to first generate a rough heat map as an original rough mask and then apply clustering algorithms to refine that mask. This rough map can be set up by simply pointing or a scribble to what is in the image and labelling it rather than explicitly label each pixel. This research outlines well the practical problems in using a semantic segmentation model,

how poor performance can be when the domain shifts compared to a human observer (Hoffman et al., 2016), furthermore how geographic regions and weather conditions can affect results significantly.

In the last few years, the transformer model first introduced in the paper "Attention is all you need" (Vaswani et al., 2017) started to become popular for semantic segmentation, as opposed to DCNN models, this approach allows for modelling the global context at the first layer of the network such as with "Segmenter" (Strudel et al., 2021). The transformer architecture started overtaking recurrent neural networks(RNN) across the natural language processing(NLP) tasks from 2017 onwards. "Segmenter" works by splitting the images into patches and uses linear patch embeddings as input tokens to the transformer encoder, upsampling is then done to get pixel level scores or as an alternative a mask transformer can be used to further improve the performance of the decoder. This method has achieved state-of-the-art results on two well known datasets; ADE20K (B. Zhou et al., 2016) and Pascal Context (2014) and was competitive on the Cityscape (Cordts et al., 2016) dataset.

2.3 Semantic segmentation metrics

Before comparing the performance of the various models already introduced, the metrics used to commonly evaluate the performance of the segmentation models will be discussed. As well as quantitative metrics to determine the accuracy of each segmentation model, the inference time and memory footprint are important measures to take into account when evaluating models (Minaee et al., 2021).

The most important metric to gauge the accuracy of a model for semantic segmentation in literature is Intersection over Union (IoU) also known as the Jaccard Index, it is " defined as the area of intersection between the predicted segmentation map A and the ground truth map B, divided by the area of the union between the two maps, and ranges between 0 and 1"(Minaee et al., 2021)

$$IoU = J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.1}$$

In this research, a land cover challenge data set provided by Demir et al. (2018) is used, the IoU is defined for each class(forest, urban, grass etc.) as:

$$IoU_j = \frac{\sum_{i=1}^{n} TP_{ij}}{\sum_{i=1}^{n} TP_{ij} + \sum_{i=1}^{n} FP_{ij} + \sum_{i=1}^{n} FN_{ij}}$$

n: number of images

j: current class

TP_{ij}: number of pixels in image i that are correctly predicted as class j

FP_{ij}: number of pixels in image i that are wrongly predicted as class j

FN_{ij}: number of pixels in image i that are wrongly predicted as any class other than class j

(2.2)

It is then averaged across all classes and entitled $mIoU$ as it occurs in all literature:

$$mIoU = \frac{1}{k} \sum_{j=1}^{k} IoU_j \tag{2.3}$$

Another common metric in the literature is Pixel Accuracy, this is the ratio of correctly classified pixels divided by the total number of pixels, however when a dataset has an imbalanced number of classes, this metric does not capture well the poor performance on some classes.

Finally, a metric which is common in papers where images are segmented into multiple classes is the F1 score also known as the Dice Similarity Coefficient. This is essentially a measure of overlap between two samples. It ranges from 0 to 1 like the IoU score, where 1 is perfect and complete overlap. The Dice Coefficient(DC) is calculated as (Taghanaki et al., 2021):

$$DC = \frac{2TP}{2TP + FP + FN} = 2\frac{|A \cap B|}{|A| + |B|} \tag{2.4}$$

2.4 Semantic segmentation model comparisons

Before comparing the different methods used to perform semantic segmentation, there are several popular semantic segmentation datasets which are commonly used for eval-

uating the models outlined previously. Below is a list of those mentioned within this piece of research, a few have been mentioned already when discussing the "Segmenter" transformer model. These first three datasets are seen as benchmark datasets in the semantic segmentation community.

- ADE20K (B. Zhou et al., 2016) - standard scene parsing dataset, which contains 20,210 images for training and 2000 images for validation, 150 total objects such as dog, cow, person, road, sky etc.
- Cityscapes (Cordts et al., 2016)- collected from 50 European cities, there is a coarse and a fine set of 2048x1024 resolution images. 30 classes in total, again made up of road, bus, sky, person etc.
- PASCAL Context (Mottaghi et al., 2014) - scene images. 400 classes with 59 which are the most common.
- EM segmentation challenge (ISBI, 2022)- biomedical imaging used in Ronneberger et al. (2015) has 30 images with binary masks with either the cell or the membrane.
- DeepGlobe land cover (Demir et al., 2018)- dataset used in this research, captured over Thailand, Indonesia, and India, 1,146 high resolution images (2,448 x 2,448), RBG format with a pixel resolution of 50cm. 7 classes.

Below in Table 2.2, a brief comparison of the methods and datasets described above is shown. In this table, over time transformers are becoming the state-of-the-art methods to do semantic segmentation. At the time of writing, a transformer based approach currently holds the best IoU score for the benchmark dataset ADE20K (B. Zhou et al., 2016). Transformers are replacing the previous state-of-the-art solution which was CNNs especially when large training sets are available. The winning authors of Z. Chen et al. (2022) have proposed a Vision Transformer Adapter (VIT-Adapter). The idea of being able to increase the receptive field is the winning ingredient. Similar to "Segmenter" (Strudel et al., 2021) and ever since 2020's paper "An image is worth

Method	**PASCAL-Context** (2014)	**CityScapes** (2016)	**ADE20k** (2016)	**EM** (2022)	**DeepGlobe** (2018)
U-Net (2015)				77.5	
Long U-Net (2021)		57.3			
Attention U-Net (2020)			44.88		
SegNet (2017)		57.0			
PSPNet (2017)		80.2	44.94		
DeepLabv1 (2014) ResNet18					43.33
DeepLabv3+ (2018)	48.5	82.7	46.47		
Segmenter (2021)	59.0	81.3	53.63		
ViT-Adapter-L (2022)			60.5		

Table 2.2: Semantic segmentation methods based on deep learning, all values reported in mIoU as a percentage.

16x16 words" (Dosovitskiy et al., 2020), transformer solutions have topped the IoU charts.

In terms of non-transformer methods, U-Net and SegNet have a similar architecture having a symmetrical design with skip connections, however DeepLabv3 has better IoU results than both those methods. The use of spatial pyramid pooling atrous convolutional layers allowing DeepLabv3 to extract dense feature maps to capture long range contexts is the main reason for better performance.

2.4.1 Semantic segmentation on satellite data

In terms of semantic segmentation in satellite imagery, the theme tended to follow the general problem of semantic segmentation, deep learning(DL) models have offered superior performance compared to SVM (Support Vector Machines), conventional NNs (Neural Networks), RF (Random Forest), CRF (Conditional Random Fields) and other supervised classification methods (Neupane et al., 2021).

However, DL based models have included parts of these traditional methods to support their models, using then either for pre-processing or post-processing. A popular post-processing method to CNN based models was to apply CRF at the end in

order to over-come the "salt-and-pepper" noise effects (Sherrah, 2016; W. Zhao et al., 2017). Within CRF, the most used is Linear Chain CRF, because the linear chain lowers the high computational cost of using CRF, in general CRF comes with a high computational cost even with Linear Chain CRF. Other studies have used de-noising filters as a post-processing step, such as Poomani et al. (2021) where a Wiener filter is applied.

As mentioned in the previous section, encoder-decoder architectures such SegNet, FCN, U-Net and DeepLabv3+ improved the problem of boundary pixel classifcation in satellite imagery. They minimized the problem by using both lower-level and higher level information coming from the skip connections. The convolutional backbones for these architectures can vary, the most popular architectures in the literature according to Neupane et al. (2021) are FCN, U-net, Segnet and DeepLab and the most popular backbones are ResNet and VGG. The figures from this review are shown below.

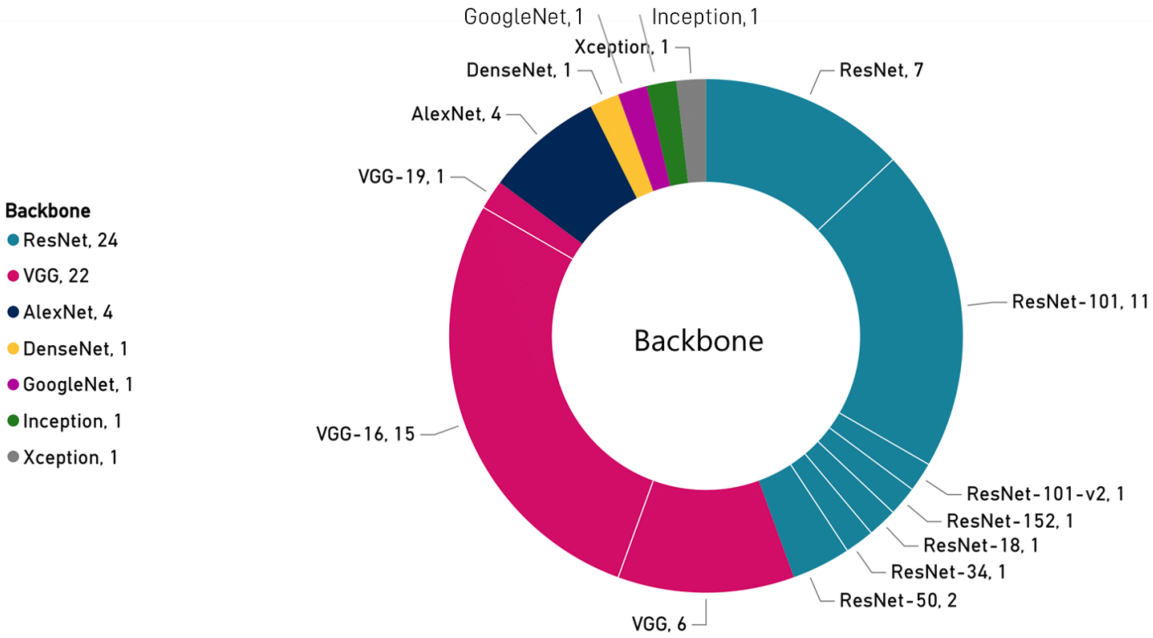


Figure 2.4: Overview of the popularity of backbones seen in papers using deep learning based semantic segmentation of urban features in satellite images. Taken from Neupane et al. (2021),

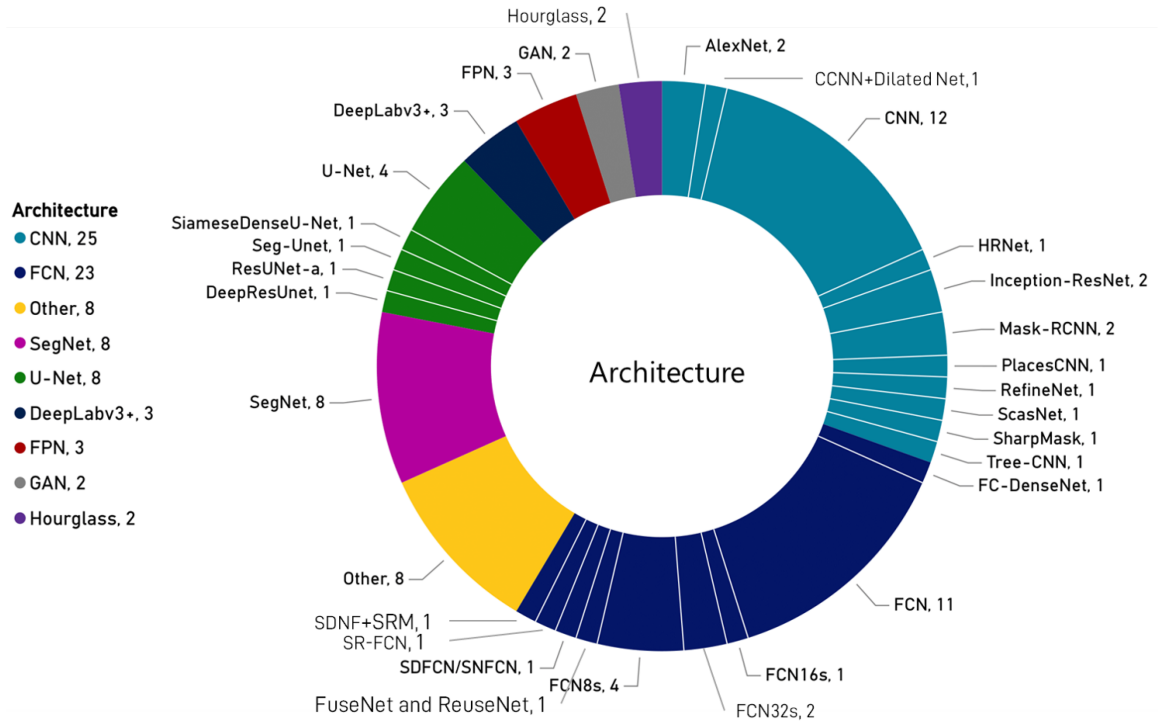


Figure 2.5: Overview of the most popular architectures utilized by papers describing deep learning based semantic segmentation of urban features in satellite images. Taken from Neupane et al. (2021),

The benchmark datasets used for satellite or remote sensing segmentation differ from those of general semantic segmentation, they can sometimes contain many more channels than the typical RGB image format, satellite data can have channel such as near-infrared, Aerosols, water vapour, cirrus cloud and more (Helber et al., 2017). A few benchmark datasets in the area of satellite imagery segmentation (or remote sening as the data may be sourced via an aeroplane) are listed below:

- Vaihingen - Germany, 33 high resolution(2,000 x 2,000), 9cm per pixel, containing 5 channels including near-infrared, red, green, DSMs, and nDSMs. 16 tiles have ground truths. 6 classes (impervious surfaces, building, low vegetation, tree, car, and clutter) are labelled.

- Potsdam - Germany, 38 high resolution(6,000 x 6,000), 5cm per pixel, 6 channels, i.e., near-infrared, red, green, blue, DSMs, and nDSMs. 6 classes labelled (same

six as Vaihingen).

- IEEE GRSS (Geoscience and Remote Sensing Society) Data Fusion Contest 2015 (Campos-Taberner et al., 2016) - Zeebruges Belgium, using airplane. 8 classes.
- Indian Pines (Baumgardner et al., 2015) - Northwestern Indiana USA, good for crop analysis. 16 classes.
- EuroSAT (Helber et al., 2017)- Sentinel-2 satellite images, 13 bands and 10 classes.
- DeepGlobe land cover (Demir et al., 2018)- data used in this research, captured over Thai- land, Indonesia, and India, 1,146 high resolution images (2,448 x 2,448), RBG format with a pixel resolution of 50cm. 7 classes.

Besides the aforementioned CNN architectures such as U-Net, DeepLabv3 etc., there are other developments which are more specific to the satellite image segmentation. These developments mainly focus on reducing the computational complexity of the method by designing a model which has fewer layers and so fewer parameters to train. DenseNet (Huang et al., 2016) is a network designed by Facebook AI Research, it does exactly this, DenseNet can reduce the number of parameters by around 5 times compared to a state-of-the-art ResNet architecture with the same number of layers. In DenseNet, each layer obtains additional inputs from all preceding layers and passes on its own feature maps to all subsequent layers. This allows the error signal to be distributed back through the network without losing that signal when having too many layers. DenseNet trains richer patterns in each layer due to receiving all this information from previous layers.

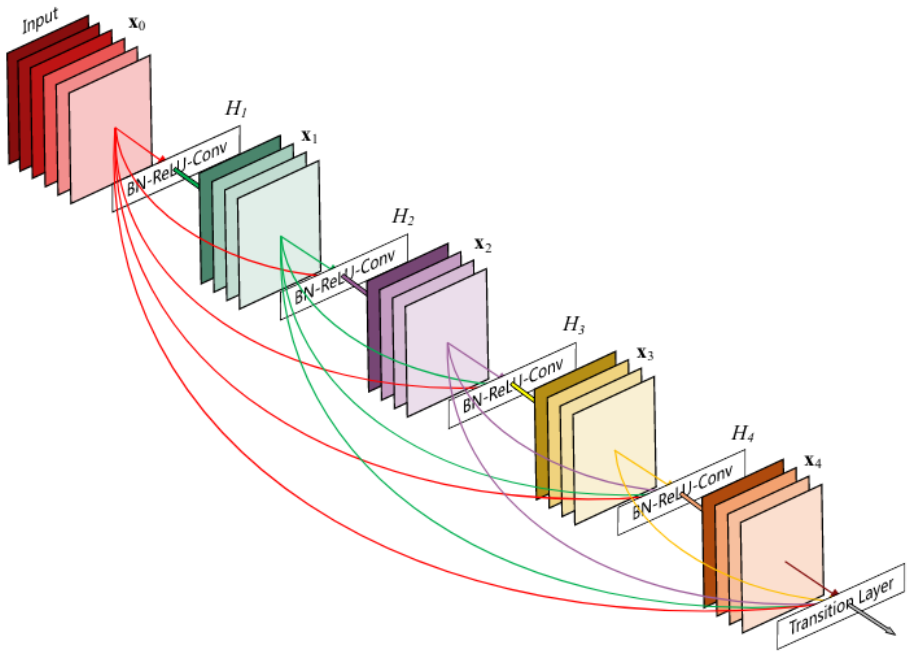


Figure 2.6: DenseNet block where each layer is connected to all previous layers, giving richer patterns. Taken from Huang et al. (2016),

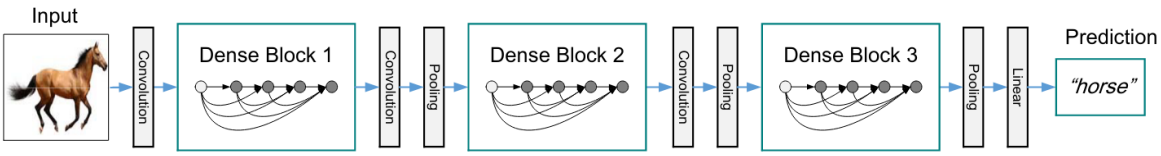


Figure 2.7: DenseNet network has multiple blocks, each block is separated by transition layers which changes the feature-maps size. Taken from Huang et al. (2016),

ShuffleNet is designed like DenseNet to also improve the computational efficiency, with mobile devices in mind. ShuffleNet claimed "The new architecture utilizes two new operations, pointwise group convolution and channel shuffle, to greatly reduce computation cost while maintaining accuracy" (Zhang et al., 2017).

One of the challenges faced in satellite semantic segmentation is the lack of training examples. Training a high quality deep network requires many examples, the labelling of these training images is usually expensive as it requires experts, whereas image classification can be helped with a crowdsourcing strategy to identify images, this is

not easily possible with remote sensing images(Yuan et al., 2021). There are many strategies to deal with this channel which will be detailed next.

To solve the issue of small training examples, Kemker et al. (2018) have used self-taught feature learning to get around the problem of using unlabelled data, once trained, the authors can then use supervised methods to fine tune the models, this is an approach similar to transfer learning. Z. Zhou and Gong (2018) have included data augmentation techniques in their training data in order to create additional labelled examples by randomly flipping and rotating images. Ma et al. (2016) proposed using a semi-supervised classification algorithm based on multi-decision labelling based on neighbours to utilize as much information as possible for unlabelled pixels, that way those labelled pixels with high confidence could be added to the supervised training.

Another challenge in applying deep learning to solve LULC segmentation tasks is using transfer learning, there are many pre-trained models available that can only be applied to the usual image format of RGB which has only 3 channels, satellite data can as shown above come in more channels, sometimes up to 200. This amount of dimensions can be computationally tough and require a lot of time to train. Novelli et al. (2017) has shown how pre-trained models do provide better accuracy as expected because these weights are often learned on very large training sets done on expensive hardware.

There is a problem with satellite segmentation models also when it comes to applying the model on different satellite data than it was trained for. In real applications of these models, limits in the training data will be exploited, changes in the test data such as different weather conditions can cause real trouble. This is called Cross-domain semantic segmentation and studied in Benjdira et al. (2019) where the overall accuracy of going from the Potsdam dataset domain to the Vaihingen domain is improved from 35% to 52%.

The trends in the results of this collection of remote sensing methods follows the general methods for semantic segmentation, albeit perhaps with a slight delay. In the general case, semantic segmentation changed to CNN model from 2012 onwards, adding residuals connections, decoder-encoder architecture using different upsampling

methods and then finally ending in the last few years with transformers.

Taking a look at results on the Vaihingen dataset, the results range from 84.46% for a U-Net (Dong et al., 2019), 85.63% for a Dense U-net(Dong et al., 2019), 87.8% for a PSPNet (Yu et al., 2018), to 90.3% for an ensemble of SegNets (Marmanis et al., 2018). With a transformer style method that could avail of capturing the global context, these authors(Wang et al., 2021) designed a UNet-like Transformer (UNetFormer) for real-time urban scene segmentation and got an overall accuracy of 91.0 with a fast inference speed to match the good accuracy. This indicts the future direction of satellite segmentation tasks, plus the score for the ensemble approach indicates its unchanged usefulness when it comes to model accuracy.

Furthermore, there are some common techniques used in semantic segmentation of remote imagery, the data augmentation technique of random cropping have proven to increase overall accuracy in a few studies (Sang & Minh, 2018; Su et al., 2022),

2.5 Ensembling Methods

The idea of ensemble learning can be traced back to a political scientist named Marquis de Condorcet who in 1785 in France proposed a theorem that said if the probability of each voter being correct is above 0.5 and the voters are independent, then the addition of more voters would increase the probability of the majority being correct until approaches 1 (De Condorcet, 1785). Also known as "the wisdom of the crowd" from Aristotle, in today's information age sites such as Wikipedia, Reddit or Quora rely on this collective knowledge.

In the area of deep learning, ensemble learning have been known to enhance the performance of models, but there are different views on the computational cost of training, the objective is to have deep ensemble models that share the best of both ensemble and deep models without being overly inefficient.

