UNIVERSITY of STIRLING

Division of Computing Science and Mathematics
Faculty of Natural Sciences
University of Stirling

Semantic Segmentation of Amazon Floodplains using Multi-sensor Satellite Imagery

Ian Scott

Dissertation submitted in partial fulfilment for the degree of
Master of Science in Artificial Intelligence

October 2021

Abstract

Floodplains are areas next to to rivers and are periodically flooded. They are important hydrologically and are ecologically productive areas that perform many natural roles. These include creating habitats for fish and other animals. Given this, it is vital to map these regions to observe and understand how they are changing over time. Due to the terrain characteristic and vast size of areas considered, it seems unreasonable to track this evolution in real time from land in the Amazon basin. Remote sensing utilising satellite data can provide a solution and improve conservation knowledge of the Amazon basin, helping to detect changes or deforestation in the region.

This dissertation aims to propose a deep learning solution to map floodplain forest areas of the Amazon basin using multi-temporal satellite imagery. A novel dataset is created consisting of optical LANDSAT 5 and L-band SAR PALSAR 1 imagery, in addition to AW3D elevation data, at 12.5m spatial resolution.

Significant advances have been made recently with Fully Convolutional Neural Networks (FCNs) in the field of image processing. Three FCN networks (FCN8s, SegNet, VGG16 U-Net) are trained and validated and a model selection performed. VGG16 U-Net is found to perform best, and achieves F1-Scores of 0.748-0.891 on certain classes, with a balanced accuracy of 0.616.

The VGG16 U-Net is then used to produce a map of an unseen satellite image by breaking the image down into smaller tiles for prediction, and then stitching the outputs back together.

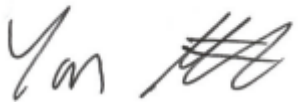
This all demonstrates the proposed architecture is capable of mapping the floodplain areas of the Amazon.

Attestation

I understand the nature of plagiarism, and I am aware of the University's policy on this.

I certify that this dissertation reports original work by me during my University project except for the following (*adjust the list below according to the circumstances*):

- The data was collated by Dr Thiago Silva, University of Stirling

Signature  **Date 15/10/2021**

Acknowledgements

I would like to thank Dr Keiller Nogueira and Dr Thiago Silva for the opportunity to work on this project.

I would also like to thank COVID-19 for inducing lock downs throughout the year, forcing me to knuckle down and study.

I would also like to thank my wife for bank rolling me through my studies for the year and managing three kids enough that I was able to bash this report out.

Table of Contents

List of Figures

1 Introduction

1.1 Motivation

The Amazon basin, Amazon thereafter, contains the largest rainforest in the world, covering nearly 7 million square kilometres of land. In the context of climate change and of reduction of CO2 emissions, the Amazon plays a vital role as a carbon sink locking away in the order of 1.1 x 10^{11} metric tonnes of carbon [20]. However, in 2021 the Smithsonian magazine reported that the Amazon now emits more greenhouse gases than it absorbs, siting the cause as human activity. Furthermore, climate change could render the Amazon unsustainable due to a severe reduction in rainfall and increased temperatures which would lead to a change of cover in the Amazon basin. This could have far reaching consequences for the entire planet. In the light of this information it seems crucial to accurately track and quantify such changes. Knowing changes in the cover of each area of the Amazon will help decision makers and scientists to assess the condition of the basin. Some areas, such as floodplains, are of specific interest. These are areas are situated adjacent to rivers that are periodically flooded. They are important hydrologically and are ecologically productive areas that perform many natural roles. As such, it is vital to map these regions and observe and understand how they are changing over time. Floodplains are constantly evolving and due to the terrain characteristic and vast size of areas considered, it seems unreasonable to track this evolution in real time from land.

Remote sensing utilising satellite data can provide a solution and improve conservation knowledge of the Amazon basin, helping to detect changes or deforestation in the region including the floodplains. Manually monitor the Amazon using remote sensing imagens is not feasible either, and automatic methods should be used.

The goal of this project is to provide automatic method for the identification of the land cover of the Amazon basin from satellite data using deep learning techniques. The specific identification and segmentation of land cover types in wetland forest situated on Amazon floodplains is the main focus of this project. In order to do this a novel dataset has been created and three Fully Convolutional Neural networks (FCNs) architectures have been trained and tested on the dataset.

Although remote sensing is essential for this type of task, the use of images composed only of visible spectrum data may not be sufficient. Atmospheric effects, particularly in tropical regions, limit what a satellite can capture in RGB images. Therefore, it is essential to use data from other parts of the spectrum as well as other sources, such as SAR. Though, the combination of images that are quite different in nature to a typical RGB image imposes new challenges.

Synthetic-aperture radar (SAR) - what is it?

Synthetic-aperture radar (SAR) is a type of radar that can be used in remote sensing to create images of landscapes. SAR requires a moving object (satellite) to provide a radar antenna above a target region. The satellite emits pulses of electromagnetic radiation with wavelengths between one meter down to several millimetres. These emissions illuminate the scene below and the distance the satellite moves during the pulse creates a synthetic antenna aperture, simulating a much larger antenna. The echo from each pulse is recorded, and signal processing is used to combine the recordings from multiple antenna positions yielding a higher resolution than possible with a conventional stationary antenna.

Remote sensing meets Deep learning

In January 2021 Zhu et al. [4] explained that most deep learning in remote sensing to date has been limited to optical data, and that synthetic aperture radar (SAR) data's potential remains locked. In their last paragraph, they summarise the need for collaboration on the subject:

"Last but not least, technology advances in deep learning in remote sensing would only be possible if experts in remote sensing and machine learning work closely together. This is particularly true when it comes to SAR. Thus, we encourage more joint initiatives working collaboratively toward deep learning powered, explainable and reproducible big SAR data analytics."

At a seminar held in March 2015, São Paulo, the following question was posed to remote sensing and vegetation mapping experts:

"Is it possible to produce annual land use and land cover maps, for the entire country, in a significantly cheaper, faster and more updated way, compared to current methods and practices, allowing the possibility of recovering the history of the last decades?"

The **Brazilian Annual Land Use and Land Cover Mapping Project** (MapBiomas) was formed in response.

This dissertation will deal with how semantic segmentation of satellite images can be used to map land cover of the amazon basin. The project sits on the machine learning side and is aimed at assisting remote sensing experts with deep learning techniques. The goal is that the findings from this project can be taken forward to contribute to the MapBiomas project.

1.2 Scope and Objectives

The aim of this project is to propose a deep learning-based model to map floodplain forest areas using multisensory satellite data.

The main activities are:

- Review literature on deep learning semantic segmentation techniques
- Learn Pytorch framework in order to be able to build and train deep learning models
- Read data from earth observation sector file formats, combine different channels together and create a new dataset to be used in training
- Propose a deep learning model to classify pixels and segment regions in sensor images into various classes.
- Build models, perform tuning of models and make performance comparisons between them
- Report on findings

1.3 Achievements

This dissertation demonstrates it is possible to map the floodplains of the Amazon basin from satellite data using deep learning techniques. Three models are proposed, tuned and evaluated against one another. VGG16-U-Net, is proposed for the task and tested. The results of the test are discussed, and conclusions drawn. A demonstration of how the model can be used to make inference on large satellite patches is given, as it would need to do in deployment.

The objectives of the project are all met and documented in this report.

1.4 Overview of Dissertation

This dissertation is arranged into eight chapters. An overview of chapter is given below:

Chapter 1 – Introduction of the dissertation, a background to the topic, scope and objectives of the project and achievements.

Chapter 2 – Background of relevant remote sensing topics, and deep learning techniques for semantic segmentation of forest area. Introduction of the reasoning for the project approach.

Chapter 3 – The sources of the novel dataset are presented here, and how it is extracted from earth observation files.

Chapter 4 – The methodology used in the project is explained and defined here.

Chapter 5 – Experimental setup is provided, stating what analysis has been performed.

Chapter 6 – Results and discussion, with an evaluation of the final model.

Chapter 7 – A demonstration of how the model can be used to make inference on large satellite patches. The discussion from 6 is continued.

Chapter 8 – A conclusion is drawn, summarising and critically evaluating the project. The limitations of the work are presented and potential areas for future work are identified.

2 Background

2.1 Introduction

This project will be applying deep learning to multisensory data of the Amazon collected via satellite. The main focus of this project will sit on the deep learning side, however, as always, it is important to understand what the data is, where it comes from and in this case why it was chosen. The following section is therefore divided into two parts, remote sensing and deep learning, providing a background for each and providing some similar works to make comparisons to in later chapters.

2.2 Remote sensing

This section outlines the theoretical background of the data types and gives an idea as to what information can be gained from them to aid in the classification of wetland forest. It assumes the reader has a good knowledge of light and vision in addition to a basic understanding of the electromagnetic spectrum and related terminology. We will talk about wavelengths in the coming section and so Figure 1 has been provided to aid the reader in relating the wavelength magnitude to the different bands of the spectrum, if they so wish.

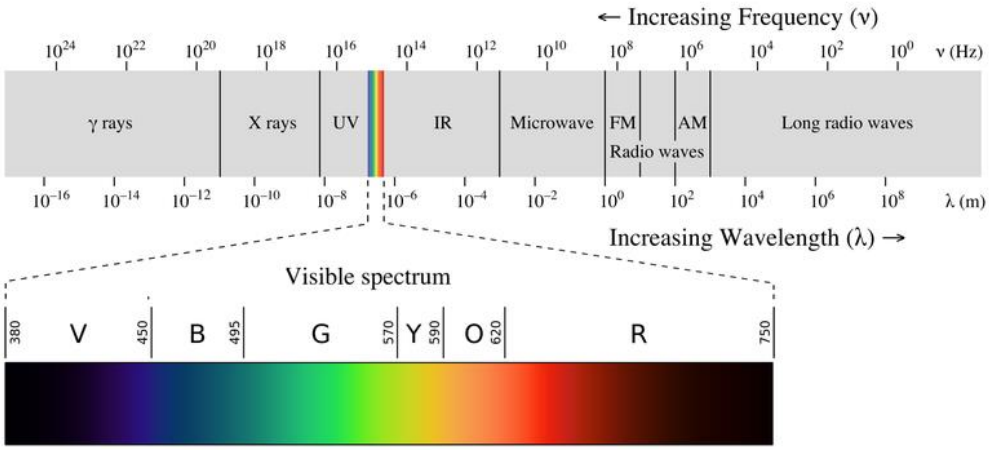


Figure 1 – The electromagnetic spectrum with wavelength and frequency, taken from https://en.wikipedia.org/wiki/Electromagnetic_radiation

What follows is not an exhaustive explanation of why each channel in the project dataset has been chosen as level of detail would fall outside the scope of an AI masters project. For further details of this nuanced domain knowledge see [15] as a starting point.

2.2.1 How can SAR help in mapping wetland forest areas in the Amazon?

As SAR relies on microwaves, it is possible to select frequencies that avoid signal attenuation due to atmospheric effects, meaning SAR functions independently of weather effects and the time of day, making it a valuable tool in remote sensing applications. The ability for SARs to function regardless of clouds is especially useful for a rain forest such as the Amazon.

The use of SAR data for studying wetland areas such as the Amazon are well established. Costa et al. [5] have used variations in signal from dry and flooded vegitation to map the extent of flooding in the Amazon floodplain. Additionally numerous studies have shown that SAR images can be used to analyse and monitor variations in aquatic vegetation.

Costa et al. [6] Used SAR images of the lower Amazon to aid in the estimation of net primary productivity of aquatic foliage. The SAR images helped determine biomass and coverage area. Kasischke et al. [7] successfully used variations in SARs backscatter during periods of flooding and non-flooding for monitoring surface hydrologic. Hess et al. [8] used a decision-tree model on SAR data as a multi-classifier. They found L-HH polarisation the most useful for distinguishing between non-flooded and flooded forest, and cross polarised L-band for separating woody and non woody vegetation.

Figure 2 shows the different microwave scattering processes that occur in wetland forested areas such as the Amazon. These processes affect the final SARs images and can be exploited to make classifications of different ground covers. When the microwaves strike a surface such as the foliage of tree canopy, a certain amount of surface back scattering occurs, sending a signal back to the satellite establishing an image. The remaining photons will be diffused in other directions possibly reflecting several times. A small amount will eventually return to the satellite, providing more signal. This is called volume back scattering.

Due to the side looking nature of the radar, when the electromagnetic waves strike the surface of a flat smooth surface, such as open water, a surface reflection occurs and the photons do not return in the direction of the sensor and are lost (specular scattering). This makes classification of open water relatively easy as it results in dark image tones in SARs images.
Flooded vegetation produces double bounce back scattering, essentially the surface reflection will further illuminate any objects in its path resulting in brighter tones in SAR images. This is in contrast to the non-flooded vegetation with only volume scattering which will generally not be as light as double bounce scattering.

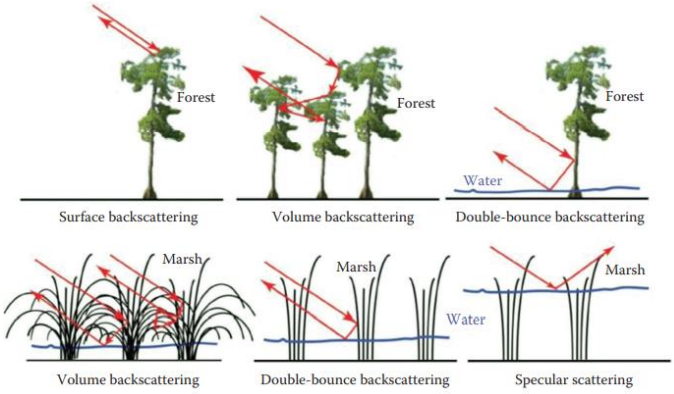


Figure 2 – Different scattering processes wetland and forest vegetation, taken from [9]

2.2.2 Wavelength and polarization

This has shown how a SAR image is affected by the presence of flood water and vegetation, but if we wish to make classifications between different types of vegetation we will need more than just variations in image tone. Up to now we have just referred to the outgoing radiation as microwave, however radar systems function in different frequency bands of the electromagnetic spectrum. These bands are labelled by letter, and generally L band is preferred for classification of woody wetland vegetation [9]. Novo et al. [10] found L band radar to be sensitive to stand height and above ground biomass, and Costa et al. [11] found L band shows double bounce for vertical tall woody wetland vegetation making it a good choice for distinguishing flooded forests from herbaceous vegetation such as grasses in the Amazon.

In addition to the frequency band, the outgoing pulse and returning signal both have polarizations (due to the nature of an antenna). These are specified as either vertical (V) or horizontal

(H). When un-polarized electromagnetic waves are reflected on a surface, vertical components are absorbed or refracted leaving only horizontal polarization to be reflected (this is why a surface reflection is not as vivid as a mirror reflection). The polarization of the radar system, be it same polarization (VV and HH) or cross-polarization (VH or HV) can highlight certain characteristics of certain targets. These characteristics can be exploited in the mapping of aquatic plants, with accuracies in the range of 65-97% being attainable [5], [10], [12] with plant variety being discernible to a degree due to structural and dielectric property differences between classes at different polarizations [13].

2.2.3 Optical

Combining optical and radar images provides useful multi-sensor data that has proven successful at mapping land use [16]. As the information content from each is diverse when fused together they can have a complimentary effect. Haack et al. [18] found increases in accuracies by as much as 10% by combining both data types. Different seasonal images can prove valuable in mapping forest areas such as the Amazon.

A useful type of optical image when dealing with vegetation is Near-infrared (NIR). NIR has a wavelength range of 0.7 to 1.0 µm, just beyond the colour red, making it invisible to the human eye. Colour-infrared (CIR) imagery shifts the primary colours so that the NIR region can be seen in the image and shows as reds. Vegetation's chlorophyll absorbs sunlight in the photosynthetically active region (PAR wavelength 0.4 to 0.7 µm) using it as a source of energy through photosynthesis, but photon wavelengths longer than around 700 nm have insufficient energy to synthesis organic molecules. This surplus energy is dissipated by the leaf cells by re-emitting it as NIR. Thus, vigorously growing dense vegetation producing a lot of chlorophyl appear dark in PAR and bright in NIR as can be seen in Figure 3. Lighter tones of red generally represent vegetation not containing much chlorophyl, such as mature standing trees or even dead or unhealthy vegetation. We can clearly see different shades of red for dense canopy and variations across different vegetation species while non vegetation such as buildings or road are noticeably different.

Figure 3 – Colour infrared satellite image (CIR) on the left, with the normal RGB image on the right, taken from https://blog.hxgncontent.com/impact-of-color-infrared-photography-and-color-infrared-images/

Remote sensing has exploited these prominent differences in plant reflectance to distinguish variations in vegetation, lending itself to further analysis of satellite images. Spectral rationing

is a form of image data transformation and are used in remote sensing as an enhancement technique. The image from one band is divided by another band in a pixelwise manner, yielding an 'index' image appropriate for a specific task.

The normalized difference vegetation index (NDVI see Figure 4) provides a simple graphical way of exploiting these effects and has been used extensively to simply and quickly identify the condition of vegetated areas, with the first such work to do so by Rouse et al. in 1973 [18]. It has been used with deep learning techniques on earth observation tasks before [45]. NDVI is most familiar and used indexes to find and analyse live plant crowns in multisensory remote sensing data. It is therefore an obvious choice to aid in the task of mapping wetland forest areas of the amazon.

$$\mathrm{NDVI} = \frac{(\mathrm{NIR} - \mathrm{Red})}{(\mathrm{NIR} + \mathrm{Red})}$$

Figure 4 – NDVI equation, taken from https://en.wikipedia.org/wiki/Normalized_difference_vegetation_index

Normalized Difference Water Index (NDWI see Figure 4) is another such index, suggested by Gao et al. [19] and enables the monitoring of water content in leaves. It does this using NIR and short-wave infrared (SWIR). Due to its strong relation to plant water content it is likely to provide a useful proxy for mapping wetland forest areas of the Amazon.

$$\mathrm{NDWI} = \frac{(Xnir - Xswir)}{(Xnir + Xswir)}$$

Figure 5 – NDWI equation, taken from https://en.wikipedia.org/wiki/Normalized_difference_water_index

2.2.4 Elevation data

Stereoscopic satellite imagery is where two images are observed of the same location at different angles by a passing satellite, generally about 45-90 seconds apart and is termed in-track stereo. By use of photogrammetric techniques, these two images can produce a 3D elevation model on an area, termed a stereo-derived elevation model.

Previous work [14] has yielded success in the classification of land cover and tree species identification using multispectral data with stereo imagery for canopy height. As liquid water obeys the rules of gravity, it is logical that areas of wetland will lie in regions of land that are lower than their surroundings. It therefore seems reasonable to assume that elevation data may assist in the mapping of wetland forests in the Amazon, and has been used in mapping marshland in other works [43].

2.2.5 Novel project dataset

We have discussed how the L band SAR can provide useful signals when spatially distinguishing wetland forest, and different combinations of polarizations highlight various characteristics.

Combining optical data with SAR can also assist in the classification of vegetation, and we have introduced the indexes NDVI and NDWI as useful optical channels especially when provided across seasons. Elevation data may also assist and is readily available. Liu et al. [43] found that higher spatial resolution can have a positive effect on classification accuracy of marshland vegetation with deep learning models, so using a high resolution can yield better results.

All three of these sources will be drawn upon in the creation of the dataset and the layers compiled into one stack ready to use for analysis.

2.3 Deep learning in Computer vision

Recently, deep learning has been gaining substatial attention in the subject of computer vision (CP). What has driven this interest is deep learning models outperforming previous state-of-the-art techniques at multiple CP tasks. This has been enabled by the appearance of large publicly available labelled datasets to work with in addition to hardware advances such as powerful GPUs accelerating training times considerably. This surge in interest has been further fuelled by powerful frameworks such as Pytorch, Tensorflow and Theano which make faster prototyping possible.

Before going into how deep learning can assist with mapping the Amazon, it is useful to have an idea of the different ways a computer can gain an 'understanding' of an image.

Image Classification

Image classification is a primary building block in many CP tasks. We provide the computer with an image and expect it to return a discrete label of what mainly is in that image (we assume only one class is present, not multiple). Figure 6 demonstrates this concept. With convolutional neural networks the model will return probabilities for each class, and the expectation is that the actual class has the highest probability. Image classification provides us with the 'what' of an image.

