# A Patch-Based Deep Learning Approach for Land-Classification of Sentinel-2 Satellite Imagery

David Smith

Department of Computer and Information Sciences

University of Strathclyde, Glasgow

August 26, 2019

# Declaration

This dissertation is submitted in part fulfilment of the requirements for the degree of MSc of the University of Strathclyde.

I declare that this dissertation embodies the results of my own work and that it has been composed by myself. Following normal academic conventions, I have made due acknowledgement to the work of others.

I declare that I have sought, and received, ethics approval via the Departmental Ethics Committee as appropriate to my research.

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to provide copies of the dissertation, at cost, to those who may in the future request a copy of the dissertation for private study or research.

I give permission to the University of Strathclyde, Department of Computer and Information Sciences, to place a copy of the dissertation in a publicly available archive. (please tick) Yes [ ] No [ ]

I declare that the word count for this dissertation (excluding title page, declaration, abstract, acknowledgements, table of contents, list of illustrations, references and appendices is .

I confirm that I wish this to be assessed as a Type 1 2 3 4 5 Dissertation (please circle)

Signature:

Date:

# Abstract

The application of machine learning to the remote sensing field has produced many exciting new advancements. One of these is in land-classification: identifying the land-cover content present in satellite imagery. The use of deep neural networks has provided state-of-the-art accuracies for land-cover classification. This project experiments with deep learning architectures and novel image handling to methods to improve the classification accuracies achievable by Global Surface Intelligence, a geospatial analytics company. By implementing segmentation and patch-based training, we successfully show that by expanding their current methodology to consider spatial content, they can achieve increased classification accuracies.

# Acknowledgements

I'd like to kindly thank:

Dr Marc Roper, Mr William Wallace and Prof Ian Ruthven from the University of Strathclyde, for their helpful guidance and support. Alexey, Mark and Jelmer from Global Surface Intelligence for their invaluable assistance throughout the project. Georgi and Jerry from L3C AI Cloud for their technical assistance and cloud services.

# Contents


**1 Introduction** **1**

1.1 Background . . . 2

1.1.1 Remote Sensing . . . 2

1.1.2 Multispectral Imagery . . . 3

1.1.3 Global Surface Intelligence Ltd . . . 5

1.1.4 Problem Area . . . 6

**2 Related Work** **12**

2.1 Computer Vision . . . 12

2.1.1 Classifying Remote Sensing Data . . . 15

2.1.2 Land-Cover Classification . . . 18

**3 Research Methods** **21**

3.1 Research Questions . . . 21

3.2 Research Methodology . . . 21

3.2.1 What is the impact of considering local regions of pixels when performing image classification activities? . . . 21

3.2.2 What would be the feasibility of GSI adopting a patch-based approach based on the proposed methods? . . . 22

**4 Implementation** **23**

4.1 Area of Interest . . . 23

4.2 Segmentation Approach . . . 28

4.2.1 Segmentation of Scene . . . 28

4.2.2 Constructing Hierarchy . . . 30

4.2.3 Data Representation . . . 32

4.2.4 Model Selection . . . 32

4.2.5 Training and Validation Method . . . 35

4.2.6 Prediction . . . 35

4.3 Patches Approach . . . 35

4.3.1 Scene Preparation . . . 36

4.3.2 Data Representation . . . 38

4.3.3 Model Selection . . . 38
4.3.4 Training and Validation . . . 39
4.3.5 Prediction . . . 40
4.3.6 Transfer Learning . . . 42

**5 Model Performance** **46**
5.1 Segmentation Approach . . . 46
5.1.1 Training Performance . . . 46
5.1.2 Validation Performance . . . 49
5.1.3 Prediction Results . . . 50
5.2 Patch-based Performance . . . 52
5.2.1 Training Performance . . . 53
5.2.2 Validation Performance . . . 55
5.2.3 Prediction Results . . . 56
5.2.4 Scene Prediction . . . 58
5.3 Transfer Learning . . . 59
5.3.1 Training and Validation Results . . . 59
5.3.2 Prediction Performance . . . 61

**6 Discussion** **63**
6.1 What is the impact of considering local regions of pixels when performing image classification activities? . . . 63
6.1.1 Segmentation Approach . . . 63
6.1.2 Patch Approach . . . 64
6.2 What would be the feasibility of GSI adopting these methods into the current methodology? . . . 67
6.3 Future Work . . . 68
6.4 Conclusion . . . 69

**Bibliography** **69**

# List of Figures


1.1 Images from the European Space Agency satellite Sentinel-2. . . . . 2
1.2 Sentinel-2 Bands 2-7, 11 and a true colour image, from top left to bottom right. . . . . 3
1.3 Sentinel-2 Bands 4 and 8, combined to display NDVI, from left to right. . 4
1.4 The CORINE ground-truth data for a Sentinel-2 scene. . . . . 6
1.5 GSI land-classification methodology. . . . . 7
1.6 Demonstrating of information loss by decreasing resolution, left to right. . 8
1.7 Constructing a hierarchy of segments. For the segment tagging, every pixel in a given numbered region is assigned that number. . . . . 9
1.8 Forming overlapping tiles around individual pixels across the AOI. This is done for every pixel in the AOI. The borders will be padded with zero values so that all patches are the same size. . . . . 10

2.1 The main components of a CNN architecture. . . . . 13
2.2 Image segmentation into distinct classes [20]. . . . . 14
2.3 A hyperspectral stack of images, where each thin layer represents a band. Hyperspectral datasets are generally composed of 100-200 bands. . . . . 16
2.4 Upsampling CNN process to produce a segmented heatmap of input dimensions. . . . . 17
2.5 Samples from commonly used aerial imagery datasets, highlighting the contrast in image content between aerial and satellite imaging. . . . . 17
2.6 Comparison highlighting the difference between remote sensing and general image datasets. The first, ImageNet, has been used in the past for remote sensing image transfer learning tasks. . . . . 19

4.1 10m resolution Sentinel-2 Bands 2,3,4 and 8. . . . . 24
4.2 True-colour image of the Sentinel-2 tile. . . . . 24
4.3 Stacking the four individual bands to form our scene raster. . . . . 25
4.4 Final AOI created by merging bands 2,3,4 and 8, cropping and increasing the image contrast. . . . . 26
4.5 CORINE ground-truth data corresponding to the cropped AOI. . . . . 26
4.6 CORINE Land-Classification Colour Legend . . . . . 28

4.7 High scale and low minimum segment size segmentation. . . . . 29
4.8 Increasing segment scaling and size, from left to right. (Scale: 100, 300, 750, minimum size: 50, 100, 250. . . . . 29
4.9 Final AOI segmentation (Scale: 300, minimum size: 250). . . . . 30
4.10 Tagging segments based on mean pixel value. . . . . 31
4.11 MLP basic architecture. . . . . 34
4.12 Forming overlapping tiles around individual pixels across the AOI. . . . . 36
4.13 A 3x3 and 5x5 patch, with buffer sizes of 1 and 2 respectively, from left to right. As can be seen the buffer is taken outwards from the central pixel, outlined in bold. . . . . 37
4.14 Range of information offered by different patch sizes of a region of interest. 37
4.15 CNN architecture for the patch-based approach. Note: filter size was 3x3 throughout, other dimensions were as labelled. . . . . 39
4.16 Padding according to the patch dimensions around the AOI. The highlighted pixels along the top row of the AOI indicate the path of travel for the patch, representing the centre pixel sliding along the row with a stride of 1, and are distinct from the external padding pixels. . . . . 41
4.17 Commonly-used remote sensing image archives. [63] . . . . . 42
4.18 Sentinel-2 Bands 2-7, 11 and a true colour image, from top left to bottom right. . . . . 43
4.19 BigEarth CNN architecture. Similar to previous CNN, with the addition of MaxPooling layers after each convolution double. Notably, the final two convolutional blocks are of similar dimensions to the previous architecture. 44

5.1 Overfitting during Test Runs 2 and 4, from left to right. . . . . 47
5.2 Results from 20% drop-out regularization and training set increase to 100,000 pixels. . . . . 48
5.3 Optimizer changed to Stochastic Gradient Descent with a learning rate of 0.01. . . . . 48
5.4 Training performance on 3x3 and 7x7 patches, from left to right. Note: epoch difference due to an early-stopper callback. As the validation accuracy began to plateau, training was stopped. This was a useful way of optimising the training process. . . . . 54
5.5 Adjusted training performance, with a learning rate decrease to 0.001 and batch size increase to 512. . . . . 54
5.6 Training on 500,000 patches for 3x3 and 7x7 patch models. . . . . 55
5.7 CNN-3x3 label predictions, corresponding to the CORINE Land-Type colour legend shown earlier. . . . . 58
5.8 CNN-7x7 label predictions, also corresponding to CORINE colour scheme. 58
5.9 CNN-3x3 prediction probabilities, with black representing low confidence and white representing high confidence. . . . . 59

5.10 CNN-7x7 prediction probabilities, similar colour-scale. . . . . 59
5.11 Initial BigEarth training performance. . . . . 60
5.12 BigEarth training after drop-out regularization. . . . . 60
5.13 Reduced BigEarth-CNN architecture performance. . . . . 61
5.14 BigEarth-CNN scene-wide predictions. . . . . 62

6.1 CNN-3x3, CNN-7x7 and ground-truth over Swindon. . . . . 65
6.2 True colour image, CNN-3x3, CNN-7x7 and ground-truth over a small airport. 65
6.3 CNN-3x3, CNN-7x7 predictions and the true-colour image over a region of roads. . . . . 66
6.4 CNN-BigEarth predictions for the Swindon area. . . . . 66

# List of Tables


4.1 CORINE Land-Types, Labels and AOI Representation . . . . . 27

5.1 Training our MLP for 100 epochs on 10,000 segment-tagged pixels. . . . . 47
5.2 Run to run performance comparison for our MLP, MLP-Pixel and Random Forest Classifier baseline. . . . . 49
5.3 Class breakdown for our best-performing MLP model, tested on our test subset of 50,000 pixels from the segment-rank dataset. . . . . 50
5.4 MLP performance on testing subset of 50,000 pixels. . . . . 50
5.5 Prediction performance for the GSI Random Forest (RF) classifier on 400,000 extracted pixels from our AOI. . . . . 51
5.6 Overall metrics for the GSI Random Forest Regressor. . . . . 51
5.7 MLP prediction performance over 400,000 pixels. . . . . 52
5.8 Prediction metrics for the MLP. . . . . 52
5.9 Range of patches considered, with dimensions, pixel contents and label number per patch. . . . . 53
5.10 Patch Size vs Validation Accuracy . . . . . 53
5.11 Class breakdown for our best-performing CNN model, tested on our testing subset of 100,000 patches. . . . . 56
5.12 Overall CNN performance on our testing set of 100,000 pixels. . . . . 56
5.13 Prediction performance for the GSI Random Forest (RF) classifier on 400,000 extracted pixels from our AOI. . . . . 57
5.14 Overall metrics for the GSI Random Forest Regressor. . . . . 57
5.15 CNN-7x7 performance over the prediction set of 400,000 unseen pixels. . . 57
5.16 Overall metrics for the GSI Random Forest Regressor. . . . . 57

# Chapter 1

# Introduction

Image classification is the primary product of Global Surface Intelligence Ltd (GSI), an Edinburgh-based geospatial analytics company. Using machine learning and statistical methods, they extrapolate commercially valuable data and insights, from an array of satellite, drone and aerial imagery. Machine learning algorithms have seen a massive rise in popularity for remote sensing image analysis in recent years. The powerful performance of deep learning neural networks has been widely adopted in the field, due to their impressive capabilities at performing common remote sensing tasks such as preprocessing, detection and classification [10, 41]. Deep neural networks, whilst not currently in use by the GSI classification methodology, have shown state of the art performance for land-classification, the primary service offered by GSI. Therefore, deep learning is a natural area of interest for GSI. A key component of the current GSI classification pipeline is image processing: how the raw satellite imagery can be transformed and modified to greater reveal its features. This project aims to expand this methodology, experimenting with a range of image handling techniques which would allow the incorporation of deep learning to the GSI work-flow. Specifically, this project investigates the use of deep neural networks trained on both segmented and patch-based imagery, aiming to determine the feasibility of GSI adopting a deep-learning approach to their land-classification methodology.