One of the main reasons ensembles work according to Fort et al. (2019) is computational wherein a learning algorithms get stuck in a local optima due to it using a local search, whereas ensemble models can overcome this issue by performing some

form of local search via a different starting point which leads to better approximation of the true unknown function.

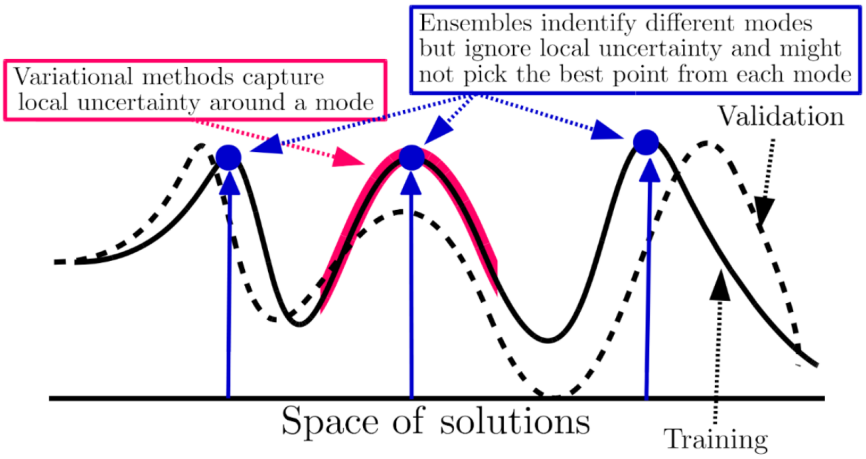


Figure 2.8: Ensembles tend to find different solutions or modes which can be missed if a model searches only locally, note the "variational methods" does poorly against validation in comparison to the ensembles. Taken from Fort et al. (2019)

The main point is if we diversify the base classifiers we have can have a successful ensemble which isn't easily prone to overfitting. These have been around for many years and there are a host of different strategies to follow this approach like bagging(bootstrap aggregating), boosting and stacking. There are some types of ensembles which are implicit to the neural network such as Dropout (Srivastava et al., 2014) where hidden nodes are randomly dropped during training.

2.5.1 Bagging (bootstrap aggregating)

The main idea in bagging is to generate independent observations by creating bags of samples with the same size and distribution as the original data. These samples may be taken with or without replacement, after training each bag of samples, the output is combined, either by majority voting for classification tasks or by averaging for regressions tasks. Some advantages include parallelization of training, decrease overall model variance and decreasing overfitting. The concerns with bagging is that it is computationally expensive.

Bagging is used by Smaida and Yaroshchak (2020) to classify medical images to help doctors diagnose eye diseases. The train three different CNN architectures; a CNN trained from scratch, a VGG16 network that was pre-trained and Inceptionv3 that was pre-trained. The authors show that combining the results improves the performance.

2.5.2 Boosting

The main idea of boosting is to convert a weak learning model into a learning model with better generalization. The methods can vary, but one popular implementation is AdaBoost (Freund et al., 1999), this solution assigns weights to the samples used to train the weak learners, it's adaptive in the sense that at each iteration, the weak learners are tweaked so that the subsequent learners focus on more difficult cases, in the end the learners are combined in a weighted sum. Boosting differs from bagging because the training can't be done in parallel, it has to been sequential and the overall model's bias is decreased and not so much it's variance. Furthermore, it is useful if a model is under-fitting.

The AdaBoost technique has been utilized with a CNN in Taherkhani et al. (2020), the proposed AdaBoost-CNN is designed to reduce the computational cost of the classical AdaBoost when dealing with large sets of training data by reducing the required number of epochs for each weak learner. AdaBoost-CNN uses transfer learning to sequentially transfer the trained knowledge of a learner to the next learner while updating the weights of the samples to improve accuracy. The authors found this approach gave them a 3.51% improvement in accuracy for the CIFAR-10 dataset (Krizhevsky, Hinton, et al., 2009) compared to a base CNN.

2.5.3 Stacking

Another approach to building an ensemble is stacking, this can be done by combining the output of multiple base models and using a classifier to compute the final predictions. In many implementations, a logistic regression mode is often used as the

combiner.

2.5.4 Ensemble Methods applied to Semantic Segmentation

Ensemble methods have been used to help improve the accuracy scores of semantic segmentation models, Shimizu et al. (2008) use an ensemble segmentation trained by AdaBoost to a liver lesion extraction problem for a competition called "Liver Tumor segmentation Challenge 2008". Their results showed the extracted regions improved as the number of weak segmentation processes T increased. They concluded that "the proposed ensemble segmentation algorithm over-extracted the surrounding tissues, such as muscle and stomach wall, when the number T of weak segmentation processes was small. However, it succeeded to reduce such false positives without increasing false negatives as the number T increased".

Nigam et al. (2018) does semantic segmentation on aerial scenes captured by a fleet of drones by using Fully Convolutional Networks(FCNs) trained with a stochastic gradient descent optimizer. The FCNs have been pre-trained on larger datasets such as ULSVRC, ADE20K and AeroScapes, so the bottom layers of these models are frozen when retrained on the aerial scenes. The idea of this paper is to study the transferability of knowledge from multiple segmentation benchmarks. These fine-tuned models from different domains are then ensembled by aggregating them to improve performance. Using these different domains with their pre-trained weights resulted in an improvement of 8.12%.

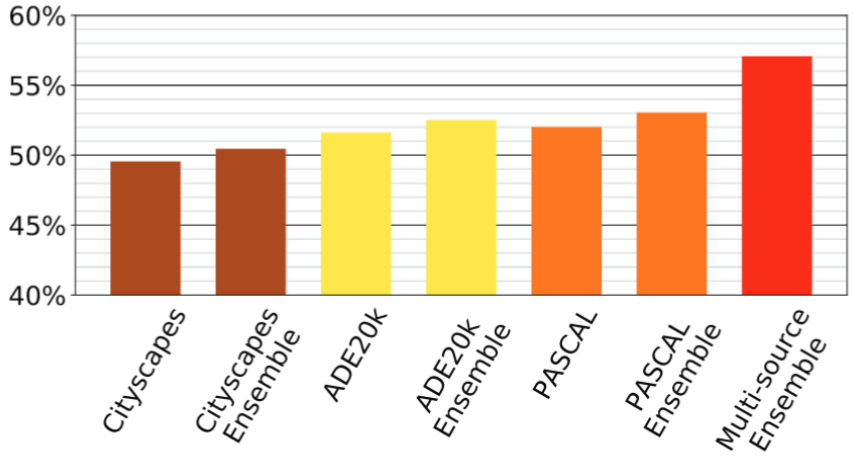


Figure 2.9: This is a comparison for accuracy scores from Nigam et al. (2018) showing the segmentation models which were either fine-tuned from a single source or multiple sources. The chart shows that an ensemble of models from a diverse source domain gives the best performance.

Tapper et al. (2020) used an ensemble on a LULC segmentation problem, the ensemble contained multiple FCNs each trained with different data, they have noted this ensemble architecture prevents overfitting and gave excellent accuracy. They claim this is a real help when training on a certain location but then focusing on how well the solution works on a global level. While this paper reports great overall accuracy figures on the 14 labels in their dataset, it does not detail how much of an impact the ensemble part had in solving their problem.

Marmanis et al. (2016) is another paper in the LULC domain which reports using an ensemble network on a ISPRS (International Society for Photogrammetry and Remote Sensing) semantic labelling benchmark dataset. Again Marmanis et al. (2016) report state-of-the-art semantic segmentation performance on aerial images, but without an assessment of the effect an ensemble had, the computational power required by an ensemble is a trade-off where the benefit needs to be known. The authors did report on the fact there is a certain amount of 'label noise' which is unavoidable in the dataset, these are errors in the dataset where the labelling has ground truth errors, it can make the dataset become saturated and obsolete. This is an important consideration for

satellite semantic segmentation since maybe it may be impossible to use an ensemble that can keep performing better if the ground truth has too much 'label noise'.

Cheng et al. (2020) have proposed a weighted ensemble of randomly cropped images to classify objects seen by a robotic arm's camera. Using a weighted ensemble improved the reliability, accuracy and stability of the image classification. The authors used a range of different architectures such as AlexNet, VGG16 and ResNet34. ResNet34 gave the authors the best individual accuracy. In order to compute the weights for the averaging of the models, the authors assigned the smaller weights to models with larger cross entropy. The question is whether this weighted ensemble approach could be extended to semantic segmentation problems.

Ensembles are popular within the MRI (magnetic resonance imaging) classification topic due to the importance of recall(or sensitivity) and to reduce as much as possible the type II errors. Shah and Madabushi (2021) used a U-net architecture to classify MRIs of the brain into tumorous and non-tumorous regions. The authors state that training models independently on different MRI modalities[1] allow each model to specialize on certain labels or regions, which could then be ensembled to achieve improved predictions. This statement was concluded to be true in their findings, which showed that ensemble gave benefits in terms of the Dice score. However, the authors did mention that there was a class imbalance in the data and suggested future work could involve data augmentation and a weighted ensemble method so as to optimally leverage individual models' strengths.

There is a lack of research on the usage of a trained weighted average on satellite imagery in the literature, it is more commonly used in the MRI and biomedical imaging disciplines, perhaps due to the lower statistical significance level in the medical industry. A type II error is much more serious in medicine and MRI diagnosis than in satellite imagery.

The area of satellite imagery may suffer from a small number of training examples for certain LULC segmentations for many reasons. The first being the data is expensive to gather, requiring great cost and coordination. The second reason for small training

[1]Modality refers to a particular type or sequence of MRI

examples is an inherit imbalance in certain LULC types, such as snow, wetlands or glaciers. It could also be imagined that more LULC types will come in the future as more land is transformed by human development.

A solution to deal with a lack of examples is to build as many training examples as possible with the usage of augmentation and use these examples to train multiple models. Two ensemble approaches have worked well in the domain of biomedical images, the first being Bagging, by averaging these models in parallel and the second is boosting, by assigning weights to each of the trained models before averaging them. Prior studies have failed to evaluate the extent to which these approaches could help LULC segmentation and in terms of the trade-off, does the high computational effort pay off?

2.6 Conclusion

Research into semantic segmentation of satellite images has mainly focused on using general semantic segmentation architectures, which are broadly thought of as an encoder network followed by a decoder network, the encoder is usually pre-trained on something like ImageNet and decoder is doing it's best to project the features learnt back onto the same pixel space as the input. A commonly used example of that is the U-Net architecture. While it is shown that an ensemble approach such as average or weighted average can help improve the metrics of IoU and accuracy, particaluarly a weighted average has been benificial to image classifations problems. It is not shown in the literature to what extent it can improve semantic segmentation for images such as satellite images. The conclusions of this literature review leads to the design experiments which will be detailed in the next chapter.

Chapter 3

Experiment Design and Methodology

Prior studies in the area of semantic segmentation of satellite imagery have failed to focus on quantifying the benefits given by using an ensemble approach. There is a high computational cost to selecting it as an approach, therefore it is of interest to know specifically what advantage an ensemble method such as bagging and boosting can have on the IoU of a benchmark satellite LULC segmentation challenge dataset.

The purpose of this study was to analyse a satellite imagery dataset and to quantify the statistical improvement of a semantic segmentation model trained with augmented images and using ensembling in terms of an IoU score. The first step was to source the data, I have chosen to use the dataset provided by the "DeepGlobe 2018, a Satellite Image Understanding Challenge" (Demir et al., 2018) as described in Section 1.2. The data can be downloaded via "Land Cover Classification" link on the challenge website [1] after subscribing to the competition or is also available via kaggle[2]. Once the dataset is extracted, there are 803 satellite images available, JPEG format images of size 2,448 x 2,448 pixels, 3 channels red, green and blue(RGB), each encoded with 8 bits for a range of 0 to 255.

For each satellite image, there is also a paired mask image. This mask is also in

[1] http://deepglobe.org/challenge.html

[2] https://www.kaggle.com/datasets/balraj98/deepglobe-land-cover-classification-dataset

RGB with 7 classes of labels, Table 3.1 below shows the method of coding for each LULC classification and is followed by a bar chart in Figure 3.1 which shows how unbalanced the LULC classifications are.

Classification	**Description**	**Red**	**Green**	**Blue**	**Colour**
Urban land	Man-made, built up areas with human artifacts	0	255	255	aqua
Agriculture land	Farms, any planned (i.e. regular) plantation, cropland, orchards, vineyards, nurseries, and ornamental horticultural areas; confined feeding operations	255	255	0	yellow
Rangeland	Any non-forest, non-farm, green land, grass	255	0	255	violet
Forest land	Any land with x% tree crown density plus clearcuts	0	255	0	green
Water	Rivers, oceans, lakes, wetland, ponds.	0	0	255	blue
Barren land	Mountain, land, rock, dessert, beach, no vegetation	255	255	255	white
Unknown	Clouds and others	0	0	0	black

Table 3.1: Satellite image LULC classifications Demir et al. (2018)

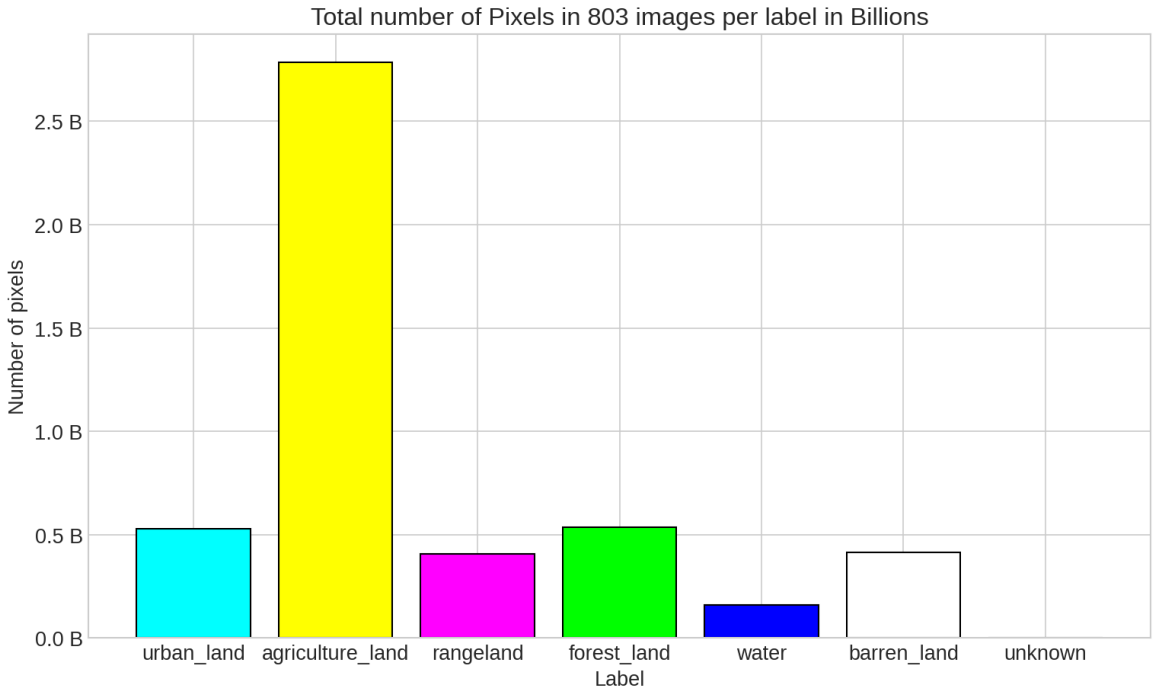


Figure 3.1: Total number of Pixels in the 803 satellite images, aggregated per label in the Billions(B) of pixels.

An example of one image and mask is shown below in Figure 3.2. The dataset

was released in 2018, taken from satellite images collected by an American company DigitalGlobe (Now Maxar Technologies). There is no information about when exactly the images were taken, whether in the summer, winter or what time of the day, we know that they were captured over the countries of Thailand, Indonesia, and India.

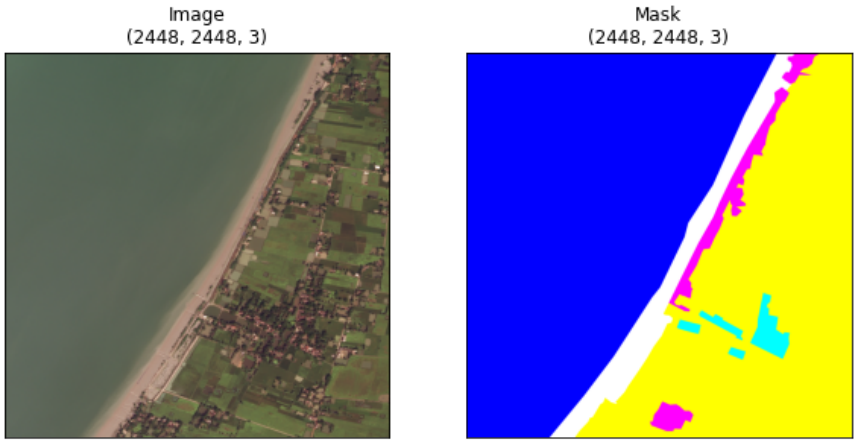


Figure 3.2: Example image and mask of image id 513968 from the Deeplobe competition dataset.

The data preparation, exploration and processing of the data was all completed with the Python v3.7.13 programming language along with the following libraries; Pandas v1.4.1 (pandas development team, 2020) for data manipulation, Matplotlib v3.5.1 (Hunter, 2007) for data visualization, TensorFlow v2.8.0 (Martín Abadi et al., 2015) for machine learning model building and experimentation, Albumentations v1.1.0 (Alexander et al., 2018) for image augmentations, Seaborn v0.11.2 (similiar to Matplotlib) (Waskom, 2021) and segmentation-models v1.0.1 (Yakubovskiy, 2019) for building pre-trained image segmentation based models. TensorFlow is the tool which performs all the underlying computation and training for deep learning, Keras[3] is the abstract level which Python makes use of to build and develop TensorFlow models. During this chapter I will discuss using some of Keras's core data structures which are layers and models. Using both Keras and TensorFlow together allows me to take full advantage of modern GPUs (graphics processing unit), these GPUs have many

[3]https://keras.io/about/

cores which allows for better computation of multiple parallel processes such as a convolutional operation.

The research was divided into two phases each with it's own goal. Phase 1 was to import the data and create many independent DCNN models, phase 2 was to use the models from phase 1 to create ensembles and evaluate them as shown in the illustration in Figure 3.3 on the next page. A short explanation of the key steps is given below:

Phase 1:

A The entire Deepglobe dataset is **split** 90:10 so that a test set is put aside.

B The training set is shuffled and **split** 90:10 into train and validation subsets. The validation set is used after each epoch to determine when to stop training, thus reducing the opportunity for overfitting.

C **Augmentation** is applied to the training set. Specifically images are randomly cropped to 512x512, with 50% vertical flips, 50% rotate 90 degrees and 20% random brightness contrast.

D N models were created using the **U-net** with a ResNet34 backbone pre-trained on ImageNet.

Phase 2:

E Ensemble method number 1, a **simple average ensemble** was created by pooling all the models and averaging their predictions at inference time.

E Ensemble method number 2, a **weighted average ensemble** was created by pooling all the models, training a new WeightedAverage Keras Layer in a training subset, using the weights learned to make predictions at inference time.

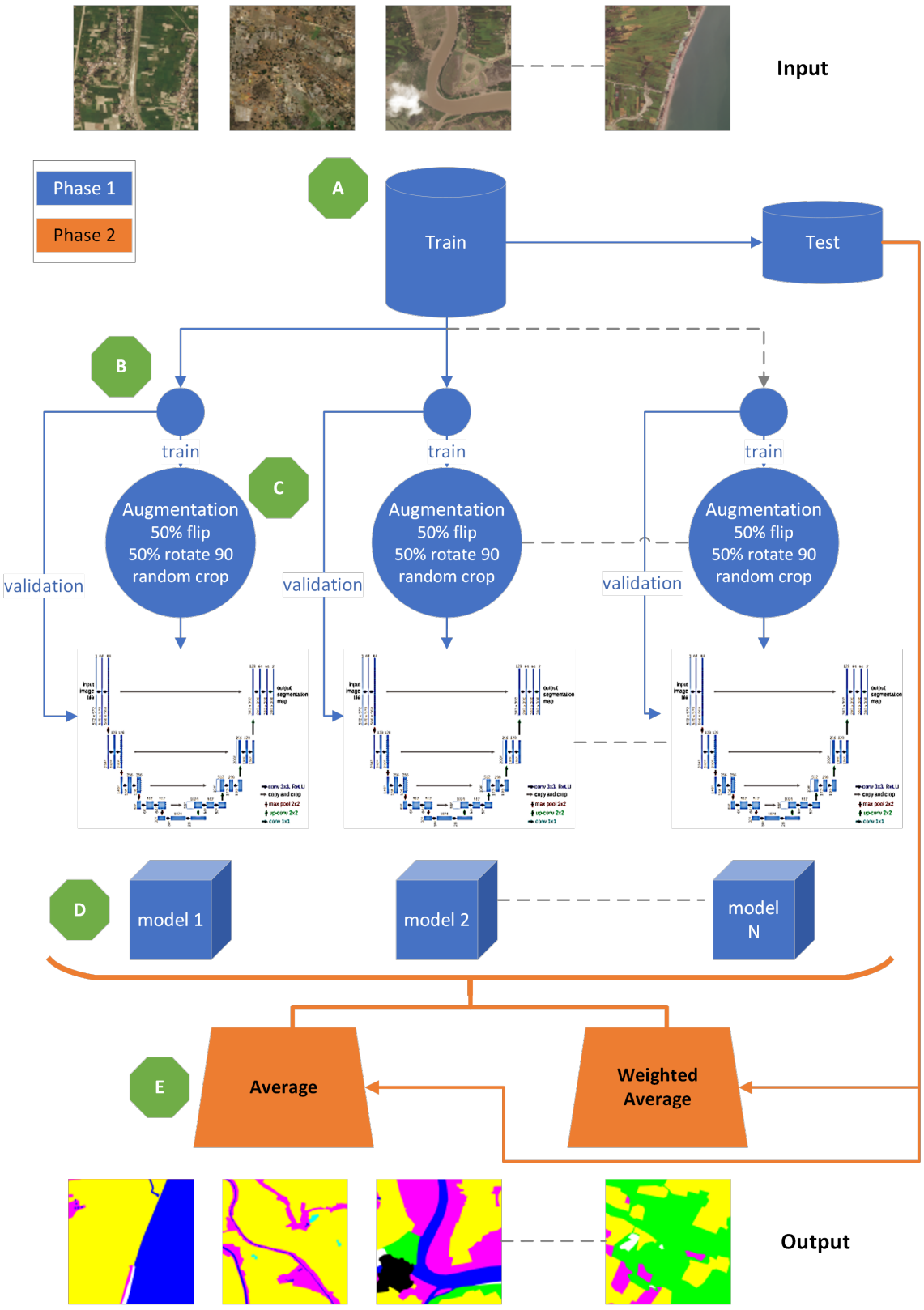


Figure 3.3: Overview of full design with explanation on previous page.