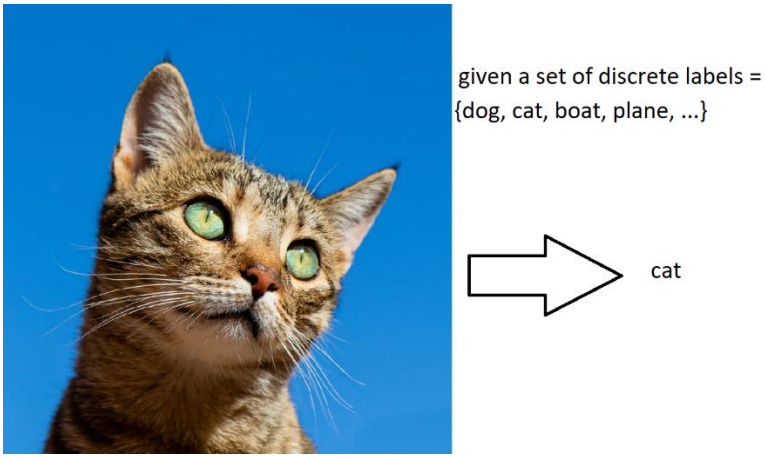


Figure 6 - Image classification, photo taken from https://www.twinkl.co.uk/teaching-wiki/cat

Classification with localisation

Figure 7 – Classification with localisation, photo taken from https://excitedcats.com/bombay-cat/

In addition to the classification, here we expect the localisation to be computed as well and is typically represented by a bounding box. This starts to combine the 'what' of image

classification with the 'where' of a box. Again the assumption is that there is only one object class in any given image.

Object Detection

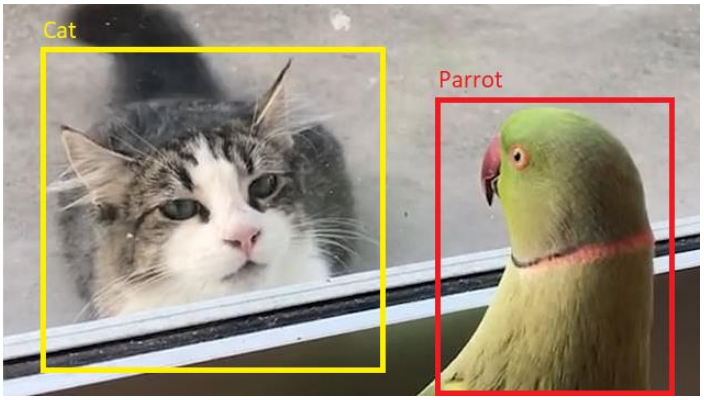


Figure 8 – Object detection, photo taken from https://www.dailymail.co.uk/news/article-6822447/Playful-parrot-loudly-squawks-Peek-boo-hides-neighbours-hungry-cat.html

Object detection (Figure 8) takes localisation to the next level by allowing the image to contain an unknown number of objects. For each known object, a bounding box is placed and a label assigned, providing 'what' and 'where' for each object.

Semantic Segmentation

With semantic segmentation there are no objects as such. Unlike the previously mentioned tasks, the output is not labels or bounding boxes, but an image that is normally the same resolution as the input. Each pixel in the image is classified to a particular class. This is why the output is commonly known as a *dense prediction*. This dense prediction provides both 'what' and 'where'.

Figure 9 – Semantic segmentation, image taken from [2]

Semantic segmentation with Deep learning

Long et al. [2] showed that convolutional network architectures that had initially been developed for image classification can be repurposed for dense prediction. For each pixel the class with the highest probability will be taken as the label.

These repurposed networks substantially outperformed the previous state-of-the-art techniques (such as simultaneous detection and segmentation [4]) on PASCAL VOC 2011 and 2012 test sets, proving the methodology.

2.3.1 Deep learning used in remote sensing

Previously, work has been performed detecting deforestation using CNNs on satellite images. Whilst this is not the aim of this project, the methodologies and findings from these works can be used to establish an approach to this projects task, and aims to show the position of this work relative to the existing literature.

De Bem et al. [22] performed binary classification for change detection of deforestation in the Amazon. They use CNN architectures, (ResUnet, SharpMask and U-Net). This was performed with Landsat-8 images with a spatial resolution of 30m and found these architectures to perform better in most metrics than two classic machine learning algorithms – random forest and multilayer perceptron. The CNN architectures produced similar results, with ResUnet obtaining good F1-Scores (0.943-0.947.) They point out a short coming of CNNs is the necessity of a 1 to 1 ground truth due to spatial context being critical. However they also suggest the addition of extra bands in remote sensing may facilitate the CNN learning process with less samples to learn from.

Similarly Mazza et al. [36] concluded that CNNs outperform random forest techniques and found their version of U-Net to perform best of several CNNs on their task of mapping forest/non-forest, yielding F1-Scores between 0.825-0.863.

Chantharaj et al. found their SegNet model [34] outperformed the state-of-the-art Gated Convolutional Neural Network [35] model on their Landsat-8 dataset.

Andrade et al. [23] also performed deforestation change detection on the Amazon forest using the state of the art DeepLabv3+ architecture and make comparisons against other deep learning methods (Early Fusion and Siamese Convolutional Network). The classified images were again Landsat-8 images with a spatial resolution of 30m. Their results demonstrated that all tested variations of the DeepLabv3+ architecture significantly outperformed the other deep learning methods in terms of F1-Score, and found the gains to be even more significant when the sample data was limited. In order to train their DeepLab-based change detection model, they used a batch size of 16 and Adam optimiser for 100 epochs. They made use of weighted focal loss due to the class imbalance the found in their data. They suggest further work applying their CNN on data from other sensors, and suggest SAR systems since cloud coverage is an issue in tropical regions.

Recently DeepLabV3 [42] has had success classifying marsh vegetation [43].

Flood et al. [24] use 3 band Earth-I imagery to map trees and large shrubs in Queensland, Australia. They use a U-Net architecture and find an accuracy of around 90% after 80 epochs, demonstrating the technique can classify beyond just trees. They use an 20 pixel overlap strategy when making inference on large satellite images, stating they 'chop up' the images with an overlap of 20 pixels. They then trim off 20 pixel from the edge of each prediction so that they can be combined to form a contiguous whole image.

Lee et al. [25] used SegNet and U-Net to classify images from Kompsat-3 into forest and non-forest. There results found the accuracies to be 74.8% for U-Net and 63.3% for Segnet.

Finally, Bragagnolo et al. [26] map alterations in cover of the forest in the Amazon using a number of CNN architectures, (Segnet, U-Net, FCN32, DeepLabV3+, PspNet, ResNet50-Segnet, and Vgg-PspNet,) finding U-net to yield the best F1-Score of 0.947, and Segnet a F1-Score of 0.861. They state their results indicate the capability of U-Nets to generate high fidelity result making them applicable to tracking of forest cover.

Interestingly, they found their U-Net model was capable of identifying patches of forest in satellite imagery that had not previously been identified in the hand labelled masks. They state this shows that the trained network can function well despite misclassifications in the training data. They suggest an iterative process could be adopted, where by the dataset could be re hand labelled with these corrections and retrain the network potentially to a higher accuracy.

They found considerable misclassification at the edge between forest and non-forest areas, stating such pixels to be tougher to classify than those on the interiors. The reason provided is because they are located at a point of changing forest density gradient, and the exact boundary is 'fuzzy' even to specialised visual inspection.

They also state the binary classification approach of lumping all objects into either forest or non-forest maybe a limitation, and say future studies are needed on their problem of mapping forest areas of the Amazon into more object classes.

Other available CNN architectures are being used for segmentation of satellite images. DenseNet [38] has been used in detecting and segmenting palm oil plantations [39] and LinkNet [40] has been compared to other methods [41] finding accuracy similar to U-Net, though not competitive in Dice similarity.

3 Novel Dataset

This section deals with how the data is taken in and prepared for use in training CNN. Firstly, the sources of the data streams are stated, with the layers of each listed. Secondly, as the data is supplied in an earth observation sector file format, it needs extracting out of each of the files and combining into a stack.

3.1 Data sources

The data used in this project, as discussed in 2.2, consists of four streams:

1. Ground truth
   The ground truth (mask) has come from two different sources. The Mamiraua is taken from [55]. The masks of the Curuai and Parintins have been upsampled (original was 100m per pixel) from the output of a pre-existing model by Hess et al. documented in [12] and [54].
2. Optical data, LANDSAT 5
   Optical imagery from the Landsat 5 Thematic Mapper sensor, consisting of six spectral bands in the visible, near and shortwave infrared electromagnetic spectrum. The red, NIR and SWIR1 bands were used to compute the NDVI and NDWI images using Google Earth Engine and taken for the dry season and the wet season. Statistical approaches have been used on a time series of images to limit atmospheric effects.
3. Elevation data, AW3D
   AW3D provide elevation data for the entire globe. This single layer was retrieved from google earth engine.
4. L-band synthetic aperture radar (SAR) data, PALSAR 1
   This data has been taken from the Alaska Satellite facility and 17 layers have been created from a time series of images with various statistic performed by pixel (the title of the file relates to the statistics that have been used). Doing this reduces noise and increases the amount of information captured by each layer.

As the available is quite limited, a number of additional bands have been used as suggested by De Bem et al. [22]. The majority of these bands are SARs images, as suggested by Andrade et al. [23]. As per Hess et al. [8] suggestion, L-band polarisations have been included.

These streams and the layers within them are shown in Table 1. The temporal coverage of the images is 2006-2011, as both sensors seized operation shortly after. More recent landsat images from Landsat 8 are available, however there is no overlap with available L-band SAR observations.

	File name
Ground truth	Class_mask_tile
1 Optical	dry_season_NDVI_tile
	dry_season_NDWI_tile
	wet_season_NDVI_tile
	wet_season_NDWI_tile
2 Elevation	DEM_median_tile
3 L-band SAR	AP1_tile_hhhv
	AP1_tile_hh_cv
	AP1_tile_hh_max

	AP1_tile_hh_mean
	AP1_tile_hh_median
	AP1_tile_hh_min
	AP1_tile_hh_range
	AP1_tile_hh_skewness
	AP1_tile_hh_std
	AP1_tile_hv_cv
	AP1_tile_hv_max
	AP1_tile_hv_mean
	AP1_tile_hv_median
	AP1_tile_hv_min
	AP1_tile_hv_range
	AP1_tile_hv_skewness
	AP1_tile_hv_std

Table 1 – The files that make up the layers of each stream to create the complete dataset stack

3.2 Reading data from earth observation files

Four patches of size 4000x4000 pixels have been used in this project; one from Mamiraua, and two from Curuai and one from Parintins. Ground truth masks are shown in Figure 10 and Figure 11 respectively. The files are in a geo TIFF format meaning they have georeferencing metadata combined with a tif file. When combined with the correct projection these files can be used to generate longitude and latitude for each pixel and thus locate objects. Gdal was used to extract the raster data, a 4000 x 4000 array or image, from each file as this is all that is required for the purposes of this project. The spatial resolution of the extracted images is 12.5m per pixel, making them fine resolution [44], which can yield better results (see 2.2.5). Each of these images were concatenated along a new axis, creating an array 23 channels deep.

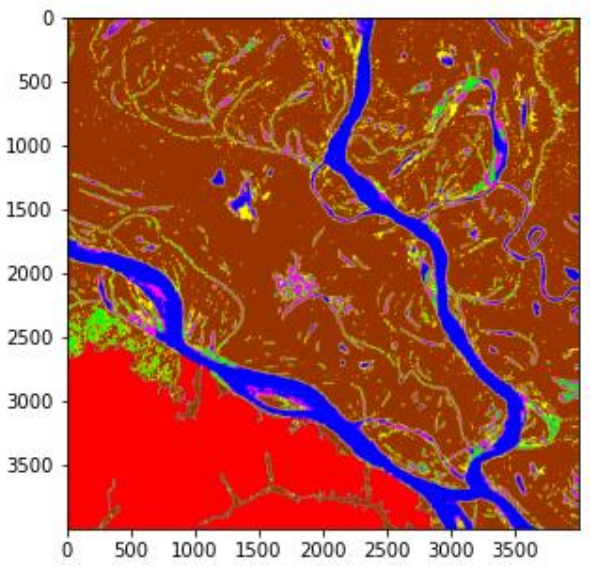


Figure 10 - Ground truth for Mamiraua patch [55]

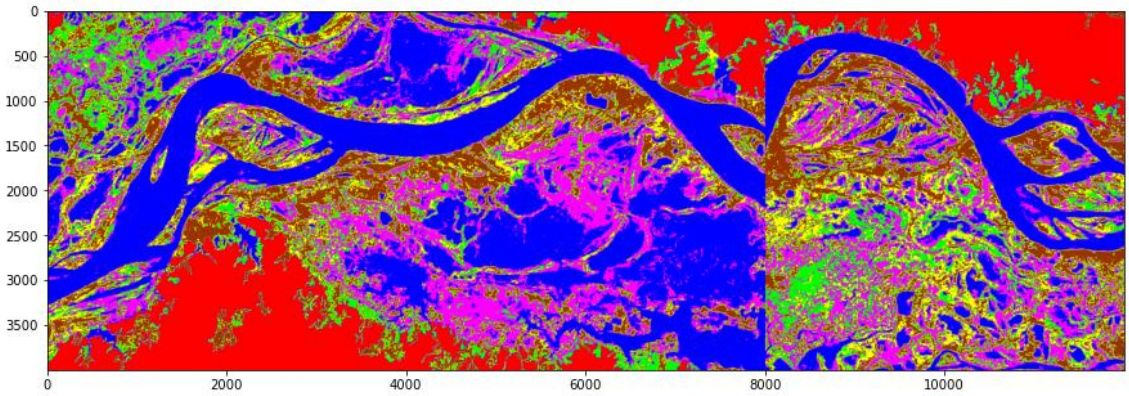


Figure 11 - Ground truth for Hess et al. [54] patches. Thereafter, from left to right, Curuai 1, Curuai 2 and Parintins.

4 Methodology

In the first instance, a working environment is defined. Then the data is put in a format the environment understands, ready for training.

The process of proposing a model commences. There is no single best model or training algorithm that works optimally for any given deep learning problem. This is sometimes referred to as the *no free lunch* theorem [52]; a set of assumptions that yield good results in one domain may perform disappointingly in another. For that reason, it is prudent to select appropriate models for the task and make comparisons between them.

Section 2 & 3 described the computer vision task at hand and the sources of the data; this defined the domain. Deep learning architectures for this task were presented and examples of them performing well in similar domains were provided. In the methodology outlined here, different models and associated hyperparameter searches will be validated allowing the best method (from those trialed) for the problem to be chosen empirically. Any further refinements can then be performed on the chosen model. Finally, the model can be tested on the unseen test data and the results presented for discussion. A further stage is performed, demonstrating how the model can be used for inference on whole satellite patches.

The project stages are provided here with the relevant chapter for reference:

4.1 Decide on environment

4.2 Prepare the data

4.3 Discuss deep learning methods

4.4 Build deep learning models

5 Train the models (experimental setup)

6 Validate and analyze the models results

6.2 Choose one model

6.3 Perform any relevant ablation studies

6.4 Test final model

7 Demonstration - Inference on large patches

4.1 Environment

Due to the nature of CNNs, a significant amount of trivial calculations are performed which would not be practical on even a multi core central processing unit (CPU) as they are designed to perform general computations. In contrast, a graphical processing unit (GPU) is good at handling computations that can be performed in parallel. A GPU can potentially have thousands of cores which it can perform computations simultaneously on. Neural networks are embarrassingly parallel, needing little to no effort to separate the task into a number of smaller parallel tasks that are independent of each other. It therefore makes sense for the work environment to have access to a GPU to make the project feasible.

4.1.1 Nvidia hardware and CUDA

Nvidia design and produce GPUs and have provided a software platform called CUDA that pairs with their GPU hardware giving developers an application programming interface (API) to the hardware. This enables the developers to build software that is accelerated by the processing power of Nvidia GPUs.

4.1.2 PyTorch and Hardware

The PyTorch framework provides an open source machine learning library that can be used for deep learning in computer vision tasks. A benefit of PyTorch is that it has CUDA 'baked in'. This enables the building and training of CNNs that can easily been trained on the GPU without the need for the user to know the CUDA API; the user simply needs to know how to code in python.

PyTorch can be thought of as NumPy with GPU support, with tensors behaving almost exactly the same as NumPy arrays but having the advantage of being moveable between the GPU and CPU ram with a simple command. Familiarity with both Python and NumPy made PyTorch an obvious choice of environment for this project.

The hardware used for this project is a server running jupyterhub. This gives access to four NVIDIA GeForce GTX 1080 T GPUs each with 12 Gb of ram and will speed up the training process considerable.

This dissertation assumes the reader is familiar with python in both the procedural and object oriented paradigms to a reasonable standard.

4.2 Data preparation - Custom Dataset creation

Pre-processing tasks such as dealing with missing data and creating smaller tiles for the training set are described here. Once the data is ready, it is loaded into a custom PyTorch dataset class, ready for use.

4.2.1 Handling missing data

Whilst the optical channels are collected from a time series of images which should reduce atmospheric effects, with a location such as the Amazon some pixels will always have seen cloud cover with every pass of the satellite overhead. Where the data is missing, the pixel value is assigned 'nan' (not a number). Due to the computational nature of deep learning, further calculations can't be performed with this value and will generate an error. Opencv INPAINT_NS (Navier-Stokes based method) is used to infill these 'nan' regions. Whilst this does give the model usable values, those values are still not correct for the associated pixels. For this reason these pixels were assigned the label -1 in the ground truth (see Figure 12) and are ignored in the training process. However, due to the nature of convolution they will influence the feature maps as they pass through the networks. As this is a known error in the dataset it will be a systematic for all models and is deemed acceptable for the project.

Figure 12 – nan regions set to -1 (black) in the ground truth mask

4.2.2 Tiling of large image

Due to the size of the patches, not only in terms of image height and width but depth through the layers, it is not reasonable to pass this amount of data through a CNN at once, given the current size of GPU ram. It would also yield a training set size of three, which is not appropriate for training purposes. What we can do to solve this is split the patches down into smaller tiles for use with the CNN.
A tile size of 256x256 has been used which is reasonably typical [45], [46]. As it is a power of 2, it can be downsampled multiple times and maintain a whole number of pixels. This will mean a tile will represent 3.2 by 3.2 km area, which should be large enough to ensure global information is not completely lost. As 256 does not fit perfectly into 4000, the final tile has been shifted such that it overlaps the previous tile sufficiently enough to keep a consistent size of 256x256. In doing so the whole tile is made use of without any waste. There is however, a small repetition of data within the dataset. This is deemed acceptable [45], [46] as again this will be systematic across the models. Mean and standard deviations are taken for each channel so the dataset can be standardised later.

4.2.3 *amazonMultiBandDataset* class

The smaller tiles are 23 channels deep and are NumPy array objects. In order to use them for training our CNN using the PyTorch framework, they will need to be transformed into a custom PyTorch dataset object. To do this a new class is defined called *amazonMultiBandDataset* which inherits from PyTorch's Dataset class. The tiles are provided to this class along with the channel means and standard deviations as arguments in the initialisation method and stored in two separate lists. The ground truth or mask is kept as Y, and the remaining channels of the tiles as X. When the instance of this class has its 'get item' method called on a certain index, it returns the corresponding tile from X that has been transformed to a tensor and standardized (normalized to a standard normal distribution [47]) using the mean and standard deviation for each channel. The ground truth from Y at that index is also returned as a tensor.

4.2.4 Dataset Organization

Data availability necessitated the creation of a dataset containing three of the four patches mentioned in 3.1. The Mamiraua patch, Figure 10, was combined with Curuai 1 and 2 to create a dataset. A barchart of the ground truth masks (see Figure 13) shows the distribution of pixels across the classes. Class imbalance is present which is typical for this type of task and will have to be addressed during training.