## 1.1 Background

### 1.1.1 Remote Sensing

The capabilities of remote sensing technologies today have allowed satellite imagery to be increasingly accurate and readily available. Out of approximately 2000 active satellites currently in Earth's orbit, 35% of these are involved in Earth Observation (EO) activities [67]. This has given rise to a huge increase in the volume and velocity of satellite imagery produced, with some satellites producing terabytes of data daily [66].

Satellite imagery can have a substantial impact on many important activities here on Earth. It's effective use can assist scientists, governments and agencies in monitoring critical assets, protecting fragile ecosystems and warning against extreme weather events [32]. Organisations and companies can check the condition of valuable assets, farmers can remotely monitor the health of their crops and city planners can observe city growth [48, 49, 6]. It is evident that there is an exceedingly diverse range of applications for satellite imagery to be exploited.

**Figure 1.1:** Images from the European Space Agency satellite Sentinel-2.

A major application of satellite imagery is in land cover classification. Land cover is defined as "the observed biophysical cover on the Earth's surface" and extends to man-made structures and population centres [52]. The use of remote sensing data for land classification extends back to the mid-1940s [44, 1]. Inspired by the success of aerial reconnaissance in the First and Second World War, the agricultural industry began using aerial imagery to monitor farmland. In 1960, the first meteorological satellite was launched, sending back the infrared images of cloud patterns. Then in 1972, a true-colour imaging satellite was launched, the first of the US Landsat programme. Europe followed with the European Space Agency (ESA) Copernicus programme [45]. This marked the start of the democratisation of space-borne data, which had previously been difficult to obtain by members of the public. Sentinel-2, part of the Copernicus constellation, produces 5TB of freely available, high-resolution imagery of Earth's surface everyday [18]. By resolution, we mean the

spatial resolution measured, that is to say the smallest linear or angular separation between two objects that the sensor is sensitive to. [30]. Ranging from hundreds of metres to hundreds of centimetres, depending on the application the level of detail that can be achieved is impressive.

Evidently, satellite imagery is a useful tool for the task of land-classification. However, with the level of new data produced everyday, keeping track of changes on the Earth's surface is no easy task. To meet this challenge, progress in the fields of artificial intelligence and remote sensing have produced effective image classification methods. These can be used to classify the content of satellite imagery, and display surface characteristics that are invisible to the human eye. This is due to the common use of multispectral image sensors, which capture images by sensing for different wavelengths of light, including those outwith the visual region. Multispectral imagery can reveal a wealth of previously unrealised information about an area of land. We'll discuss how this works in the next section.

### 1.1.2 Multispectral Imagery

A multispectral image is a collection of several monochrome images of the same scene, each taken with a different sensor and representing a different region on the electromagnetic spectrum. Each image is referred to as a band. EO satellites operate sensors which are designed to capture a range of bands, allowing for a wider range of visualisation. We can see this in Figure 4.18, which shows a selection of the bands detected by ESA's Sentinel-2 satellite.

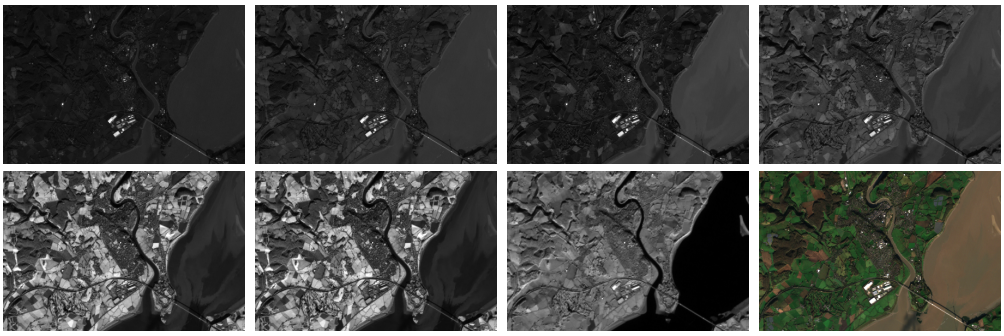


**Figure 1.2:** Sentinel-2 Bands 2-7, 11 and a true colour image, from top left to bottom right.

We can see in the top row, it is hard to identify regions in the image. The bottom row shows the borders of a town in the centre of the image, and the outline of surrounding fields. We can also see the outline of waterways more clearly. The first three images represent bands 2, 3 and 4. These represent the visual region of

the electromagnetic spectrum. Combining these gives a coloured image, seen in the last image. The rest of the tiles represent different regions of the spectrum. The contrasting differences between the tiles show the amount of information that can be revealed by detecting multiple spectra.

Each individual band gives a different representation of the scene. However, by combining certain bands, we can reveal even more information. By examining the proportions of difference between certain bands, we can give a new representation of the scene at hand. One of these is the Normalized Difference Vegetation Index (NDVI), which is calculated with Bands 4 and 8, as seen below:

$$NDVI = \frac{B08 - B04}{B08 + B04}$$

B04 represents the pixel values for Band 4, subtracted from B08, the pixel values in Band 8. We can visualise the outcome of this operation below.

Band 4 Band 8 NDVI Index

**Figure 1.3:** Sentinel-2 Bands 4 and 8, combined to display NDVI, from left to right.

Healthy vegetation reflects more near-infrared light, the wavelength of light picked up by Band 8, and absorbs more red light, that picked up by Band 4. Therefore by calculating NDVI we can monitor vegetation levels, where a high NDVI presence indicates healthy vegetation. We can see in Figure 4.18, the NDVI was calculated and plotted using a colour range, where black represents low NDVI and light pink represents high. We plot it over a colour range as a greyscale monochrome image is not very useful for visualization. It shows a large amount of fields on the left-hand side with varying levels of NDVI. We can clearly see the boundaries of the fields and the outline of towns, roads and waterways. Interestingly, we can also distinguish between less and more healthy patches of vegetation, shown by the differing shades of pink across the fields. We can see a bright patch at the lower centre of the image, showing a very-healthy cluster of vegetation. This is not seen in the other two bands.

This shows that inclusion of multispectral imagery can reveal previously unrealised features of a scene. Applications for this new information have been found across many industries, with clear uses in agriculture, forestry and urban planning apparent just from the scene discussed above. A company taking advantage of this potential

is Global Surface Intelligence Ltd.

### 1.1.3 Global Surface Intelligence Ltd

Applications for remote sensing data have risen in recent years, as the popularisation of big data has meant it's great potential can be exploited on a larger-scale. A rising awareness of these applications have produced an increasing demand for satellite-derived analytics [22]. A company taking advantage of this demand is Global Surface Intelligence Ltd (GSI), an Edinburgh-based geo-spatial analytics company.

GSI are a data services company which provides machine learning and predictive analytics on large and complex remote sensing datasets. They use aerial, drone and satellite imagery to derive key insights, intelligence and decision making tools, delivering them to customers across major industries such as agriculture, forestry, energy and more. Typical GSI services would be using satellite imagery to assist a farmer monitoring crop growth, identifying different tree species for a forestry commission, or detecting pipeline breaches for an oil company, for example.

The underlying activity in all of these services is classification. Effective classification is an integral part of the GSI work-flow: achieving accurate identification is vital in providing reliable advice to customers. In order to classify a customer's area of interest (AOI), GSI train machine learning models and use them to predict over the AOI. These models are selected and designed by GSI engineers.

**A brief machine learning overview:** Machine learning is a branch of artificial intelligence (AI). It addresses the fallacy in assuming humans have complete knowledge of the factors determining an event's outcome. Instead, machine learning uses computer systems to find and "learn" these determinants. For example, a traditional application of AI is email spam detection. Based on human observation, an AI system would be told what features in an email to look for, and, if spotted, to flag the email as spam. This makes the assumption that human observation captures all the possible features that distinguish an email as spam. The machine learning approach would be to expose the system to emails already identified as spam. This process of "training" allows the system to establish for itself the important features, which can then be looked for in new, unseen emails. Machine learning algorithms have proved effective at determining underlying features and patterns within datasets, ones that humans may miss [64, 53, 55].

Whilst the final performance of a machine learning model is weighted on the predictions it makes, these are determined by the quality of training. Training a machine learning model effectively relies on the quality of the training data. In the realm of

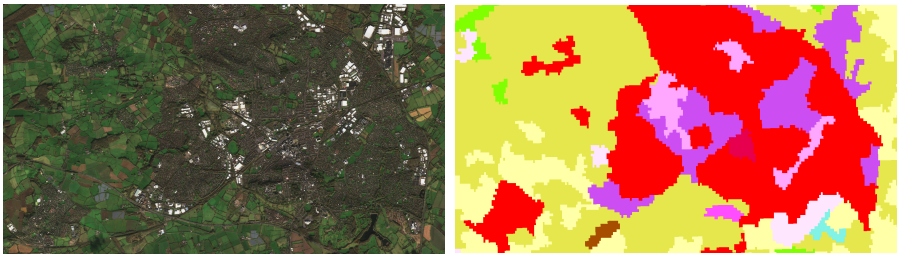
image-classification, the quality of the training data depends on how much information is contained in each image. To maximise this, an image can be manipulated to greater reveal its features. This allows the model to more easily connect these features to a given label, or in our case a given land-type, thus improving it's ability to classify that land-type whenever it next sees those features. This makes the manner in which the image is manipulated prior to training an important factor to consider.

### 1.1.4 Problem Area

Our project relates to the manner in which GSI process their imagery before using it to train machine learning models. This section will provide an overview of the current GSI land-classification methodology, before discussing some methods which this project proposes, forming the basis of an alternative methodology.

#### GSI Current Methodology

The current GSI training methodology is as follows. Satellite imagery is acquired from Landsat-8 and Sentinel-2 satellite sources. Normally it contains multispectral optical and radar imagery, representing both active and passive sensing data. The resolution of the imagery is between 10m and 60m, depending on source and sensor. The data is cleaned and filtered, with adjustments made for atmospheric corrections (e.g. cloud coverage, haze, snow). Ground-truth data is acquired. Ground-truth represents the actual land-cover of the area of imagery we are training on. GSI use CORINE land-cover maps, which are an established set of 45 different land-cover types. CORINE is used across Europe and is well-known.