All code execution for this research was split between two machines. The first machine is a Dell OptiPlex 5040 with an i5-6500 CPU @ 3.20GHz for data preparation, exportation, testing and ensemble work. The second machine is a Google Colab VM (Google, 2022a) with a Xeon(R) CPU @ 2.20GHz which has a Tesla p100 16GB GPU which was used for model training[4].

In the next two sections I detail the design and the steps taken for phase 1 and phase 2 of this research, followed by a description of the statistical test which was carried out to either accept or reject the null hypothesis.

3.1 Phase 1: Model creation

3.1.1 Split dataset and choose architecture

The very first step in phase 1 was to put aside a group of images and corresponding masks which were going to be the test set used for evaluation at the end of model creation and ensemble creation. It was vital that none of the trained models had at any time seen these images. There were 803 total images, 80 or 10% of those were put away for testing, leaving 723 images for training.

After the test set was dropped from the dataset, the procedure for creating independent deep neural networks could be started. This procedure needed to be replicated for each model creation. I will now focus on the design of this procedure.

As outlined in the literature review, the model that is created here needs to be able to predict a label for each pixel in the input image, a task known as semantic segmentation, deep convolutional neural network(DCNN) have demonstrated particularly good results in this area. The issue of getting the error gradient back through the network during backpropagation to change the value of the weights in a DCNN has also been overcome by using Residual networks(ResNet), these networks can allow error gradients to skip one or more layers and lessen the problem of the vanishing gradient problem as seen by He et al. (2015).

[4]Available with a Colab Pro licence

An architecture which takes advantage of this skip connection modification is U-Net, it was designed for biomedical images, but it has proved to be a useful architecture for all sorts of segmentation applications, it has be shown to work fast. A segmentation of a 512 x 512 image took less than a second on a GPU according to the U-net paper (Ronneberger et al., 2015), it also worked well with low amounts of training images thanks to the usage of augmentation. As discussed in the literature review, it has achieved competitive results on the benchmark Vaihingen dataset (Dong et al., 2019), it was one of the four most popular models according to Neupane et al. (2021). The research aim was not to find the highest possible IoU score for the DeepGlobe dataset but to explore the contribution of an ensemble approach to improve that metric; therefore the choice of architecture is in fact not a vital component to achieve the research aim.

Another aspect of deep learning techniques which is useful is transfer learning. As highlighted in the literature, transfer learning means we can transfer pre-trained weights from one problem to another, it allows faster and better convergence of the model during training.

A convenient Python library which combines these 3 aspects, namely; ResNet, U-net and pre-trained weights is the 'segmentation-models' library by Yakubovskiy (2019). The segmentation-models library comes with many backbones such as different models of VGG, ResNet, Inception and DenseNet all trained on the 2012 ILSVRC (ImageNet) dataset. The library is built to work with the TensorFlow Keras framework.

A medium-sized ResNet encoder with 34 layers was chosen for this study, since ResNet was one of the most popular backbone options according to Neupane et al. (2021). Inference time and accuracy results for an ImageNet classification from the figures reported by the library authors also suggested ResNet-34 gave a good compromise when compared to other models (Pavel, 2022). This ResNet34 backbone was used in a U-net architecture in my implementation as seen in Figure 3.3.

3.1.2 U-Net architecture

The structure of the U-Net architecture is highly symmetrical, it consists of a contracting (decoding) path and an expansive (encoding) path making up a "U" shape. There are typical repeated uses of convolutions followed by rectified linear units(ReLU) and max pooling on the decoding. Then on the encoding side, there are repeated uses of convolutions, ReLUs and up-sampling. The idea is that during decoding, the feature information is increased and the spatial information in reduced. There are connections which combine this feature information through upsampling with the spatial information from downsampling, giving an overall architecture which is good at identifying high resolution features required for a semantic segmentation task.
The U-Net as implemented in Keras is made up of the following layers types:

- Downsampling side: BatchNormalization, ZeroPadding2D, Conv2D, Activation, MaxPooling2D
- Upsampling side: BatchNormalization, ZeroPadding2D, Conv2D, Activation, Concatenate and UpSampling2D.

Batch normalization is designed to apply a transformation that maintains the mean output close to 0 and having a standard deviation close to 1. The zero padding is required to allow the convolutions to work on the edges of the image by adding padding.

The Conv2D layer will apply many kernels to the whole image in order to extract features, they can come in different sizes such as 3x3 or 5x5, in the chosen U-Net all kernel size's were 3x3, except for the very first which was a 7x7. The amount of filter kernel increases as the model downsamples, each filter was trying to find out some interesting information in the image, going from 64 to 128, to 256 and to 512. This is a well-known feature of the CNN architecture which I won't detail here as O'Shea and Nash (2015) provides a great overview.

The next layer used in U-Net is the Activation layer, this layer simply applies an activation function to the input and it can also reside in the convolutional layer itself.

For CNN architectures, this is almost always a Rectified Linear Unit (ReLU) ever since AlexNet.

The final ingredient on the decoder side is the Max Pooling layer (MaxPooling2D), this Layer reduced the dimensionality of the input image, so the model has fewer parameters to train and so that the final feature map at the bottom of the "U" combines maps learned from previous layers, giving it a bigger window on the image. On the upsampling side of the "U", Keras used a UpSampling2D layer to expand the feature map information by doubling it and a Concatenate layer to add the spatial information from the downsampling side of the "U".

This U-Net architecture had pre-trained weights from ImageNet, so when using it for this LULC segmentation, all those decoder layer weights were frozen so as not to re-train them. Only the BatchNormalization layers were left unfrozen in order to allow it to adjust the mean and standard deviation seen during that current batch in both training and inference. In this way, BatchNormalization works slightly different to Layer type classes.

In order to train U-Net for the DeepGlobe satellite images, the upsampling section of the network was set as trainable. This means from the 1st UpSampling2D layer until the final Softmax Layer of the U-Net, all weights are trainable. Most of these weights that need training are filter weights on the Conv2D layers, as the network reaching it's output Softmax layer, the numbers of filters get smaller and smaller, ending with 7 filters. A Softmax layer in my context is a function which will return an array of 7 probability scores summing to 1. Each score will give the probability that for each pixel, the label is one of 'urban_land', 'agriculture_land', 'rangeland', 'forest_land', 'water', 'barren_land' and 'unknown'.

The entire model has 24,457,024 parameters of which 3,167,930 were trainable, allowing me to take advantage of all the filters learned on ImageNet. Alternative fine tuning policies may in practice lead to better optimisation of individual models, but this is left for future optimisation work.

3.1.3 U-Net optimizer and loss function

An Adam optimizer was selected to update the model via gradient descent after each batch, Adam has demonstrated considerable gains in terms of training cost and performance (Kingma & Ba, 2014), the optimizer updated the network based on a custom loss function which was made up of a Dice loss and Categorical Focal Loss. Dice loss is designed to deal with class imbalance, coming from the Dice coefficient (explained in Section 2.3), it measures the similarity between the predicted mask and the true mask. Categorical Focal loss tries to penalize more heavily samples which are easier to classify, therefore making the model pay more "attention" to labels which are hard to classify.

3.1.4 U-Net metrics

During training, after each epoch which is a full set of the batched images, I had an opportunity to use the validation set to keep an eye on how the model was performing on new data. The most important metric to monitor was IoU as mentioned in the Literature Review, this metric is widely used for semantic segmentation. Keras provide a categorical accuracy and an IoU (Intersection over Union) metric which can monitor the mean IoU of all the labels as well as monitor the IoU of each individual label. The segmentation-models library which provided the pre-trained weights for the decoder provided a IoU score metric which was also used. There is a slight difference in how the segmentation-models library and Keras library calculates their IoU score which I describe in Figure 3.4 using Equation 2.1. As seen in the Figure, the 3rd channel has zero intersection and zero union, this is a division by zero. To handle this, the segmentation-models library treats the IoU for this channel as 1.0 and therefore artificially increases the overall IoU for all channels. For this reason Keras's 'OneHotMeanIoU' was used as it did not include empty intersections. The 3rd channel in Figure 3.4 has no union or intersection similar to "Unknown" in the DeepGlobe dataset.

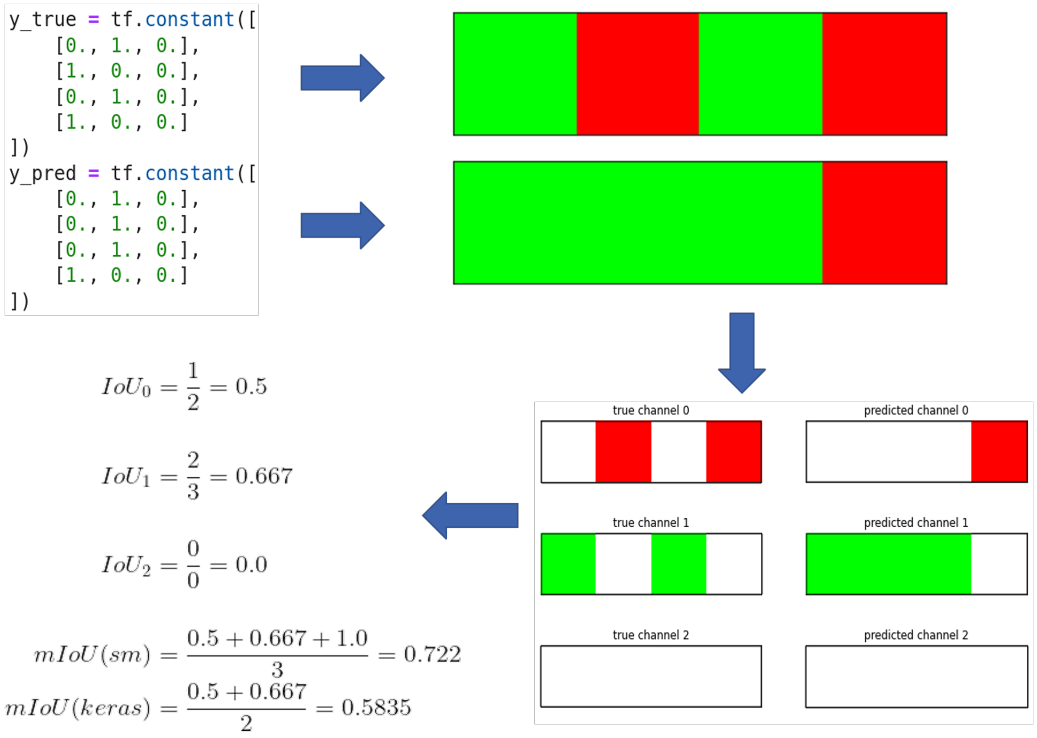


Figure 3.4: Example calculation of an IoU, using 3 channels, depicted here as Red, Green and Blue only for illustration (these could be water, urban and forest). The difference between how the segmentation-models(sm) library and Keras library calculate IoU is shown.

3.1.5 U-Net Image Processing and Augmentation

In order to train this U-Net architecture, firstly the images need to be processed and prepared for model tuning. In order to be memory efficient, in Keras I created a "Sequence" class which can perform preprocessing and batch the training samples. Using the "Sequence" class allows for only loading into memory those images currently required for training. To use the DeepGlobe training set, each image and corresponding labelled mask needed to be a size which was divisible by 32, a size of 512 x 512 was chosen based a few experiments with other values and the conclusions of Hirahara et al. (2020) which indicated 256 x 256 performed better over a 128 x 128, the authors did note that increasing the resolution will increase the training time, which was noted also by this study. Another reason for 512 x 512 being chosen was due the augmentation

performed, random cropping was to be done to each image meaning only 512 x 512 of a total 2,448 x 2,448 was to be used for each sample.

There are two data subsets for each model trained; a validation set and a training set. Both subsets are taken from the set of 723 images. The validation set was split into patches, no augmentation was carried in order to match the test(which was put away at the start of the experiment) as much as possible. The segmentation-models library has a preprocessing function that would prepare the image, however for ResNet-34, this simply returned the same image unprocessed.

The training set was subjected to data augmentation functions each time a new batch was called by the model fit function using a library called Albumentations. As indicated earlier, random cropping of the image to 512 x 512 is carried out, as well as a 50% probability vertical flip, a 50% probability of a rotation by 90°, and finally a 20% probability of a random brightness and contrast change as seen in Figure 3.5. For images for which there is no clear notion of a top like this DeepGlobe dataset, the Albumentations library recommends adding only transformations that don't cause a loss of information, non-rigid transformations were avoided.

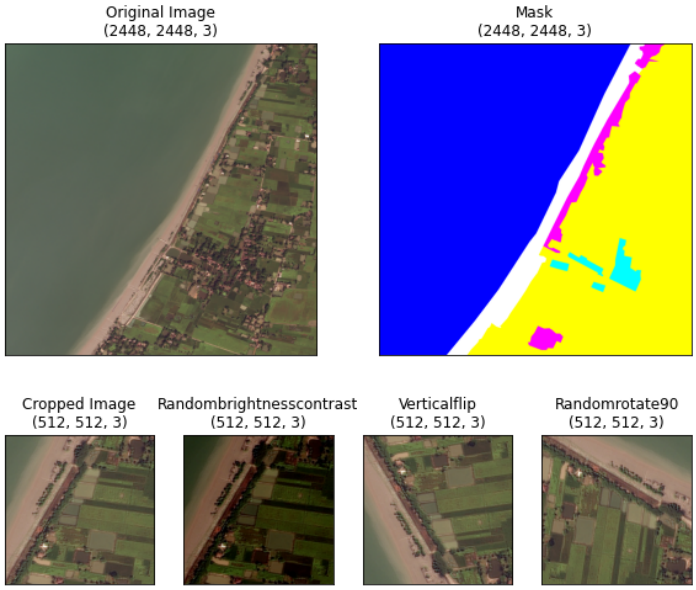


Figure 3.5: Example of augmentations on image_id 513968 satellite image with transformations made by Albumentations; cropping, random brightness and contrast, vertical flip and rotation by 90°.

3.1.6 U-Net Mask Processing

To prepare the mask labelling image for training, the RGB format needs to be encoded from these 3 channels into the 7 channels allowed by the Anderson Classification. The details of these RGB encodings were already shown in Table 3.1.

One-hot encoding was used to convert each pixel of the mask into a 7 channel image with each channel representing one segmentation label as shown below in Figure 3.6.

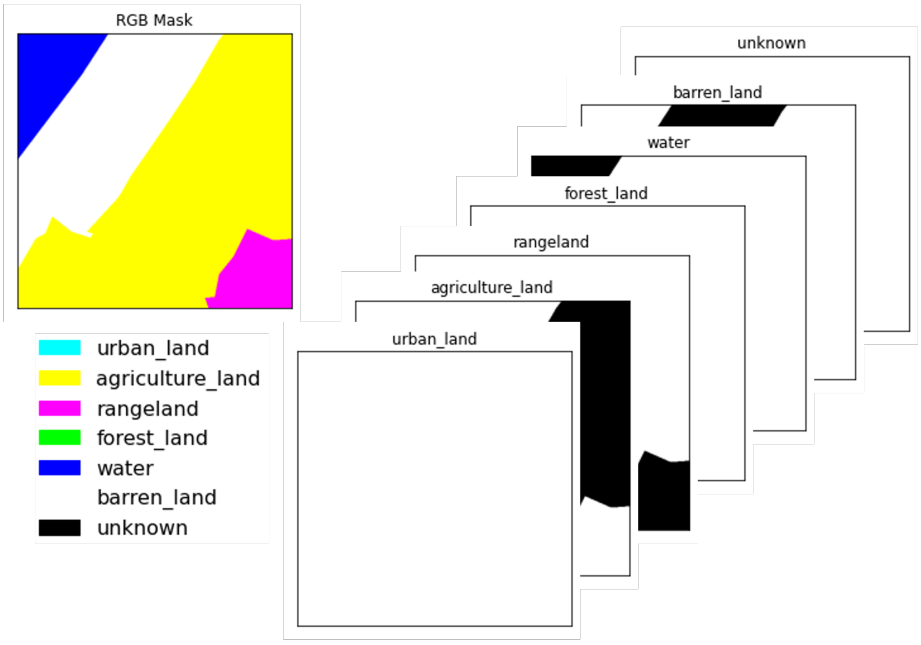


Figure 3.6: Each colour represents a colour in RGB space and is converted to a one-hot encoding 7 channel image in order to be used by the model to calculate the error gradients.

3.1.7 U-Net Training Models

Model training was carried out on Google Colab's VM. The script would firstly download the pre-trained ResNet-34 decoder by using the Segmentation Models library (resnet34_imagenet_1000_no_top.h5). 723 satellite images and corresponding masks were divided into a training and validation set as shown in Figure 3.3–meaning each model had 650 examples to train with and 73 for validation. These examples were shuffled for each model.

After the U-net model was compiled and ready to be fine-tuned to the satellite data, the 650 training images (which go through the augmentation explained previously) were fed in batches of 8, along with 73 full images (or 1168 patches) for validation, as there is no augmentation applied to the validation. This meant each training epoch would get slighty different data during each epoch, due to the augmentations.

After training was started on Colab's Tesla P100 16GB GPU, the Keras library provided a Callback API which can perform certain actions at various stages during training. I used this callback API to save the model whenever the validation set reported the highest one-hot mean IoU as reported by `tf.keras.metrics.OneHotMeanIoU`, I also used this API to make sure training stops once this one-hot mean IoU has stopped improving thereby saving needless excess computation.

After training, each model was saved to disk using the HDF5 file format with file extension '.h5'. Each model was manually inspected to make sure that each model was minimizing the loss function and that the training did not reach the max number of epochs without distinguishing some labels. In Chapter 4, I will detail the distribution of results for these models.

3.2 Phase 2: Apply Ensemble

Phase 2 of this experiment used the models saved from phase 1. There were two approaches to ensembling investigated. The first was based on the bagging ensemble, all models from phase 1 are averaged at inference time. The second is based on a boosting approach, all models from phase 1 are trained in a new model where I created a weighted average layer in Keras, this allows the best models to have more say on the final result.

The first ensemble (referred to as 'average') simply pooled together the prediction for every pixel on a test image and returned the ARGMAX[5] result of the average of all those models as shown in Figure 3.7. This model required no training. This average only ensemble model is implemented again using TensorFlow and Keras, the Average

[5]Returns the indices of the maximum values along an axis.

class in Keras takes as input a list of patches which are produced by the models, and returns a single image as output also of the same shape.

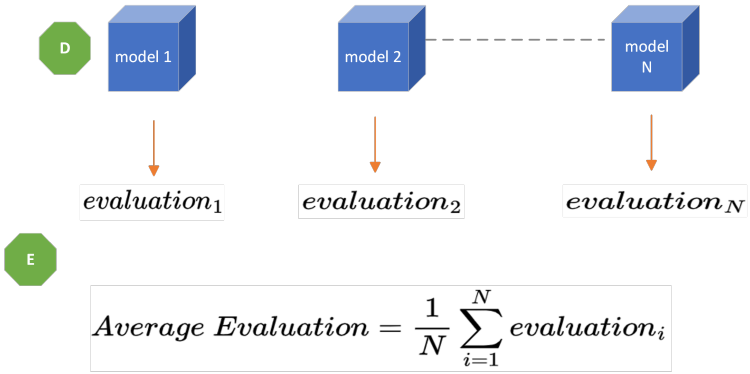


Figure 3.7: Illustration of Ensemble 1: simple average of all models. (D) Models from phase 1. (E) Ensemble

The second ensemble (referred to 'weighted') was trained, the training found a set of weights for each model which gave the maximum one-hot mean IoU score. During this training, only the 'WeightedAverage' Layer was set to trainable, while the rest of the model was frozen. In this boosting approach, the training will try to allocate weights to each 'weak' learner so that learners which give a good classification result are assigned a higher weight.

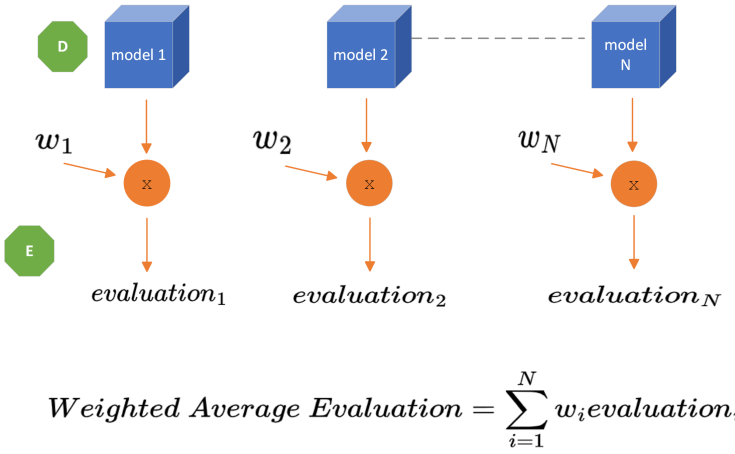


Figure 3.8: Illustration of Ensemble 2: weighted average of all models. (D) Models from phase 1. (E) Ensemble. The sum of the weights must be normalised so that they add up to 1.