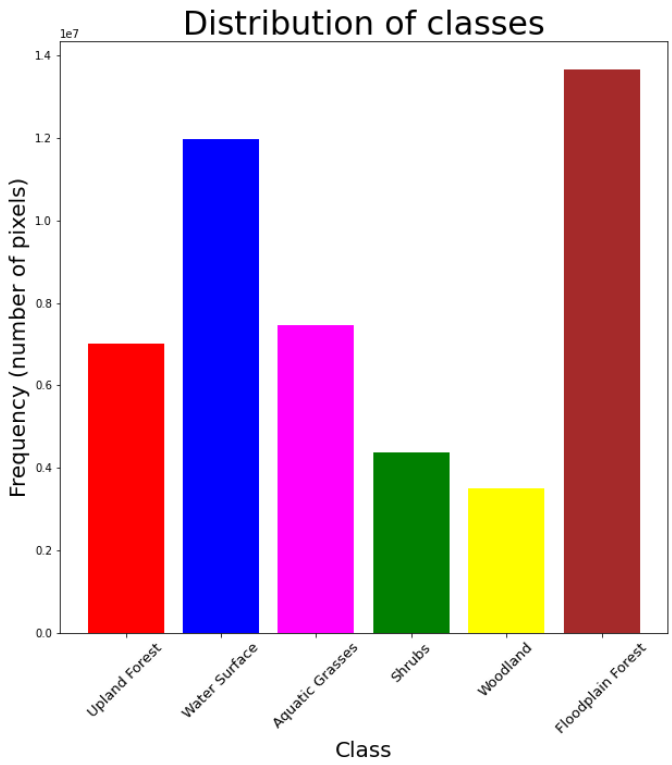


Figure 13 - Bar chart showing distribution of pixels across the classes in the dataset. Class imbalance is present

In order to perform training, validation and testing, the dataset is randomly split into three subsets on a 80/10/10 split. As the models will be trained at different times, a manual seed was set to ensure consistency. The size of the subsets are shown in Table 2.

Subset	Length
Training	614
Validation	77
Testing	77
Total	768

Table 2 – length of each subset

Figure 14 shows the distribution in each subset. Whilst the best approach with machine learning is to train on a balanced dataset, in many cases this is not possible or practical. Often the minority classes are of the most interest and methods are required to improve their recognition rates [48]. The training subset has is unbalanced with the largest class being 3.8 times the size of the smallest class. As stated before, class imbalance is quite common in semantic segmentation tasks and focal loss (see 4.5) can be used to mitigate the issue.

It is recommended to validate and test with data that has a similar mix as to what will be seen in production. Whilst the validation and test distributions are not identical to that of the whole dataset, they are similar. The distribution of the classes across the entire Amazon region is not known (within this project) and so these distributions are deemed acceptable for the purpose of making performance comparisons between different architectures.

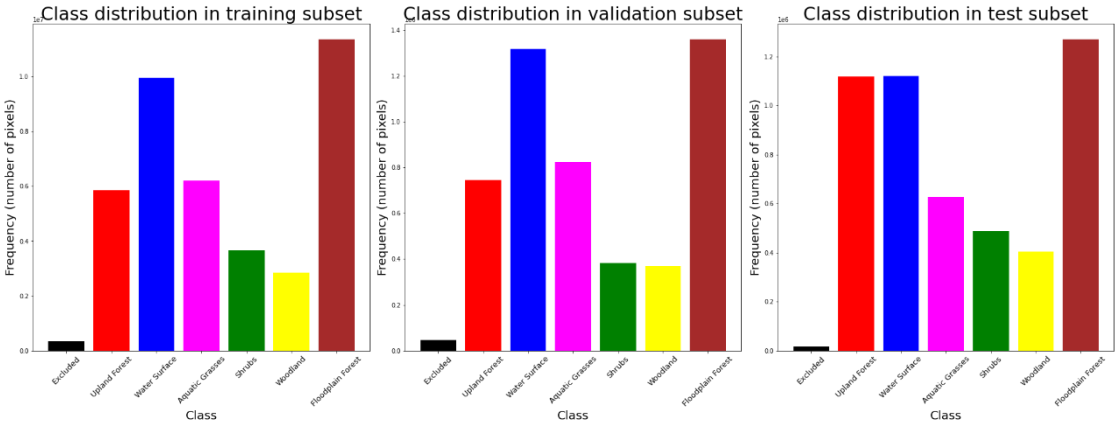


Figure 14- Class distribution across subsets

4.2.5 DataLoader PyTorch class

The dataset object has been created and loaded with the data, and then split into the various subsets. As mentioned in 4.1.2, the models are run on a GPU. In order for this to work, everything needed for the calculations must be kept in the GPU ram. This means both the network and data must exist simultaneously on the GPU ram. It is therefore very easy to run out of GPU ram, and one of the roles of the dataloader object is to prevent this. The Dataloader object accesses the data from the dataset in batches, thereby only holding a smaller size of data on the GPU at any one time. If the batch size is set too large, the GPU can run out of ram and an exception is raised. Yann LeCun gave his perspective on this stating "Friends don't let friends use minibatches larger than 32".

4.3 Discussion on deep learning methods

Computer vision is a fast-growing field with significant amounts of research applying convolutional neural networks (CNNs) to computer vision tasks being published in recent years. As stated in 2.3.1, CNNs have been shown to perform better than other classic machine learning approaches [22], [36] as well as other deep learning methods [34], [23] when working with satellite imagery. As this has already been established, this project will involve only CNNs.

This section outlines methods using CNNs to perform semantic segmentation of images and assumes the reader has a good working knowledge of neural networks and related terminology at least to the level of building and training an simple image classifier.

Semantic segmentation - Ground truth

Training any deep learning model requires 'correct' examples for the model to learn from. These labels are called *ground truth* and their format varies depending on the task. When an image classification is performed, we expect a simple label, such as a string as a ground truth. It is relatively trivial to build a dataset for image classification as all that is required is for a human to observe the image and assign a label. With semantic segmentation a dense prediction (or mask) is required. The labelling task of the ground truth for the dataset is therefore different. The 'labeller' is required to essentially colour code or segment the image into the different classes, and the task of labelling becomes considerably harder and more involved.

With this in mind, it is not surprising that datasets for semantic segmentation tend to be scarce and small. This is why we look at the idea of repurposing an image classifier as it enables us to perform transfer learning. The first important topic to discuss when looking beyond straight forward image classifiers is the notion of transferring learned parameters from one task to another, as the necessity to do this shapes the architectures of semantic segmentation.

4.3.1 Transfer learning

Training deep learning models from scratch is time consuming and requires large datasets. One machine learning approach used to side-step these problems is transfer learning; a pre-trained model is used as the starting point for training on a second task. This has the effect of speeding up training times and can improve the performance of the model also.

Donahue et al. [28] evaluate if features extracted from a trained convolutional neural network can be repurposed to a novel generic task, even when the new task differs significantly from the original. They find the first layers learn 'low level' features, and later layers learn 'high level' semantic features, and show that transfer learning is extremely effective when the datasets are similar. This can mean you can move weights from trained large dataset, (they use ImageNet), to small dataset and achieve good accuracies without fine tuning (the transferred layer's weights are frozen).

The early layers of deep learning networks tend to train similar filters, such as Gabor filters (see [28] Figure 15,) and are reasonably general. As we pass deeper through the network the learned filters become specific to the task.

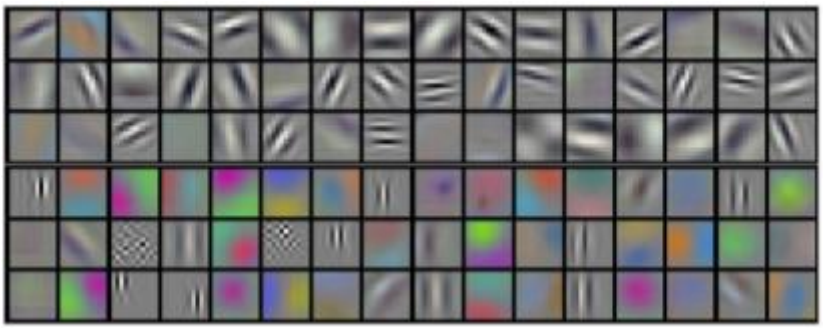


Figure 15 – Kernels of the first convolutional layer of a CNN, taken from [28]

Yosinka et al. [27] aimed to demonstrate where this transition occurs, and to what degree it affects the final accuracy in their graph Figure 16.

The blue points represent a so called 'selfer', or a model that has been trained on a subset of task B, then trained again on another subset of task B (BnB). This represents training on two very similar tasks and shows the pre-trained model works well as a feature extractor (frozen weights in the pretrained layers). There is a drop at around layer 4 which demonstrates fragile co-adaption between these layers, however they show fine tuning (blue with +, BnB+ meaning transferred layers not frozen) recovers these interactions.

The red points demonstrate transfer learning between two different tasks, AnB, with the transferred layers being locked. What we can see here is that the earlier layers generalise well, however later layers (when frozen) are too specific to task A and do not transfer well to task B. Again, when the network is fine-tuned (AnB+) the accuracy recovers. In fact it actually recovers past the BnB+ situation, indicating that there is extra accuracy to be gained by allowing the network to see more diverse data, and that some effects of task B training have lingered in the network. We can see this more generally in the graph below, as '5: Transfer + fine-tuning improves generalization'.

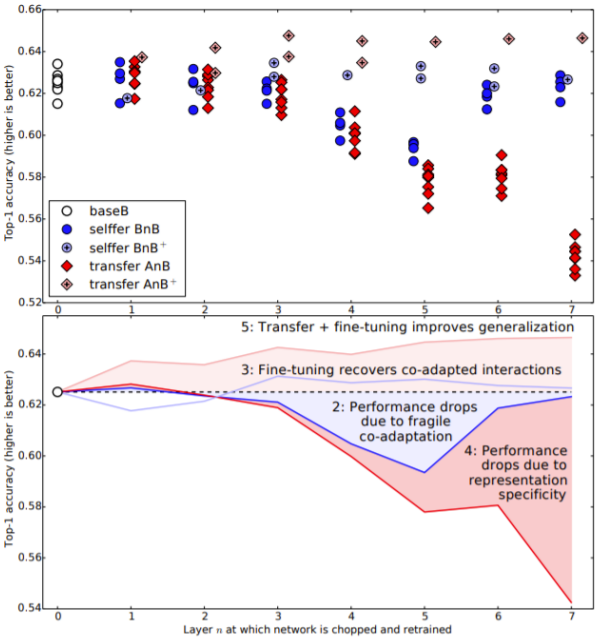


Figure 16 – How transfer learning varies by layers, and depending on task similarity, taken from [27]

They also demonstrate the effectiveness of transferability is related to how similar the tasks are, but that even when tasks are significantly different transfer learning yields better results than random initialization of weights.

The models used in this dissertation will therefore use transfer learning where possible. In order to do this, we need a way of repurposing image classifiers to produce dense prediction.

4.3.2 A change in architecture – fully convolutional networks FCN

As stated before, an image classifier uses global information of the image to resolve 'what' in an image. Local information is required to resolve 'where'.

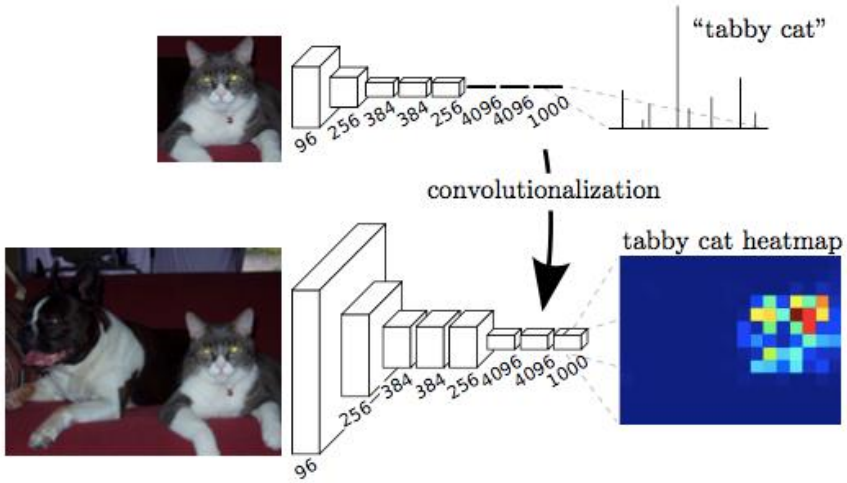


Figure 17 – Convolutionalization of a classifier, image taken from [2]

Figure 17 (top) shows how an image classifier will take an image, crop it to only include the cat and pass it through a series of convolutional layers with max pooling reducing the size. It would then meet fully connected layers the last of which will provide a score for each class, with the dominant class score providing the prediction ('tabby cat' in this case).
In 2015 Long et al. [2] published a paper that proposed the fully connected layers can be reinterpreted as convolutions whereby the kernel size is the same as the entire input tensor,

resulting in a 1x1 tensor. Further convolutions with kernel size of 1 can be used with the final one having the same number of kernels as classes. This then provides the same classification score as before, with the field of view of just the cat image.

Convolutions do not depend on the resolution of the input image, as the weights (learned parameters) are simply held within the kernel which is a fixed size tensor. As the network now consists only of convolutions, the input image size is no longer limited by the size of the fully connected layers. This means the entire image can be passed to the FCN, we don't have to focus solely on the cat for example. Once we pass a larger image through, we will no longer end up with a 1x1 sized tensors, but a feature map, as can be seen in Figure 17 (bottom).

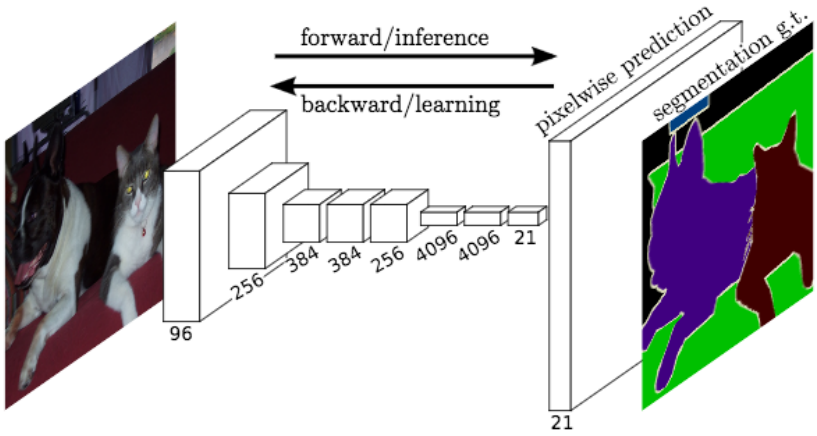


Figure 18 – Changing the architecture of a classifier for dense prediction tasks, image taken from [2]

Figure 18 shows how the last layers depth can be changed to match the number of classes in the dataset the network will be used with. This provides the 'what' aspect. We then require the image to be up sampled to the original resolution, providing the 'where'. This can be done with transposed convolutions.

4.3.3 Fractionally strided convolutions

It is assumed the reader is familiar with convolutional layers, which are used in the contracting path of a CNN. The opposite of this (not in the strict mathematical sense,) are fractionally strided convolutions (transposed convolution, up convolution). These are used to up sample a feature map to a larger feature map. As it is a convolution, there are trainable parameters. Figure 19 shows how a 2x2 input padded with a 2x2 border can have a 3x3 kernel convolved a stride of 1 to increase the size to 4x4.

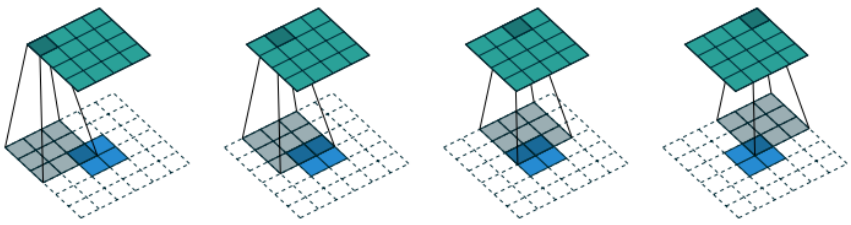


Figure 19 - Illustration of a fractionally strided convolution, taken from [29]

4.3.4 Different back bones

As stated before, the FCN architecture takes a CNN image classifier (back bone) and converts it into a dense predictor using transfer learning. Table 3 is taken from [2] and shows how three FCN with different back bones compare. We can see FCN-VGG16 performed best in mean IU, however it in terms of forward time it is not as efficient as the other two. It also has a considerable number of parameters.

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

Table 3 – Evaluation metrics of FCN32 with different 'back bone' architectures, taken from [2]

As the task in this dissertation is not time or memory critical, VGG16 will be used as a back bone where appropriate.

4.3.5 FCN8s - Getting the 'where' right

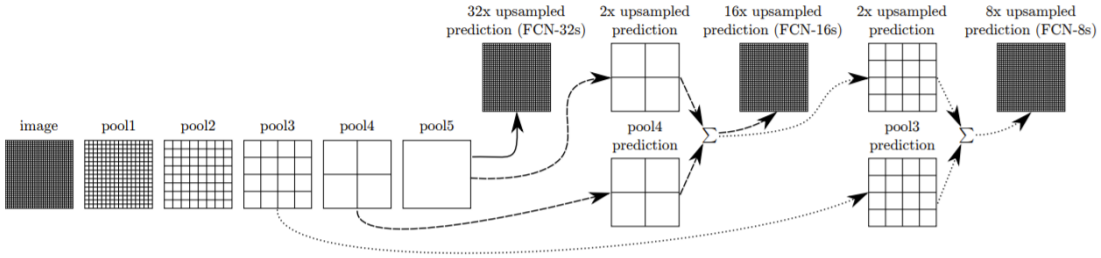


Figure 20 – How the skip layers of FCN architecture work, taken from [2]

Figure 20 shows how the pooling layers reduce the resolution of the image as it passes through the network. The results shown in Table 3 are when the size of the image remains at pool5 size. The prediction is up sampled by a factor of 32, hence the name FCN-32s. If the prediction is up-sampled twice, it can then undergo elementwise addition (fusion) with pool4. This can then be up sampled again to provide FCN-16s. If we do a similar fusion processes with pool3, we can then achieve **FCN-8s**.
The idea of this is that there is more local information in pool3 compared to pool4, and again to pool5. By passing or 'skipping' these layers through from earlier in the network and fusing them during the up-sampling, local information is gained and provides more 'where' to the final prediction. Figure 21 shows how the prediction improves with each 'skip' model, with FCN-8s visually looking the closest to the ground truth.

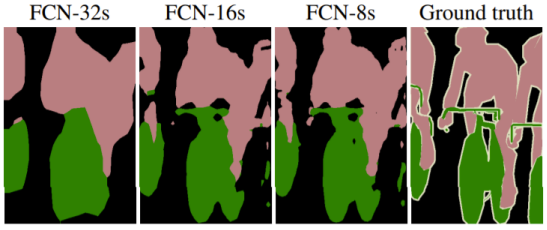


Figure 21 – The effect of adding the fusions or skip connections, taken from [2]

Table 4 confirms this with the evaluation metrics. As FCN8s performed the best, it shall be a model in this dissertation.

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**

Table 4 – Comparison of 'skip FCNs', taken from [2]

4.3.6 U-Net

As mentioned in 2.3.1, U-Nets have performed well previously at mapping forested areas [36], [24], [25], [26]. U-net is a FCN with an encoder decoder type architecture, where the image shape is down sampled by the decoder and then up sampled by the encoder. In Ronneberger et al. [30] own words, Unet is a more elegant architecture than FCN. The inclusion of skip layers again helps provide 'where' to the prediction. The expansion or decoder also has a large number of feature channels enabling the network to convey local details to later resolution layers. This results in the decoder looking almost symmetrical to the encoder and is what earned it the name U-net, as it is a U-shaped network (see Figure 22).