Sentinel-2 scene CORINE Land-Cover Map

**Figure 1.4:** The CORINE ground-truth data for a Sentinel-2 scene.

Next, in order to gain a more generalised representation of both the imagery and the land-cover maps, they are both sampled to the same resolution. This means having to degrade the resolution of the imagery to that of CORINE, which is 400m. After resolution matching, the imagery can then be labelled pixel to pixel with CORINE.

Once labelled, training can begin on the 400m image pixels. The model learns the relationship between the pixel band values and the assigned CORINE land-cover label.

GSI use a machine learning algorithm known as a Random Forest, a collection of tree-based algorithms where each tree makes individual classifications for what a pixel could be. These are then aggregated, using the “wisdom of the crowd" to establish the overall highest-confidence prediction [9]. We will talk further about algorithm choice later in this report.

Finally, once the model has reached a certain level of training accuracy, that is to say how well the model has learned the training set, we can test it on new imagery. This would come in the form of a separate AOI, which could be determined by GSI or by a customer. We can visualise this end-to-end process below.

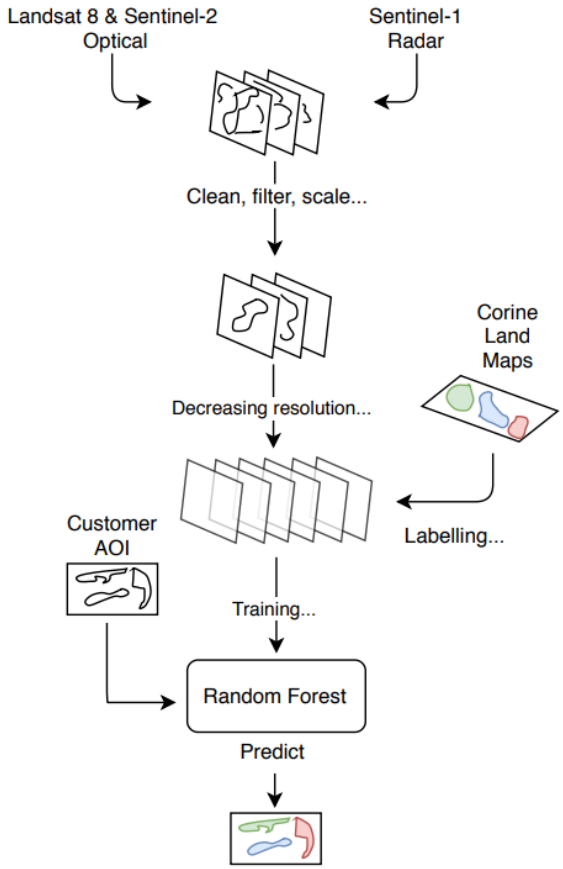



**Figure 1.5:** GSI land-classification methodology.

After training on 400m resolution imagery, the model takes in the pixels which make up the AOI. The model will then provide a prediction per pixel of the AOI.

The distinct feature of this method is the downsampling. There are merits to downsampling of this nature. By reducing the pixel size resolution to 400m, you effectively smooth the local region into one average value. Therefore the CORINE label which is matched with the 400m pixel represents the average class representation of that small region. However, this could be viewed as a rough approach. The pixel variability within a $400m^2$ area of 10m resolution imagery is high. By smoothing this to one value, a large amount of information would be lost. Any zones of heterogeneous pixels, which are common in satellite imagery, would be smoothed out and assigned the same label. Information loss due to changes in resolution can be visualised below in Figure 4.18.

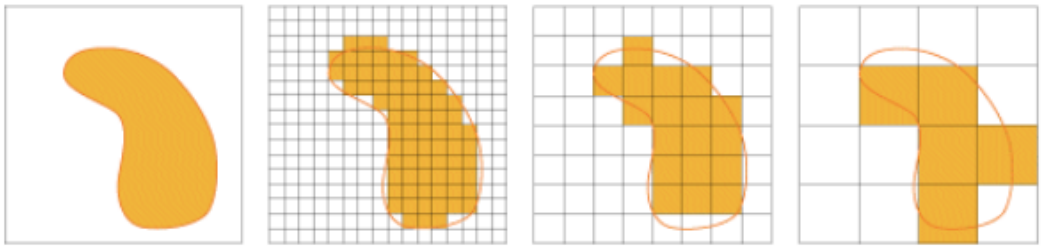


**Figure 1.6:** Demonstrating of information loss by decreasing resolution, left to right.

Whilst aggregating pixel values to averages over a region can provide useful information, they fail to inform on spatial content or texture within that region. These average would miss small details, which can often be key identifiers. For example, distinguishing between airport runways or urban green spaces. These land types are most identified by their intricate shapes, which would be overlooked when taking an average over the wider area.

### This Project

Considering that pixels in remote sensing data are normally surrounded by pixels of similar value, it would be logical to exploit this relationship when predicting individual pixels [13]. Also, determining how local regions of a scene belong to the global hierarchy of objects or patterns has long been identified as a key component of human visual perception [69]. Under these assumptions, this project proposes two main methods to incorporate into the GSI image classification methodology.

**1) Hierarchical Graph-based Segmentation**: This method uses a graph-based approach break an image into regions of pixel-similarity, or segments. Applying this to our AOI, we can produce a mosaic of distinct shapes. Each shape contains pixels of similar value. Recording the pixel values and count of each shape allows us to form a hierarchy of segments, allocating segments that contain similar groupings of pixels to the same level. A given pixel's *segment rank* represents the rank of

the segment it is contained within. Using this hierarchy, we can form a training set using band values and *segment rank* values, for every pixel in our AOI. With this additional information on a pixel's local region, we can achieve improved pixel classification results. This method operates under the assumption that two distant pixels which are of the same *segment rank* are likely to have similar values, and therefore belong to the same land-cover type.

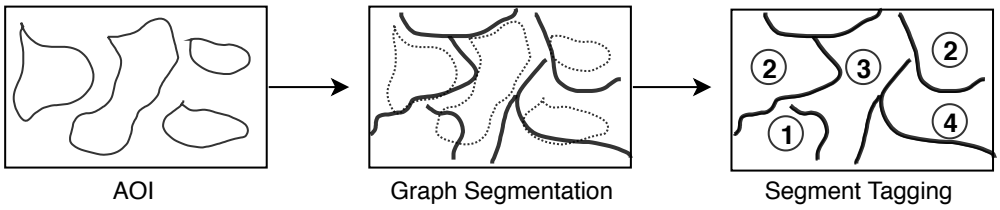



**Figure 1.7:** Constructing a hierarchy of segments. For the segment tagging, every pixel in a given numbered region is assigned that number.

This method uses graph-based segmentation to form similar pixels into regions. It involves plotting each pixel as a node on a graph, with connections linking to each of its neighbouring pixel nodes. A weight on each connection is set, depending on the similarity between nodes. Similarity is quantified as pixel intensity, colour or a similar attribute. These weights are then adjusted by an adaptive, greedy-algorithm, meaning local regions of nodes are prioritised when changing weights.

Unlike singular pixels, the segments provide shape and contextual information, as well as rich textural features. This extra knowledge can help exploit the benefit of considering local neighbourhoods, whilst still providing a classification for individual pixels.

**2) Patch-to-Pixel Classification**: The shape of the local region around a pixel can be a key indicator of its class. For instance, a grey pixel could represent an industrial warehouse, or be part of a road network. Without accounting for the shape of surrounding pixels, a confident prediction would be hard to make. Accounting for the shape of its local region would help identify distinguishable features, in this case the thin curvature of a road or rectangular warehouse structure, making it easier to classify. Under this assumption, this method involves partitioning our AOI into overlapping tiles of pixels, or "patches".

By selecting a pixel and forming a surrounding patch, we can see local shapes and texture which the central pixel is a part of. By associating the patch contents with the pixel, we can gain more information on the locations where the central pixel class appears. Using a predictive model, we will learn the relationships between the content of each patch and its centre. This information is then used to provide a

classification for that central pixel. Similar to segmentation, this method allows us to retain the high-resolution classification of a pixel-based approach, whilst improving overall accuracy due to the additional knowledge of spatial content. We can visualise this below.

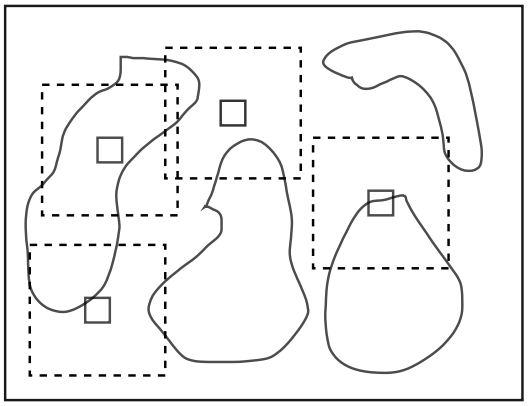


**Figure 1.8:** Forming overlapping tiles around individual pixels across the AOI. This is done for every pixel in the AOI. The borders will be padded with zero values so that all patches are the same size.

These methods may require selection of a new machine learning algorithm. Whilst the GSI methodology involves the use of a Random Forest Regressor, we may explore other options. When considering clusters of pixels, as opposed to one, the representation of the training data changes dramatically. An input no longer consists of a singular value, but multiple values. Additionally, the positioning of the values within one input relative to each other is a factor in that input's labelling. This increases the complexity of the relationship between an instance of the training set and it's label. For any model attempting to learn these relationships, the task becomes harder.

To meet this challenge, this project utilises "deep learning", a form of machine learning which utilises Artificial Neural Networks (ANN). ANNs are highly-suited to inputs of a multi-dimensional nature, such as our 2-dimensional patches. ANNs are biologically-inspired computing systems which are named after the structure of the brain. Using a vast network of artificial neurons, they can detect underlying features and patterns over a variety of inputs, without specifically being told what to look for. The networks are capable of "learning" the features themselves, picking up patterns which are hard to find manually [34, 68, 2, 50, 23, 15]. This project makes use of various ANN architectures when implementing the proposed methods. We'll discuss these in detail later in this report.

To conclude, our project proposes an alternative training methodology for the land-

classification activities undertaken at GSI. This methodology will be produced in a way that it is verifiable, by shared metrics, and comparable to the current GSI methodology. This project will also involve recreating the GSI methodology to form an imitation model to act as a baseline. This is required as we do not have access to the complete current methodology, so to act as a temporary comparison this project will implement a similar pixel-based classifier. This will be used during the development of our models as a soft comparison. We do, however, possess metrics from the full GSI methodology performance, gathered from results of a certain testing scenario. By recreating these conditions, we'll be able to make a direct comparison of the final version of our methods. Using the results of this validation, the project will be able to produce final recommendations for the proposed methods use at GSI.

# Chapter 2

# Related Work

This section contains a review of current image classification methodologies. We will first consider methods commonly found across general computer vision projects, before concentrating on remote sensing classification, such as including aerial imagery, and finishing with a review of satellite image land-cover classification.

## 2.1 Computer Vision

Computer vision is an extensive field with many techniques used to improve accuracy. With a range of different activities including object detection, scene classification, semantic segmentation or change detection, the main task is classification of some form. With so many different applications, there have emerged a wide range of methods and approaches one can take to image classification. We will discuss some relevant ones here.