3.3 Statistical Tests

In order to satisfy the research hypothesis, I needed to determine whether there was a statistically significant relationship between a single model from phase 1 (also known as the base model) and an ensemble method from phase 2.

I have stated that the Null hypothesis is that there is no significant difference between the base U-Net model and an ensemble model approach. The alternative hypothesis is that there will be a statistically significant difference in terms of an IoU score. In order not to contaminate the final t-test result, the test set which was put aside at the beginning of this chapter was only used for finding these group distributions and was not used to decide which model from the collection would be chosen as the best for comparison. The method used to select this best stand-alone model is described next. Before using this 'put away' test set, the two base models needed to be selected from the pool of 64 models. A random sample of 160 images was chosen from the complete training set of 723 images in order to select a model with the best mIoU ('base_max') and average mIoU ('base_avg'). An illustration of how both base models were selected is shown in Figure 3.9.

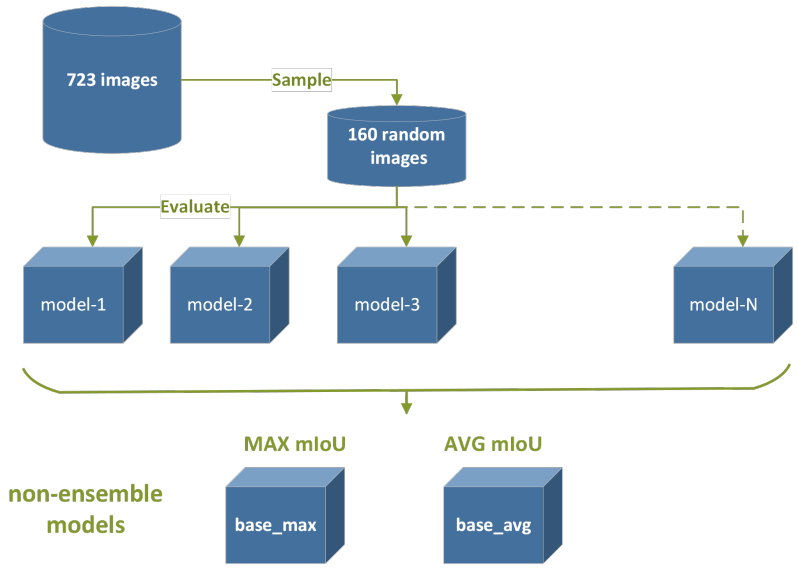


Figure 3.9: Illustration of method used to select both base models which were lated used to test the research hypothesis.

The first base model ('base_max') was the one which reported the max mIoU score for this random set was chosen for comparison, this way the alternative hypothesis would be more difficult to prove, since the model selected for comparison was above the average. Futhermore, the computational time required to generate this pool of models is as large as the ensemble, but the inference time of the single will be much faster so if the effect is not large, this comparison would not be practical. This was an important metric to understand according to Minaee et al. (2021) when evaluating models and comparing them.

The second base model ('base_avg') was chosen by finding the model which had the closest mIoU to the mean of all the models. This single model would simulate a training of one model, which has a short training period and also a short inference time.

To quantitatively analyse the data gathered from the 2 distributions (distribution 1 being the single U-net, distribution 2 being the ensemble), a paired student t-test was chosen to calculate the probability of obtaining the observed difference between the two distributions and telling us whether the observed difference is due to chance or is real. For both distributions, the results were shown to follow the normal distribution as we will see in Chapter 4.

The 2 distributions were created by splitting the test dataset which was put aside at the beginning of this Chapter. The test set contains 80 full 2,448 x 2,448 images, there are 16 patches in each image (using a 512 x 512 size patch). To fill these distributions with IoU metrics, the test images were firstly split into patches, secondly shuffled and thirdly divided into 40 test sets. Each of the 40 test sets were evaluated by all saved models; 'base_max', 'base_avg', 'average' and 'weighted'.

Figure 3.10 below describes the process to compare models to test the research hypothesis.

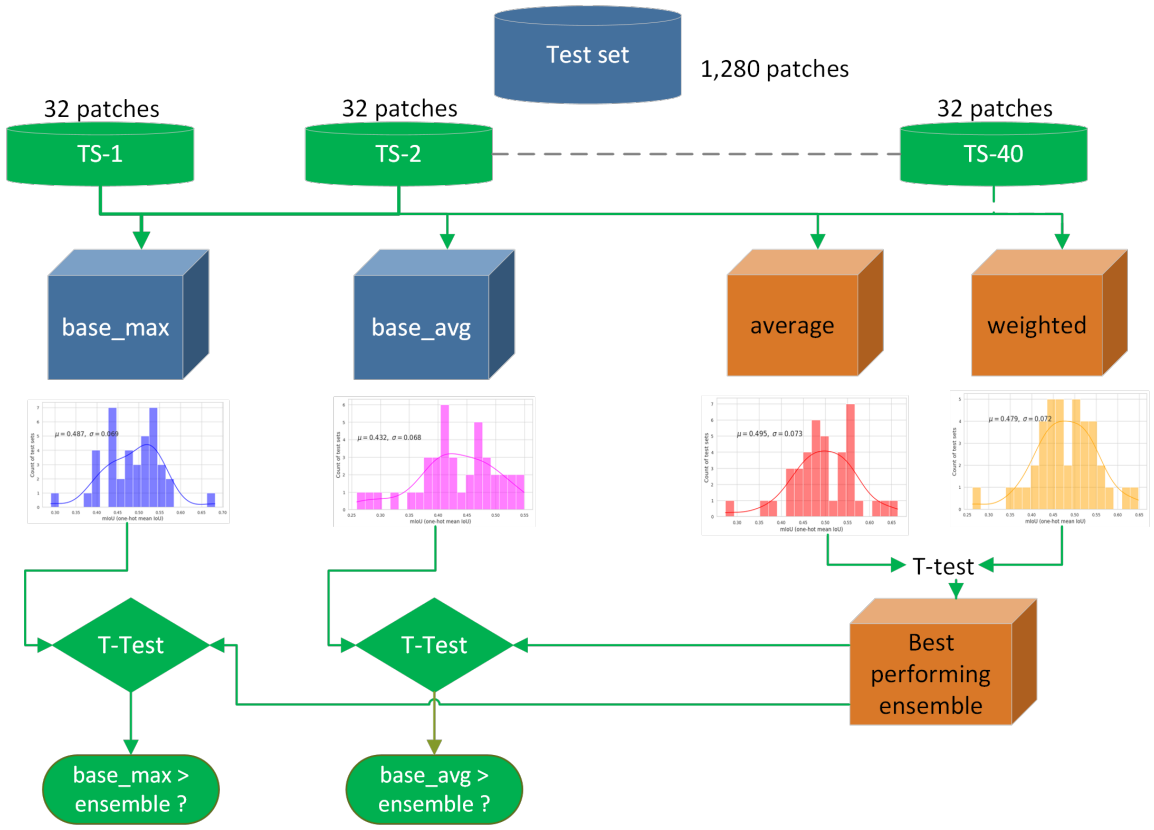


Figure 3.10: Illustration of the statistical test to accept or reject the Null hypothesis using a split of the test data. The non-ensemble model could be chosen by average or maximum mIoU and the ensemble method could be an average or a weighted average method.

The statistical test produced a test statistic and a p value, the p value gave the probability of obtaining the result of the null hypothesis, remembering that the null hypothesis for this study is that there is no effect visible on the mIoU score of the test data when using ensemble methods.

A significance level(alpha) of 5% was chosen for the t-test, meaning the risk of rejecting a true null hypothesis that I am willing to take is 5%.

Chapter 4

Results, Evaluation and Discussion

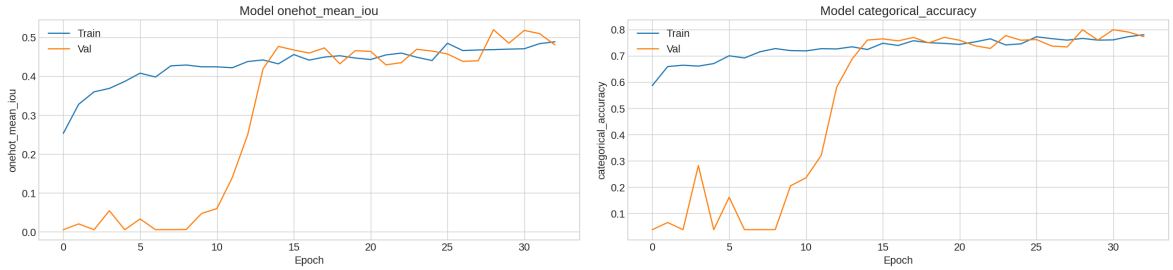
This chapter presents the key results from the experiments described in the previous chapter. The chapter is divided into the following sections; Phase 1, Phase 2 and Statistical Tests. Phase 1 reports the results from training the semantic segmentation U-Net models on the satellite images and how the two base models were selected. Phase 2 details the evaluation of the two ensemble methods on the test set and compares their performance using a statistical test in order to uncover the model which performs best. The Statistical Test section addresses the main hypothesis of this research, that an ensemble approach has a positive effect on the mean intersection over union metric for LULC classification of satellite images. The chapter concludes with a reiteration of the research questions and a discussion into how the findings address those research questions, including any shortcomings found during the work

4.1 Phase 1

4.1.1 U-Net Models

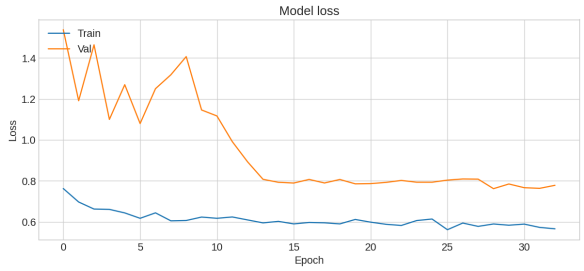
The U-Net models were trained on Google Colabs GPU to speed up training. 64 models in total were trained, each separately on training sets which were randomly sampled. During training, each epoch took about 300 seconds, the average number of epochs over all the models was 26.7 with a standard deviation of 3.4. Figure 4.1 is an

example of the training of one model, similar results are seen for all models.



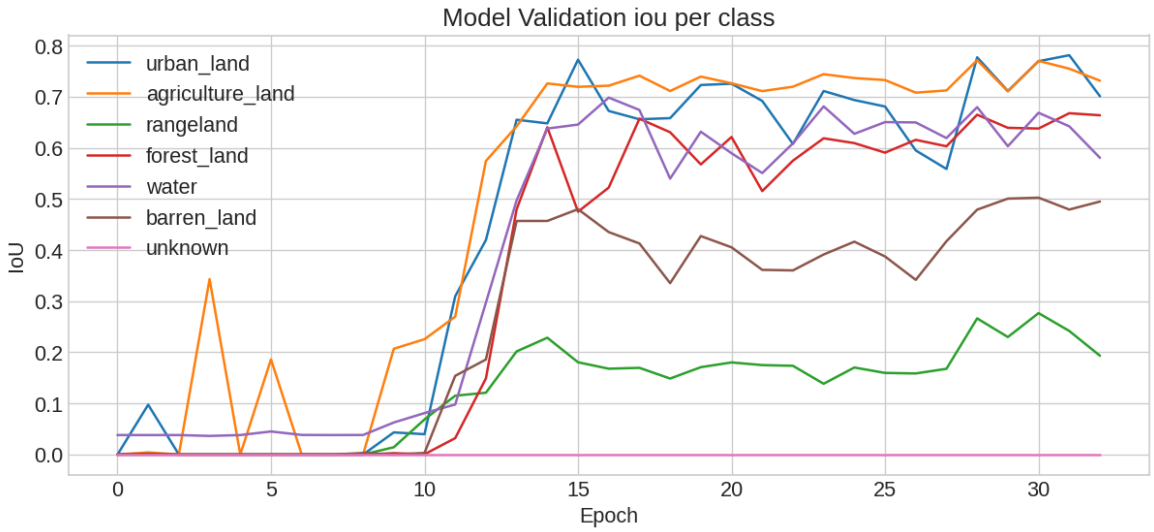
(a)

(b)



(c)

(d)



(e)

Figure 4.1: Model training metrics for model '10_05_2022_10_06_52'. (a) Mean one-hot IoU for all labels. (b) Categorical accuracy for all labels, provided by Keras's CategoricalAccuracy. (c) Dice loss + Categorical Focal Loss (d) IoU per label on train set. (d) IoU per label on validation set.

Range land and barren land tend to be the most difficult to identify in the images, agriculture land and urban land tend to show the highest IoU scores. There is a big imbalance in the data set in terms of LULC classifications as seen in Figure 3.1. Looking into the dataset at classifications such as barren land, there are example of poor labelling in the data, for example Figure 4.2 shows an area of Land predicted to be barren land by a model but labelled as unknown in the ground truth.

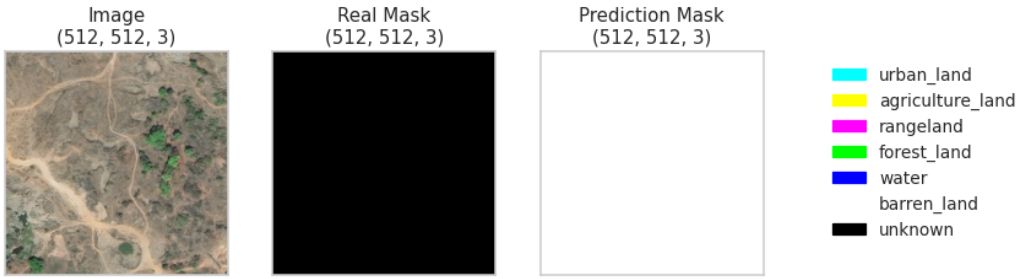


Figure 4.2: Example of a 512 x 512 image with ground truth labelling the pixels as Unknown, but the U-Net model labels this as barren land.

Below in Figure 4.3, an image and a real segmentation mask of agricultural land is shown, however the model '10_05_2022_10_06_52' could not determine it to be agricultural because the identifying features were perhaps outside the receptive field for the model.

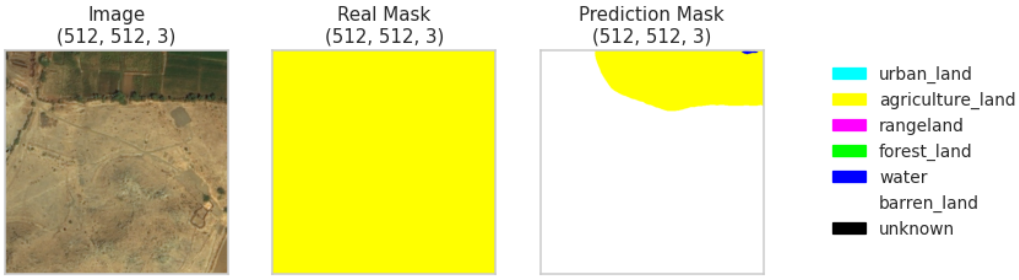


Figure 4.3: Example of a 512 x 512 image with ground truth labelling the pixels as agricultural land, the model incorrectly predicts the centre to be barren land, a possible sign of a small receptive field, whereby the model is looking too local.

4.1.2 Choosing Base Models

After these 64 models were trained, the models needed to be evaluated to find a base model that can be used for comparison in the research hypothesis. Two base models will be found in this analysis, the first was the strongest model and the second was the model which gave an average result on the 160 random images from the entire training dataset.

Inference took on average 78.2 seconds to evaluate the 160 sample images on a Dell OptiPlex 5040 with an i5-6500 CPU @ 3.20GHz in order to find the best model to choose for comparison. Below is a histogram (Figure 4.4) of that sample test set for each of the models showing the mIoU score (M = 0.534, SD = 0.035). This mean score is not important, only the distribution is interesting as it suggests each model is classifying the images differently.

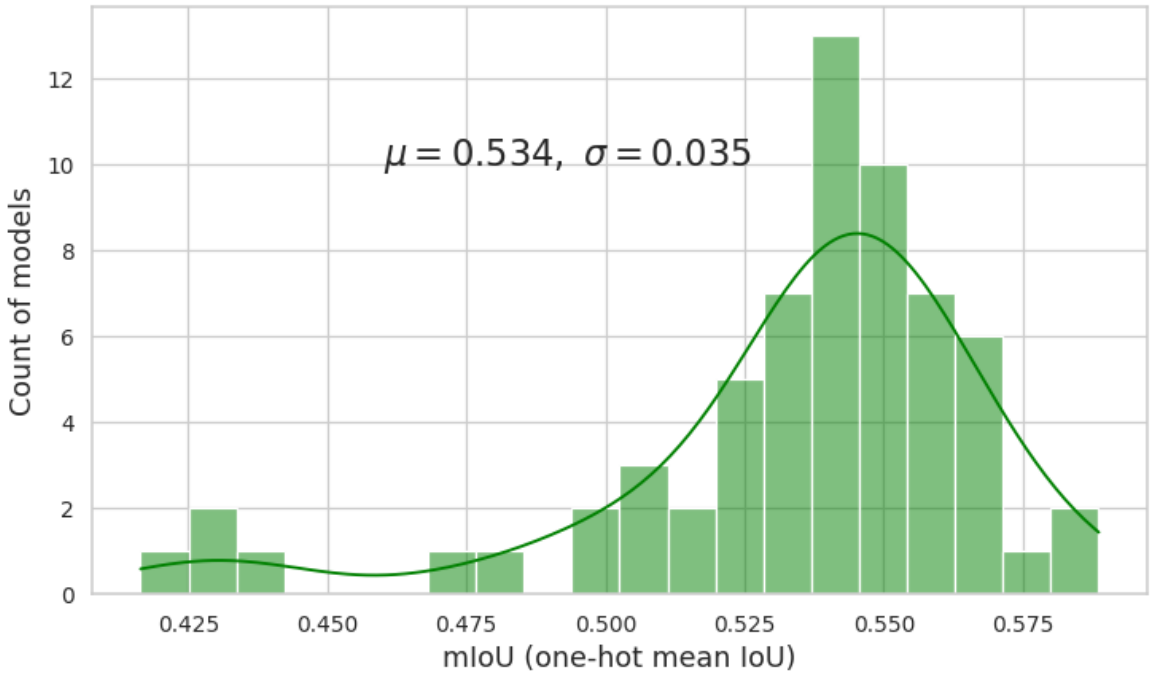


Figure 4.4: In order to find the base models, all models were evaluated on a sample of 160 images, this is the histogram of those models.

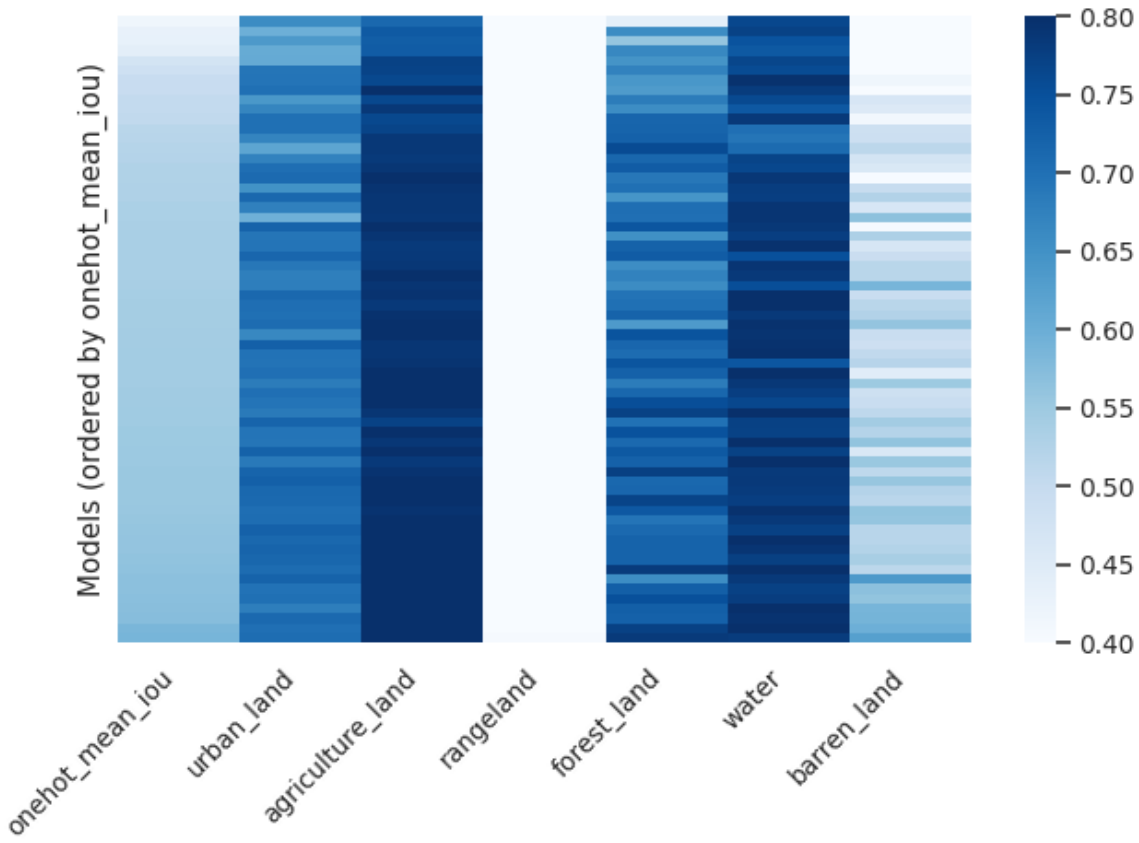


Figure 4.5: All the models stacked vertically sorted by mIoU, note the differences between models for LULC labels such as 'Urban Land', 'forest land' or 'Barren Land'

name	one-hot mIoU	urban_land	agriculture_land	rangeland	forest_land	water	barren_land
17_05_2022_08_29_54	0.588658	0.705544	0.819414	0.404692	0.784299	0.780707	0.625949
16_05_2022_22_43_50	0.585234	0.701949	0.815242	0.392973	0.774620	0.814029	0.597823
10_05_2022_17_48_13	0.572695	0.712100	0.814956	0.373251	0.724915	0.794473	0.589171
04_05_2022_15_53_41	0.570572	0.680878	0.830271	0.366442	0.728061	0.800582	0.587771
04_05_2022_10_20_05	0.570104	0.700735	0.825532	0.373503	0.750537	0.778792	0.561630
10_05_2022_00_42_26	0.567812	0.698342	0.823781	0.383553	0.727410	0.774283	0.567315
17_05_2022_19_00_42	0.567747	0.720394	0.810108	0.365464	0.656823	0.784826	0.636614
10_05_2022_10_06_52	0.565446	0.707801	0.817192	0.337859	0.781462	0.800904	0.512905
04_05_2022_13_31_29	0.563826	0.714180	0.838799	0.359373	0.720672	0.775943	0.537818
18_05_2022_11_05_04	0.560473	0.719233	0.825496	0.342427	0.721027	0.791292	0.523834
07_05_2022_23_04_07	0.560160	0.713423	0.833312	0.328841	0.721867	0.805791	0.517888
17_05_2022_11_15_35	0.559083	0.724208	0.832857	0.353974	0.711515	0.773757	0.517272
05_05_2022_04_11_14	0.556463	0.705605	0.816232	0.333772	0.695220	0.785903	0.558509
04_05_2022_21_00_41	0.556007	0.704008	0.793892	0.298216	0.737922	0.795956	0.562055
07_05_2022_15_15_30	0.554349	0.711155	0.803825	0.304066	0.768886	0.776957	0.515554
10_05_2022_15_27_39	0.554306	0.713396	0.814748	0.331041	0.716009	0.782452	0.522495
18_05_2022_03_43_22	0.554175	0.725014	0.816633	0.281701	0.713110	0.785541	0.557227
10_05_2022_12_51_56	0.553570	0.719477	0.794450	0.292391	0.775739	0.782912	0.510020
06_05_2022_23_50_36	0.552412	0.687827	0.785261	0.315610	0.723740	0.800812	0.553630
06_05_2022_16_12_57	0.552232	0.723311	0.821838	0.351396	0.736860	0.772437	0.459784

Table 4.1: Results for top 20 models evaluated on 160 random samples from the 723 image training set. '17_05_2022_08_29_54' will be the base model.