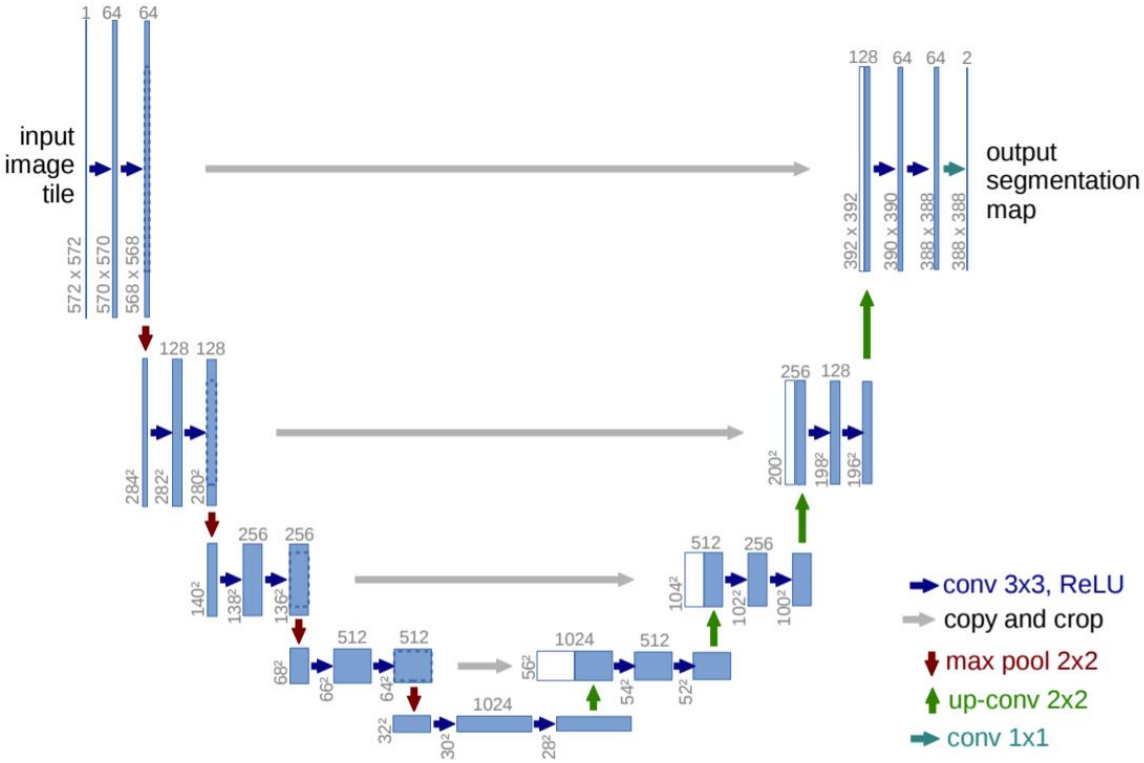


Figure 22 – U-net architecture, taken from [4]

It is apparent from Figure 22 that no padding is applied to the convolutions (they are valid), resulting in the loss of 1-pixel loss at the border at each convolution. For this reason, the skip layers require cropping to match the dimensions in the corresponding decoder stage. This yields a smaller output segmentation map than the input image size, which removes artifacts from the border region due to the field of view with the idea being a larger image can be stitched back together.

4.3.7 Modified U-net

Modifications can be made to the U-net architecture whilst still retaining the essence of it. The use of padded convolutions that do not reduce the image dimensions not only negate the need for cropping of the skip layers, but also allow the output image resolution to be the same as

the input. This enables the use of a standard data loader and avoids the need for mirror padding. This has been used considerably as a modification to u-net, with high-ranking versions seen in Kaggle competitions such as the Carvana image masking challenge so it seems unlikely it would have a detrimental effect on accuracy.
Batch normalisation [31] can also be included and Relu activation functions can be replaced by exponential linear units (elu), both of which Iglovikov et al. [32] found to be more robust to noise during learning on satellite images with their version, Multispectral U-Net.
Chhor et al. [37] also used batch normalisation and 'same' padding and found their U-Net model to be very efficient at dense prediction.
The modifications stated here shall be adopted for the U-net model in this dissertation.

4.3.8 Segnet

The two models we have already discussed have relied on up-sampling with convolutions or simple bilinear interpolation. Badrinarayanan et al. [33] proposed a different method of increasing the resolution. They kept the idea of a fully convolutional network with an encoder-decoder type architecture, but rather than skip connections passing entire feature maps from earlier on in the network, they pass the pooling indices.

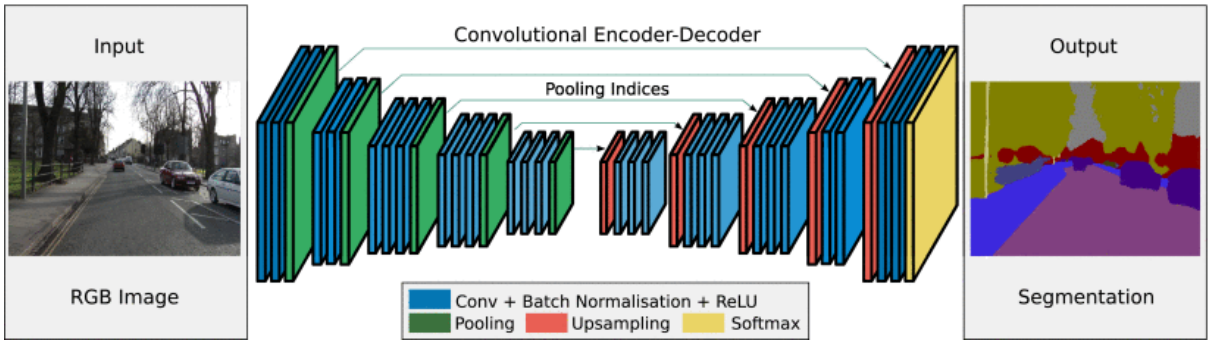


Figure 23 – The SegNet architecture, taken from [33]

Figure 23 illustrates the SegNet architecture. As mentioned in 2.3.1, Segnets have performed well previously when applied to satellite image data [34],[25],[26].

The encoder is still essentially VGG16. When a max pooling operation happens during the down sampling, the indices of the maximum values are retained and passed further on in the network to the corresponding un-pooling layer. Figure 24 shows how these indices are then used to create a sparse feature map. Subsequent convolutions then densify the feature map.

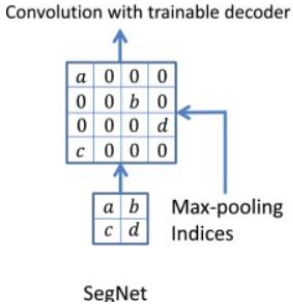


Figure 24 – How the pooling indices are used to up sample, taken from [33]

4.4 Build models

This section provides details of the architectures as built for the project. With the exception of the pre trained VGG16 model, all other models have been written from scratch and as such will differ slightly from the models in the original paper.

An update that is required on all models is the number of channels in and number of channels out. These are set to 22 and 6 respectively. This does however present a problem with the transfer learning method, as the VGG16 model was trained on image net and as such has weights for 3 in channels only. The solution used for this was to simply reuse the weights multiple times in the first convolution and allow fine tuning to assist thereafter. This method is motivated by the idea that filters obtained by the pretraining are generic to all vision tasks, and can therefore be redeployed to any channel. However, this may not be the case, but for the purposes of comparing the model performance it was deemed acceptable as any additional error caused will be systematic across all three models.

4.4.1 VGG16 – the back bones

The architecture of VGG16 is shown in Figure 25. This architecture is relevant as its weights are transferred to the other architectures and have influenced the design of them to accommodate. The pre trained model is taken from the torchvision.models package and is trained on ImageNet.

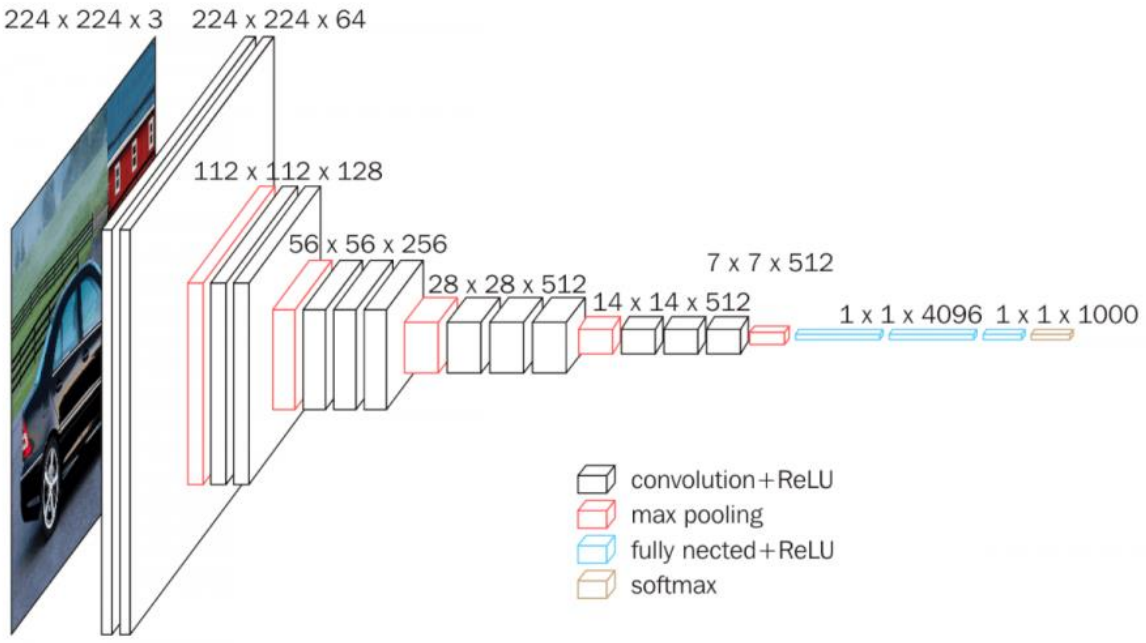


Figure 25 – VGG16 architecture, taken from https://neurohive.io/en/popular-networks/vgg16/

4.4.2 FCN8

As discussed in 4.3.5, an FCN8 model will be used. The architecture is shown in Figure 26. In 4.3.4 it was explained how the original paper [2] found the VGG16 back bones to obtain the best mean IU, and so it was necessary for the architecture to be derived from VGG16. When compared to Figure 25, we can see the encoder is the same as VGG16.

In torchvision it is normal to simply output the logits, and so the softmax layer is not implemented in the model (it is often performed in the loss calculation).

The original implementation of FCN8s added 100 pixels as padding to the incoming image and was cropped off later in the network. This enabled smaller images to pass through the network. The dataset used in this project has a fixed image size, and so it was not deemed necessary to include it this padding, as it only complicated the model and would have required more memory.

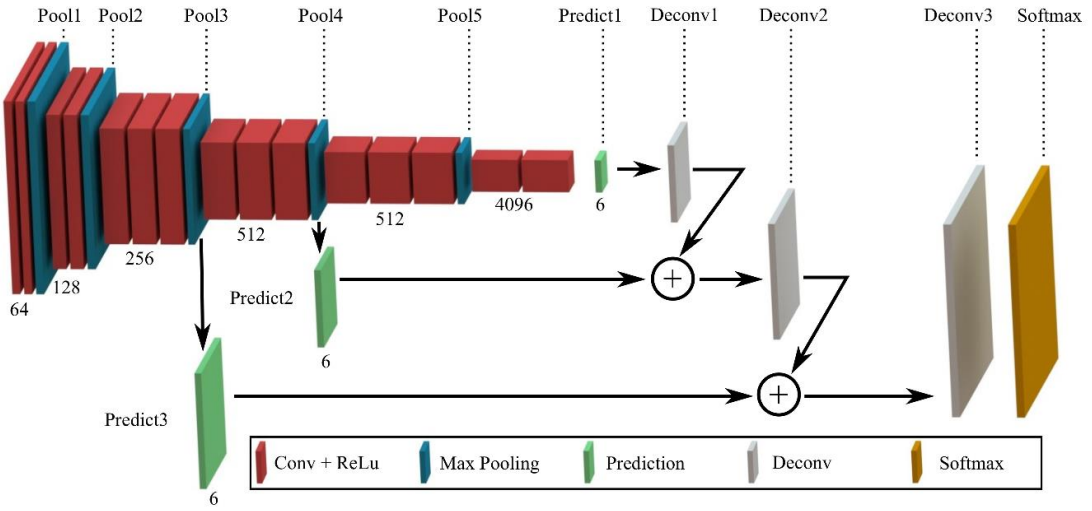


Figure 26 – FCN8s architecture as used in this project, image taken from https://awesomeopensource.com/project/alokwhitewolf/Guided-Attention-Inference-Network

4.4.3 VGG16 U-Net

Section 4.3.7 proposed some modifications of the U-net architecture firstly to bring it up to date but secondly to accommodate the pre-trained weights of the VGG16 back bone model. Previous work using these modifications successfully was also presented, along with the logic as to why they are used. The changes and the resulting model are presented here.

Firstly, in order to produce an output image the same size as the incoming image, padding was added (padding of 1 for 3x3 kernel resulting in padding 'same'.) This means the only size change is the max pooling and up convolutions (transposed convolution) hence, the skip layers no longer require cropping.

Batch norm has been used after each 3x3 convolution and are followed by elu, rather than Relu.

In order to accommodate the pretrained weights of the VGG16 model, it was necessary to modify the encoder to have three triple convolution blocks. This was deemed better than cherry picking weights to suit and loosing layers as this would break the fragile co-adaption between the layers (see section 4.3.1). The lowest point (bottle neck) of the encoder is also not as deep as the original U-net, having a depth of 512, again to match the weights from the VGG16 model.

U-net is traditionally symmetrical, however the decoder in this model does not have any triple convolution blocks, making it unsymmetrical. This is firstly because of the obvious point that there are no weights to transfer from the VGG16 model, and secondly, as the original model did not utilize them it is unlikely they will add much value.

The updated architecture schematic is shown in Figure 27.

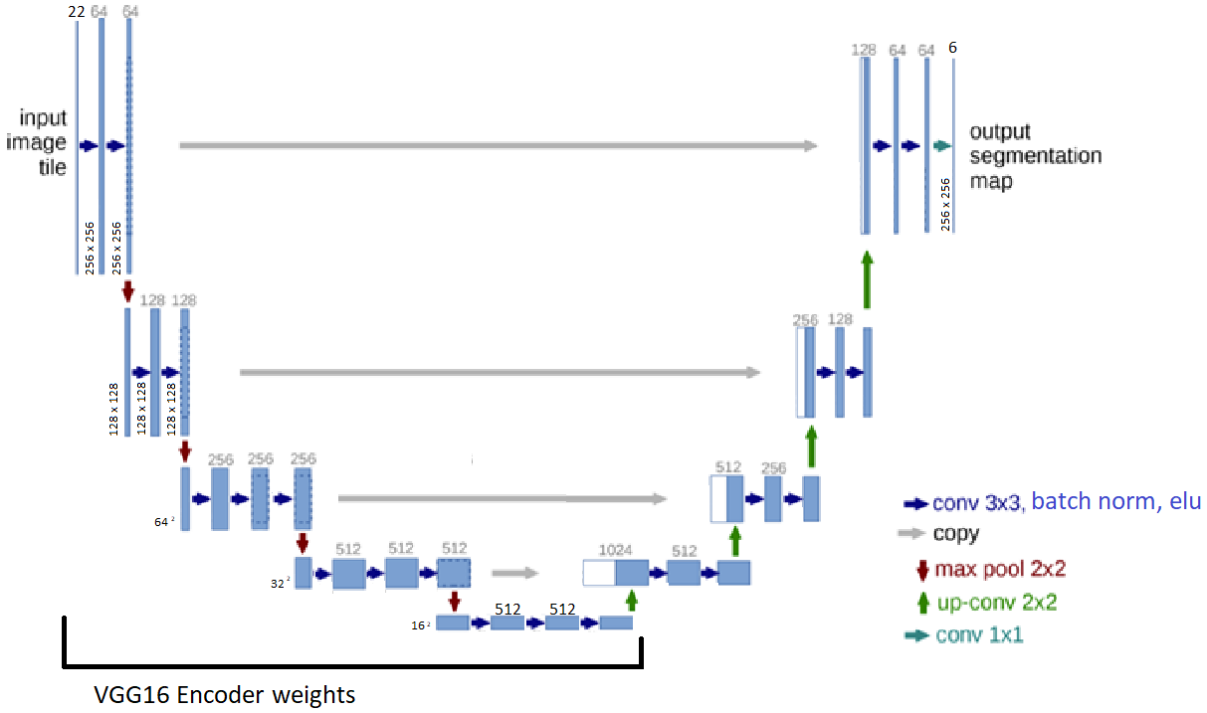


Figure 27 – VGG16U-net architecture, image modified from [4]

4.4.4 Segnet

As discussed in 4.3.8, a SegNet model shall be used. As can be seen again when compared to Figure 25, the encoder is taken from VGG16, allowing the weights to be transferred.

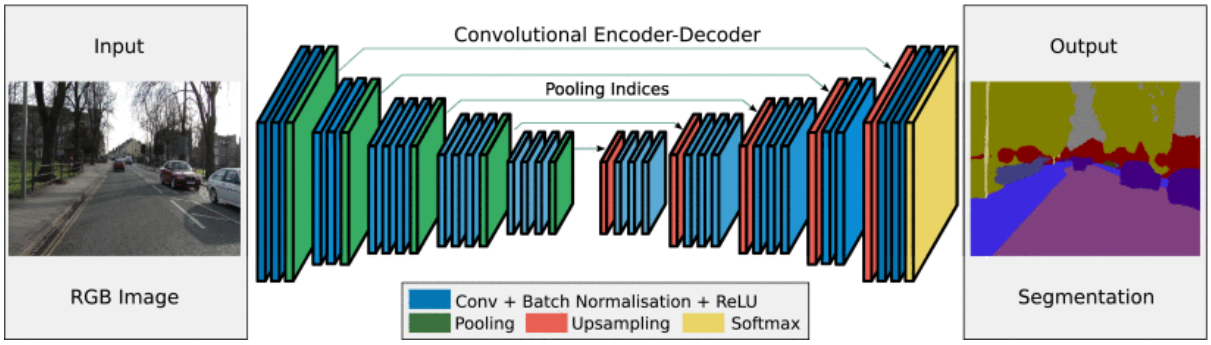


Figure 28 – The SegNet architecture, taken from [33]

4.5 Training Methods

4.5.1 Focal loss

When a model is trained on an imbalanced dataset, the model will be biased towards the majority class. One idea to take care of class imbalance in a dataset is focal loss [49]. Focal loss is an extension of balanced cross entropy adjusting the loss function with a modulating gamma term, as can be seen in Figure 29. This effectively down weights easy examples, focusing the training onto the harder minority classes. The equation for alpha balanced focal loss is Equation 1.

Andrade et al. [23] found focal loss effective at dealing with their class imbalance issue, and so it is adopted as a loss function in this project.

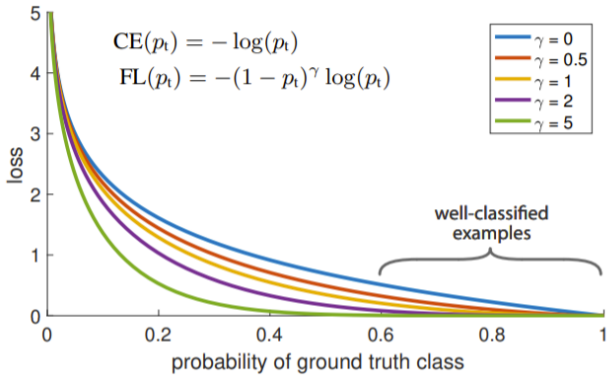


Figure 29 – effect of varying the gamma term on the loss function curve. Note, gamma = 0 is equivalent to cross entropy, taken from [49]

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t)$$

Equation 1 – Alpha balanced focal loss, taken from [49]

4.5.2 Hyperparameters

As with many machine learning models, deep learning has many parameters that control the learning process, called hyperparameters. They are distinct from model weights in that they are not inferred while the model is fitting to the dataset. Changing hyperparameters can have a dramatic impact on how the models learn and how well the trained model makes predictions. There are many hyperparameters, presenting an unreasonably large search space however most variation in a models performance can generally be attributed to just a few of them [50]. Due to time, resource limitations and in some cases simple logic, some hyperparameters were fixed and some varied through a limited sample space in a grid search manner. Both are described here. Any hyperparameters not mentioned in this section are left as the PyTorch default.

4.5.2.1 Fixed

Loss function

In preliminary runs focal loss was found to outperform cross entropy loss and so has been used in all runs.

Class weights - alpha

As discussed in 4.5, focal loss takes a weighting alpha for each class. Class weight has been used here, and for the ith class is defined as:

$$ClassWeight_i = n_{samples} / (n_{classes} * n_{samplesi})$$

This has been calculated for the each dataset.

Gamma

As stated in 4.5, focal loss uses a gamma term to adjust the shape of the loss curve. A gamma v alue of 2 has been used as the class imbalance is not extreme, and has been shown to work well in practice [49], [23].