Many computer vision projects involve the use of segmentation [61, 25, 46, 37]. Gould et al. incorporate object geometries into decision-making when splitting an image into regions [25]. They follow work done by Hoi et al., who imitate the way humans use local object orientation, perspective and positioning to understand the layout of a wider scene [28]. Using similar visual cues, Gould et al. form regions of comparably positioned objects in their image. They evaluate pixels based on local geometry, merging together regions of horizontal or vertical pixels (e.g. pavements and trees). This method achieves impressive segment purity: a metric describing the extent homogeneity within each boundary. This innovative approach relies on the presence of large, well-defined objects in the image material. The project also required a large amount of hand-labelling to identify key features, an expensive process.

The use of hand-crafted features for classification has largely been outdated by the arrival of "deep learning" [35, 70, 31]. Deep learning refers to the use of the ANNs mentioned earlier. An ANN is built of sequential layers, and with these layers being stack-able and of no fixed size, ANNs are easily customised. Due to this, there have emerged many different architectures in use today. Most computer vision projects involve the use of a convolutional neural network (CNN). Well suited to image classification, CNNs account for image structure by handling images as multidimensional inputs, and explicitly consider the spatial context of pixels through their convolutional layers. These layers contain a number of a filter matrices, where each matrix represents a pattern. These filters are "convolved" over the input, searching for their respective patterns, and produce "feature-maps" based on the presence of these patterns. We can visualise the basic layout of a CNN below.

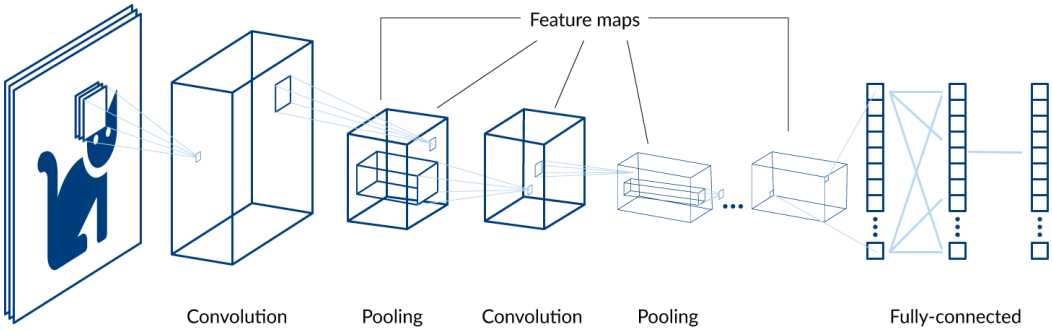



**Figure 2.1:** The main components of a CNN architecture.

Stacked layers of these maps allow feature vectors to be formed and learned by the network, which can then predict the input's class whenever it detects similar features. These feature maps form the input to each subsequent layer, and are normally shrunk as they move through the network, represented by the pooling blocks in Figure 2.1. This ability to form high-level representations of an image's structure mean CNNs are clearly a tool worth exploring in this project.

A project by Farabet et al. uses CNNs in conjunction with a similar style of segmentation as done by Gould et al, this time on an image dataset for autonomous vehicle training [20]. They use three separate CNNs, scaling the same image to three different degrees to form the different inputs. In parallel, they segment the original image into regions, creating a hierarchy of segments ranked by purity. Combining the feature-vectors from the high to low-level CNN outputs, they account for the different class densities, indicated by the segment hierarchy, by weighting low-level feature-vectors more in regions of low homogeneity. This method achieves impressive results over several benchmark computer vision datasets. This shows the suitability

of the technique to datasets containing large, distinct objects, with low mixing of classes; a common feature of computer vision datasets. Remote sensing imagery generally has high granularity, therefore our approach has to account for this.

**Figure 2.2:** Image segmentation into distinct classes [20].

Some projects use CNNs as the method of carrying out segmentation [10]. In a traditional CNN, the feature maps are “downsampled": a pooling operation is carried out to reduce the input size, aiming to determine the main features by taking the highest or average feature within a local regions on the maps. Eventually, the features get flattened into a final one-dimensional vector, which represents the class prediction. Here, the Deeplab-CNN differs by instead upsampling the prediction vector, returning its dimensionality and size until it returns to its original dimensions. This goes from a single or multiple labels for the image, determined by the one-dimensional vector, to producing a prediction for every pixel in the image, resembling a segmented heat-map. The model demonstrates leading results on the PASCAL VOC dataset, a benchmark computer vision dataset [19]. Again, these datasets contain images with small numbers of clearly defined objects. Producing heat-maps of larger objects is more straightforward than with fuzzy, overlapping objects containing multiple classes.

Several other works use segmentation with neural networks [61, 37]. Socher et al. describe the use of a recursive neural network (RNN), a deep learning architecture traditionally used in the field of natural language processing. They breakdown the image in a manner similar to sentence deconstruction; considering key segments and features in a similar fashion to how adjectives and nouns provide textual context. This approach relies on the presence of a small number of dominant features, whereas in our AOI there is a high number of similar features.

Liu et al. implement transfer learning to segment an animal image dataset. Transfer

learning accounts for the use of a neural network that has already been trained. It's network of neurons have been configured for use on a previous dataset. Using it to train on another dataset means the network may already be familiar with colours or certain shapes, which could be detected quicker and with more confidence in the new data. This relies on a level of similarity between the training and new dataset. In this case, they use a famous computer vision archive, ImageNet [14]. However, they also address a common theme which applies to all the works considered thus far, that a level of cross-domain similarity between the features used for training and the data one wants to predict must be present. Remote sensing data varies significantly when compared to computer-vision datasets. Whilst the methodologies considered are clearly applicable for general image classification, we will discuss their application to remote sensing data in the next section.

### 2.1.1 Classifying Remote Sensing Data

Image classification becomes more complex when used for the remote sensing field, mainly due to the presence of multispectral imagery, as well as the high level of detail and irregularity in the image [41, 7, 75]. Processing the extra image layers requires computational power and memory. The techniques still operate under the same principles, however must be performed on each band separately. We'll examine projects who implement some of the methods discussed so far for the remote sensing domain in the following section.

Ma et al. completed a comprehensive meta-analysis of algorithms used for analysing remote sensing data, with the focus on deep learning applications. They discuss some activities that can increase the image resolution, such as using autoencoders, another deep learning architecture [75]. Zeng et al. merge high-resolution panchromatic (black and white) imagery with low-resolution multispectral imagery, to enhance spatial resolution and retain multispectral information; an activity known as image fusion [29]. Liu et al. propose a similar technique, fusing high-resolution optical imagery with LiDAR data [38], to improve classification of the Vaihingen aerial image dataset [57]. Whilst our project has a single data-stream, the concept of fusing different spectral images together to create a hybrid of high-res multispectral content is useful for us, as our image sources are multispectral.

Another paper working with high-resolution imagery is a project by Zhu et al., which proposes the inclusion of "hyperspectral" imagery to provide a greater resolution of spectral information [76]. They also experiment with a Generative Adversarial Network (GAN), a deep learning architecture which aims to produce a more representative encoding of a dataset [24]. Using GANs in this manner prevented overfitting

occurring; a common occurrence with hyperspectral data due to the high feature count.

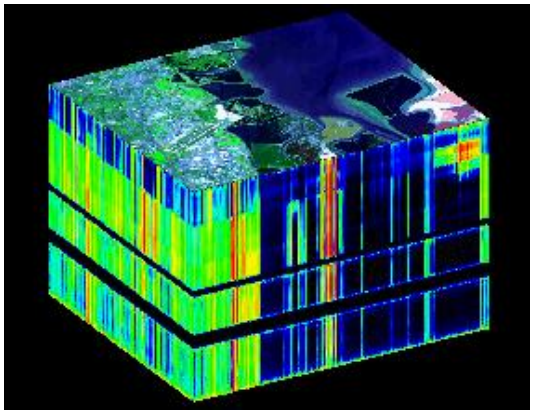


**Figure 2.3:** A hyperspectral stack of images, where each thin layer represents a band. Hyperspectral datasets are generally composed of 100-200 bands.

Similarly, in an attempt to address the trade-off between spatial-spectral content present in hyperspectral imagery, Hamida et al. propose a CNN architecture optimised for the 3D nature of multispectral imagery [4]. By treating each pixel as an $n$ x $n$ x $n$ volume, they implement 3D convolutions when filtering over the input. This is done based on the assumption that different bands may display different features for the same pixel. An innovative method of addressing the "curse of dimensionality" in the hyperspectral image context. Including more bands to our project would make this a worthwhile technique to experiment with.

Another image-handling approach, this time for high-resolution aerial imagery in the UC Merced dataset [72], uses a CNN for unsupervised learning [43]. By removing image labels, Marmanis et al. train a model with a greater knowledge of high-level image features, after allowing it to first explore them freely, before retraining with specified labels. Another CNN project by Liu et al. introduce the concept of upsampling CNN inputs to output a segmented heatmap of the image, discussed earlier by Farabet et al. They design a CNN with this "hourglass" architecture, aiming to detect cloud content in RGB remote sensing data [36]. We can visualise this CNN architecture in Figure 2.4.

Yu et al. propose an alternative preprocessing technique, which involves applying three operations to each image in their collection: flip, translation and rotation [74]. Applying these to each image and retaining the original increases the total size of the training set, and when applied to certain classes, can achieve a more balanced dataset. They go on to apply a CNN on the augmented data using benchmark classification datasets, to achieve state-of-the-art results. Data augmentation is a

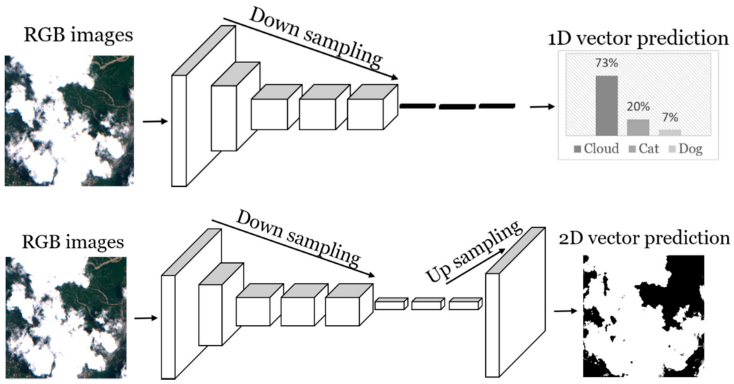



**Figure 2.4:** Upsampling CNN process to produce a segmented heatmap of input dimensions.

common activity in image classification [12, 5, 11]. Data augmentation is appropriate for use on small, misrepresented datasets, and whilst its advantages for increasing diversity and class-balance are powerful, it does not create new information for a given model, it only increases visual variability for existing imagery.

With further applications of deep learning found in object detection, land-use classification and semantic segmentation, classification has emerged the main activity within the remote sensing field [41] . This refers to both land-cover and land-use classification, with several papers employing CNNs on the land-use UC Merced dataset [7, 40]. In fact, the use of aerial imagery is common, forming many benchmark remote sensing archives (WHU-RS19, RSSCN7, NWPR VHR-10 and the Vaihingen dataset [26, 77, 16, 57]).

UC Merced Vaihingen WHU-RS19

**Figure 2.5:** Samples from commonly used aerial imagery datasets, highlighting the contrast in image content between aerial and satellite imaging.