From Table 4.1, model '17_05_2022_08_29_54' was taken as the first base model to be used for comparison against the ensemble methods. This was referred to as the 'base_max' model and a summary of the Keras structure is shown in the Appendix A.0.2. The second base model ('30_04_2022_01_28_23') will be the closest to average according to this 160 random sample test and this model was referred to as 'base_avg', details on how this model is selected will be shown below.

Next, the test set which was put away from the beginning was used by each of these model to produce distributions. This test set was divied into 40 subsets as described in Chapter 3. Figure 4.6 below shows the proportions of each label in terms of pixels counts, it matches the imbalance seen also in the training set.

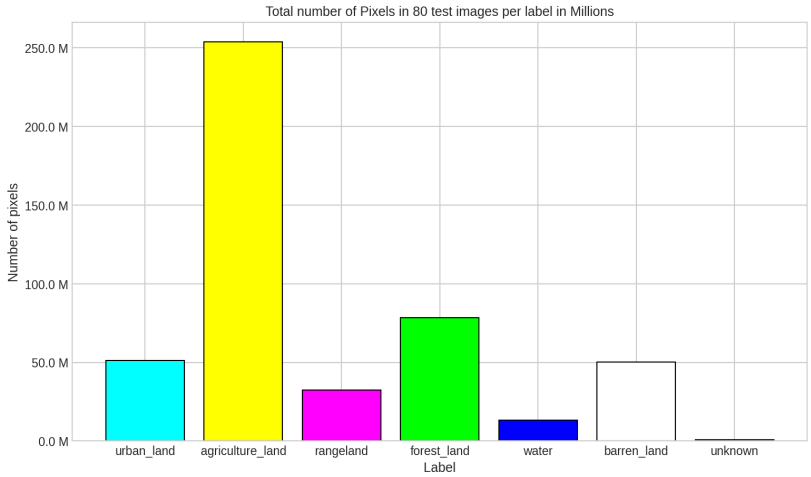


Figure 4.6: The test set containing 80 images which put away from the beginning which was split into 40 subsets. The barchart shows the pixel count for each LULC label.

Base Model ('base_max')

Inference for this 'base_max' model took on average 16.88 seconds to evaluate one of the 40 test sets on a Dell OptiPlex 5040 with an i5-6500 CPU @ 3.20GHz. Below is a histogram (Figure 4.7) that shows the spread of test results (M = 0.487, SD = 0.070).

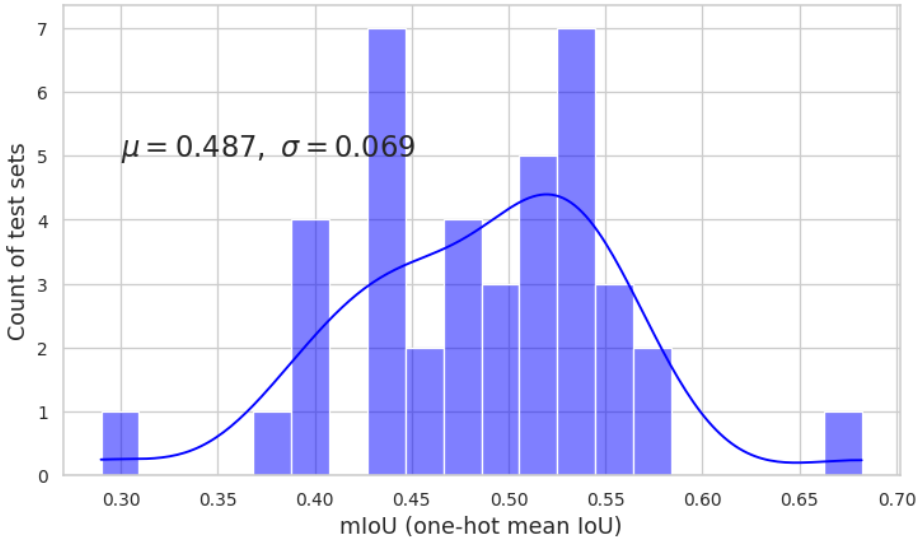


Figure 4.7: The 'base_max' model results for 40 test sets of 32 patches.

Base Model ('base_avg')

To find the 'base_avg', the model was the closed mIoU was chosen using python command shown in Figure 4.8 and the results are shown in Table 4.2.

```
> df_close_to_mean = model_results_df.iloc[
  (model_results_df['onehot_mean_iou']-0.534).abs().argsort()[:1]
  ]
```

Figure 4.8: Find the closest model to the average for all 64 models on the random test set.

name	30_04_2022_01_28_23
loss	0.593985
categorical_accuracy	0.809373
onehot_mean_iou	0.533991
iou_0	0.596772
iou_1	0.787853
iou_2	0.292681
iou_3	0.702586
iou_4	0.791186
iou_5	0.566859
iou_6	0.0
sm_iou_score	0.714951

Table 4.2: Model '30_04_2022_01_28_23' with the closest to average mIoU value. Model is entitled 'base_avg'. In this Table, all the metric which were recorded during evaluation are shown. 'onehot_mean_iou' is the metric used for selection. 'sm_iou_score' comes from the segmentation-models Python library, which is calculated differently as explained in Chapter 3.

Inference for this 'base_avg' model took on average 22 seconds to evaluate one of the 40 test sets on a Dell OptiPlex 5040 with an i5-6500 CPU @ 3.20GHz. Below is a histogram (Figure 4.7) that shows the spread of test results (M = 0.432, SD = 0.068).

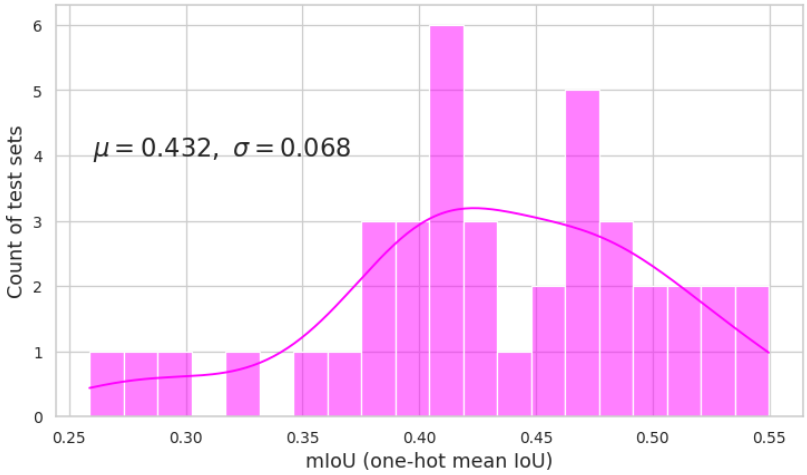


Figure 4.9: The 'base_avg' model results for 40 test sets of 32 patches.

4.2 Phase 2

4.2.1 Average Ensemble

Inference for the 'average' ensemble took on average 572.55 seconds to evaluate one of the 40 test sets on a Dell OptiPlex 5040 with an i5-6500 CPU @ 3.20GHz. Below is a histogram (Figure 4.4) of those tests showing the mIoU score for each(M = 0.495, SD = 0.073).

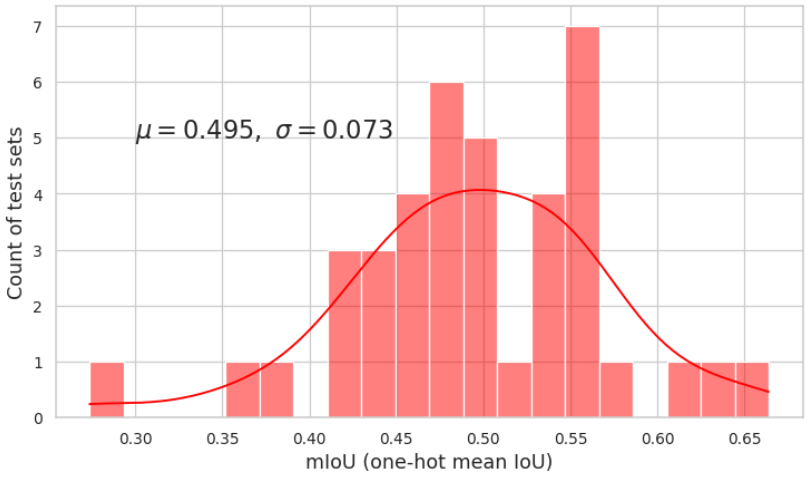


Figure 4.10: The 'average' ensemble model results for 40 test sets of 32 patches.

4.2.2 Weighted Average Ensemble

In order to train a weighted model, the first attempt was to use all 64 models trained from phase 1, however the memory resources required for such an effort were not possible on average desktop PC such as the Dell Optiplex described above, more than 30 GB of RAM is required to met the demand of training the weights on the final WeightedAverage Layer.

The second attempt was to lower the total number of U-Net models connected to the WeightedAverage Layer, the top 20 models as seen from the random sample of 160 images. This attempt however could not find any other optimum other than given each model as equal proportion or weight in the final classification as shown in Figure 4.11, meaning the result would match the ensemble using only average.

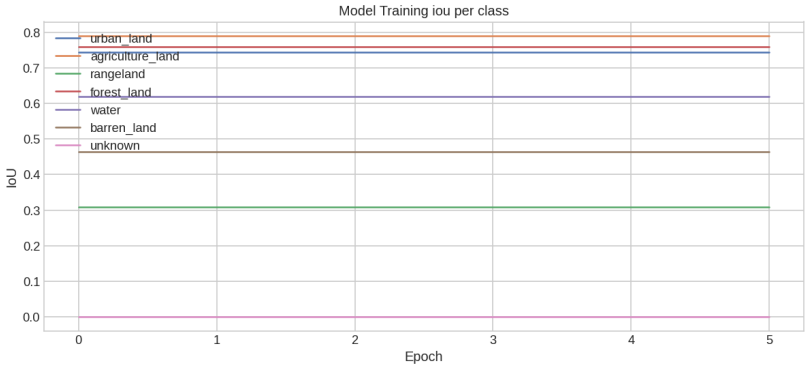
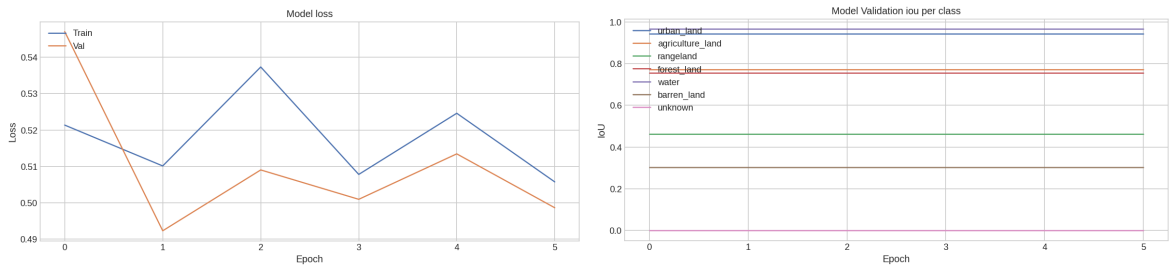


Figure 4.11: Training a weighted average ensemble model using the top 20 U-Net models from phase 1. Because the models were so similar, the training could not find a better configuration other than an equal proportion for each model.

The third attempt was to take a random sample of 4 models from the pool of 64 U-Net models available. The training took about 200 seconds on a training set of 100 random images from the training set. It was trained for 6 epochs, there was no improvement in mIoU or model loss during this time as seen in Figure 4.12. This trained model is equivalent to an average of four models.



(a) (b)

Figure 4.12: Training a weighted average ensemble model using 4 model sampled at random.

Inference for the 'weighted' model took on average 40 seconds to evaluate one of the 40 test sets on the same i5-6500 CPU. Below is a histogram (Figure 4.13) of that sample test set for each of the models showing the mIoU score (M = 0.479, SD = 0.072).

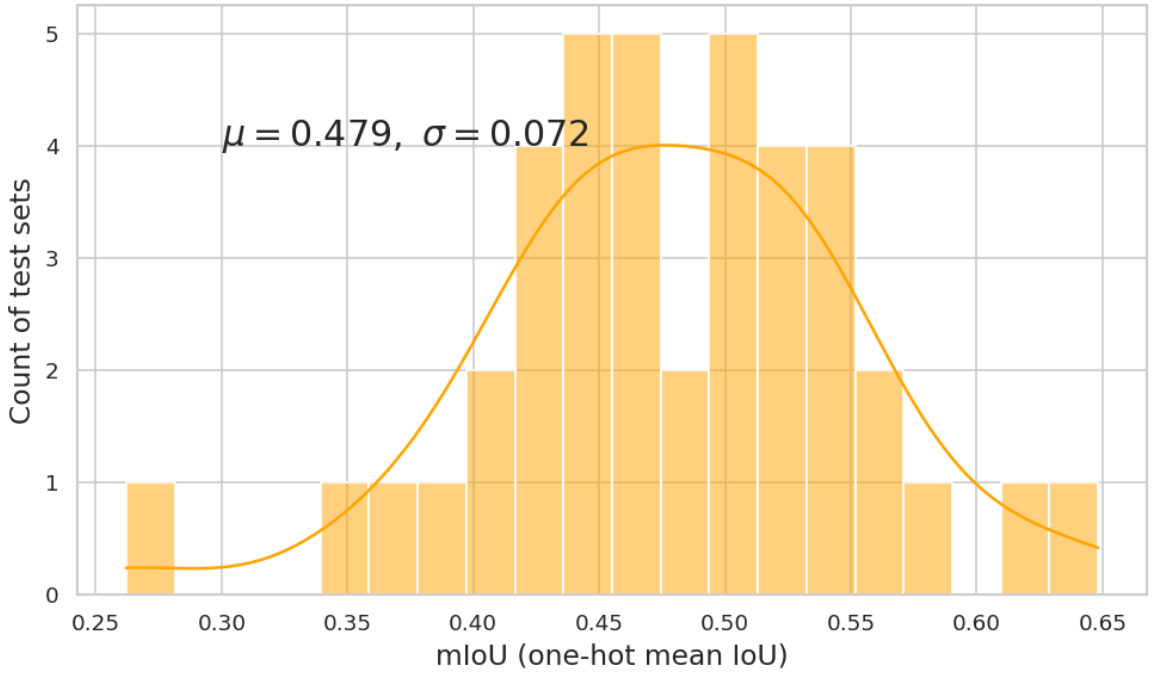


Figure 4.13: Results from the 40 test sets for the 'weighted' average model which uses 4 models combined.

```
> tf.nn.softmax(w_ensemble_model.get_weights()[-1]).numpy()
array([[[0.2411845 , 0.25571018, 0.25089592, 0.2522094 ]]],
      dtype=float32)
```

Figure 4.14: A softmax representation of the weights on the final layer of the 'weighted' average ensemble. The proportions are almost equal.

Sample predictions are enclosed in the Appendix under section A.0.1.

4.3 Statistical Tests

Restating the statistical hypothesis used to test the effect of the ensemble approach:

- **Null hypothesis**: If data augmentation and an ensembling technique are used during training then the testing IoU score of a LULC semantic segmentation model is **not** statistically significantly different from the testing IoU score of a model with data augmentation and without an ensembling technique.
- **Alternative hypothesis**: If data augmentation and an ensembling technique are used during training then the testing IoU score of a LULC semantic segmentation model is statistically significantly different from the testing IoU score of a model with data augmentation and without an ensembling technique.

In summary, the null hypothesis is that ensembles won't help considerably, that there is no effect. As discussed in Chapter 3, a paired student t-test was used to test for this effect. The paired student t-test comes with a few assumptions which the needed to be confirmed in the data.

The first assumption is that there are no significant outliers in the distribution and the second is that the distribution is 'Normal'. To complete this, I calculated the standardised score for the IoU variable and counted what percentage are outside an acceptable range. I chose the 97.5th percentile point as a cut off point, this is approximately 1.96 standard deviations from mean and 95% of the data should lie

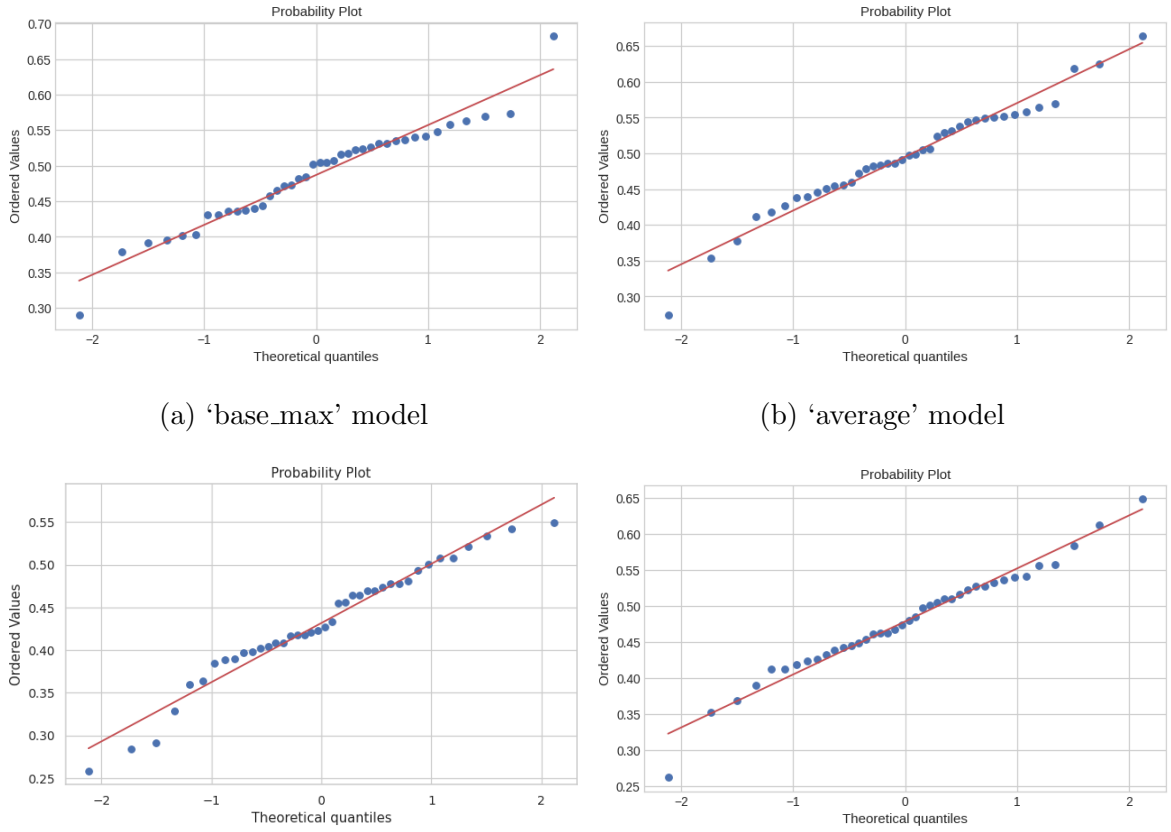
between 1.96 of the mean each way for a standard normal distribution. For the second assumption, I determined the distribution to be normal by inspecting the skew and kurtosis of each distribution.

I inspected the distribution from the results above, those being the base model('base_max'), the second base model based on the closest to mean('base_avg'), the average model('average') and the weighted average model('weighted'). A table to summarize the results is shown below.