Batch size

Batch size was already introduced in 4.2.5. It is the number of samples that pass through a model prior to the error gradient estimation and the update of the model weights. Smaller batch sizes are noisy, which can help generalisation, and they also enable a whole batch of training data to fit in GPU memory. A batch size of 32 is common to use [58] especially when it is not tuned, and it has been shown that batch sizes of 32 or smaller yield good results [59].

When training, a batch size of 32 would not fit onto the GPU memory with the models simultaneously, and so the batch size was set to 16. Andrade et al. [23] also trained their model with a batch size of 16 obtained good results.

Epoch

The number of times the dataset will pass through the model during training. If we allow the model to train for too long, the model may not generalize well and will overfit the training data. Too short and it may not of converged fully.

Whilst this is a variable hyperparameter, the approach taken here is to ensure the training runs sufficiently long enough (50 epoch) to use the log to make a judgement how the training has progressed.

4.5.2.2 Varied

Optimiser

The optimizer aims to reduce the error rate (defined by the loss function) by backpropagating it through the network and updating the weights. They should always move in a downwards direction leading to gradient descent. Many optimisers are available, but stochastic gradient descent (SGD) and Adam are commonly used [22], [23] and will both be part of the search space.

Learning rate

Often regarded to be the most important hyperparameter, the learning rate controls the rate the model weights are updated as a response to the error, essentially providing step size. As with any optimization problem, if the step size is too small it can take a long time to converge, potentially failing to train or may get stuck in a local minima. However if the step size is too large it may miss the minima completely. This is further complicated by the use of pretrained weights; They will hopefully place the model in a good local minima to start with, so jumping out of this with a high learning rate would be unwise. However, maybe the domain is too different and they place us in a bad local minima, and leaving it may be better.

It is clearly highly problem specific and in this project is handled by empirically. The values considered initially are 0.01, 0.001 and 0.0001. However if validation finds the best model to be

at one end of the search space, a further order of magnitude shall be used until a peak is found.

4.6 Evaluation metrics

As with all machine learning tasks, there is a need to have evaluation metrics in order to present results of how well the semantic segmentation models a performing. TP refers to true positives, FN refers to false negatives, n_{cl} is the number of classes.

Class Precision

Precision measures the proportion of positives that are actually positive. In a multi-class case, precision for each class (k) is calculated by :

$$\boldsymbol{Precision_k} = \frac{TP_k}{TP_k + FP_k}$$

Intuitively, class precision tells us how much can we trust a prediction of that class.

Class Recall

Recall measures the proportion of true positives that are correctly classified. In a multi-class case, recall for each class (k) is calculated by :

$$\boldsymbol{Recall_k} = \frac{TP_k}{TP_k + FN_k}$$

Intuitively class recall answers the question "how likely will a pixel of that class be correctly classified?" It measures the models ability to find all the positives of that class in the dataset.

F1-Score

F1-Score aggregates precision and recall using the idea of harmonic mean. It is calculated as:

$$\boldsymbol{F1 - Score} = \mathbf{2} \cdot \left(\frac{precision \cdot recall}{precision + recall}\right)$$

As F1-Score is provides a useful trade off between both precision and recall, and can help spot weak points of a classifier.

Balanced accuracy

Balanced accuracy in a multiclass classification problem is good with imbalanced datasets. It is defined as the average recall of each class, thereby giving each class equal importance regardless of its sample size:

$$\boldsymbol{Balanced\ accuracy} = \left(\frac{1}{n_{cl}}\right)\sum_k \frac{TP_k}{TP_k + FN_k}$$

Where k = {1…n_{cl}}

If a dataset is balanced, accuracy and balance accuracy tend to converge on the same value [51].

Mean IOU

This is the mean of the intersection over union across the classes.

$$\boldsymbol{mIOU} = \left(\frac{1}{n_{cl}}\right)\sum_{k}\frac{TP}{(TP + FN + FP)}$$

Where k = {1…n_{cl}}

5 Experimental Setup

This section defines the experiment and clearly states what work has been performed.

A hyperparameter search is conducted and evaluated on the validation subset. It is used to identify the best model for the task and its associated tuning.

The selected model and tuning then undergoes two ablation studies. The first is check the sensitivity to the training set size. Secondly, the gamma variable which was considered fixed will be varied to assess the models sensitivity to it. Thirdly, the model is allowed to train for a longer period of time so that overfitting is detected and an epoch to stop at identified.

Finally, the selected model and tuning (with the results of the ablation studies also considered) is tested on the unseen test subset.

As explained in 4.2.1, a class exists for pixels with erroneous data in at least one of the layers of the dataset. All training will be set to ignore this class, effectively excluding it from the model.

The results to all the investigations are presented in 6.

5.1 Hyperparameter Search Space

As discussed above, hyperparameter tuning has been performed on the models. The search space is presented in Table 5. As can be seen, 3x2x3 = 18 runs are performed initially for 50 epochs each. A further two runs were performed to ensure the best learning rate was found for each architecture and is documented under the respective model.

Name	Values	# of
Architecture	FCN8, Segnet, VGG16 U-Net	3
Loss function	focal	1
Class weights	[1.1403, 0.6682, 1.0715, 1.8280, 2.2786, 0.5857]	1
Gamma	2	1
Batch size	16	1
Epoch	50	-
Optimiser	SGD, Adam	2
Learning rate	0.01, 0.001, 0.0001, (0.00001)	3

Table 5 – Hyperparameter search space

The models generated from the hyperparameter search were trained using the training dataset. They were evaluated using mIOU and Balanced accuracy, as detailed in 4.6, with the validation dataset at each epoch and the results logged for discussion.

5.2 Ablation study

5.2.1 Dataset Size

Further analysis has been performed to assess the sensitivity of the size of the training dataset on validation accuracy. In order to do this the training set was randomly split to have 0.75 and 0.5 of the original training dataset.

5.2.2 Gamma

In the hyperparameter search, gamma was assumed to be a fixed value of 2. As gamma determines how much the models attention is turned towards minority classes, it should be set to match the imbalance of classes.

A sensitivity is performed with gamma values of 0.5, 1, 2, and 5.

5.2.3 Number of epochs

When we have highly flexible models that can fit complex problems, care is needed to ensure overfitting does not occur. It is undesirable to allow training to run for too long as the model will attempt to fit every minor variation in the data input, which is most likely to be noise than a true signal. An over fitted model will not generalise well to out of sample data, and knowing when to stop training is important.
The model is allowed to train for a larger number of epochs in order to establish exactly where overfitting starts to occur. [24] and [23] used 80 and 100 epochs respectively to train, and so the models will be trained upto 100 epochs.

5.3 Testing

The selected model and tuning is then trained using both the training and validation data. In order to mitigate the chance of overfitting, training is allowed to continue to the epoch number ascertained in 5.2.3.

In deployment, the model would need to make predictions on areas with infilled data. To simulate this, the test subset is regenerated without the excluded class in the ground truth. It is possible to do this as the seed was manually set during splitting.

The trained model is then used to make predictions on the unseen test subset. The evaluation metrics in 4.6 are then calculated on the ground truth and the prediction and the results presented for discussion.

6 Results

Results are presented for the various studies, (see 5 for experimental procedure). Graphs are generated through tensor board and provided here to support the discussion and selection rational. Validation accuracy is computed using balanced accuracy, and supported with mIOU as a further validation. Both metrics are described in 4.6.

As the training process is stochastic, there can be a lot of fluctuation on validation which makes it hard to distinguish trends between models, especially when they are very similar. Tensorboard provides a 'smoothing' facility which reduces this variability and highlights trends. Where this function has been made use of will be stated, and the original un smoothed graph displayed transparently in the background for reference.

6.1 Results by Architecture

The best tuning for each model is picked here. These runs are then used in 6.2 to support model selection.

6.1.1 FCN8s

Figure 30 shows validation accuracy for all runs performed on the FCN8s architecture. The SGD optimiser failed to train under these conditions, as shown by the flat lines at the bottom of the graph. For the Adam optimiser, two runs did train, the learning rates were 0.001 and 0.0001. Whilst the curves are stochastic, it can be seen that a learning rate of 0.001 had a tendency to perform better. This was further shown in Figure 31 where the validation mIOU trends better for a learning rate of 0.001.

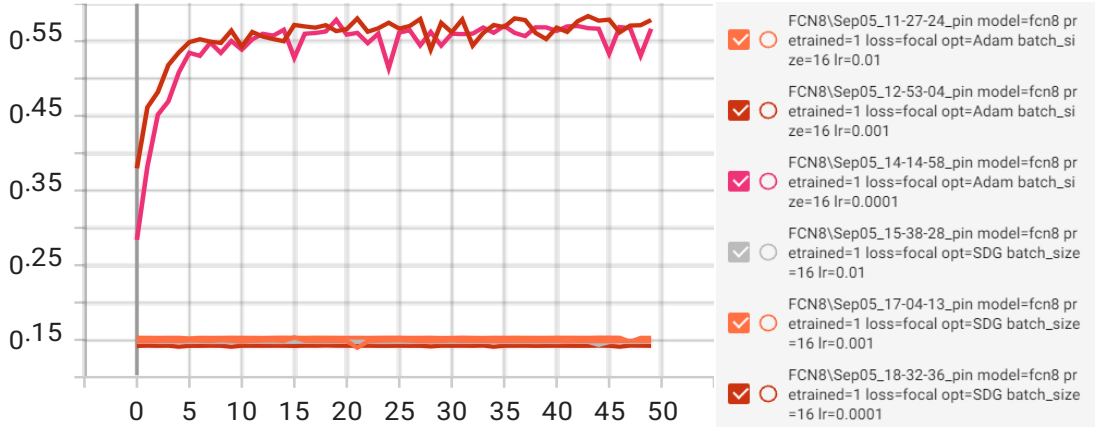


Figure 30 – Validation accuracy for all FCN8s models

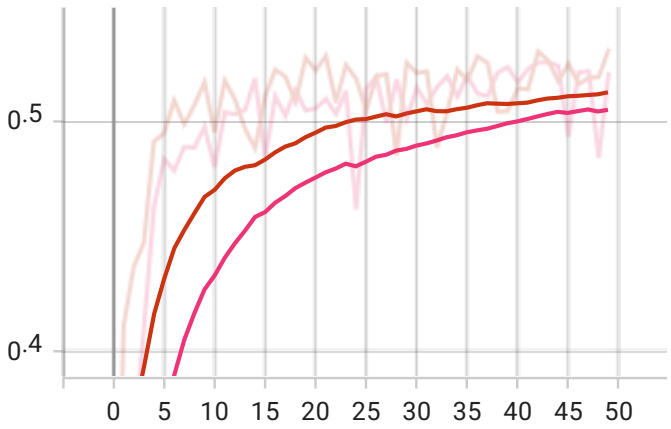


Figure 31 – Validation mIOU on FCN8, Adam opt., lr = 0.001 and 0.0001, smoothing applied

The best tuning of FCN8s was deemed to use an Adam optimiser with a learning rate of 0.001.

6.1.2 Segnet

Figure 32 and Figure 33 show the validation accuracy and mIOU respectively for all Segnet runs. Again we can see that the SGD optimiser failed to train under these conditions. It can be seen that Adam optimiser with a learning rate of 0.0001 trended highest in both validation accuracy and mIOU. A learning rate of 0.00001 was also trained but did not perform as well as 0.0001.

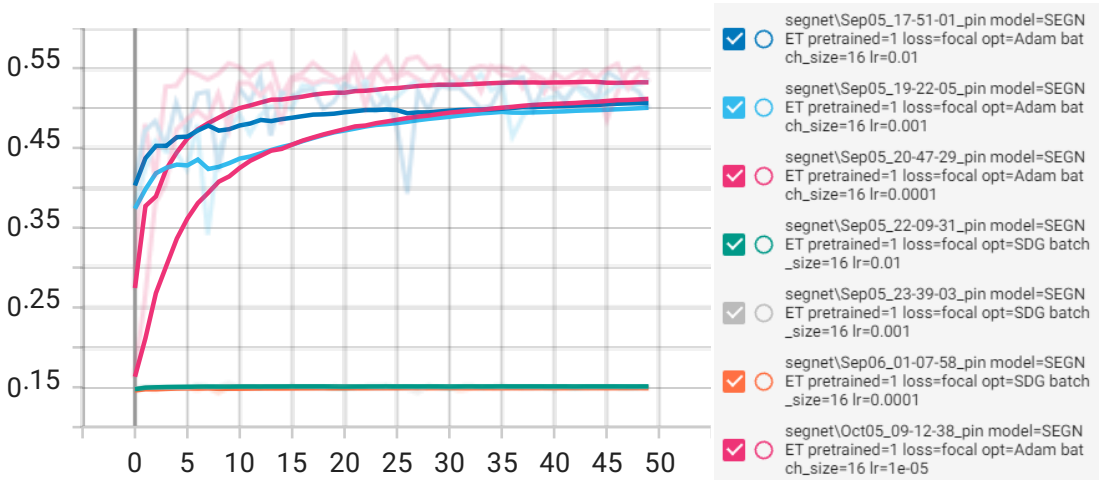


Figure 32 - Validation accuracy for all Segnet models, smoothing applied

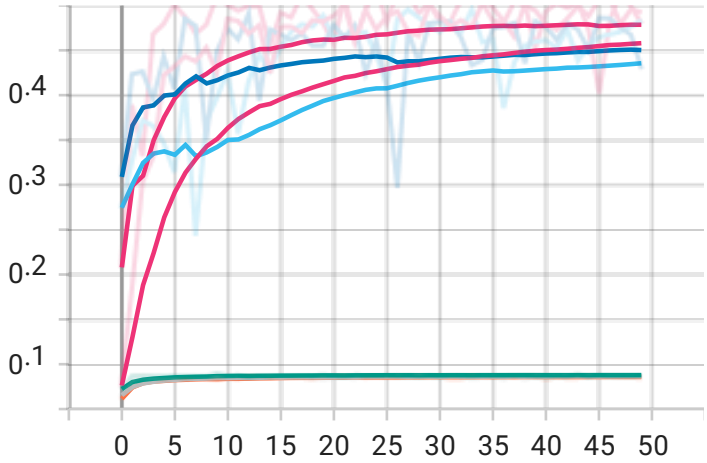


Figure 33 - Validation mIOU for all Segnet models, smoothing applied

The best tuning of Segnet was deemed to use an Adam optimiser with a learning rate of 0.0001.

6.1.3 VGG16 U-Net

Figure 34 shows validation accuracy for all runs performed on the VGG16 U-Net architecture. In this case, all runs trained. In order to identify the highest trending model, the low performing runs have been removed leaving just two and smoothing added, see Figure 35. The top two performers are trained using Adam with a learning rate of 0.0001 and SGD with a learning rate of 0.01. Adam with a learning rate of 0.0001 appears to trend slightly higher, and the mIOU graph (see Figure 36) supports this, however it is very close. A learning rate of 0.00001 was also trained but did not perform as well as 0.0001.

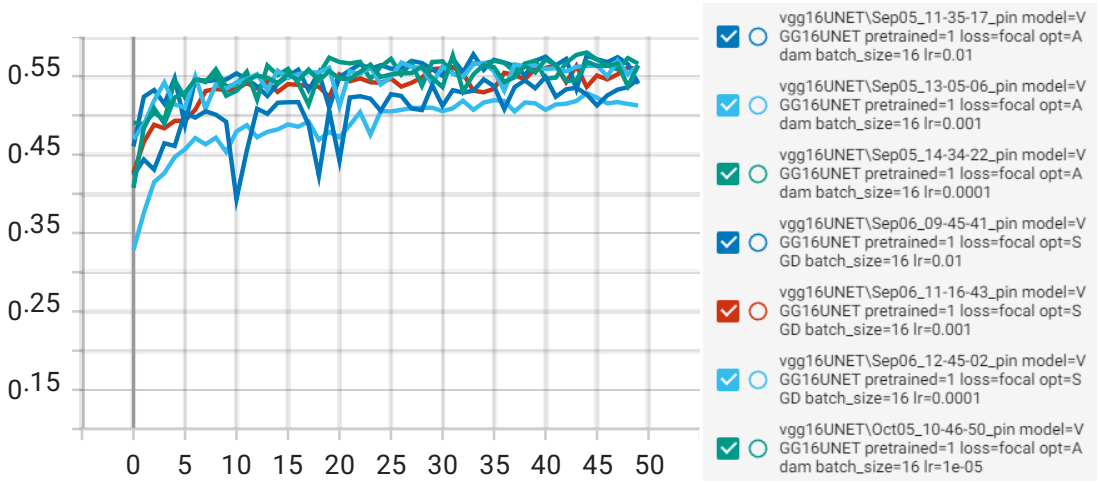


Figure 34 - Validation accuracy for all VGG16 U-Net models

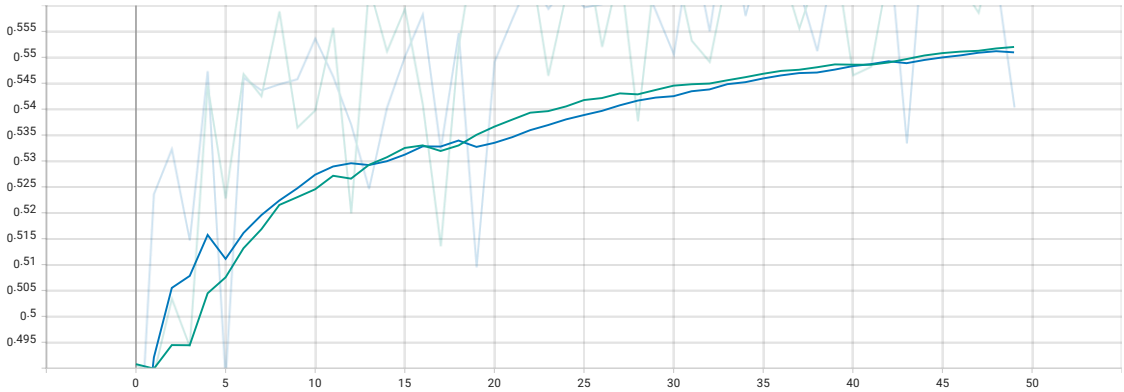


Figure 35 - Validation accuracy for VGG16 U-Net Adam lr=0.0001 and SGD lr=0.01 models, smoothing applied

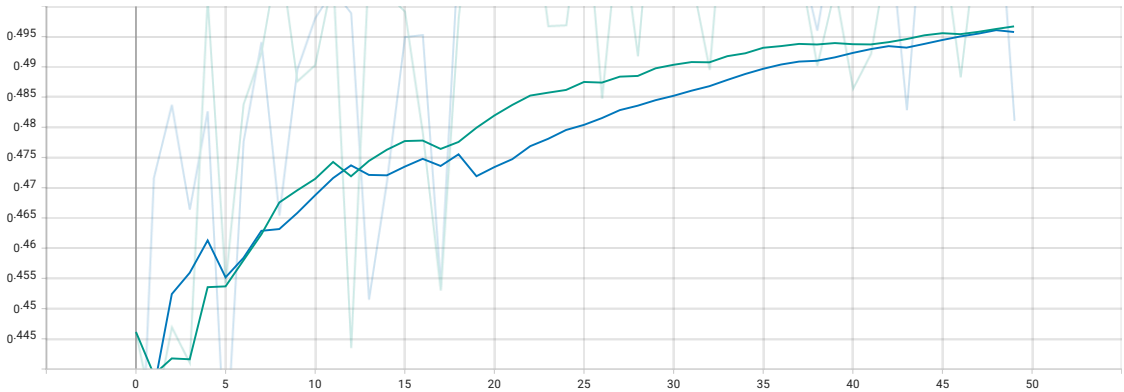


Figure 36 - Validation mIOU for VGG16 U-Net Adam lr=0.0001 and SGD lr=0.01 models, smoothing applied

The best tuning of VGG16 U-Net was deemed to use an Adam optimiser with a learning rate of 0.0001.

6.2 Model Selection

The best performing models validation accuracy are presented in Figure 37. Segnet appears to have performed the worst on this task, however it is not that far behind. FCN8s and VGG16 U-Net are almost equal, with FCN8s appearing to be marginally ahead. As this is a stochastic process and each model has only been trained once, it is not possible to confirm one model is better than the other with the difference being so minimal. Instead Figure 34 is referred to, showing VGG16 U-Net to perform well in all hyperparameter searches. In doing so, the architecture has shown stability on this task and demonstrated its robustness.