Thus far, we have discussed image manipulation techniques used in remote sensing classification, primarily through the use of CNNs. These works propose suitable approaches to take in our project, however differ in some regards. Many of the works

are classifying aerial imagery. This level of resolution is not as common in satellite imagery. Whilst the methods are similar, our project has different requirements:

- Our data is strictly **multispectral and RGB satellite imagery**, not aerial or single-band images.
- Our application is strictly **land-cover classification**, not land-use, object detection, image fusion or scene classification.

We will now look at projects which are more aligned with the above criteria.

### 2.1.2 Land-Cover Classification

For classifying land-cover, the dominance of neural networks continue. In general, images with detailed spatial content, such as high-resolution satellite imagery, are well suited to CNNs [41]. As mentioned, considering one pixel at a time makes it hard to take advantage of this spatial content [62]. Song et al. state how classification of high-resolution imagery is more accurate on a patch-basis compared to pixel. Particularly for land-cover mapping, objects perform better as identifiers when compared to pixels [8, 17, 42]. This provides strong rationale for the techniques proposed by our project. We examine works who have implemented similar approaches here.

The concept of training models on patches is first introduced by Sharma et al. Using Landsat data at 30m resolution, they create patches by sampling individual pixels and forming a 5x5 surrounding tile. Assigning the centre pixel's label to the entire patch, they create a labelled dataset of patches. They train a CNN using this patch-based approach and run a comparison pixel-classifier, which trains on individual pixels [58]. They highlight the advantages of considering surrounding spatial content, shown in the superior accuracies of the patch-based approach. However there are some discrepancies: the patch-based and pixel-based classifiers were trained on different pixels, so a direct comparison cannot be made.

Song et al. address this shortcoming by proposing a CNN architecture designed for land-classification of multispectral imagery. Directly comparing this with the approach posed by Sharma et al., along with several pixel-based classifiers training on the same imagery [62]. They also include a "heterogeneity value", which describes the complexity and contextual property of the land-cover [59]. Specifically, it's value for a given pixel depends on the diversity of neighbouring pixels. A high value would mean a pixel is surrounded by dissimilar pixels. Song et al. go on to find that patch-based classifiers outperform pixel classifiers on pixels of low to medium heterogeneity. This is to be expected, as pixel classifiers are not concerned with local

regions. This explains the superior performance of pixel-wise classifiers for pixels with high heterogeneity. However, they conclude to state that overall accuracy was higher with the patch-based approach.

Another project recognising the advantage of patch-based training was the work done in creating the EuroSAT dataset, assembled by Helber et al. The EuroSAT dataset consists of 27,000 28x28 multispectral images created from Sentinel-2 data for the purpose of land-classification [27]. Helber et al. also implement a CNN to provide a baseline measure of the dataset potential, and achieve an impressive 98.57% testing accuracy, notably when training only on the datasets RGB bands. Therefore it is clear CNNs can achieve impressive results when training on patches. However, it is not mentioned how the model generalised on patches of satellite imagery outwith the dataset. Our project requires good generalisation, as the content GSI has to predict depends on customer requirements. Therefore, a large enough dataset is required to expose the model to a high variety and quantity of imagery.

ImageNet sample.

Remote sensing sample

**Figure 2.6:** Comparison highlighting the difference between remote sensing and general image datasets. The first, ImageNet, has been used in the past for remote sensing image transfer learning tasks.

Whilst EuroSAT provides a good starting point for patch-based training, in order to generalise well, CNNs need a high number of annotated images. To make up for the relatively small size of EuroSAT, Helber et al. employ a common tool: transfer learning on ImageNet. As previously mentioned, ImageNet (17 million images belonging to 22,000 classes [14]) is useful to allow the network to get an initial feel for general colours, shapes and textures. However, there are fundamental differences between the everyday items and scenes in ImageNet and remote sensing data, as clearly seen above.

To address the lack of publicly-available annotated high-resolution satellite imagery, BigEarthNet was created [63]. BigEarthNet consists of over 500,000 multi-labelled high-resolution Sentinel-2 images, and forms the largest archive of Sentinel patches

to date (2019). To demonstrate the advantage of training “from-scratch" on this dataset, i.e. without transfer learning, Sumbul et al. train a CNN on BigEarthNet patches, and compare the results with a state-of-the-art CNN, partially trained on ImageNet and then fine-tuned on BigEarthNet [63]. They achieve a 20% higher F1 score when training from-scratch. Whilst not being able to compare this approach to a pixel-classifier, it still gives us strong rationale to experiment with patches.

A project that has made good use of transfer learning, however, is an analysis of city slums by Wurm et al. Using a network trained on QuickBird [33], Wurm et al. perform semantic segmentation on Sentinel-2 data. By creating a hierarchy of patch sizes to train on, they find that transfer learning can be beneficial for Sentinel-2 imagery, if trained on data of similar resolution and content [71]. Similar to Farabet et al.'s multi-scale three-way CNN discussed in Section 3.1, Wurm et al. experiment with different patch-sizes. They find a larger patch size gives better predictions, however this is only shown for the prediction of inter-city slums. This also could explain the impressive results after transfer-learning on QuickBird, which is a very-high-resolution (VHF) 0.65m imaging satellite, suitable for detecting complex urban landscapes [39]. However, patch-size experimentation would be an appropriate comparision to make during our work, as it could yield a range of results.

Another study with patch-size was undertaken by Tong et al., classifying high-resolution imagery from the Chinese Gaofen-2. They proposes a novel hybrid pixel-patch approach. They perform hierarchical segmentation on an image, then after sorting and classifying the imagery by patch, they integrate the patch classification with the segmentation boundaries through a major-voting strategy [65]. They consider the most frequently appearing class within each segmented region, and use this to inform the patch classification for that corresponding region. If multiple patches are present in a segment, then only the region pixels contained by that patch are used. They also experiment with different patch and segmentation scales. They concluded that segmentation scale depends on the resolution and scale of objects in the training image. Contrary to Wurm et al., they state a smaller patch size yielded optimum results, due to each patch having a single label meaning more information was lost as you increased the patch size.

Evidently, there is a lot of potential to experiment with the concepts of patches and segmentation in the GSI land-classification methodology. In both general computer vision projects, aerial imagery and satellite multispectral imagery, these techniques have been applied to considerable success, in a range of applications. We have seen different forms of both approaches, including variable patch sizes, different spectral bands, innovative metrics and a hybridisation of the two techniques. Therefore there is a clear rationale for GSI to experiment with these methods.

# Chapter 3

# Research Methods

## 3.1 Research Questions

- What is the impact of considering local regions of pixels when performing image classification activities?
- What would be the feasibility of GSI adopting a patch-based approach based on the proposed methods?

## 3.2 Research Methodology

### 3.2.1 What is the impact of considering local regions of pixels when performing image classification activities?

This question will be answered in two parts. First, we must understand the performance of image classification when only considering individual pixels. This involves examining the current GSI methodology in detail, and understanding its performance across the range of different features in our AOI. There may be some land-types a pixel-based approach excels at, compared to our methods. There may also be indicators of how to improve on it. Using standard machine learning evaluators, such as classification reports, we will be able to understand the performance of the current methodology. Additionally, we'll implement a similar baseline classifier, that resembles the current GSI methodology. This is because this project did not have access to the full working GSI methodology, so in order to inform our model development, a similar model was recreated.

We'll also be able to communicate with GSI engineers on the key design choices

that went into the legacy method's design, as well as any other comments on its performance. By maintaining a good communication link and making the most of the wealth of relevant knowledge that exists at GSI, we will be able to determine how to improve on the current approach.

After understanding the impact of pixel-only classification, we can properly assess the effects of implementing our proposed methods. Using satellite imagery provided by GSI, we will experiment with the segmentation and patch-based techniques by experimenting with models trained on this data. Then, we'll train our model on this augmented data. This will require ground-truth data for all of our imagery, so that the model can learn the relationship between the image content and its land-types. By evaluating the training and prediction performance of our machine learning model, using similar metrics to the evaluation of the current methodology, we can then fully understand the impact of the proposed methods..

### 3.2.2 What would be the feasibility of GSI adopting a patch-based approach based on the proposed methods?

A key factor in evaluating this project's methods will be estimating the feasibility of efforts to integrate the proposed methodology to the actual GSI work-flow. To check the viability of the method, it's performance will have to be evaluated using shared metrics, to compare to the results of the current method. We can easily achieve this by using our answers from the previous questions. These will provide a quantifiable basis, as well as overall comments, on which GSI can base their decision on.

If the proposed method passes the validation procedures, then GSI will likely examine the method further, making adjustments and changes to allow it to be streamlined into the GSI work-flow. GSI would be using the method as part of a back-end system designed to classify land-cover in satellite imagery. It would **not** be used as a customer interface, user experience, or application. The method is strictly a method of training machine learning models for a specific application.

If GSI choose not the use the proposed method, they may still examine the work done, and extrapolate any useful information and research. This could be used to inform changes they themselves go on to make to the current methodology, or any work done developing similar software. Or, it may be the work done by this project simply indicates to GSI that this problem area is not worth exploring, or that resources are better allocated in a different manner, or a different area. It could validate their existing methodology, highlighting its advantages and areas that are hard to improve.

# Chapter 4

# Implementation

This section describes how our project approached the research questions posed in the previous section. Our main goal was to determine the viability of our proposed training methods. To do this, it was logical to use imagery provided by GSI. This would allow for a more direct comparison afterwards.

## 4.1 Area of Interest

The scene provided by GSI can be seen below. It represents a tile captured by the Sentinel-2 satellite. We had access to the three different resolutions captured by Sentinel-2: 60m, 20m and 10m. Across these, Sentinel captures multispectral imagery spanning 13 different bands.

We decided to use the 10m resolution imagery, or bands 2, 3, 4 and 8. For experimenting with different pixel-handling methods, it was logical to use bands which were all of the same resolution. This would make processing and handling easier. Additionally, the 10m resolution imagery provided the greatest level of detail, and would ideally be the most informative to use. This level of resolution was deemed an appropriate level to extract information on both the region and biome-level [30]. Therefore this design choice was both practical and would access the greatest level of information per pixel.

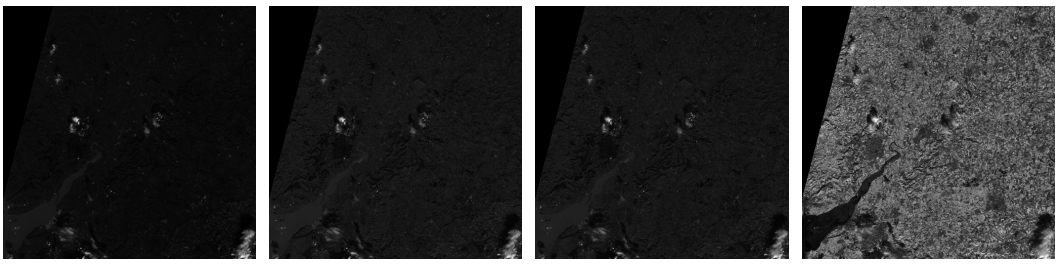


**Figure 4.1:** 10m resolution Sentinel-2 Bands 2,3,4 and 8.

**Figure 4.2:** True-colour image of the Sentinel-2 tile.

We can see the 10m resolution bands of our scene in Figure 4.1, and a true-colour image, compiled from bands in 2,3 and 4, in Figure 4.3. Due to adjustments made for the inclination of the Sentinel-2 orbit, we can see the effects of the image cropping, shown by the black section on the left-hand side. Spanning an area of approximately 10000km$^2$, Figure 4.3 displays South-West England, with Bristol and Swindon as the main urban centres in the centre and lower-right of the scene, respectively. The rest of the image is predominantly fields, vegetation and water.