Model	sample size	M	SD	outside +/- 1.96 SD	skew	kurtosis
base_max	40	0.487	0.070	0.05	-0.175	4.294
base_avg	40	0.432	0.068	0.07	-0.556	3.271
average	40	0.495	0.074	0.05	-0.430	4.276
weighted	40	0.479	0.073	0.05	-0.348	4.247

Table 4.3: Summary of the four distributions. M stands for mean, SD stands for standard deviation, outside +/- 1.96 SD is the proportion of test sets which returned a result whihc was more than 1.96 standard deviations from the mean.

From Table 4.3, we saw the skewness of each distribution is concentrated on the right tail, the value is not large(<1), the kurtosis for the three distributions was greater than 3 and can be called leptokurtic, meaning it is highly concentrated around the mean. Visible inspection of Figure 4.17 confirms this high kurtosis. The base model has the highest kurtosis which would be expected, since the model is not generalizing like the average or weighted models.



(a) 'base_max' model

(b) 'average' model

(c) 'base_avg' model

(d) 'weighted' average

Figure 4.15: Q-Q plot for each model on the 40 test sets.

To test my data matched that of a normal distribution, a Shapiro Wilk test was performed on all three distributions, all returning a p-value larger than 0.05 (0.23, 0.23, 0.49 and 0.705 for base_max, base_avg, average and weighted respectively), the Shapiro Wilk test is a statistical test used to check if a variable like mIoU follows a normal distribution. The null hypothesis states that the variable is normally distributed, this is not rejected by my tests, so for the purposes of a t-test, I did consider each distribution to be treated as a normal distribution.

4.3.1 Compare ensemble models

Before conducting the research hypothesis t-test, I needed to determine which ensemble model was giving the best results and since I had two distributions I could make a paired t-test to compare. The null hypothesis is that there is not a difference and the

alternative is that weighted produces a higher mIoU score.

As stated in Chapter 3, the significance level(alpha) was set to 0.05 or 5% for these experiments.

A Paired samples t-test was conducted to determine the effect of using a weighted average in an average ensemble approach. The results indicate there is significant difference between a simple 'average' model (M=0.495; SD=0.074) and 'weighted' average model (M=0.479; SD=0.073); [t(39) = -8.88, p <0.001].

The 95% confidence interval of the difference between the means ranged from [-0.021 to -0.013] and it indicated a difference between the means of the samples. I, therefore, reject the null hypothesis that there is no difference between the means and conclude that there is a difference. The difference is negative suggesting the 'average' model has a mean difference of 0.017 more than the 'weighted'.

```
Paired t-test


data:  w_ensemble and ensemble
t = -8.8838, df = 39, p-value = 6.537e-11
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.02066541 -0.01300032
sample estimates:
mean of the differences
            -0.01683287
```

Figure 4.16: T-test from R's stats library (R Core Team, 2013), to determine the effect of using a weighted average vs an average ensemble approach.

Visually inspecting the kernel density estimation (KDE) plot in Figure 4.17 below would support this statistic also.

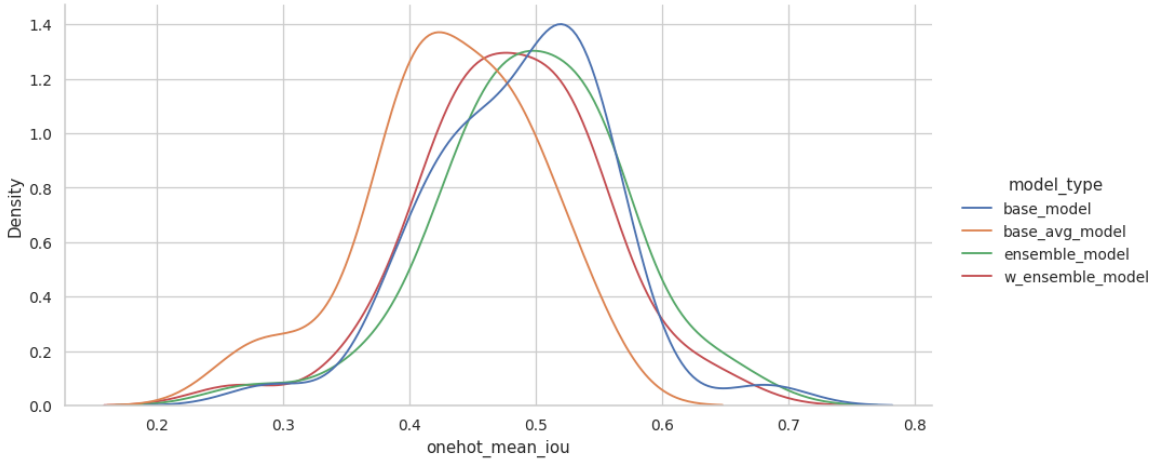


Figure 4.17: Kernel density estimation (KDE) plot for all 4 model distributions.

In the next section it will be shown how the 'average' model compared against the two base models ('base_max' and 'base_avg') and finally test the research hypothesis.

4.3.2 Research Hypothesis

Reporting Average Ensemble and Max Base Model T-test

A Paired samples t-test was conducted to determine the effect of using an average ensemble approach on a LULC semantic segmentation task. The results indicate a significant difference between the mIoU score of the best model from a group of models without an ensemble (M=0.487; SD=0.070) and the mIoU score with an average ensemble (M=0.495; SD=0.074); [t(39) = 2.0912, p = .04306].

The 95% confidence interval of the difference between the means ranged from [0.0002 to 0.0163] and did indicate a difference between the means of the samples. I, therefore, reject the null hypothesis that there is no difference between the means and conclude that there is an effect of using an average ensemble on the mIoU score of this LULC semantic segmentation task. The effect is positive and the means increased by 0.0083.

```
Paired t-test


data:  ensemble and base
t = 2.0913, df = 39, p-value = 0.04306
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  0.0002717852 0.0162913785
sample estimates:
mean of the differences
             0.008281582
```

Figure 4.18: T-test from R's stats library (R Core Team, 2013), to determine the effect of using an average ensemble approach.

	T	dof	alternative	p-val	CI95%	cohen-d	BF10	power
T-test	2.091324	39	two-sided	0.043061	[0.0, 0.02]	0.114826	1.206	0.109189

Table 4.4: Results of T-Test from Python's Pingouin Library (Vallat, 2018), conducted to determine if there is an effect to using an average ensemble approach. dof is the degrees of freedom, BF10 stands for the Bayes Factor, cohen-d is the Cohen d effect size and power is the achieved power of the test (1 - type II error).

According to Sawilowsky (2009), a cohen-d effect of 0.11 is very small. Furthermore, note the power of the t-test, 0.11 is very low. This value ranges from 0 to 1, the lower this value the higher the probability of making a type II error on the null hypothesis. Moreover, the BF10 value of 1.21 reported by Pingouin's Library is interpreted as "Not worth more than a bare mention" according to Wei et al. (2022).

Reporting Average Ensemble and Average Base Model T-test

A Paired samples t-test was conducted to determine the effect of using an average ensemble approach on a LULC semantic segmentation task. The results indicate a

significant difference between the mIoU score on an average model without an ensemble (M=0.432; SD=0.068) and the mIoU score with an averaged ensemble (M=0.495; SD=0.074); [t(39) = 12.61, p <0.001].

The 95% confidence interval of the difference between the means ranged from [0.053 to 0.074] and did indicate a difference between the means of the samples. I, therefore, reject the null hypothesis that there is no difference between the means and conclude that there is an effect of using an average ensemble on the mIoU score of this LULC semantic segmentation task. The effect is positive and the means increased by 0.0636.

```
Paired t-test


data:  ensemble and base
t = 12.609, df = 39, p-value = 2.457e-15
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  0.05340081 0.07380764
sample estimates:
mean of the differences
               0.06360422
```

Figure 4.19: T-test from R's stats library (R Core Team, 2013), to determine the effect of using an average ensemble approach.

	T	dof	alternative	p-val	CI95%	cohen-d	BF10	power
T-test	-12.608689	39	two-sided	2.456895e-15	[-0.07, -0.05]	0.890699	2.66e+12	0.999792

Table 4.5: Results of T-Test from Python's Pingouin Library (Vallat, 2018), conducted to determine if there is an effect to using an average ensemble approach. dof is the degrees of freedom, BF10 stands for the Bayes Factor, cohen-d is the Cohen d effect size and power is the achieved power of the test (1 - type II error).

According to Sawilowsky (2009), a cohen-d effect of 0.89 is considered large to very large. Furthermore, note the power of the t-test, 0.99 is very high. This value ranges

from 0 to 1, the lower this value the higher the probability of making a type II error on the null hypothesis. Moreover, the BF10 value of 2.6 trillion reported by Pingouin's Library is interpreted as "Very Strong" according to Wei et al. (2022).

4.4 Discussion

This research has stated the importance of mIoU accuracy when classifying LULC satellite images. There are many architectures developed over the years to solve the problem of semantic segmentation and applying them to LULC classification, the literature has shown there has been minimal use of the ensemble technique in other to maximize the mIoU of the segmentation by grouping these many architectures and turning many weak models into a strong model.

The U-Net model training highlighted some inconsistencies in the labelling of the dataset, there are many examples of the U-Net models predicting barren land when the actual ground truth label reported the land as barren land. There also existed a lot of confusion between barren land and range land, even with human vision, the differences between barren and range are difficult to determine and finally there were examples of segmentation masks predicted where it looks like the model didn't have a big enough receptive field, there were agricultural fields where inside the field, the model determined it to be barren as the size of the field was too large for the receptive field. Perhaps this is an area where an architecture with a more global sense such as a transformer could preform better.

There was also a successful use of transfer learning in this experiment, the weights pre-trained on ImageNet proved useful for semantic segmentation of satellite images, this greatly increased the computational cost of training, however it didn't allow the final U-net models to be as broad in their range, which would have widened the differences between the ensembles weak learners.

There was evidence that the patience setting for the early stopping mechanism was too large on the training of the U-Net models, training could have been faster if the parameter in the callback API was set to a lower value, it was set to 15, when analysis

of the validation plots would indicate this number was not required.

In the Introduction Chapter, I introduced the research questions; 1) to find out which ensemble methods performs the best, 2) can this ensemble method perform better than a single stand-alone model?, 3) to what extend can it perform better?, 4) is the ensemble method better than the best single model from a group of models? and 5) to what extend is that better?

The reason for using two base models, is that an average chosen from the U-Net group of models simulates the training of one random model, whereas a selected max IoU score model needs all models to be fully trained first and so requires the same training time as an ensemble but has a quicker inference time.

To answer the first research question, when a comparison was done between the two ensemble approaches, the results show that a simple ensemble has performed better than a weighted ensemble model. This is clear from the t-test which showed there existed a mean difference of 0.17 between the models. The 'average' model used all 64 models to produce a pixel label during evaluation whereas due to hardware limits, the 'weighted' model used just 4 models. The training of these 4 models was unsuccessful since the weights of these models did not diverge far from equal proportions.

The training of the individual models were already using a lot of generalizing techniques such as augmentation, so each model carried a lot of equal knowledge, this meant that the training of 'weighted' model did not determine that one model should carry far more weight than another.

To answer the second research question, I need to see what the distribution of all models was and select an average model from the group. This selection process relied on the random evaluation set of 160 images which was done at random on the full 723 images. A different selection of 160 images could have resulted in a different average model being selected.

The two distributions; 'base_avg' model and 'average' model were compared side by side in KDE plot, visually the impact of an ensemble was clear, moving the distribution of the mIoU score up by 0.06 on average across all the test sets. The t-test confirmed this, with a large to very large cohen-d effect, a very high test power and a BF10 value

described as very strong, this addresses the 3rd research question.

The fourth research question asks a question about large inference time required by an ensemble approach, essentially is it worth it? The training time required by an ensemble is almost the same as that required to pick the best of a bunch of models. The results show each model took about 300 seconds per epoch and there was about 26.7 epoch on average. With the limitation, that these figures are very much specific to each GPU or CPU hardware setup, for my setup, a 4 core CPU took 78.2 seconds per model to evaluate 160 satellite images.

$$\begin{aligned} 300\ seconds\ \times 26.7\ epochs\ \times 64\ models\ &= 512,640\ seconds \quad (142.4\ hours) \\ 78.2\ seconds\ \times 64\ models\ &= 5,005\ seconds \quad (1.39\ hours) \\ 512,640 + 5,005 &= 517,645\ seconds \quad (143.8\ hours) \end{aligned}$$

That's 143.8 hours to train and select the best model and 1.39 hours less to train for an ensemble, not much of a difference. However, the real difference is the inference time, 573 seconds to evaluate 32 patches(2 images) with an ensemble versus about 17 seconds for a single model. GPU parallelization could help reduce that difference. There is at least an order of magnitude in the difference. It should be said that the length of time spent training could be decreased by training the models in parallel, this is an advantage of this type of bagging ensemble approach. Another advantage to this bagging method is that the ensemble model could be updated easily when new labelled data is available, by simply training new models and adding them.

The answer to the fourth research question about whether an ensemble of a group can be better than the best from a group, it is important to keep into account the inference time. The results from the t-test for the best base model and the ensemble show a p-value of 0.43, which is close to 5%, meaning 5% of the time we would see no difference in the means. The cohen-d and Bayes Factor confirm this result indicating the result is "Not worth more than a bare mention". There is not strong enough evidence to suggest using an ensemble instead of taking the best model from the pool of U-Nets, especially since the inference time is an order of magnitude larger. However, as the test data drifts further from the training data, this effect could increase, in this

experiment the test and training data do not diverge greatly.

The firth research question asks what extend is an ensemble better than the best selected single base model? From the evidence we have, the conclusion would be that it is not showing any difference worth the effort.

The findings here did meet my expectations, there is evidence that an ensemble of U-Net models that use a simple average can give better performance than an average single U-Net. These findings agree with previous research discussed in the Literature Review such as Nigam et al. (2018) and Tapper et al. (2020) in terms of preventing overfitting and proving the benefits of an ensemble. The challenges presented by using an ensemble agree with those mentioned in Benbriqa et al. (2021), that is the high time-cost of this solution and the difficulty in choosing members of the ensemble.

The weighted ensemble approach did not prove to be a success, this could be due the use of augmentation, using augmentation prevents a model from overfitting too much, if many of these weak learner U-Net model are generalizing quite well, then there is no success is trying to identify the strengths and weakness of each. Moreover, the computer resources required to train a large weighted ensemble have proven to be considerable, this could be an explanation for the lack of literature on the topic of weighted ensembles in the semantic segmentation of satellite images.

The results show that a weighted ensemble approach might be good for an image classifcation task where there is only one or two prediction required, but in a semantic segmentation task, an output needs to have 512 x 512 predictions. Without augmentation there has been improvements shown in the medical imaging domain when using a weighted ensemble approach such Dang et al. (2021). So possibly, the generalising of individual models is weakening the impact of a weighted approach.

Chapter 5

Conclusion

This chapter will explain the outcomes inferred in this research. A brief overview of the research is given, followed by a summary of the problem definition, after that the experiment design and evaluation is detailed and finally there are few suggestions for future work in this area.

5.1 Research Overview

This research aimed to identify an ensemble approach which could improve the mean intersection over union(mIoU) metric for a pixel labelling problem known as semantic segmentation. Semantic segmentation is critical in the area of Land Use Land Cover(LULC) for guiding land management, land planning and sustainable development.

An ensemble approach consists of pooling together the predictions of many models to produce better predictions and in doing so, better mIoU scores. There has been many CNN model architectures developed over the years which have become deeper and deeper with the help of residual connections to solve these semantic segmentation problems.A popular architecture called U-Net was chosen in this research to predict labels for a dataset from the DeepGlobe 2018 Land Cover Classification Challenge.

This research addresses a need to understand how much an ensemble approach can effect the mIoU, the literature provided detailed evidence of its use in the medical

imaging area. Pre-trained ResNet-34 decoder weights were utilized to transfer learning from an ImageNet dataset and apply it to a satellite segmentation task.

Two ensemble methods were designed, one by using a simple average of 64 models and another by using a weighted average of 4 models. These ensembles were tested to find out which performed better, and that ensemble method was then compared against the performance of a single U-Net model.The results focus not only on the mIoU comparisons but also the computational effort of the ensemble approach which is an important consideration.

5.2 Problem Definition

The benefits to applying an ensemble approach to the semantic segmentation of satellite imagery is not well understood. Lots of research exists applying many of the state-of-the-art architectures such as DeepLabv3+, U-Net or SegNet to satellite segmentation problems and achieving significant results. There is a penalty to applying an ensemble in terms of computational time, to make an ensemble solution practical, this needs to be also taken into consideration.

The problem is posed by asking the following question: Can independently trained well-performing models pooled together produce significantly better results than a single model for a semantic segmentation of the DeepGlobe LULC dataset?

5.3 Design/Experimentation, Evaluation & Results

The dataset taken from the DeepGlobe 2018 Land Cover Classification Challenge was used to evaluate the impact of an ensemble approach. The data was split into a test set and training set, with the training set, 64 U-Net models were trained using a ResNet-34 decoder with pre-trained weights from ImageNet.

The ensembles were built using these U-Net models, the first ensemble used a simple average of the 64 models, the second ensemble used just 4 U-Net models but trained a new Keras Layer called WeightedAverage to find the optimum weight for

each U-Net model. Both these ensembles were evaluated with the test set, a weighted ensemble approach did not improve the mIoU score when compared to a simple average ensemble.

Two base models were selected by taking an average U-Net model and a U-net model which performs the best on the train set. Both models were compared to the average ensemble method using a Student's t-test. The results of an average ensemble gave a statistically significant greater mIoU than a single average U-Net model, the Cohen-d effect was found to be very large and power of the test was very high. This finding confirmed our main alternative hypothesis for this study, that an ensemble will outperform any average single model.

There was very small effect found when comparing an average ensemble model to the best performing model from a pool of models. However, the computational time was an order of magnitude higher for an ensemble in comparison to a single model. This result was surprising and indicates that there is a benefit in training many models and choosing the best one rather than using on ensemble which has a much longer inference time, it could also suggest that each model is generalising and is not looking at slightly different aspects but the similar aspects due to the use of augmentations. This results could also be suggesting that ensemble with pre-trained weights can't be as independently different from each other as an ensemble approach would like them to be.

So in summary, this research clearly illustrates the benefits of an ensemble approach in comparison to a single training of a model, but it also raises the question whether it is better to select the best model in a pool rather than do an ensemble because of the impact an ensemble has on the inference time.

5.4 Contributions and impact

This research clearly identified and quantified the effect of using an average ensemble approach to solve the semantic segmentation of satellite images.

Importantly, the results of this research provide evidence that pooling together

single independently trained models and averaging them has a very large effect on the mean intersection over union metric in the area of satellite imaging just it has shown in the area of medical imaging (Shah & Madabushi, 2021; Shimizu et al., 2008).

The results are consistent with ensemble studies in the area of LULC semantic segmentations (Marmanis et al., 2016; Tapper et al., 2020), this research however goes further to quantify the effect of an ensemble in comparison to a single model, demonstrating a mean difference of 0.17 in terms of the mean class intersection over union for a 7 classes LULC satellite imagery dataset.

In addition, it is now clear that a weighted ensemble approach using very well generalised independent models is not a better approach than a simple averaged ensemble. Training the weights applied to each model in a pool did not have a positive effect on the mIoU metric.

Studies which have used an ensemble approach have failed to mention the disadvantage an ensemble has at inference time, a single model was shown to be at least on order of magnitude faster than an ensemble. Taking in account this computation at inference, this research has provided evidence that there is no benefit to using an ensemble if many models are trained and the best performing model is used.

Studies have shown a weighted ensemble approach is more accurate in an image classifcation problem (Cheng et al., 2020; Harangi et al., 2018) however when applied to an image segmentation problem, those benefits are not as clear.

5.5 Future Work & recommendations

An area of difficulty in using the ensemble approach to semantic segmentation is developing many independent models, in this research 64 models were training similarly, albeit with augmentation that changes the training input. There are more methods which could be used to give a more diverse set of models, such as using different architectures or decoder backbones as seen in Cheng et al. (2020) and T. Zhou et al. (2021) for image classification. The more each model finds a local solution, the greater the ensemble. Each DCNN architecture has its own strength and weakness which can be

taken as an advantage in an ensemble approach. If an ensemble of different backbones with different pre-trained weights were utilized, the diversity of the ensemble would be greater and thus the power of the ensemble could be greater.

Model selection was not considered in this study, Ganaie et al. (2021) writes "Finding a criterion for model selection in ensemble deep learning should be an important target for researchers in the next few years." The authors stress there has been little attention paid to this topic. In this research the high number of models in the ensemble has lead to extremely high inference times, an interesting research question which could look asked is–what is the minimum number of models which could be utilized withouth dropping the performance significantly?

As mentioned in the literative review chapter, transformer based architectures based on the Vision Transformer (Dosovitskiy et al., 2020) are becoming the state-of-the-art approach in the semantic segmentation domain. They don't use convolutions and can capture more contextual informational. A key comparison would be between a transformer based approach such as the Segmenter (Strudel et al., 2021) and the ensemble approach, does a single transformer outperform an ensemble?