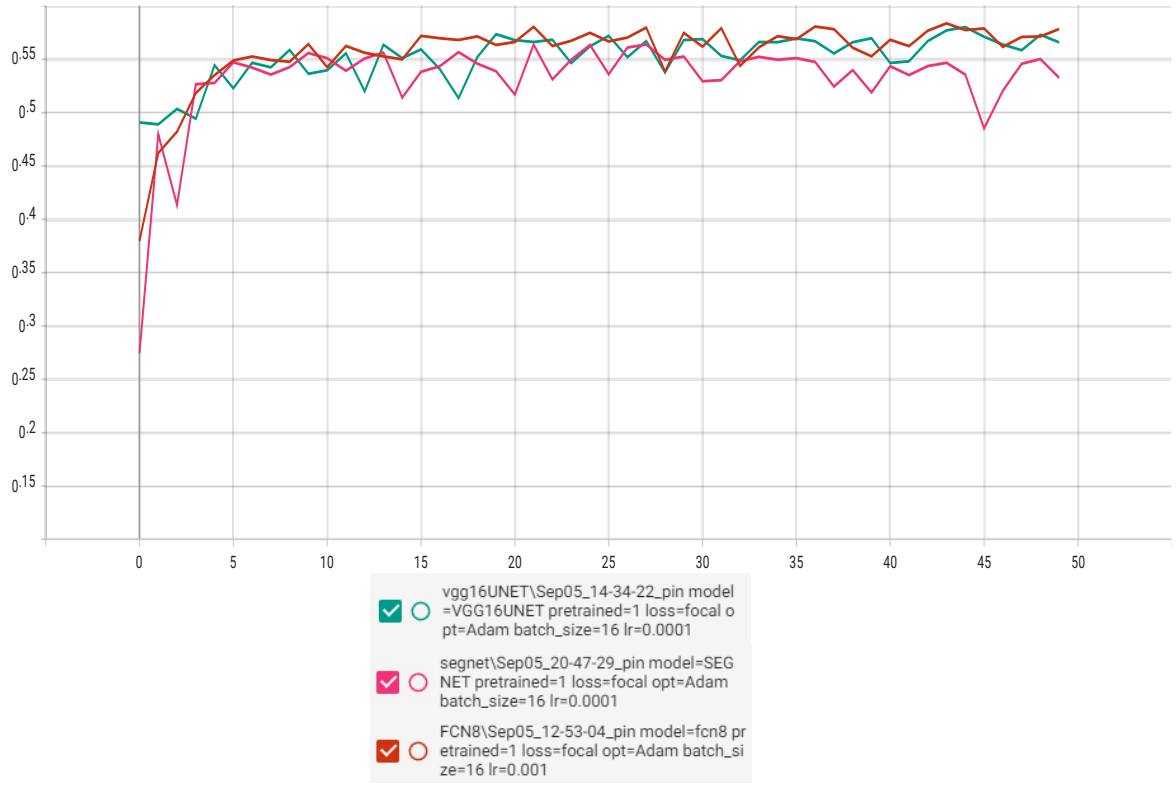


Figure 37 - Validation accuracy for the best performing models of each architecture

The best model architecture is found to be VGG16 U-Net. It was trained with an Adam optimiser with a learning rate of 0.0001. This model will be used in the sensitivity checks.

Other works [36],[24],[25],[26] have also found U-Net to perform best and so this outcome is consistent with the literature reviewed.

6.3 Ablation study

These studies are conducted using the VGG16 U-Net trained with an Adam optimiser with a learning rate of 0.0001.

6.3.1 Dataset Size

The results of the dataset size sensitivity check on validation accuracy are shown in Figure 38. Validation accuracy is found to be sensitive to the training dataset size, indicated more data may yield higher classification performance.

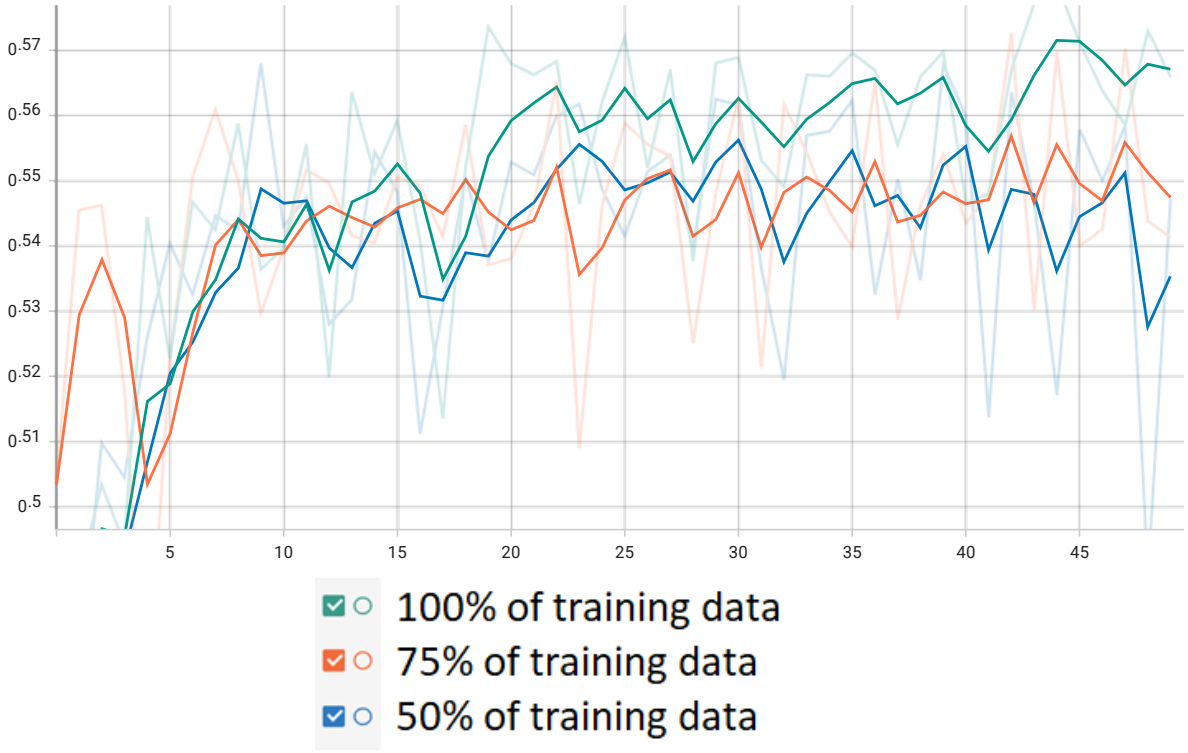


Figure 38 - Sensitivity of varying dataset size on validation accuracy, smoothing applied

6.3.2 Gamma

A sensitivity is performed with gamma values of 0.5, 1, 2, and 5.
The results are shown in Figure 39. A gamma value of 2 appears to trend best, however validation accuracy is found to not be that sensitive to gamma.

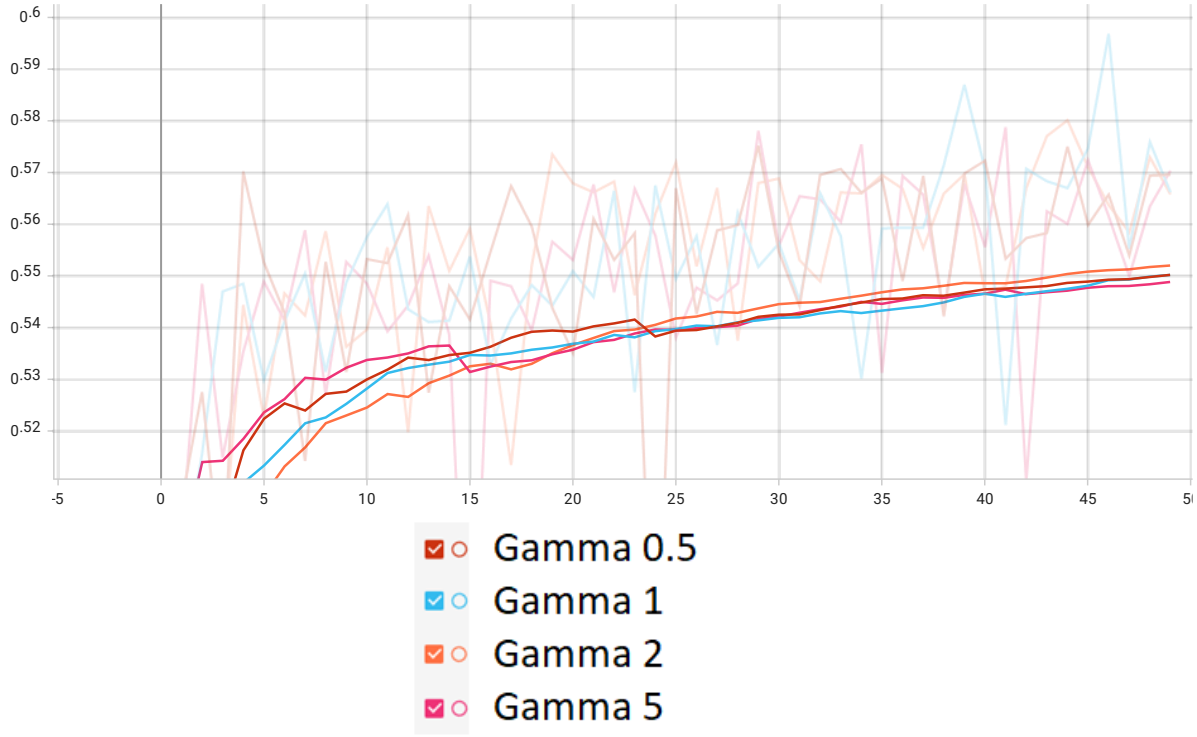


Figure 39 - Sensitivity of varying gamma on validation accuracy, smoothing applied

6.3.3 Number of epochs

A run for 100 epochs was performed to establish when to stop training. Figure 40 shows that the training loss is continually decreasing and would do after epoch 100. Figure 41 shows that the validation loss levels out at around epoch 80, and increases with additional epochs. The model is over fitting the training data, and so epoch 80 is deemed a good place to stop the training process.

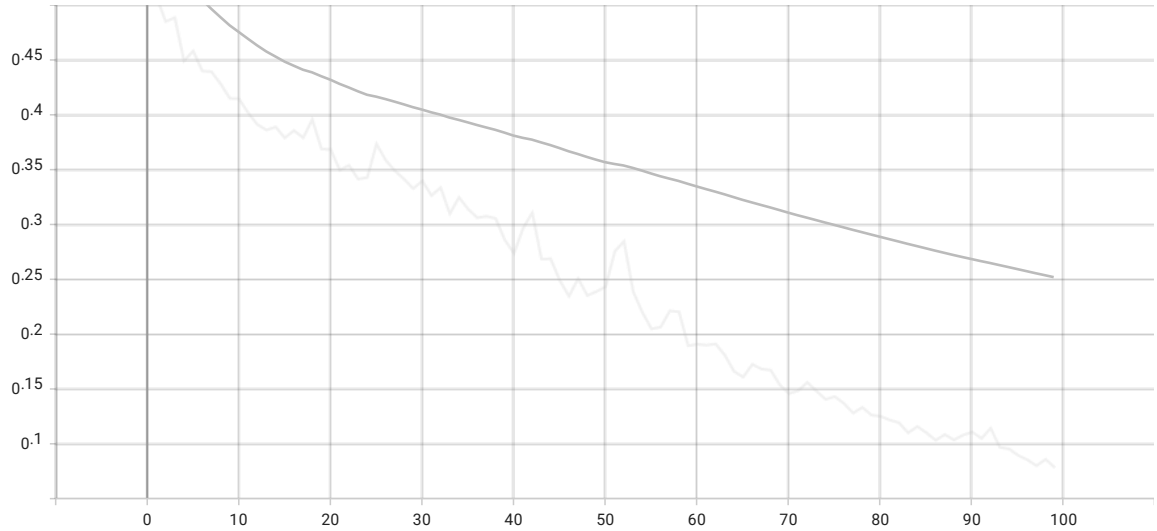


Figure 40 – Training loss upto 100 epochs

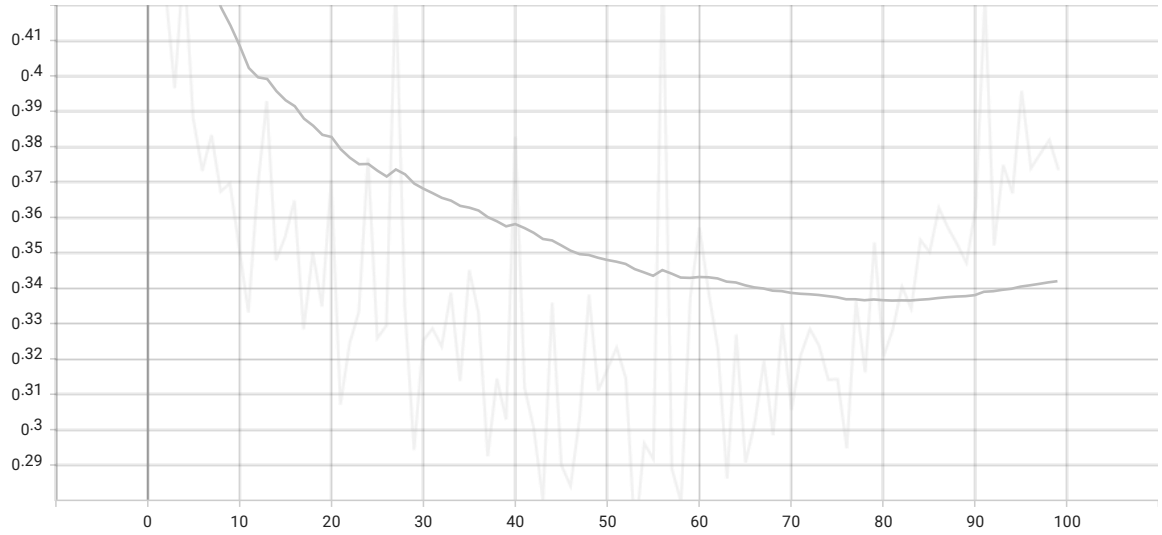


Figure 41 – Validation loss upto 100 epochs

6.4 Test - Performance Evaluation

The results of the test data on the trained VGG16 U-Net are presented here (see 5.3 for procedure). Class precision, recall and F1-Scores are shown in Table 6. The confusion matrix across the classes is shown in Table 7. The evaluation metrics from 4.6 are shown in Table 8.

Class	Class		
	Precision	Recall	F1-Score
Upland Forest	0.855	0.931	0.891
Water Surface	0.860	0.831	0.845
Aquatic Grasses	0.492	0.441	0.465
Shrubs	0.336	0.439	0.381
Woodland	0.251	0.439	0.319
Floodplain Forest	0.952	0.616	0.748

Table 6 - Class Precision, Recall and F1 Scores

Precision scores vary across the classes from 0.251 to 0.952, and recall scores vary from 0.439 to 0.931. However the F1-Score gives us a better general idea of the models strengths and weaknesses.

The F1-Scores highlight the models strengths; it is good at predicting water surface, upland forest and reasonable at reasonable at floodplain forest.

As discussed in 2.2.1, open water exhibits specular scattering making its classification relatively easy to distinguish from the other classes, as the results show. Unflooded vegetation will also appear darker in SARs images, and the combination of the elevation data and NDWI index may explain why the model predicts upland forest well. Floodplain forest is also predicted reasonably well which could be due to L band double bounce on its vertical tall woody stems helping to distinguish it from other flooded vegetation, as suggested by other works [9], [10], [11].

The F1-Scores for aquatic grasses, shrubs and woodland show the models weaknesses, finding it hard to distinguish these classes. The confusion matrix confirms this also, with relatively high numbers of pixels misclassified between these classes. These classes are fairly homogeneous and as such can be similar in appearance. Within a 12.5 m^2 it is entirely possible that these classes co-exist and so we can expect to see some blending of the classes. This effect is also seen with aquatic grasses being predicted as open water surface and provides a good example

of this problem. How many blades of grass would need to be present to predict it as aquatic grass rather than open water? If there was considerable more rain one year is it possible aquatic grass or shrubs could be submerged at the time of data acquisitions and go undetected?

This is consistent with previous works with Hess et al. [54] finding macrophytes, non-flooded forest, non-flooded shrubs and barely emergent flooded shrubs difficult to distinguish due to similarities in backscattering.

There is also a reasonable possibility of actual differences between the ground truth, itself coming from an imperfect model, and the cover at the time the data was captured.

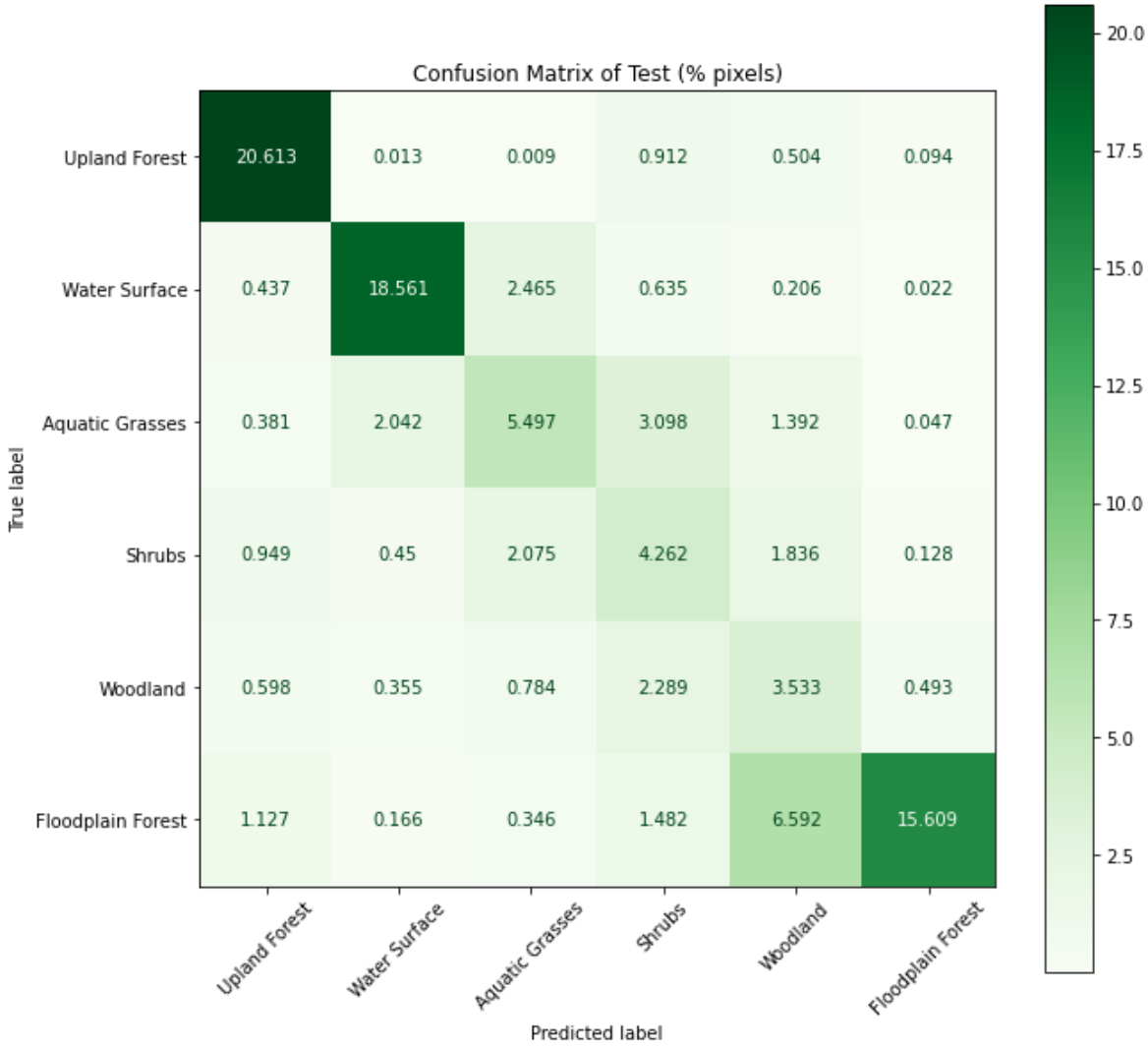


Table 7 - Confusion matrix

VGG16 U-Net achieved an overall balanced accuracy of 61.6% and a mIOU of 0.477. This reasonable considering the confusion amongst aquatic grasses, shrubs and woodland. A possible solution to this maybe to find a new band that can help better separate these classes, and include it in the dataset. Until this is available, a solution could be to collapse them into one class. This would be likely to yield higher evaluation metrics.