The true-colour image only incorporated three of the four bands we had. In order to consider all bands, we had to merge the four bands into a single object. A commonly used datatype for geographical information sciences is a *raster*. A raster is a matrix of cells, where each cell position corresponds to a position in an image, map, or

other spatial representation. The cell value represents some real-world phenomena, such as colour, elevation or temperature. Whilst each individual band represented a raster, rasters can be combined into multi-dimensional objects, and in our case the scene would be combined into the following dimensions:

$$SceneRaster = height * width * band\text{-}count$$

Here, height and width represent the dimensions of the individual band rasters. Each band was stacked, in the order of 2,3,4 then 8. We can visualise this process below.

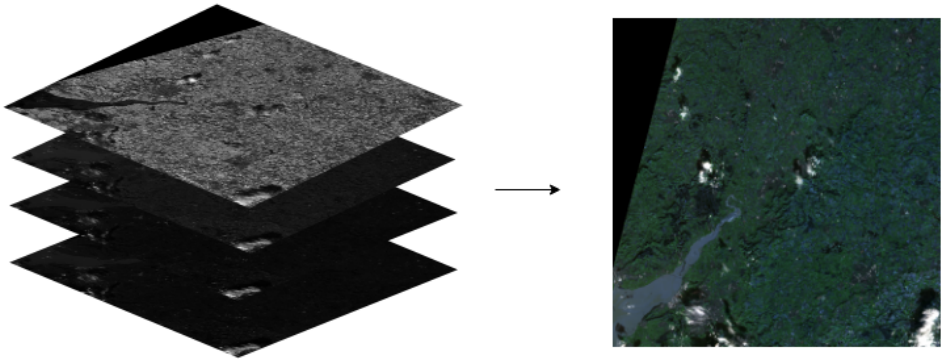


**Figure 4.3:** Stacking the four individual bands to form our scene raster.

Once the bands were merged, we wanted to reduce the overall size of the scene. At 10980 by 10980 pixels, it was a large, memory intensive object. In their readable TIFF format, the four bands required 1.6GB of storage space. To reduce this to a more manageable size, we decided to crop the scene. Using coordinates provided by GSI, the scene was reduced to an AOI of approximately 440km$^2$, or 40km by 11km. The AOI was also brightened to improve scene visibility. We can visualise this process below.

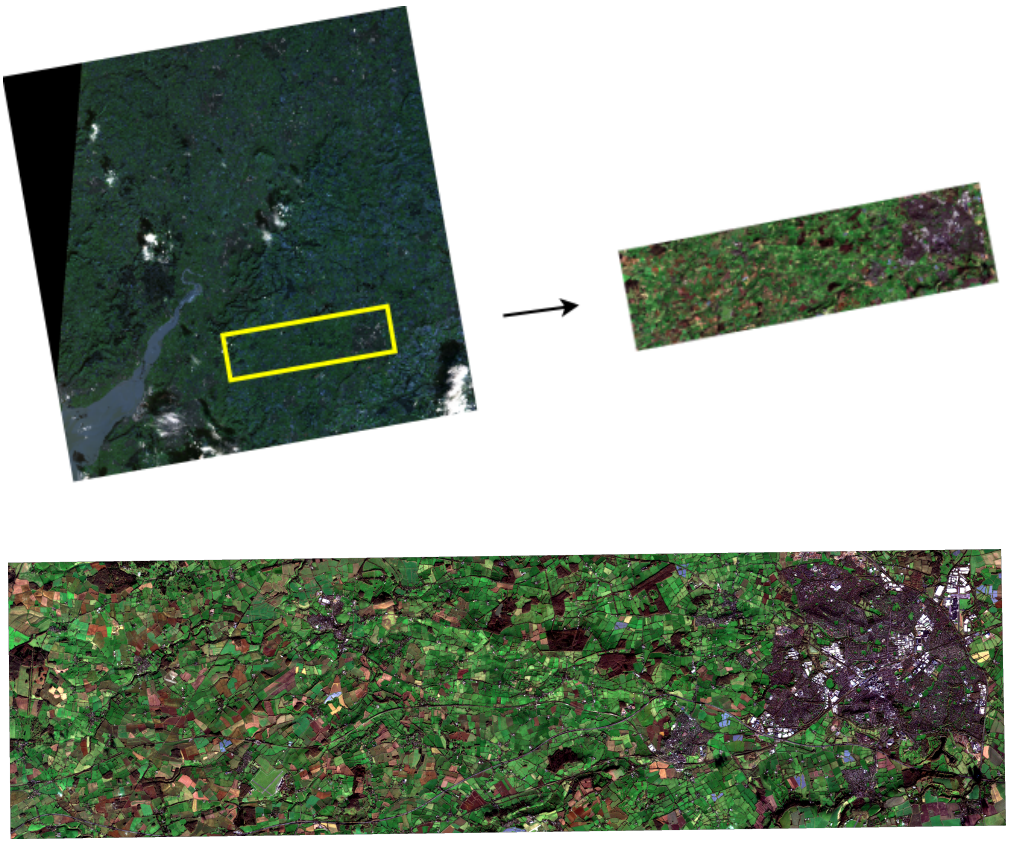


**Figure 4.4:** Final AOI created by merging bands 2,3,4 and 8, cropping and increasing the image contrast.

We required ground-truth for the new AOI. This would also be provided by GSI. It was practical to use the same ground-truth classification system, CORINE Land-Cover, as the current methodology, to allow for a direct comparison to be made.

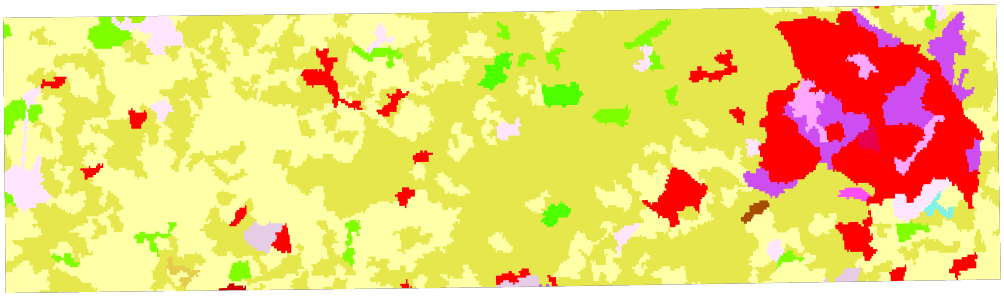


**Figure 4.5:** CORINE ground-truth data corresponding to the cropped AOI.

We now had our finalised AOI in the form of a three-dimensional, 4400x1100x4 raster, representing the AOI width, height and band count. Every pixel in the AOI represented a vector of length four, with one value for each band.

The CORINE ground-truth data was a two-dimensional raster, where each pixel value, or colour, represented a land-type classification, and corresponded to an actual geographical coordinate. In order to make pixel to pixel comparisons, both the AOI and the ground-truth had to be on the same coordinate reference system (CRS). A CRS is a standardised coordinate-based global system used to identify and locate spatial entities. Establishing a uniform CRS was essential if we wanted to compare a location in the AOI with it's position in the ground-truth. Our project used the World Geodetic System (WGS), which was established in 1984, and is the global standard for cartography and satellite navigation [54]. We'll refer to it as WGS84 from here on.

The AOI contained 17 of the 45 CORINE Land-Cover classifications. We can observe the class content and their representation in the scene below.

| **CORINE Land-Cover Classification** | **Label** | **% AOI** |
|---|---|---|
| *No data* | 0 | 1 |
| *Continuous urban fabric* | 1 | 4 |
| *Discontinuous urban fabric* | 2 | 12.5 |
| *Industrial or commercial units* | 3 | 4 |
| *Road and rail networks and associated land* | 4 | 2.5 |
| *Airports* | 6 | 0.5 |
| *Dump sites* | 8 | 3 |
| *Construction sites* | 9 | 4 |
| *Green urban areas* | 10 | 1.5 |
| *Sport and leisure facilities* | 11 | 4 |
| *Non-irrigated arable land* | 12 | 22 |
| *Pastures* | 18 | 27 |
| *Land occupied by agriculture and vegetation* | 21 | 3 |
| *Broad-leaved forest* | 23 | 6 |
| *Mixed forest* | 25 | 2 |
| *Transitional woodland-shrub* | 29 | 1 |
| *Water bodies* | 41 | 2 |

**Table 4.1:** CORINE Land-Types, Labels and AOI Representation

At first glance, it is clear there is an imbalanced class representation. Some land-types cover over 20% of the scene, and many represent less than 5%. This is more clearly visible below in Figure 4.6. Class imbalance could skew our model's understanding of the overall dataset, making it more biased when predicting the classes it has seen more of. We'll discuss the effects of class imbalance later on in this report.

Once we had both our AOI and ground-truth prepared, we could begin implementing our methods. We'll discuss this process in the next section.

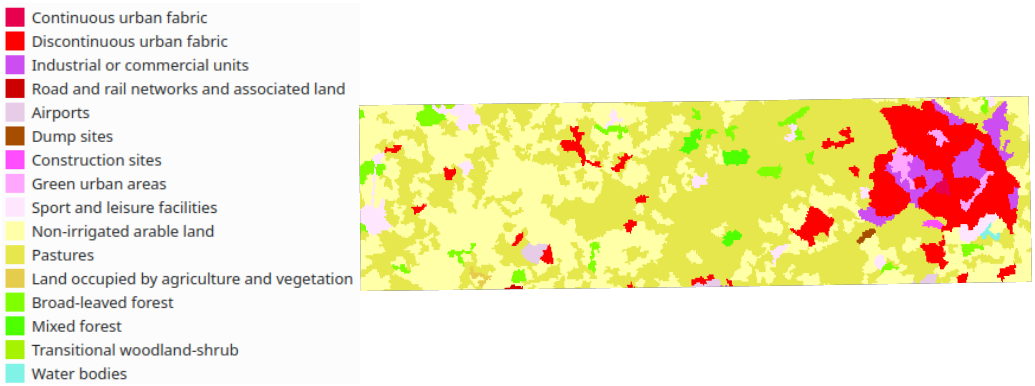



**Figure 4.6:** CORINE Land-Classification Colour Legend

## 4.2 Segmentation Approach

After cropping, acquiring ground-truth and establishing the correct coordinate system, we could first begin by segmenting the AOI. Using the graph-based similarity measure, discussed in Chapter 1, we could apply this to our merged AOI, breaking it into segmented objects. We can see the result of this below.

### 4.2.1 Segmentation of Scene

We can see the first segmentation of our AOI below in 4.7. The pixel band values were aggregated to a single value per pixel. This value represented a pixel intensity, or colour. Graph-based segmentation plots each of these values as a node on a graph, and assigns connections between each node, with the strength of the connection correlating with how similar the pixel colours are. Based on this similarity measure, pixels joined by strong connections are merged into an object, or segment. Certain parameters dictated the threshold for the minimum strength required for two pixels to be merged, as well as the minimum pixel count for a segment. These parameters were the *minimum segment size*, and segment scale. Let's see how initial settings for these parameters affected the AOI.