Bibliography

Alexander, B., I., I. V., Eugene, K., Alex, P., Mikhail, D., & A., K. A. (2018). Albumentations: Fast and flexible image augmentations. *ArXiv e-prints*. https://albumentations.ai/

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, *8*, 1–74. https://doi.org/10.1186/s40537-021-00444-8

Anderson, J. R., Hardy, E. E., Roach, J. T., & Witmer, R. E. (1976). A land use and land cover classification system for use with remote sensor data. *Professional Paper*. https://doi.org/10.3133/PP964

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*, 2481–2495. https://doi.org/10.1109/TPAMI.2016.2644615

Barnes, D. K. (2022). Cop26: How much real progress? *Antarctic Science*, *34*, 1–2. https://doi.org/10.1017/s0954102022000098

Baumgardner, M. F., Biehl, L. L., & Landgrebe, D. A. (2015). 220 band aviris hyperspectral image data set: June 12, 1992 indian pine test site 3. https://doi.org/doi:/10.4231/R7RX991C

Benbriqa, H., Abnane, I., Idri, A., & Tabiti, K. (2021). Deep and ensemble learning based land use and land cover classification. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12951 LNCS*, 588–604. https://doi.org/10.1007/978-3-030-86970-0_41

Benjdira, B., Bazi, Y., Koubaa, A., & Ouni, K. (2019). Unsupervised domain adaptation using generative adversarial networks for semantic segmentation of aerial images. *Remote Sensing*, *11*, 1369. https://doi.org/10.3390/rs11111369

Camara, G. (2022). On the semantics of big earth observation data for land classification. *Journal of Spatial Information Science*, *20*, 21–34. https://doi.org/10.5311/JOSIS.2020.20.645

Campos-Taberner, M., Romero-Soriano, A., Gatta, C., Camps-Valls, G., Lagrange, A., Saux, B. L., Beaupere, A., Boulch, A., Chan-Hon-Tong, A., Herbin, S., Randrianarivo, H., Ferecatu, M., Shimoni, M., Moser, G., & Tuia, D. (2016). Processing of extremely high-resolution lidar and rgb data: Outcome of the 2015 ieee grss data fusion contest-part a: 2-d contest. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *9*, 5547–5559. https://doi.org/10.1109/JSTARS.2016.2569162

Carruthers, J. (2019). The anthropocene. *South African Journal of Science*, *115*. https://doi.org/10.17159/sajs.2019/6428

Chattopadhyay, S., & Basak, H. (2020). Multi-scale attention u-net (msaunet): A modified u-net architecture for scene segmentation. http://arxiv.org/abs/2009.06911

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. http://arxiv.org/abs/1412.7062

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2016). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. https://doi.org/10.48550/arxiv.1606.00915

Chen, L.-C., Papandreou, G., Schroff, F., & Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. http://arxiv.org/abs/1706.05587

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). *Encoder-decoder with atrous separable convolution for semantic image segmentation.* https://github.com/tensorflow/models/tree/master/

Chen, Z., Duan, Y., Wang, W., He, J., Lu, T., Dai, J., & Qiao, Y. (2022). Vision transformer adapter for dense predictions. http://arxiv.org/abs/2205.08534

Cheng, B., Wu, W., Tao, D., Mei, S., Mao, T., & Cheng, J. (2020). Random cropping ensemble neural network for image classification in a robotic arm grasping system. *IEEE Transactions on Instrumentation and Measurement*, *69*, 6795–6806. https://doi.org/10.1109/TIM.2020.2976420

Cireşan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. http://arxiv.org/abs/1202.2745

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., & Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. https://doi.org/10.48550/ARXIV.1604.01685

Dang, T., Nguyen, T. T., Moreno-García, C. F., Elyan, E., & McCall, J. (2021). Weighted ensemble of deep learning models based on comprehensive learning particle swarm optimization for medical image segmentation. *2021 IEEE Congress on Evolutionary Computation, CEC 2021 - Proceedings*, 744–751. https://doi.org/10.1109/CEC45853.2021.9504929

De Condorcet, M. (1785). *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. de l'Imprimerie Royale.

Demir, I., Koperski, K., Lindenbaum, D., Pang, G., Huang, J., Basu, S., Hughes, F., Tuia, D., & Raska, R. (2018). Deepglobe 2018: A challenge to parse the earth through satellite images. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2018-June*, 172–181. https://doi.org/10.1109/CVPRW.2018.00031

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2010). Imagenet: A large-scale hierarchical image database, 248–255. https://doi.org/10.1109/cvpr.2009.5206848

Development of a 2001 national land-cover database for the united states. (2004). *Photogrammetric Engineering and Remote Sensing*, *70*, 829–840. https://doi.org/10.14358/PERS.70.7.829

Díaz, S., Settele, J., Brondízio, E. S., Ngo, H. T., Agard, J., Arneth, A., Balvanera, P., Brauman, K. A., Butchart, S. H., Chan, K. M., Lucas, A. G., Ichii, K., Liu, J., Subramanian, S. M., Midgley, G. F., Miloslavich, P., Molnár, Z., Obura, D., Pfaff, A., ... Zayas, C. N. (2019). Pervasive human-driven decline of life on earth points to the need for transformative change. *Science*, *366*. https://doi.org/10.1126/science.aax3100

Dong, R., Pan, X., & Li, F. (2019). Denseu-net-based semantic segmentation of small objects in urban remote sensing images. *IEEE Access*, *7*, 65347–65356. https://doi.org/10.1109/ACCESS.2019.2917952

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. http://arxiv.org/abs/2010.11929

Fort, S., Hu, H., & Lakshminarayanan, B. (2019). Deep ensembles: A loss landscape perspective. http://arxiv.org/abs/1912.02757

Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, *14*(771-780), 1612.

Ganaie, M. A., Hu, M., Malik, A. K., Tanveer, M., & Suganthan, P. N. (2021). Ensemble deep learning: A review. https://doi.org/10.48550/arxiv.2104.02395

Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., & Garcia-Rodriguez, J. (2017). A review on deep learning techniques applied to semantic segmentation. http://arxiv.org/abs/1704.06857

Google. (2022a). Google colaboratory [Accessed on 31.05.2022]. https://colab.research.google.com/

Google. (2022b). Ml practicum image classification by google developers [Accessed on 31.05.2022]. https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks

Harangi, B., Baran, A., & Hajdu, A. (2018). Classification of skin lesions using an ensemble of deep neural networks. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, *2018-July*, 2575–2578. https://doi.org/10.1109/EMBC.2018.8512800

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. https://doi.org/10.48550/arxiv.1512.03385

Helber, P., Bischke, B., Dengel, A., & Borth, D. (2017). Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. http://arxiv.org/abs/1709.00029

Hirahara, D., Takaya, E., Takahara, T., & Ueda, T. (2020). Effects of data count and image scaling on deep learning training. *PeerJ Computer Science*, *6*, e312.

Hoffman, J., Wang, D., Yu, F., & Darrell, T. (2016). Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. http://arxiv.org/abs/1612.02649

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2016). Densely connected convolutional networks. https://doi.org/10.48550/arxiv.1608.06993

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

International geosphere biosphere programme (igbp) - surface types [Accessed on 21.05.2022]. (2022). https://www.cso.ie/en/releasesandpublications/ep/p-eii/eii2016/lu/

ISBI. (2022). Segmentation of neuronal structures in em stacks challenge - isbi 2012 [Accessed on 28.05.2022]. https://imagej.net/events/isbi-2012-segmentation-challenge

Jebamikyous, H.-H., & Kashef, R. (2021). Deep learning-based semantic segmentation in autonomous driving. *2021 IEEE 23rd Int Conf on High Performance Computing and Communications*, 1367–1373. https://doi.org/10.1109/HPCC-DSS-SmartCity-DependSys53884.2021.00206

Kemker, R., Luu, R., & Kanan, C. (2018). Low-shot learning for the semantic segmentation of remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing*, *56*, 6214–6223. https://doi.org/10.1109/TGRS.2018.2833808

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. https://doi.org/10.48550/arxiv.1412.6980

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

Land use - cso - central statistics office of ireland [Accessed on 21.05.2022]. (2022). https://ceres.larc.nasa.gov/data/general-product-info/#ceres-surface-type-ids

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*, 2278–2323. https://doi.org/10.1109/5.726791

Ma, X., Wang, H., & Wang, J. (2016). Semisupervised classification for hyperspectral image based on multi-decision labeling and deep feature learning. *ISPRS Journal of Photogrammetry and Remote Sensing*, *120*, 99–107. https://doi.org/10.1016/j.isprsjprs.2016.09.001

Marmanis, D., Schindler, K., Wegner, J. D., Galliani, S., Datcu, M., & Stilla, U. (2018). Classification with an edge: Improving semantic image segmentation with boundary detection. *ISPRS Journal of Photogrammetry and Remote Sensing*, *135*, 158–172. https://doi.org/10.1016/j.isprsjprs.2017.11.009

Marmanis, D., Wegner, J. D., Galliani, S., Schindler, K., Datcu, M., & Stilla, U. (2016). Semantic segmentation of aerial images with an ensemble of cnns. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *III-3*, 473–480. https://doi.org/10.5194/isprsannals-iii-3-473-2016

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, ... Xiaoqiang Zheng. (2015). TensorFlow: Large-scale machine learning on het-

erogeneous systems [Software available from tensorflow.org]. https://www.tensorflow.org/

Minaee, S., Boykov, Y. Y., Porikli, F., Plaza, A. J., Kehtarnavaz, N., & Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2021.3059968

Mo, Y., Wu, Y., Yang, X., Liu, F., & Liao, Y. (2022). Review the state-of-the-art technologies of semantic segmentation based on deep learning. *Neurocomputing*, *493*, 626–646. https://doi.org/10.1016/j.neucom.2022.01.005

Mottaghi, R., Chen, X., Liu, X., Cho, N. G., Lee, S. W., Fidler, S., Urtasun, R., & Yuille, A. (2014). The role of context for object detection and semantic segmentation in the wild. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 891–898. https://doi.org/10.1109/CVPR.2014.119

Neupane, B., Horanont, T., & Aryal, J. (2021). Deep learning-based semantic segmentation of urban features in satellite images: A review and meta-analysis. *Remote Sensing*, *13*, 1–41. https://doi.org/10.3390/rs13040808

Nigam, I., Huang, C., & Ramanan, D. (2018). Ensemble knowledge transfer for semantic segmentation. *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018*, *2018-January*, 1499–1508. https://doi.org/10.1109/WACV.2018.00168

Novelli, A., Aguilar, M., Aguilar, F., Nemmaoui, A., & Tarantino, E. (2017). Assesseg—a command line tool to quantify image segmentation quality: A test carried out in southern spain from satellite imagery. *Remote Sensing*, *9*, 40. https://doi.org/10.3390/rs9010040

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. https://doi.org/10.48550/arxiv.1511.08458

pandas development team, T. (2020). Pandas-dev/pandas: Pandas. https://doi.org/10.5281/zenodo.3509134

Pavel, I. (2022). Github qubvel classification models trained on imagenet keras [Accessed on 31.05.2022]. https://github.com/qubvel/classification_models

Poomani, M., Sutha, J., & Soundar, K. R. (2021). Wiener filter based deep convolutional network approach for classification of satellite images. *Journal of Ambient Intelligence and Humanized Computing*, *12*, 7343–7351. https://doi.org/10.1007/s12652-020-02410-3

R Core Team. (2013). R: A language and environment for statistical computing [ISBN 3-900051-07-0]. http://www.R-project.org/

Raina, R., Madhavan, A., & Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. *ACM International Conference Proceeding Series*, *382*, 1–8. https://doi.org/10.1145/1553374.1553486

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9351*, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Russia planning massive military offensive against ukraine involving 175,000 troops, u.s. intelligence warns - the washington post [Accessed on 15.05.2022]. (2022). https://www.washingtonpost.com/national-security/russia-ukraine-invasion/2021/12/03/98a3760e-546b-11ec-8769-2f4ecdf7a2ad_story.html

Sang, D. V., & Minh, N. D. (2018). Fully residual convolutional neural networks for aerial image segmentation. *ACM International Conference Proceeding Series*, 289–296. https://doi.org/10.1145/3287921.3287970

Satellite companies join the hunt for russian war crimes - politico [Accessed on 15.05.2022]. (2022). https://www.politico.com/news/2022/04/06/satellite-russian-war-crimes-00023386

Satellite database — union of concerned scientists [Accessed on 15.05.2022]. (2022). https://www.ucsusa.org/resources/satellite-database

Sawilowsky, S. S. (2009). New effect size rules of thumb. *Journal of modern applied statistical methods*, *8*(2), 26.

Shah, B., & Madabushi, H. T. (2021). Efficient brain tumour segmentation using co-registered data and ensembles of specialised learners. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12659 LNCS*, 15–29. https://doi.org/10.1007/978-3-030-72087-2_2

Shelhamer, E., Long, J., & Darrell, T. (2017). Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*, 640–651. https://doi.org/10.1109/TPAMI.2016.2572683

Sherrah, J. (2016). Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. http://arxiv.org/abs/1606.02585

Shimizu, A., Narihira, T., Furukawa, D., Kobatake, H., Nawano, S., & Shinozaki, K. (2008). Ensemble segmentation using adaboost with application to liver lesion extraction from a ct volume.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. https://doi.org/10.48550/arxiv.1409.1556

Smaida, M., & Yaroshchak, S. (2020). Bagging of convolutional neural networks for diagnostic of eye diseases.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

Strudel, R., Garcia, R., Inria, I. L., & Inria, C. S. (2021). *Segmenter: Transformer for semantic segmentation*. https://github.com/rstrudel/segmenter

Su, Z., Li, W., Ma, Z., & Gao, R. (2022). An improved u-net method for the semantic segmentation of remote sensing images. *Applied Intelligence*, *52*, 3276–3288. https://doi.org/10.1007/s10489-021-02542-9

Sultana, F., Sufian, A., & Dutta, P. (2020). Evolution of image segmentation using deep convolutional neural network: A survey. *Knowledge-Based Systems*, *201-202*. https://doi.org/10.1016/j.knosys.2020.106062

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going deeper with convolutions. https://doi.org/10.48550/arxiv.1409.4842

Taghanaki, S. A., Abhishek, K., Cohen, J. P., Cohen-Adad, J., & Hamarneh, G. (2021). Deep semantic segmentation of natural and medical images: A review. *Artificial Intelligence Review*, *54*, 137–178. https://doi.org/10.1007/s10462-020-09854-1

Taherkhani, A., Cosma, G., & McGinnity, T. M. (2020). Adaboost-cnn: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning. *Neurocomputing*, *404*, 351–366. https://doi.org/10.1016/j.neucom.2020.03.064

Tapper, G., Sundelius, C., & Haglund, L. (2020). Global semantic land use/land cover based on high resolution satellite imagery using ensemble networks. *International Geoscience and Remote Sensing Symposium (IGARSS)*, 1070–1073. https://doi.org/10.1109/IGARSS39084.2020.9324267

Ukraine war: Satellite images appear to contradict russian denials over bucha atrocities — euronews [Accessed on 15.05.2022]. (2022). https://www.euronews.com/2022/04/05/ukraine-war-satellite-images-appear-to-contradict-russian-denials-over-bucha-atrocities

Vallat, R. (2018). Pingouin: Statistics in python software • review • repository • archive. https://doi.org/10.21105/joss.01026

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. https://doi.org/10.48550/arxiv.1706.03762

Wang, L., Li, R., Zhang, C., Fang, S., Duan, C., Meng, X., & Atkinson, P. M. (2021). Unetformer: An unet-like transformer for efficient semantic segmentation of remote sensing urban scene imagery. https://doi.org/10.48550/arxiv.2109.08937

Waskom, M. L. (2021). Seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Wei, Z., Yang, A., Rocha, L., Miranda, M. F., & Nathoo, F. S. (2022). A review of bayesian hypothesis testing and its practical implementations. *Entropy*, *24*(2), 161.

Wu, M., Zhang, C., Liu, J., Zhou, L., & Li, X. (2019). Towards accurate high resolution satellite image semantic segmentation. *IEEE Access*, *7*, 55609–55619. https://doi.org/10.1109/ACCESS.2019.2913442

Yakubovskiy, P. (2019). Segmentation models. https://github.com/qubvel/segmentation_models

Yu, B., Yang, L., & Chen, F. (2018). Semantic segmentation for high spatial resolution remote sensing images based on convolution neural network and pyramid pooling module. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *11*, 3252–3261. https://doi.org/10.1109/JSTARS.2018.2860989

Yuan, K., Zhuang, X., Schaefer, G., Feng, J., Guan, L., & Fang, H. (2021). Deep-learning-based multispectral satellite image segmentation for water body detection. *IEEE Journal of Selected Topics in Applied Earth Observations*

and Remote Sensing, *14*, 7422–7434. https://doi.org/10.1109/JSTARS.2021.3098678

Zhang, X., Zhou, X., Lin, M., & Sun, J. (2017). Shufflenet: An extremely efficient convolutional neural network for mobile devices. https://doi.org/10.48550/arxiv.1707.01083

Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid scene parsing network, 2881–2890.

Zhao, W., Du, S., Wang, Q., & Emery, W. J. (2017). Contextually guided very-high-resolution imagery classification with semantic segments. *ISPRS Journal of Photogrammetry and Remote Sensing*, *132*, 48–60. https://doi.org/10.1016/j.isprsjprs.2017.08.011

Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., & Torralba, A. (2016). Semantic understanding of scenes through the ade20k dataset. http://arxiv.org/abs/1608.05442

Zhou, T., Lu, H., Yang, Z., Qiu, S., Huo, B., & Dong, Y. (2021). The ensemble deep learning model for novel covid-19 on ct images. *Applied Soft Computing*, *98*, 106885. https://doi.org/10.1016/j.asoc.2020.106885

Zhou, Z., & Gong, J. (2018). Automated residential building detection from airborne lidar data with deep neural networks. *Advanced Engineering Informatics*, *36*, 229–241. https://doi.org/10.1016/j.aei.2018.04.002

Zhu, H., Meng, F., Cai, J., & Lu, S. (2016). Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation. *Journal of Visual Communication and Image Representation*, *34*, 12–27. https://doi.org/10.1016/j.jvcir.2015.10.012

Zhu, X. X., Tuia, D., Mou, L., Xia, G. S., Zhang, L., Xu, F., & Fraundorfer, F. (2017). Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Geoscience and Remote Sensing Magazine*, *5*, 8–36. https://doi.org/10.1109/MGRS.2017.2762307

Appendix A

Additional content

A.0.1 Example model predictions

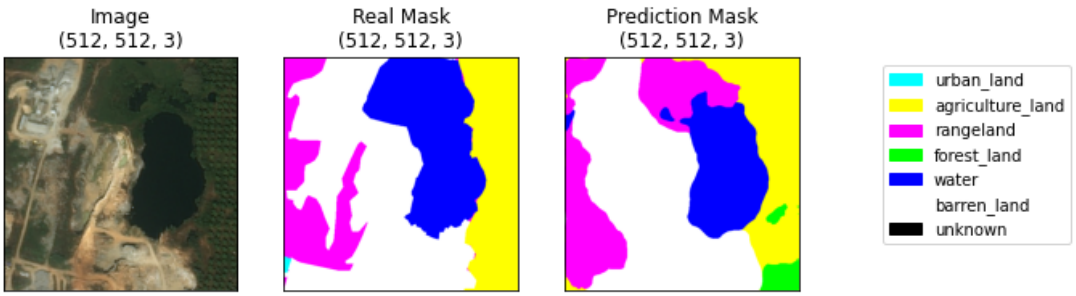


Figure A.1: Sample 1, Model 'average' ensemble

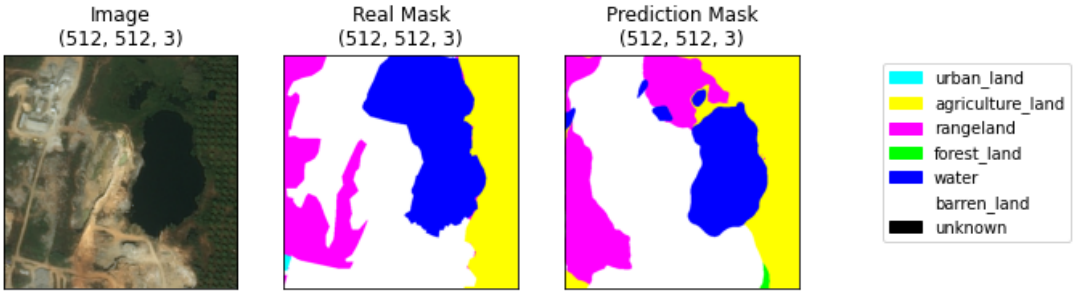


Figure A.2: Sample 1, Model 'base_avg'

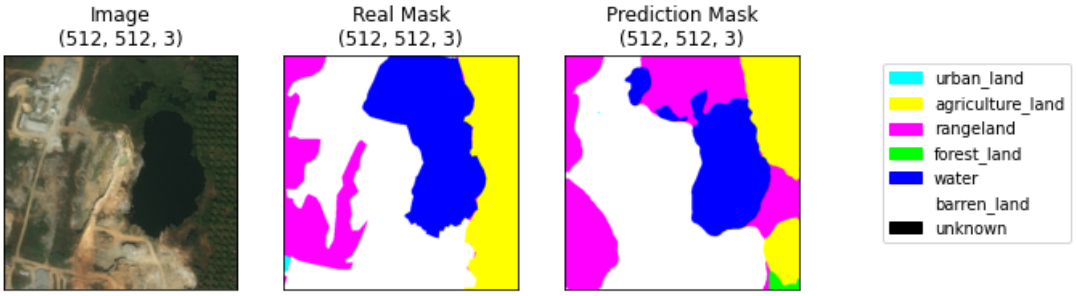


Figure A.3: Sample 1, Model 'base_max'