Set	Network	Bal. Acc.	mIOU
Test	VGG16 U-Net	0.616	0.477

Table 8 – Results of each evaluation metric for the chosen multi-classifier model when making predictions on the test set

7 Demonstration - Inference on large patches

The work performed thus far has proposed a deep learning method that classifies pixels and segments regions from sensor images of Amazon floodplain areas. The model selected was VGG16 U-Net, it has been trained as per 5.3 and tested 6.4. The weights of this model have been saved for making inferences.

What has not been addressed so far is how this trained model can be used to make inferences on whole patches (4000x4000 pixels), as it would need to in deployment. As stated before, the ground truth map for Curuai 1, Curuai 2 and Parintins are derived from the output of Hess et al. [54] work; Their model produced predictions with 100m per pixel resolution for the entire low-land Amazon basin circa 1990s. The work conducted in this section revolves around the idea of using the proposed deep learning classifier to "update" these maps using newer and better spatial resolution data.

7.1 Pipeline on trial patch - Parintins

The Parintins was described in section 3, but is thus far unused in the project. It shall be used here to make a prediction of a whole patch. As shown in Figure 42, at least one of the layers has significant missing data, most likely due to atmospheric effects as explained in 4.2.1. This is the same as could be expected during deployment and the model will need to be robust enough to handle such issues. The data is put through the same pipeline as before and the 'nan' values are infilled.

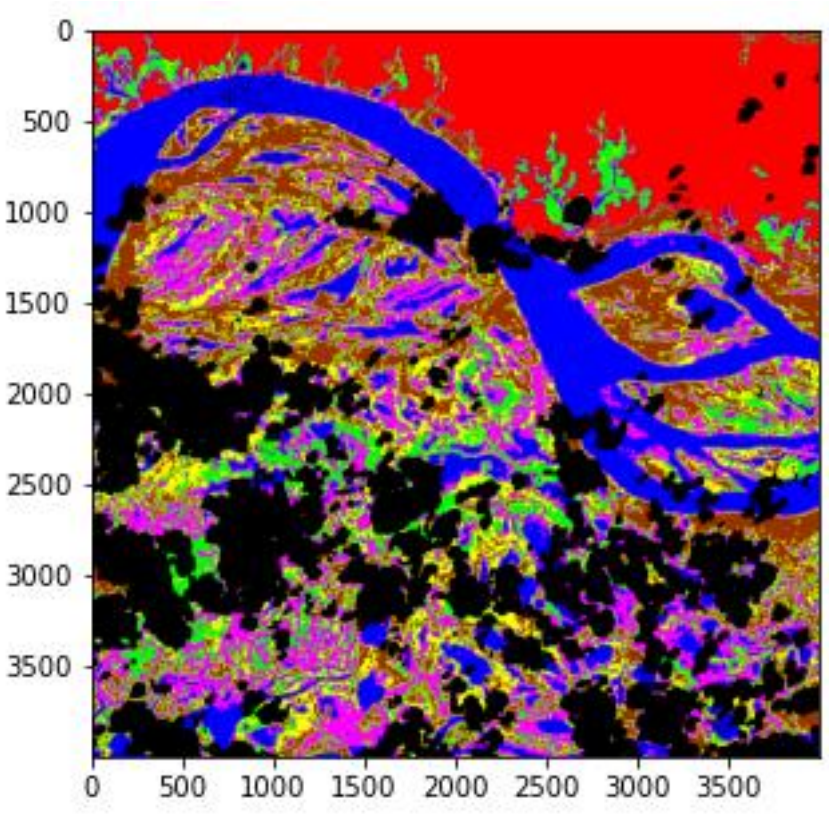


Figure 42 - The Parintins patch, black areas are due to missing data in the layers

7.2 Overlapping strategy

As explained in, 4.3.7, VGG16 U-Net is a modified version of original U-net architecture that uses padding to maintain image size after each convolution throughout the network. This means the prediction is the same dimensions as the input, however the borders of each

prediction are based on non-valid parts of the convolutions. Ronneberger et al. [30] model produced smaller output predictions and they explained an overlapping tile strategy is needed to make predictions on large images.

As explained in 2.3.1, Flood et al. [24] used such an overlapping strategy with success. A similar overlapping tile strategy is adopted here. The infilled patch data is broken into smaller overlapping tiles, using a stride of 156 pixels. The tiles are 256x256x22 making them suitable for use with the trained VGG16 U-Net model.

The tiles are then loaded into the *amazonMultiBandDataset* class as described in 4.2.3, using the mean and standard deviations from the original dataset from 4.2.4, and a dataloader object is created. The VGG16 U-Net model with its associated trained weights is loaded and set to evaluation mode. The model creates predictions for each tile in the dataloader.

Each individual prediction tile is cropped, removing 50 pixels from the border. The resulting cropped predictions are then stacked back together to create the prediction image. It should be noted that the dimensions of this reassembled prediction image are 3900x3900 due to the removal of the edges on the border of the satellite patch.

7.3 Result

The resulting prediction image of the Parintins patch is shown in Figure 44, with the ground truth shown in Figure 45 for comparison. A colour legend is provided in Figure 43.

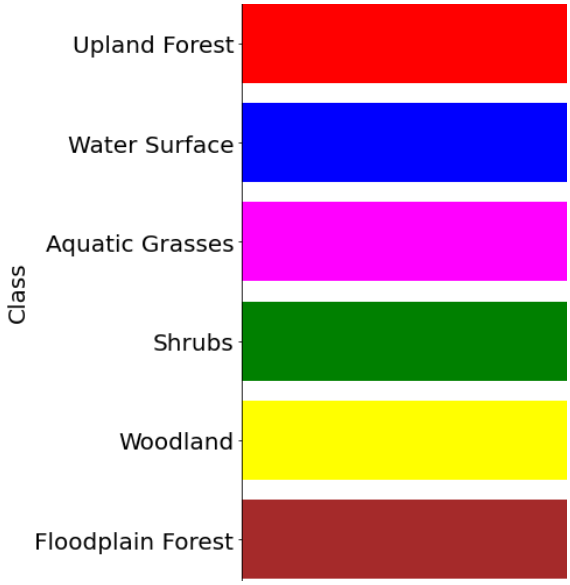


Figure 43 – Colour legend across the classes

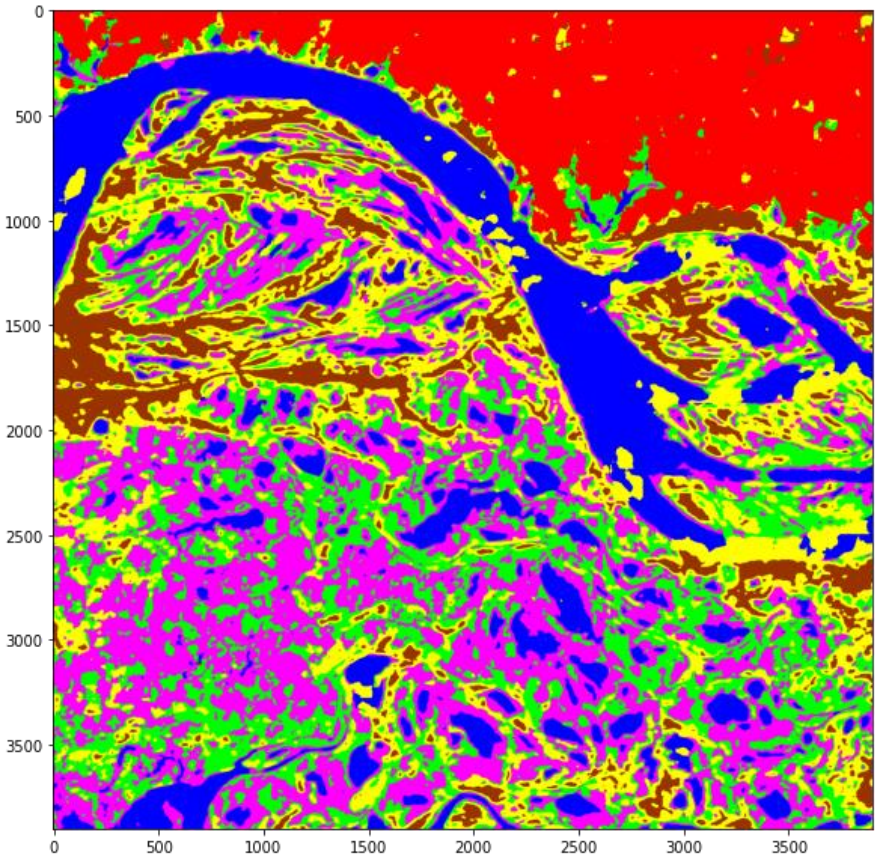


Figure 44 – Prediction map of Parintins created with the proposed deep learning model VGG16 U-Net

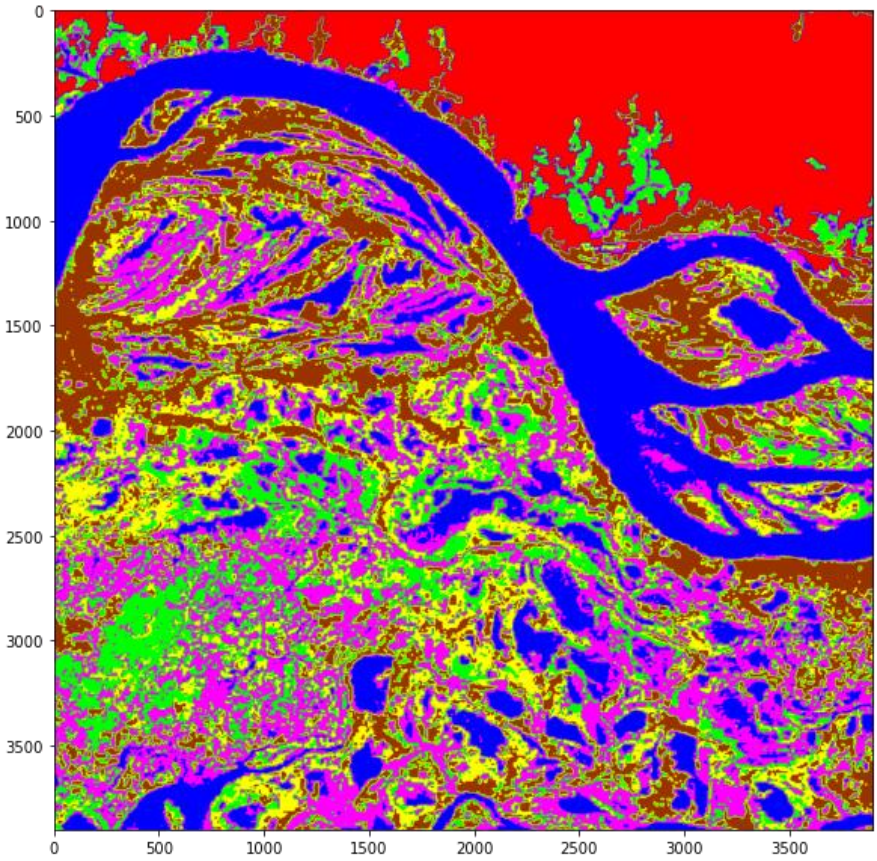


Figure 45 - Parintins ground truth, cropped to match the size of the prediction map

7.4 Evaluation

Border artifacts are not immediately obvious in the prediction patch (Figure 44). However on closer inspection, unnatural straight lines can be seen, indicating slight errors arising from the tiling process. The overlap strategy may need some refinement such as averaging over multiple predictions as other works have [45].

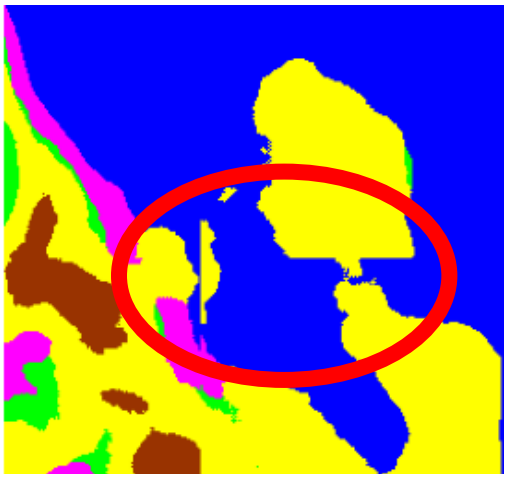


Figure 46 – Artifact from overlap strategy, note the sharp edges and the false river bank.

Continuing discussion from 6.4, it is clear that the model predicts upland forest well, however a few patches of woodland do appear (see top right of Figure 44.) Flood plain forest is also reasonable, with most of the areas being predicted, however the size of the clusters appears to be smaller, being eroded by woodland (this can be seen in Figure 47). This is not surprising as woodland is also a floodplain class based on trees, the only difference being the density of the canopy. It is also consistent with the reviewed literature with Bragagnolo et al. [26] stating the exact boundary is 'fuzzy' between the classes. This situation highlights the point made in 6.4 of inaccuracies in the ground truth. It has been up sampled from 100m per pixel, and was created with a pixel based classifier, and so it is not unreasonable to entertain the idea that the CNN model, that uses spatial features, could be more accurate than the ground truth in this instance, as has been found before [26].

Open water is classified well also, however, as shown in Figure 47 some of the cut through channels are missing, with the model often replacing them with woodland or shrub classes. Figure 48 shows example SAR and optical images for the same crop; it is not entirely clear that the water channels were actually present during the data acquisition.

As discussed before in 6.4, there is blending of aquatic grasses, shrubs and woodland shown in the confusion matrix (Table 7). The ground truth in Figure 49 highlights the homogeneous nature of these classes being blended together. The prediction captures the presents of these classes in the same area, but doesn't match them pixel by pixel.

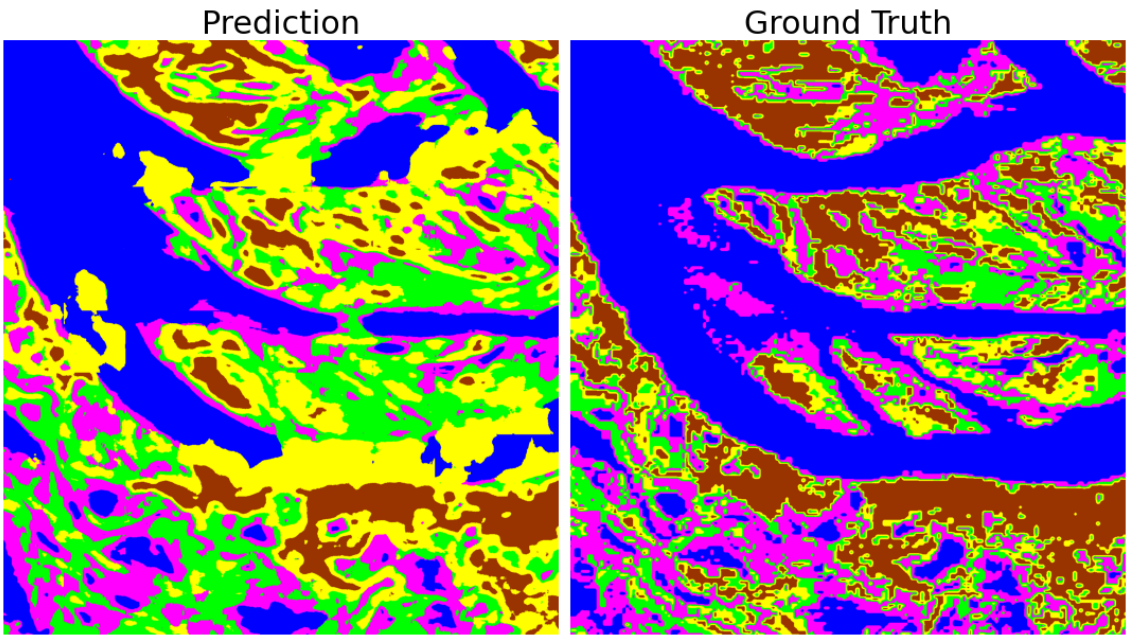


Figure 47 – Cropped images of prediction left, and ground truth right.

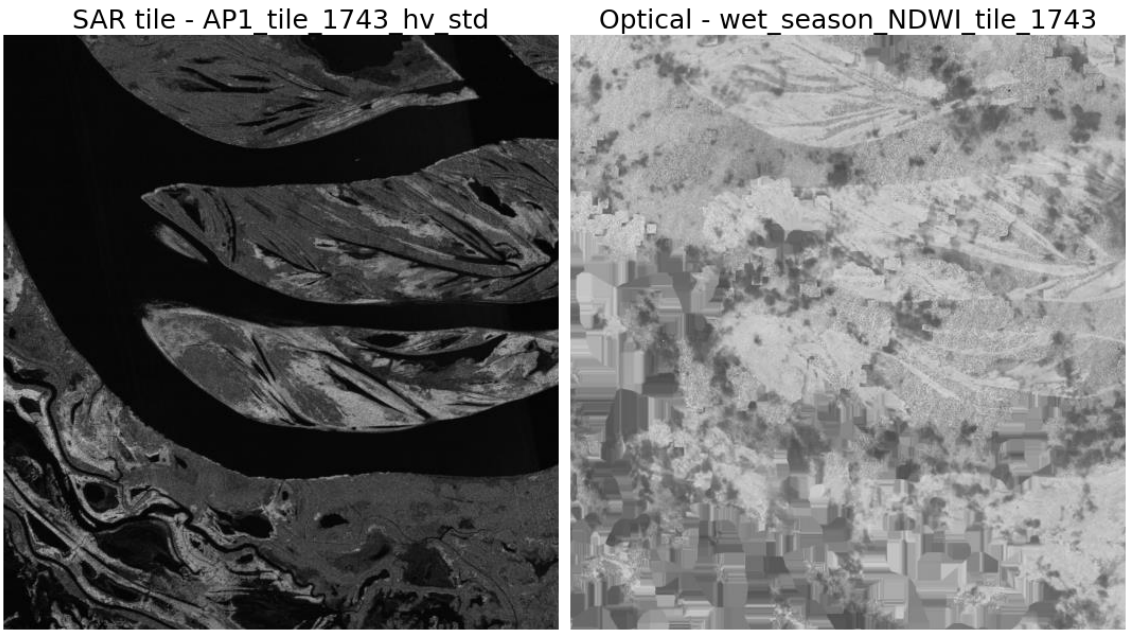


Figure 48 – Cropped images of example SAR image left, and Optical image right.

Again, these misclassifications are consistent with other works, with Liu et al. [43] finding water misclassified as vegetation along with mistakes in classifications between shallow and deep marsh vegetation. They found that images with higher spatial resolutions (2m – 10m) improved classification accuracy of marsh vegetation which could be a potential solution. As discussed earlier, there may also be a different band that would separate these classes better. Figure 50 shows the same location in both a SARs channel and an optical channel, it is not immediately obvious how these classes would be separated out from these images. It is fairly obvious however, even to a layman, that it is wetland.

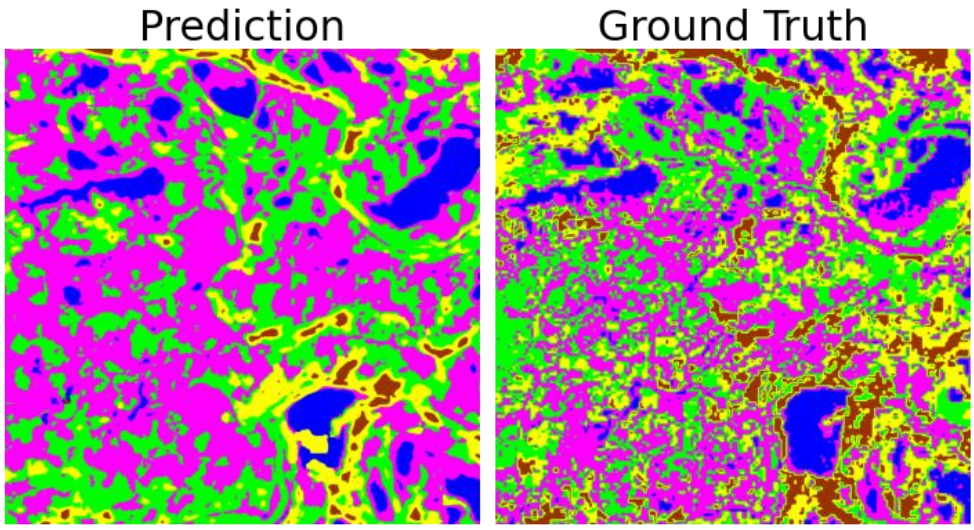


Figure 49 - Cropped images of prediction left, and ground truth right.