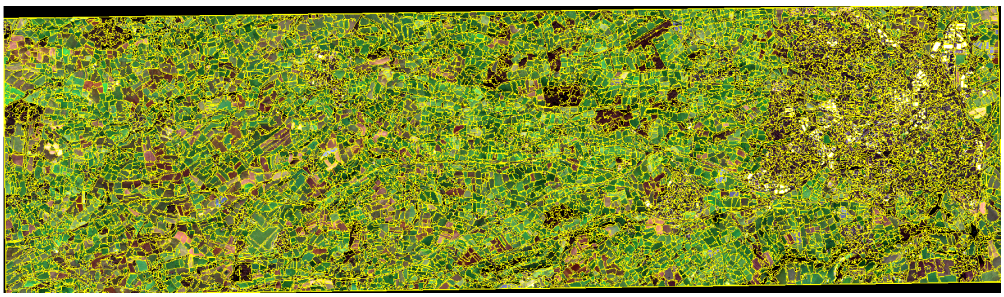


**Figure 4.7:** High scale and low minimum segment size segmentation.

Figure 4.7 shows a lot of activity. Our minimum segment size was set too low. We increased this, and reduced the level of scaling; increasing the number of segments relative to their size. We can see the effects of this below over the Swindon region of the AOI.

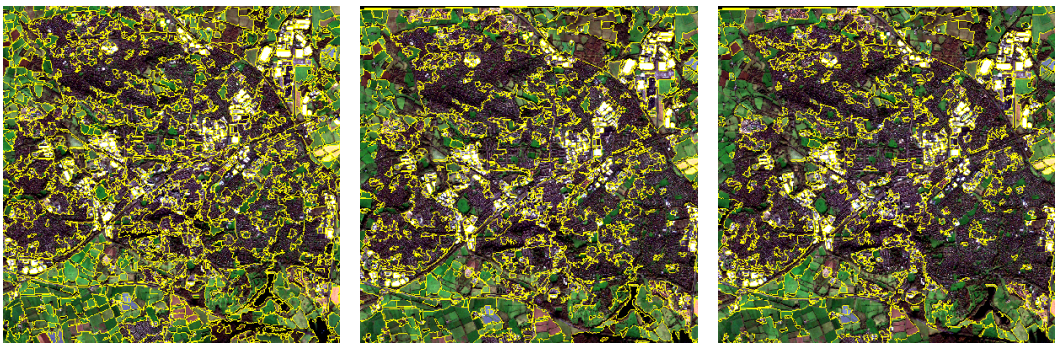


**Figure 4.8:** Increasing segment scaling and size, from left to right. (Scale: 100, 300, 750, minimum size: 50, 100, 250.

We can see in Figure 4.8 the effect of increasing segment scale. As our overall goal was to create a hierarchy of segments, ideally we would have a broad range of segment across the hierarchy. It would be optimal to have enough segment types to clearly differentiate between, but not so many as to saturate the hierarchy with too many overly-similar levels. Additionally, we wanted to capture large groups of the same class in one segment. Areas such as clusters of forests, or urban blocks, would ideally be contained within the same segment. We can see the final settings for the AOI segmentation below.

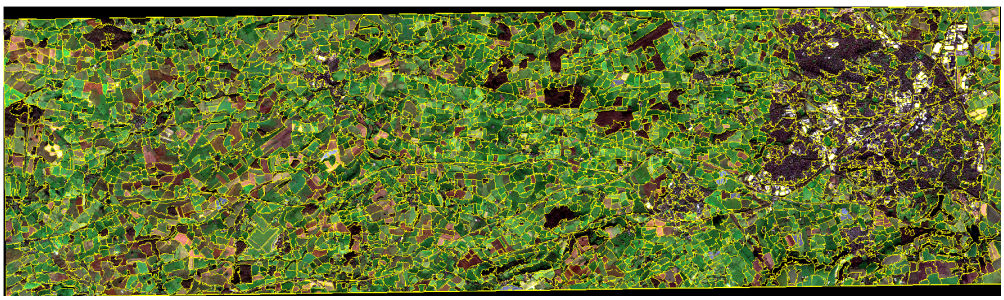


**Figure 4.9:** Final AOI segmentation (Scale: 300, minimum size: 250).

We can see how the final settings break up the scene in a more visibily appealing manner. Areas of forest and field are segmented from one another, however similarly coloured fields are grouped together. The outline of Swindon is captured, with the main residential neighbourhoods north and west of the centre grouped into their own segments. This seemed an adequate level to fix our parameters for. After segmenting the scene, the next step was to build our segment hierarchy. To do this, we had to quantify individual segments based on their local pixel content, and create a global scaling system.

### 4.2.2 Constructing Hierarchy

Our first step to constructing this segment hierarchy was by calculating the mean pixel value in each segment. We constructed a dataframe of our segments, with each row a “polygon" shape object. As polygons are made up of straight lines, we could define a segment as a collection of coordinates based on where these lines connected. Using a statistical Python package, we extracted the mean pixel value, as well as the pixel count, for every polygon. Next, we calculated the $10^{\text{th}}$, $25^{\text{th}}$, $50^{\text{th}}$, $75^{\text{th}}$ and $90^{\text{th}}$ percentile mean values across all polygons. Finally, by performing interval comparison, we could group our polygons by mean pixel value, establishing which percentile each polygon was closest to.

Based on which percentile a polygon was nearest to, a segment variable, or *segment-rank* was assigned. All polygons which had a mean pixel value below 194, the $10^{\text{th}}$ percentile, were assigned a 1. Those between 194 and 239, the $25^{\text{th}}$ percentile, were assigned a 2. This was done until every segment had a *segment rank* value between 1 and 6. We can visualize this process below.

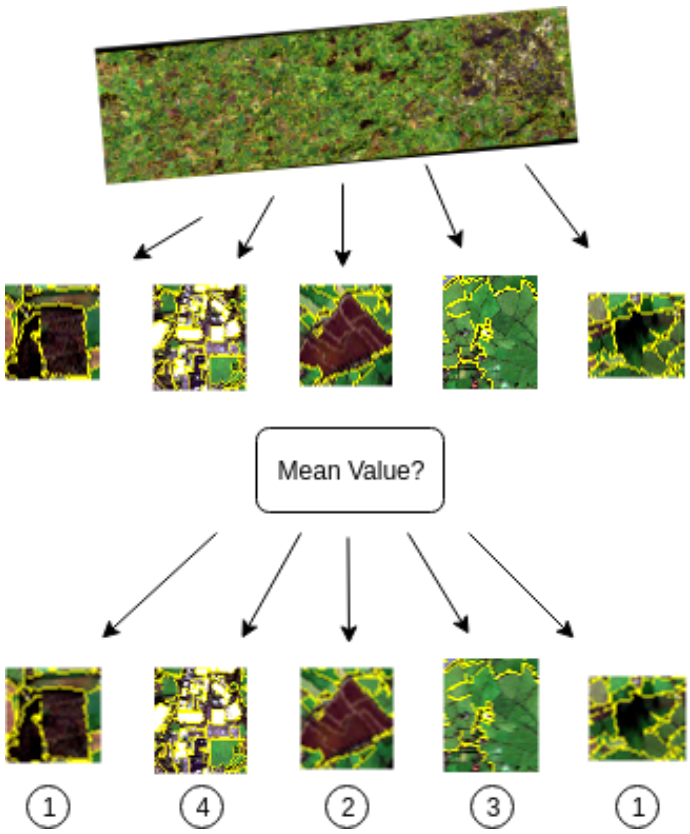



**Figure 4.10:** Tagging segments based on mean pixel value.

Notice in Figure 4.10 how the far left and right segments are assigned the same value. This is meant to represent how segments of similar colour are given the same *segment rank*.

Using our segmented scene, we would now begin constructing a dataset of pixels. To remind the reader of the overall goal, we were aiming to improve classification accuracy of these pixels, based on knowledge of surrounding pixels. In addition to existing knowledge of the pixel band values, we could now add a *segment rank* value, for every pixel in the scene. This represented the segment rank of the segment that the pixel was contained by. The segmentation allowed us to account for local regions of similarly coloured pixels, and apply this information to the pixel-level. This approach operated under the assumption that pixels of similar colour were likely to have similar *segment rank*, and therefore were likely to belong to the same land-type class. A model learning the relationship between the pixel band values and land-type would ideally be more confident to predict if it had another variable connecting the two.

For additional information, we also calculated the NDVI index, discussed in Chapter 1. This would be a tertiary indicator of a pixel's class. As a reminder, areas of high

NDVI indicated the presence of high vegetation content, which is a distinguishing feature of land-types such as cropland or forests.

### 4.2.3 Data Representation

After segmenting our image, producing a segment hierarchy and calculating NDVI, we could now construct our dataset. We had our AOI in the form of a three-dimensional raster, with ground-truth data for the same positioning and located along the same CRS. This meant we could access the same pixel position from both rasters with a single set of longitude and latitude coordinates. Randomly selecting $n$ coordinates meant we could extract that coordinates band values from the AOI, and using the same position, extract that pixel's land-type from the ground-truth. Then, once we had a pixel's band values, we could calculate the NDVI value, based on the values for band 4 and 8, as discussed earlier.

Using this method, we constructed a dataframe of pixels, where a single row represented one pixel, and the features were band 2,3,4 and 8 values, *segment-rank*, and NDVI Index. Each row also had a label, representing the CORINE Land-Cover classification.

Once our dataset was created, we wanted to train a machine learning model on the relationship between a pixel's band, segment-rank and NDVI values, and it's CORINE label. Our approach would be validated if the use of the *segment-rank* variable, an indicator of the local pixel neighbourhood, had a positive affect on classifying the pixel's label.

### 4.2.4 Model Selection

Due to the nature of our data representation, the CNN models discussed earlier would not be appropriate, due to the lack of any dimensionality within our dataset.

We also wanted to create our soft-baseline, as mentioned at the end of Chapter 1. This would provide a pixel-based baseline classifier to weight our developing models against. The set-up of our dataset meant that inclusion of the *segment-rank* dictated whether or not we wanted to incorporate spatial information to the scene or not. This meant that by not including *segment-rank*, there was zero spatial element. We decided to clone our dataset, but removing the *segment-rank*, as the dataset for our baseline measure. This would provide an indication of the effect of including local image information against solely relying on the pixel band values.

**Baseline** We first selected our baseline algorithm, the model which would not account for *segment-rank*. We decided to use a Random Forest Classifier, the same algorithm as used by GSI. However, the variant GSI used was for regression, where a continuous output is predicted. For our task, we wanted the classifier to output a categorical value, representing a distinct CORINE land-classification. Random Forest Classifiers are commonly used in remote-sensing classification [3, 47, 21]. They have been found to be powerful classifiers of remote sensing data, due to their ability to handle high data dimensionality and multicolinearity. They use decision trees on random subsets of a dataset, and by aggregating each tree classification, produce a final prediction. By increasing the number of trees and thus voting power, accuracy can be increased.

**Segmentation Model** Due to it's high suitability to image classification, we still wanted to explore deep-learning, however could not use a CNN. We selected a basic neural network for this task, a multi-layer perceptron (MLP). These were similar to CNNs, except did not contain any convolutional layers. They consisted of sequential layers of "fully-connected nodes", where each node represents a neuron, as discussed earlier. Each node in a layer is connected to every node in the following layer, a "dense" network of connections. We can visualise the architecture of our MLP below.