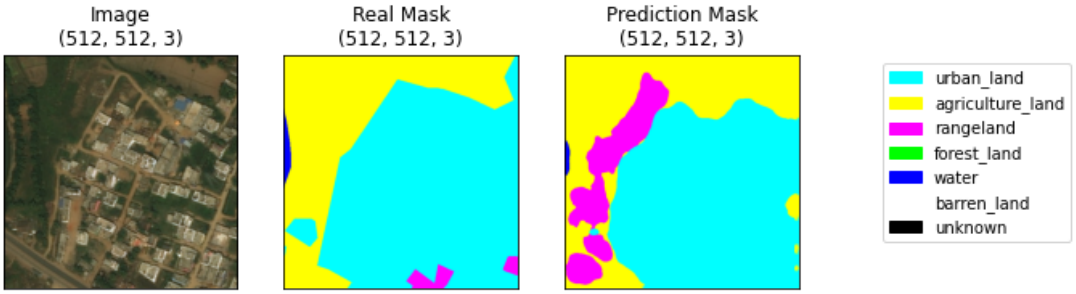


Figure A.4: Sample 2, Model 'average' ensemble

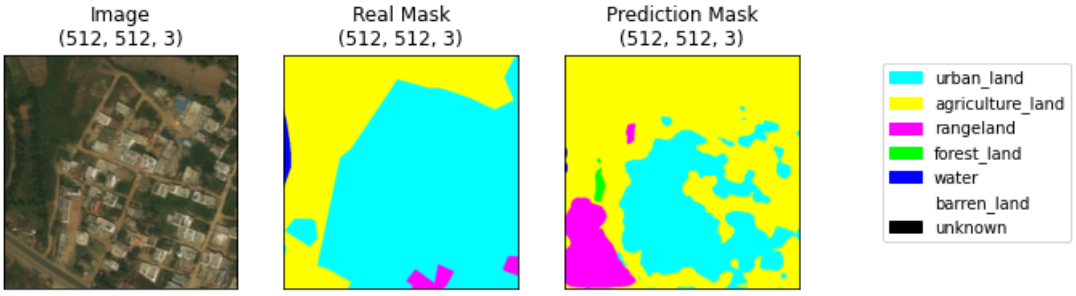


Figure A.5: Sample 2, Model 'base_avg'

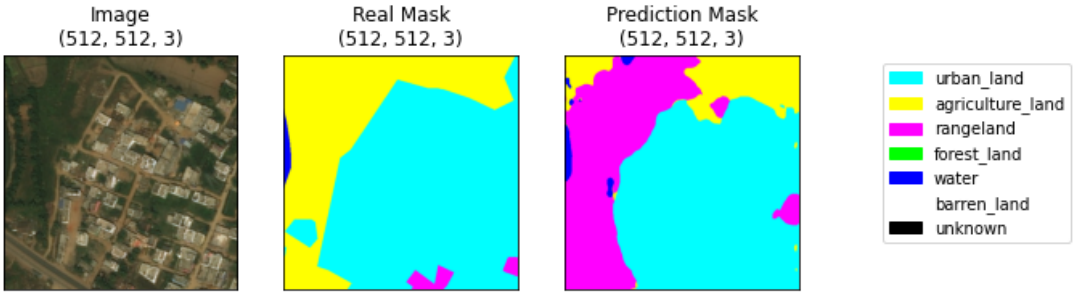


Figure A.6: Sample 2, Model 'base_max'

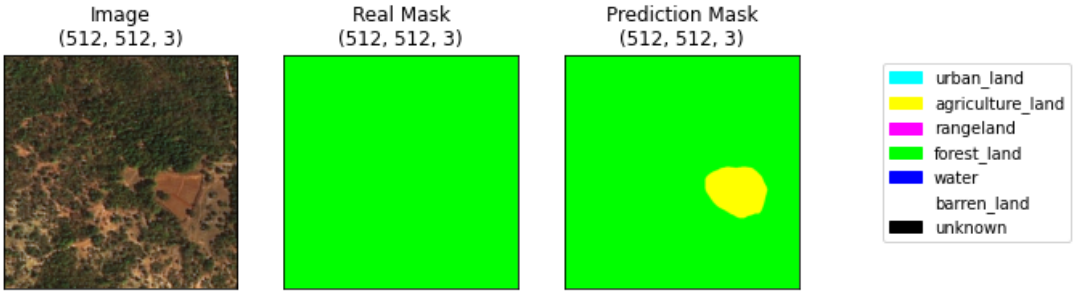


Figure A.7: Sample 3, Model 'average' ensemble

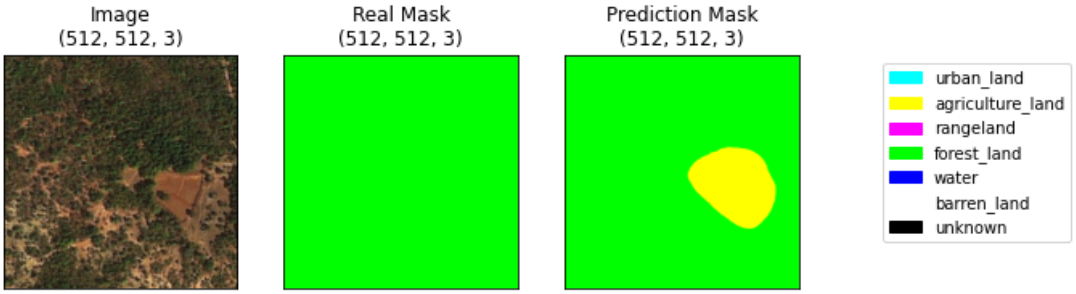


Figure A.8: Sample 3, Model 'base_avg'

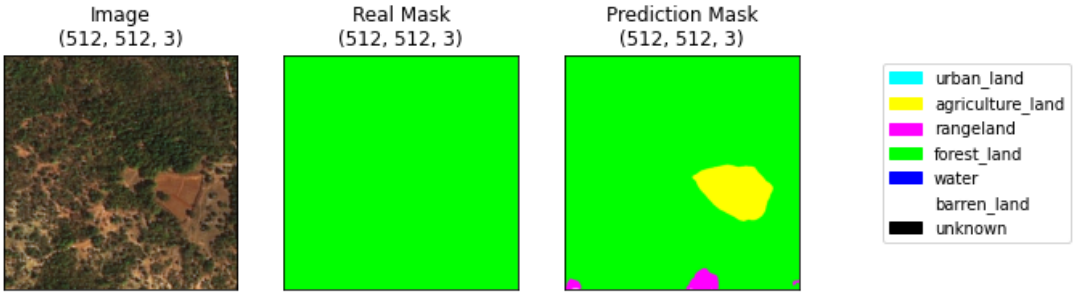


Figure A.9: Sample 3, Model 'base_max'

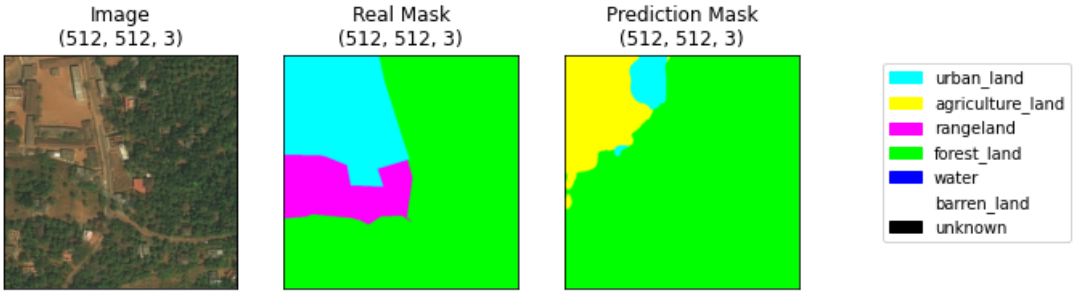


Figure A.10: Sample 4, Model 'average' ensemble

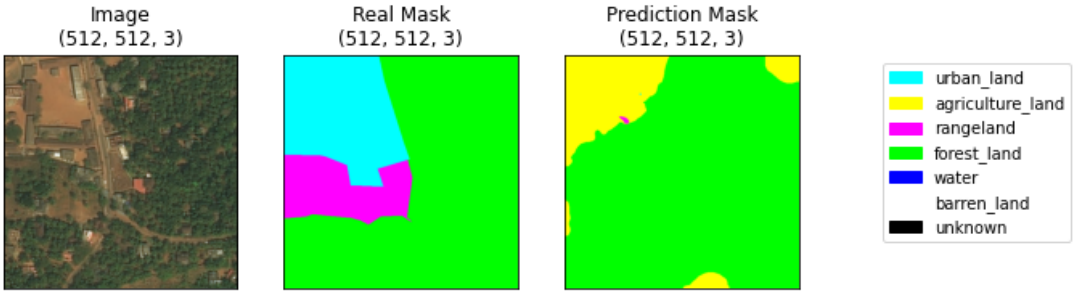


Figure A.11: Sample 4, Model 'base_avg'

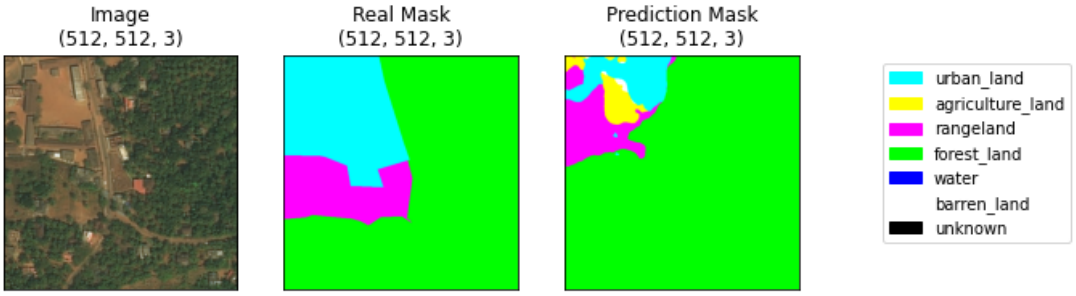


Figure A.12: Sample 4, Model 'base_max'

A.0.2 Example structure of U-Net Keras model 'base_max'

layer.name	layer.__class__	trainable	count_params	filters	kernel size
data	InputLayer	True	0		
bn_data	BatchNormalization	True	9		
zero_padding2d_136	ZeroPadding2D	False	0		
conv0	Conv2D	False	9408	64	(7, 7)
bn0	BatchNormalization	True	256		
relu0	Activation	False	0		
zero_padding2d_137	ZeroPadding2D	False	0		
pooling0	MaxPooling2D	False	0		
stage1_unit1_bn1	BatchNormalization	True	256		
stage1_unit1_relu1	Activation	False	0		
zero_padding2d_138	ZeroPadding2D	False	0		
stage1_unit1_conv1	Conv2D	False	36864	64	(3, 3)
stage1_unit1_bn2	BatchNormalization	True	256		
stage1_unit1_relu2	Activation	False	0		
zero_padding2d_139	ZeroPadding2D	False	0		
stage1_unit1_conv2	Conv2D	False	36864	64	(3, 3)
stage1_unit1_sc	Conv2D	False	4096	64	(1, 1)
add_64	Add	False	0		
stage1_unit2_bn1	BatchNormalization	True	256		
stage1_unit2_relu1	Activation	False	0		
zero_padding2d_140	ZeroPadding2D	False	0		
stage1_unit2_conv1	Conv2D	False	36864	64	(3, 3)
stage1_unit2_bn2	BatchNormalization	True	256		
stage1_unit2_relu2	Activation	False	0		
zero_padding2d_141	ZeroPadding2D	False	0		
stage1_unit2_conv2	Conv2D	False	36864	64	(3, 3)
add_65	Add	False	0		
stage1_unit3_bn1	BatchNormalization	True	256		

Table A.1: Model 'base_max' structure in Keras. Part 1

layer.name	layer.__class__	trainable	count_params	filters	kernel size
stage1_unit3_relu1	Activation	False	0		
zero_padding2d_142	ZeroPadding2D	False	0		
stage1_unit3_conv1	Conv2D	False	36864	filters:64	(3, 3)
stage1_unit3_bn2	BatchNormalization	True	256		
stage1_unit3_relu2	Activation	False	0		
zero_padding2d_143	ZeroPadding2D	False	0		
stage1_unit3_conv2	Conv2D	False	36864	64	(3, 3)
add_66	Add	False	0		
stage2_unit1_bn1	BatchNormalization	True	256		
stage2_unit1_relu1	Activation	False	0		
zero_padding2d_144	ZeroPadding2D	False	0		
stage2_unit1_conv1	Conv2D	False	73728	128	(3, 3)
stage2_unit1_bn2	BatchNormalization	True	512		
stage2_unit1_relu2	Activation	False	0		
zero_padding2d_145	ZeroPadding2D	False	0		
stage2_unit1_conv2	Conv2D	False	147456	128	(3, 3)
stage2_unit1_sc	Conv2D	False	8192	128	(1, 1)
add_67	Add	False	0		
stage2_unit2_bn1	BatchNormalization	True	512		
stage2_unit2_relu1	Activation	False	0		
zero_padding2d_146	ZeroPadding2D	False	0		
stage2_unit2_conv1	Conv2D	False	147456	128	(3, 3)
stage2_unit2_bn2	BatchNormalization	True	512		
stage2_unit2_relu2	Activation	False	0		
zero_padding2d_147	ZeroPadding2D	False	0		
stage2_unit2_conv2	Conv2D	False	147456	128	(3, 3)
add_68	Add	False	0		
stage2_unit3_bn1	BatchNormalization	True	512		
stage2_unit3_relu1	Activation	False	0		
zero_padding2d_148	ZeroPadding2D	False	0		
stage2_unit3_conv1	Conv2D	False	147456	128	(3, 3)

Table A.2: Model 'base_max' structure in Keras. Part 2

layer.name	layer.__class__	trainable	count_params	filters	kernel size
stage2_unit3_bn2	BatchNormalization	True	512		
stage2_unit3_relu2	Activation	False	0		
zero_padding2d_149	ZeroPadding2D	False	0		
stage2_unit3_conv2	Conv2D	False	147456	128	(3, 3)
add_69	Add	False	0		
stage2_unit4_bn1	BatchNormalization	True	512		
stage2_unit4_relu1	Activation	False	0		
zero_padding2d_150	ZeroPadding2D	False	0		
stage2_unit4_conv1	Conv2D	False	147456	128	(3, 3)
stage2_unit4_bn2	BatchNormalization	True	512		
stage2_unit4_relu2	Activation	False	0		
zero_padding2d_151	ZeroPadding2D	False	0		
stage2_unit4_conv2	Conv2D	False	147456	128	(3, 3)
add_70	Add	False	0		
stage3_unit1_bn1	BatchNormalization	True	512		
stage3_unit1_relu1	Activation	False	0		
zero_padding2d_152	ZeroPadding2D	False	0		
stage3_unit1_conv1	Conv2D	False	294912	256	(3, 3)
stage3_unit1_bn2	BatchNormalization	True	1024		
stage3_unit1_relu2	Activation	False	0		
zero_padding2d_153	ZeroPadding2D	False	0		
stage3_unit1_conv2	Conv2D	False	589824	256	(3, 3)
stage3_unit1_sc	Conv2D	False	32768	256	(1, 1)
add_71	Add	False	0		
stage3_unit2_bn1	BatchNormalization	True	1024		

Table A.3: Model 'base_max' structure in Keras. Part 3

layer.name	layer.__class__	trainable	count_params	filters	kernel size
stage3_unit2_relu1	Activation	False	0		
zero_padding2d_154	ZeroPadding2D	False	0		
stage3_unit2_conv1	Conv2D	False	589824	256	(3, 3)
stage3_unit2_bn2	BatchNormalization	True	1024		
stage3_unit2_relu2	Activation	False	0		
zero_padding2d_155	ZeroPadding2D	False	0		
stage3_unit2_conv2	Conv2D	False	589824	256	(3, 3)
add_72	Add	False	0		
stage3_unit3_bn1	BatchNormalization	True	1024		
stage3_unit3_relu1	Activation	False	0		
zero_padding2d_156	ZeroPadding2D	False	0		
stage3_unit3_conv1	Conv2D	False	589824	256	(3, 3)
stage3_unit3_bn2	BatchNormalization	True	1024		
stage3_unit3_relu2	Activation	False	0		
zero_padding2d_157	ZeroPadding2D	False	0		
stage3_unit3_conv2	Conv2D	False	589824	256	(3, 3)
add_73	Add	False	0		
stage3_unit4_bn1	BatchNormalization	True	1024		
stage3_unit4_relu1	Activation	False	0		
zero_padding2d_158	ZeroPadding2D	False	0		
stage3_unit4_conv1	Conv2D	False	589824	256	(3, 3)
stage3_unit4_bn2	BatchNormalization	True	1024		
stage3_unit4_relu2	Activation	False	0		
zero_padding2d_159	ZeroPadding2D	False	0		
stage3_unit4_conv2	Conv2D	False	589824	256	(3, 3)
add_74	Add	False	0		
stage3_unit5_bn1	BatchNormalization	True	1024		
stage3_unit5_relu1	Activation	False	0		
zero_padding2d_160	ZeroPadding2D	False	0		

Table A.4: Model 'base_max' structure in Keras. Part 4

layer.name	layer.__class__	trainable	count_params	filters	kernel size
stage3_unit5_conv1	Conv2D	False	589824	256	(3, 3)
stage3_unit5_bn2	BatchNormalization	True	1024		
stage3_unit5_relu2	Activation	False	0		
zero_padding2d_161	ZeroPadding2D	False	0		
stage3_unit5_conv2	Conv2D	False	589824	256	(3, 3)
add_75	Add	False	0		
stage3_unit6_bn1	BatchNormalization	True	1024		
stage3_unit6_relu1	Activation	False	0		
zero_padding2d_162	ZeroPadding2D	False	0		
stage3_unit6_conv1	Conv2D	False	589824	256	(3, 3)
stage3_unit6_bn2	BatchNormalization	True	1024		
stage3_unit6_relu2	Activation	False	0		
zero_padding2d_163	ZeroPadding2D	False	0		
stage3_unit6_conv2	Conv2D	False	589824	256	(3, 3)
add_76	Add	False	0		
stage4_unit1_bn1	BatchNormalization	True	1024		
stage4_unit1_relu1	Activation	False	0		
zero_padding2d_164	ZeroPadding2D	False	0		
stage4_unit1_conv1	Conv2D	False	1179648	512	(3, 3)
stage4_unit1_bn2	BatchNormalization	True	2048		
stage4_unit1_relu2	Activation	False	0		
zero_padding2d_165	ZeroPadding2D	False	0		
stage4_unit1_conv2	Conv2D	False	2359296	512	(3, 3)
stage4_unit1_sc	Conv2D	False	131072	512	(1, 1)
add_77	Add	False	0		
stage4_unit2_bn1	BatchNormalization	True	2048		
stage4_unit2_relu1	Activation	False	0		
zero_padding2d_166	ZeroPadding2D	False	0		
stage4_unit2_conv1	Conv2D	False	2359296	512	(3, 3)
stage4_unit2_bn2	BatchNormalization	True	2048		

Table A.5: Model 'base_max' structure in Keras. Part 5

layer.name	layer.__class__	trainable	count_params	filters	kernel size
stage4_unit2_relu2	Activation	False	0		
zero_padding2d_167	ZeroPadding2D	False	0		
stage4_unit2_conv2	Conv2D	False	2359296	512	(3, 3)
add_78	Add	False	0		
stage4_unit3_bn1	BatchNormalization	True	2048		
stage4_unit3_relu1	Activation	False	0		
zero_padding2d_168	ZeroPadding2D	False	0		
stage4_unit3_conv1	Conv2D	False	2359296	512	(3, 3)
stage4_unit3_bn2	BatchNormalization	True	2048		
stage4_unit3_relu2	Activation	False	0		
zero_padding2d_169	ZeroPadding2D	False	0		
stage4_unit3_conv2	Conv2D	False	2359296	512	(3, 3)
add_79	Add	False	0		
bn1	BatchNormalization	True	2048		
relu1	Activation	False	0		
decoder_stage0_upsam UpSampling2D	True	0	size: (2, 2)		
decoder_stage0_conca Concatenate	True	0			
decoder_stage0a_conv Conv2D	True	1769472	filters:256	(3, 3)	
decoder_stage0a_bn	BatchNormalization	True	1024		
decoder_stage0a_relu Activation	True	0			
decoder_stage0b_conv Conv2D	True	589824	filters:256	(3, 3)	
decoder_stage0b_bn	BatchNormalization	True	1024		
decoder_stage0b_relu Activation	True	0			
decoder_stage1_upsam UpSampling2D	True	0	size: (2, 2)		
decoder_stage1_conca Concatenate	True	0			
decoder_stage1a_conv Conv2D	True	442368	filters:128	(3, 3)	
decoder_stage1a_bn	BatchNormalization	True	512		
decoder_stage1a_relu Activation	True	0			
decoder_stage1b_conv Conv2D	True	147456	filters:128	(3, 3)	
decoder_stage1b_bn	BatchNormalization	True	512		
decoder_stage1b_relu Activation	True	0			

Table A.6: Model 'base_max' structure in Keras. Part 6

layer.name	layer.__class__	trainable	count_params	filters	kernel size
decoder_stage2_upsam UpSampling2D	True	0	size: (2, 2)		
decoder_stage2_conca Concatenate	True	0			
decoder_stage2a_conv Conv2D	True	110592	filters:64	(3, 3)	
decoder_stage2a_bn	BatchNormalization	True	256		
decoder_stage2a_relu Activation	True	0			
decoder_stage2b_conv Conv2D	True	36864	filters:64	(3, 3)	
decoder_stage2b_bn	BatchNormalization	True	256		
decoder_stage2b_relu Activation	True	0			
decoder_stage3_upsam UpSampling2D	True	0	size: (2, 2)		
decoder_stage3_conca Concatenate	True	0			
decoder_stage3a_conv Conv2D	True	36864	filters:32	(3, 3)	
decoder_stage3a_bn	BatchNormalization	True	128		
decoder_stage3a_relu Activation	True	0			
decoder_stage3b_conv Conv2D	True	9216	filters:32	(3, 3)	
decoder_stage3b_bn	BatchNormalization	True	128		
decoder_stage3b_relu Activation	True	0			
decoder_stage4_upsam UpSampling2D	True	0	size: (2, 2)		
decoder_stage4a_conv Conv2D	True	4608	filters:16	(3, 3)	
decoder_stage4a_bn	BatchNormalization	True	64		
decoder_stage4a_relu Activation	True	0			
decoder_stage4b_conv Conv2D	True	2304	filters:16	(3, 3)	
decoder_stage4b_bn	BatchNormalization	True	64		
decoder_stage4b_relu Activation	True	0			
final_conv	Conv2D	True	1015	7	(3, 3)
softmax	Activation	True	0		

Table A.7: Model 'base_max' structure in Keras

A.0.3 Source Code

GitLab: https://gitlab.com/brendankent/mythesiscode