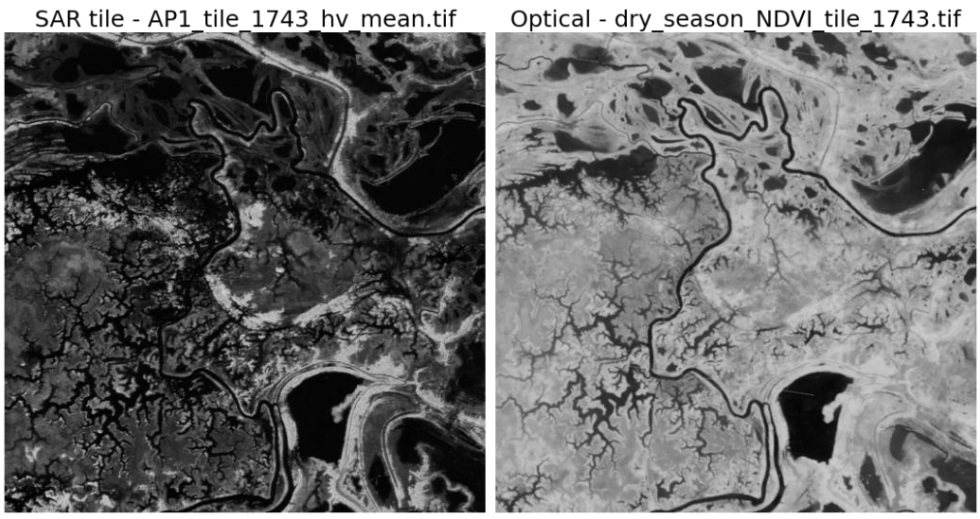


Figure 50 - Cropped images of example SAR image left, and Optical image right.

A further suggestion that is demonstrated in 6.3.1 is simply the use of more training data. The number of samples in the training dataset is considerably less than stated in similar works [43], [46]. Increasing the amount may also negate the negative effect of inaccuracies in the ground truth, effectively preventing the model from overfitting on the noise. This also presents a further option, to somehow clean the ground truths reducing misclassified pixels and strengthening the signal. This could be done in a manner of ways, such as a denoising algorithm on the current ground truth, or visual interpretation and hand labelling of high resolution images. Hand labelling could also be performed given correction from the model and the model retrained in a iterative manner, as suggested in other works [26]. Ultimately ground surveys would limit these errors.

The hyperparameter search was also limited in scope due to the timescales of the project, and a better tuning including more hyperparameters may benefit the results. The loss function is known to be extremely important as it instigates the learning process [56] and could be regarded as a hyperparameter. Combining loss functions has also been shown to work well with semantic segmentation of multispectral images [32].

A decision point in the path of this project was how to go about transfer learning. There is the obvious issue that pre trained models generally work with 3 channel RGB images, and the

strategy used of duplicating the weights to match the number of channels as described in 4.4, is just one option. Preliminary experiments showed dramatic improvements on training times using this transfer learning strategy as well as increases in classification accuracy. However, other strategies do exist [57] and could lead to better results.

A further possibility might be to try a different architecture altogether on the dataset. 2.3.1 mentioned other architectures that have been found useful. DeepLabV3+ for example has produced good results classifying marsh vegetation [43] and forest [23]. However, others [45] have found minimal to little improvement with more sophisticated CNN architectures. U-net architectures have been found to perform well with satellite imagery [32], [36], [24], [25], [26] and so it is likely the architecture is capable of more and not currently a limiting factor.

8 Conclusion

8.1 Summary

This project has succeeded in proposing a deep learning based method to map floodplain forest areas using multisensory data.

A novel dataset was created consisting of optical LANDSAT 5 and L-band SAR PALSAR 1 imagery, in addition to AW3D elevation data, at 12.5m spatial resolution.

Three FCN architectures, FCN8s, SegNet and VGG16 U-Net were trained and validated across a number of hyperparameters.

VGG16 U-Net wass found to perform best, achieving F1-Scores of 0.748-0.891 on certain classes, with a balanced accuracy of 0.616.

The project demonstrated that the VGG16 U-Net model generalised well by making predictions on unseen test data. It also went a stage further than this by demonstrating how the model could be used in deployment to make predictions on larger satellite image patches and did so on an unseen patch. The prediction generated on this patch was then used to demonstrate the strengths and weaknesses of the model and how it might be improved upon.

While the results of this project are successful and demonstrate the potential of deep learning to make dense prediction on multispectral satellite imagery of the Amazon basin, further work is needed to refine the model and the chosen input data before deployment should be considered. Either that or accept a reduction in classes. It would also be prudent to retest the model on actual ground truth data to confirm its classification accuracies are within tolerance.

8.2 Evaluation

In order to be able to build and train deep learning models it was necessary to learn the PyTorch frame work. This had knock on effects because a lot of PyTorch is object oriented, and it was necessary to learn this programming paradigm. Once training had commenced it became obvious that there was a need to understand and use GPU hardware in order for the project to be successfully completed.

Understanding of the earth observation sector and GIS world was undertaken early on in the project. Grasping the how the projection of the satellite image is handled and the concepts of the geo information being stored within the files structure understood. Combining the extracted raster information to create a new dataset was also achieved successfully, requiring problem solving skills and computer vision knowledge to handle missing values.

Three deep learning models capable of semantic segmentation were coded in Pytorch, ensuring cognition processes within the architectures. A creative solution for applying transfer learning when the input channels do not match the incoming trained weights tensor was developed.

Tuning of the models was performed via a hyperparameter search. Time and computational power restraints limited the extent of this search, however it was deemed sufficient to make performance comparisons between the three architectures and base model selection on. Ablation studies were then performed on VGG16 U-Net to assess its sensitivity to certain variables such as dataset size.

VGG16 U-Net was then trained accordingly with the training and validation data and tested on the unseen test set. As the unseen test was taken from the dataset at random, it did not consist of consecutive tiles. This means the border inaccuracies caused by non-valid parts of the convolutions have been included in the results, and actual results are likely to be higher.

Results were obtained by cropping the tiles in an attempt to only include the valid part of the predictions, however the gains didn't seem worth the loss of test data. A better solution would to simply of had more satellite patches to test, however data availability restricted this approach.

A further patch was obtained and used to demonstrate potential deployment and how the model generalised, providing a good visual output to aid discussion.

8.3 Limitations

The VGG16 U-Net model was evaluated against a prediction mask produced by another model. It should be tested on more accurate hand labelled ground truths. Due to this, the work in this project provides a proof of concept, and justifies further work rather than a model fit for deployment.

A limitation that should be noted is the scope of the hyperparameter search. As stated before, time and resources dictated what was feasible, and many more hyperparameters exist.

A different loss metric could aid the learning process better. Dice loss could be tried or even combinations of loss functions [56].

The technique used for transfer learning may be a limiting factor and a better approach may exist.

Despite the limitations, the model could be used to estimate changes in the land cover of the Amazon basin, helping to detect changes or deforestation in the floodplain region.

8.4 Future Work

Further work on collapsing the classes into more general categories is already underway but was not ready intime for inclusion in this report. Something that would have been extremely beneficial is a method to evaluate the significance of each layer in making inferences. The idea of an additional sensitivity study where each layer gets dropped in turn and the test inference repeated may yield valuable insights into the significance of each layer to each class prediction. Another option that could be investigated is semi-supervised or even unsupervised training. This would have the obvious benefit of requiring little or no ground truth. It would be interesting to see how a significantly different architecture would perform on the task, and future work could include applying DeepLabV3+ or a vision transformer, such as TransUNet to the task. Figure 51 shows the extent of the area covered by patch data. In the future it is intended to deploy the model to perform inference on the whole region.

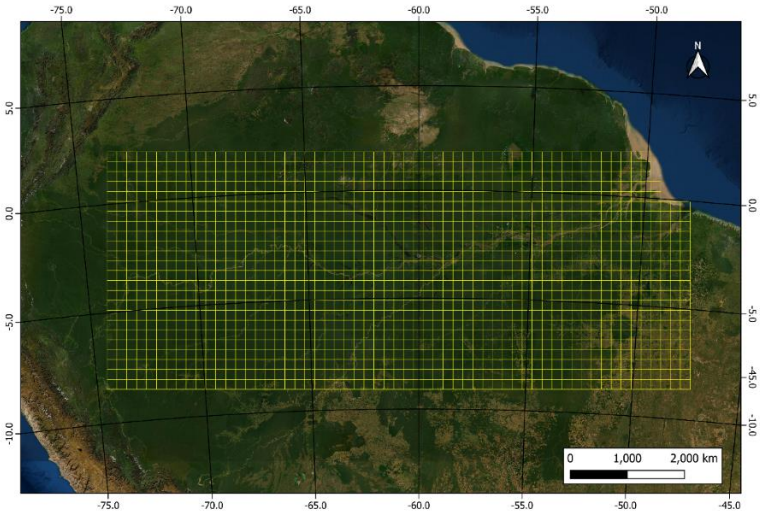


Figure 51 - Entire coverage of satellite patch data that the project will go on to make inference on.

References

[1] Noh, Hyeonwoo, Seunghoon Hong, and Bohyung Han. "Learning deconvolution network for semantic segmentation." Proceedings of the IEEE international conference on computer vision. 2015.

[2] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[3] Zhu, Xiao Xiang, et al. "Deep learning in remote sensing: A comprehensive review and list of resources." IEEE Geoscience and Remote Sensing Magazine 5.4 (2017): 8-36.

[4] Zhu, Xiaoxiang, et al. "Deep learning meets SAR: concepts, models, pitfalls, and perspectives." IEEE Geoscience and Remote Sensing Magazine (GRSM) (2021).

[5] Costa, M. P. F. "Use of SAR satellites for mapping zonation of vegetation communities in the Amazon floodplain." *International Journal of Remote Sensing* 25.10 (2004): 1817-1835.

[6] Costa, Michael. "Estimate of net primary productivity of aquatic vegetation of the Amazon floodplain using Radarsat and JERS-1." *International Journal of Remote Sensing* 26.20 (2005): 4527-4536.

[7] Kasischke, Eric S., et al. "Effects of seasonal hydrologic patterns in south Florida wetlands on radar backscatter measured from ERS-2 SAR imagery." *Remote sensing of environment* 88.4 (2003): 423-441.

[8] Hess, Laura L., et al. "Delineation of inundated area and vegetation along the Amazon floodplain with the SIR-C synthetic aperture radar." *IEEE transactions on geoscience and remote sensing* 33.4 (1995): 896-904.

[9] Ralph W. Tiner, Megan W. Lang, Victor V. Klemas - Remote Sensing of Wetlands Applications and Advances-CRC Press (2015)

[10] Novo, E. M. L. M., et al. "Relationship between macrophyte stand variables and radar backscatter at L and C band, Tucuruí reservoir, Brazil." *International Journal of Remote Sensing* 23.7 (2002): 1241-1260.

[11] Costa, M. P. F., et al. "Biophysical properties and mapping of aquatic vegetation during the hydrological cycle of the Amazon floodplain using JERS-1 and Radarsat." *International Journal of Remote Sensing* 23.7 (2002): 1401-1426.

[12] Hess, Laura L., et al. "Dual-season mapping of wetland inundation and vegetation for the central Amazon basin." *Remote sensing of environment* 87.4 (2003): 404-428.

[13] Noernberg, M. A. "The use of biophysical indices and coefficient of variation derived from airborne synthetic aperture radar for monitoring the spread of aquatic vegetation in tropical reservoirs." *International Journal of Remote Sensing* 20.1 (1999): 67-82.

[14] Xie, Zhuli, et al. "Classification of land cover, forest, and tree species classes with Ziyuan-3 multispectral and stereo data." *Remote Sensing* 11.2 (2019): 164.

[15] Silva, Thiago SF, et al. "Remote sensing of aquatic vegetation: theory and applications." *Environmental monitoring and assessment* 140.1 (2008): 131-145. Simard, Marc, et al. "Mapping height and biomass of mangrove forests in Everglades National Park with SRTM elevation data." *Photogrammetric Engineering & Remote Sensing* 72.3 (2006): 299-311.

[16] Haack, Barry N., and E. Terrance Slonecker. "Merged Spaceborne Radar and Thematic Mapper Digital Data for Locating Villages in." *Photogrammetric engineering & remote sensing* 60.10 (1994): 7253-1.

[17] Haack, Barry, and Matthew Bechdol. "Integrating multisensor data and RADAR texture measures for land cover mapping." *Computers & Geosciences* 26.4 (2000): 411-421.

[18] Rouse, John Wilson, et al. "Monitoring vegetation systems in the Great Plains with ERTS." *NASA special publication* 351.1974 (1974): 309.

[19] Gao, Bo-Cai. "NDWI—A normalized difference water index for remote sensing of vegetation liquid water from space." *Remote sensing of environment* 58.3 (1996): 257-266.
[20] Tian, Hanqin, et al. "Climatic and biotic controls on annual carbon storage in Amazonian ecosystems." *Global Ecology and Biogeography* 9.4 (2000): 315-335.
[21] Cox, Peter M., et al. "Acceleration of global warming due to carbon-cycle feedbacks in a coupled climate model." *Nature* 408.6809 (2000): 184-187.
[22] De Bem, Pablo Pozzobon, et al. "Change detection of deforestation in the Brazilian Amazon using landsat data and convolutional neural networks." *Remote Sensing* 12.6 (2020): 901.
[23] Andrade, R. B., et al. "Evaluation of semantic segmentation methods for deforestation detection in the amazon." *ISPRS Archives; volume 43, B3* 43.B3 (2020): 1497-1505.
[24] Flood, Neil, Fiona Watson, and Lisa Collett. "Using a U-net convolutional neural network to map woody vegetation extent from high resolution satellite imagery across Queensland, Australia." *International Journal of Applied Earth Observation and Geoinformation* 82 (2019): 101897.
[25] Lee, Seong-Hyeok, et al. "Classification of Landscape Affected by Deforestation Using High-Resolution Remote Sensing Data and Deep-Learning Techniques." *Remote Sensing* 12.20 (2020): 3372.
[26] Bragagnolo, L., Roberto Valmir da Silva, and José Mario Vicensi Grzybowski. "Amazon forest cover change mapping based on semantic segmentation by U-Nets." *Ecological Informatics* 62 (2021): 101279.
[27] Yosinski, Jason, et al. "How transferable are features in deep neural networks?." arXiv preprint arXiv:1411.1792 (2014).
[28] Donahue, Jeff, et al. "Decaf: A deep convolutional activation feature for generic visual recognition." International conference on machine learning. PMLR, 2014.
[29] Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." arXiv preprint arXiv:1603.07285 (2016).
[30] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and computer-assisted intervention*. Springer, Cham, 2015.
[31] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *International conference on machine learning*. PMLR, 2015.
[32] Iglovikov, Vladimir, Sergey Mushinskiy, and Vladimir Osin. "Satellite imagery feature detection using deep convolutional neural network: A kaggle competition." *arXiv preprint arXiv:1706.06169* (2017).
[33] Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017): 2481-2495.
[34] Chantharaj, Sirinthra, et al. "Semantic segmentation on medium-resolution satellite images using deep convolutional networks with remote sensing derived indices." *2018 15th International joint conference on computer science and software engineering (JCSSE)*. IEEE, 2018.
[35] Wang, Hongzhen, et al. "Gated convolutional neural network for semantic segmentation in high-resolution images." *Remote Sensing* 9.5 (2017): 446.
[36] Mazza, Antonio, et al. "TanDEM-X forest mapping using convolutional neural networks." *Remote Sensing* 11.24 (2019): 2980.
[37] Chhor, Guillaume, Cristian Bartolome Aramburu, and Ianis Bougdal-Lambert. "Satellite image segmentation for building detection using U-net." *Web: http://cs229. stanford. edu/proj2017/final-reports/5243715. pdf* (2017).

[38] Huang, Gao, et al. "Densely connected convolutional networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[39] Abdani, Siti Raihanah, and Mohd Asyraf Zulkifley. "Densenet with spatial pyramid pooling for industrial oil palm plantation detection." *2019 International Conference on Mechatronics, Robotics and Systems Engineering (MoRSE)*. IEEE, 2019.

[40] Chaurasia, Abhishek, and Eugenio Culurciello. "Linknet: Exploiting encoder representations for efficient semantic segmentation." *2017 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2017.

[41] Khryashchev, Vladimir, et al. "Comparison of different convolutional neural network architectures for satellite image segmentation." *2018 23rd conference of open innovations association (FRUCT)*. IEEE, 2018.

[42] Chen, Liang-Chieh, et al. "Rethinking atrous convolution for semantic image segmentation." *arXiv preprint arXiv:1706.05587* (2017).

[43] Liu, Man, et al. "Comparison of multi-source satellite images for classifying marsh vegetation using DeepLabV3 Plus deep learning algorithm." *Ecological Indicators* 125 (2021): 107562.

[44] Liang, Wang, *Advanced Remote Sensing (Second Edition)*, Academic Press, 2020, Pages 1-57

[45] Audebert, Nicolas, Bertrand Le Saux, and Sébastien Lefèvre. "Semantic segmentation of earth observation data using multimodal and multi-scale deep networks." *Asian conference on computer vision*. Springer, Cham, 2016.

[46] Martins, José Augusto Correa, et al. "Semantic segmentation of tree-canopy in urban environment with pixel-wise deep learning." *Remote Sensing* 13.16 (2021): 3054.

[47] Boas, *Mathematical Methods in the Physical Sciences (Third Edition),* Wiley, 2006, p. 763.

[48] Haibo and Yunqian, *Imbalanced Learning: Foundations, Algorithms, and Applications (1st Edition)*, Wiley, 2013

[49] Lin, Tsung-Yi, et al. "Focal loss for dense object detection." *Proceedings of the IEEE international conference on computer vision*. 2017.

[50] Hutter, Frank, Holger Hoos, and Kevin Leyton-Brown. "An efficient approach for assessing hyperparameter importance." *International conference on machine learning*. PMLR, 2014.

[51] Grandini, Margherita, Enrico Bagli, and Giorgio Visani. "Metrics for multi-class classification: an overview." *arXiv preprint arXiv:2008.05756* (2020).

[52] Wolpert, David H., and William G. Macready. "No free lunch theorems for optimization." *IEEE transactions on evolutionary computation* 1.1 (1997): 67-82.

[53] Masters, Dominic, and Carlo Luschi. "Revisiting small batch training for deep neural networks." *arXiv preprint arXiv:1804.07612* (2018).

[54] Hess, Laura L., et al. "Wetlands of the lowland Amazon basin: Extent, vegetative cover, and dual-season inundated area as mapped with JERS-1 synthetic aperture radar." *Wetlands* 35.4 (2015): 745-756.

[55] Ferreira-Ferreira, Jefferson, et al. "Combining ALOS/PALSAR derived vegetation structure and inundation patterns to characterize major vegetation types in the Mamirauá Sustainable Development Reserve, Central Amazon floodplain, Brazil." *Wetlands Ecology and Management* 23.1 (2015): 41-59.

[56] Jadon, Shruti. "A survey of loss functions for semantic segmentation." *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, 2020.

[57] Wollmann, Thomas, et al. "Multi-channel Deep Transfer Learning for Nuclei Segmentation in Glioblastoma Cell Tissue Images." *Bildverarbeitung für die Medizin* 2018 (2018): 316-321.

[58] Bengio, Yoshua. "Practical recommendations for gradient-based training of deep architectures." *Neural networks: Tricks of the trade*. Springer, Berlin, Heidelberg, 2012. 437-478.

[59] Masters, Dominic, and Carlo Luschi. "Revisiting small batch training for deep neural networks." *arXiv preprint arXiv:1804.07612* (2018).