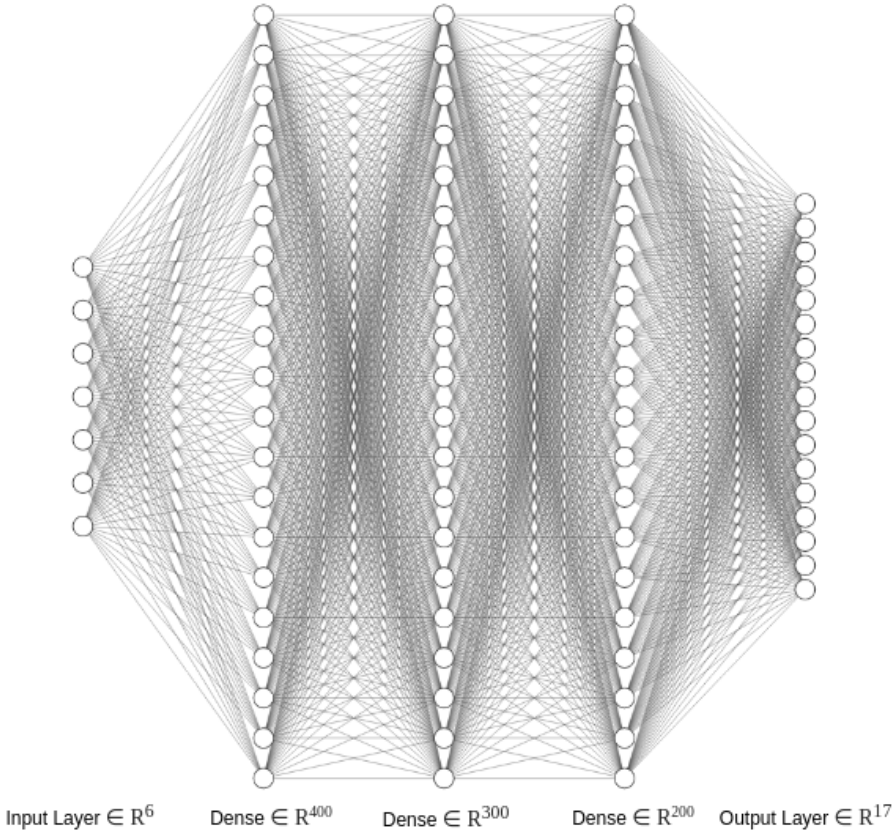



**Figure 4.11:** MLP basic architecture.

Our input feature vector was an array of length 6, representing our four band values, pixel NDVI and segment-rank value. These values would be passed through the four dense layers, with each node in the every layer representing a number between 0 and 1. The node count for the three dense layers was 400, 300 and 200, respectively. These layers would transform the array of values as it passed through the network. The nodes will determine which value has an effect on the final label through backpropagation. This refers to the process of updating the node values, and their individual biases, by accounting for the difference between the prediction and actual value. The final layer outputs an array of length 17, representing the class count of the AOI. The array is made up of 17 probabilities between 0 and 1, each representing the confidence of prediction for that class. As there is only one classification per feature vector, these probabilities will sum to 1. This makes our problem a multi-class classification task, as each input belongs to one class from a set of different classes. Therefore, a confident prediction for one class would affect the prediction probabilities for the remaining classes.

**Model Evolution** There exists a wide search space of parameters for neural net-

works, with many factors influencing the quality of their outputs. This project experimented with a selection of different parameters, and the evolution of the model performance is discussed in detail in the next chapter.

### 4.2.5 Training and Validation Method

The next step was to begin training the above models on our pixel dataset. Machine learning models require both training and validation data. This is so that learning can take place iteratively; training on one subset of the data, making predictions on another, and repeating the process. One pass of this process is known as an epoch. A separate subset of the data is held back before this split, in order to test the final model after a certain number of epochs have passed.

We used a 60/20/20 training, validation and testing split, meaning that for a given dataset of 100,000 pixels, the model would train on 60,000, predict 20,000, calcuate the error through backpropagation, and repeat the process, for however many epochs were set. At the end of training, it would predict 20,000 new, unseen samples. The model could then be evaluated based on the result of the final predictions.

As well as model configurations, we wanted to explore the effect of pixel-count, feature-importance, and different degrees of segmentation. These will be discussed in the performance section in the next chapter.

### 4.2.6 Prediction

To make a direct comparison, we wanted to use our model to predict in a similar manner to the GSI methodology. We would do this be recreating the same testing conditions for which the GSI-supplied metrics were achieved. We can examine the results of this in the next chapter.

## 4.3 Patches Approach

Our patch-based approach was the second method include local and spatial information to the classification process. This method involved partitioning our AOI into overlapping square tiles, or patches. We can see a quick reminder of the process below.

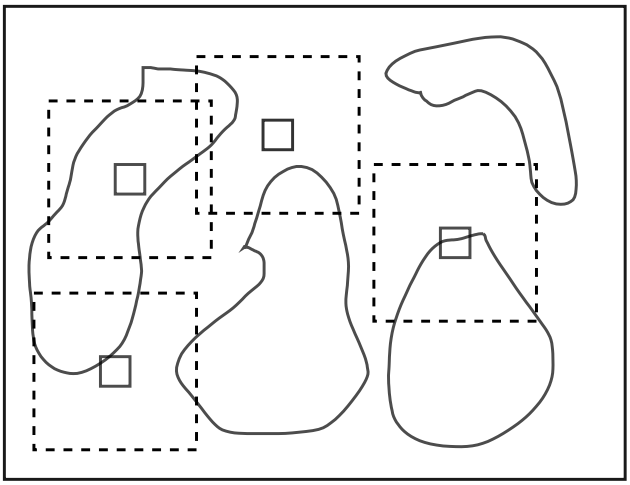


**Figure 4.12:** Forming overlapping tiles around individual pixels across the AOI.

Our aim with this approach was to train on a patch surrounding a centre pixel, learn the relationships within the patch that determines it's label, and to be able to provide a prediction for an unseen centre pixel after scanning it's patch. We'll first discuss how we extracted the above patches, before moving on to model choice and configuration, and then our training and validation methodology.

### 4.3.1 Scene Preparation

#### Patch Construction

In order to construct our patches across the AOI, we first had to extract centre pixels. In an identical manner as the segmentation approach, we gathered our pixels and labels by coordinate. However, instead of only extracting the pixel, we applied a *buffer* zone. This was applied by adding and subtracting the buffer value along the height and width of the AOI, starting from the position of the centre pixel. This formed a square of pixels. We can visualise this below.

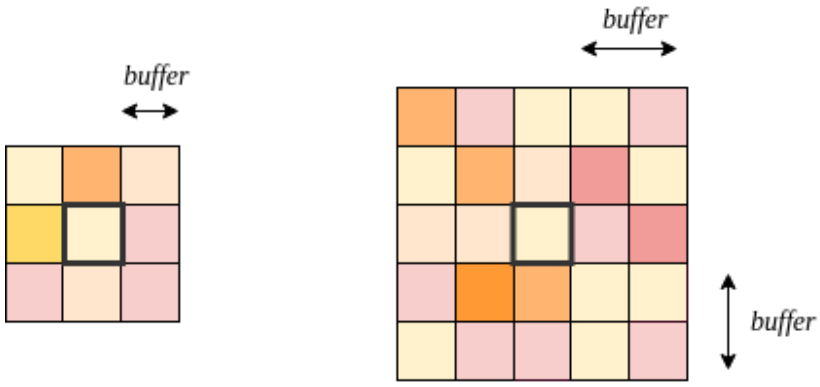



**Figure 4.13:** A 3x3 and 5x5 patch, with buffer sizes of 1 and 2 respectively, from left to right. As can be seen the buffer is taken outwards from the central pixel, outlined in bold.

The buffer size dictated the amount of information contained by a patch. This method used the information learned from patches to make a prediction for a pixel. Therefore, the amount of information in each patch could be a critical for prediction accuracy. We can get some context for how patch-size could affect predictions in our AOI by examining some sample patches below.


**Figure 4.14:** Range of information offered by different patch sizes of a region of interest.

The above patch selection was extracted from the highlighted zone in Figure 4.14, from a close-up of a neighbourhood in Swindon, taken from our AOI. It shows a small park nested within a residential area. The ground-truth for the area would be classed as a *green urban space*. When forming patches around the pixels in this area, we can see the potential range of information that a model would be trying to connect with this label. The 3x3 patch shows only the green area. The 7x7 patch shows the beginnings of its boundaries, and the 21x21 fully highlights the shape of the park, indicated by the surrounding urban fabric. Therefore, one would think a model trained on this 21x21 patch would provide the most confident prediction, as it clearly shows the parks position within the block. This would imply that the 3x3 patch would cause the model to incorrectly classify the park, possibly as a field or vegetation. However, there is also the possibility of overloading the model with too much information. The 21x21 patch may be classified as urban fabric, due to the inclusion of the surrounding homes. The 7x7 patch may be classified as a field, with its faint boundary assumed to be hedging. The 3x3 patch may represent the correct clustering of green pixels which are commonly found in parks, leading to a correct classification.

Evidently the patch-size would be a factor to consider when experimenting with this method. We'll explain the remainder of the patch approach next.

### 4.3.2 Data Representation

We extracted our patches as discussed. An individual patch was stored as a multi-dimensional array, with the dimensions *height* by *width* by *band-count*. These were kept in a dataframe. Along with the patch arrays, the CORINE labels for every pixel in each patch were also stored. Although we were predicting the centre label only, it was useful to keep the whole patch label content. This informed us on the homogeneity of each patch, shown by the label diversity along one row. Using the dataframe format meant we could easily eliminate poorly represented classes and outliers within the data.

### 4.3.3 Model Selection

For this approach, the model used was a CNN. Our inputs were three-dimensional patches, with a high spatial content. Our CNN used three repeating convolutional blocks, where each contained two convolutional layers, with 3x3 sized filters. The number of filters doubled at each layer. After six repeating convolutional layers,

the input was fed to a fully-connected dense layer of 1000 nodes, and finally to an output layer of size 17. We can visualize our final architecture below.

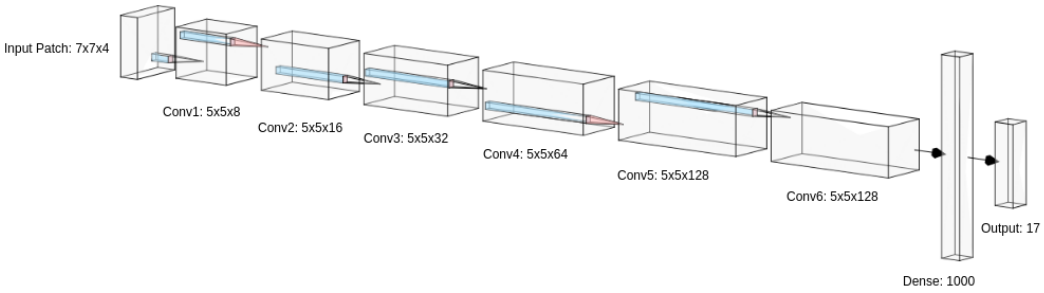



**Figure 4.15:** CNN architecture for the patch-based approach. Note: filter size was 3x3 throughout, other dimensions were as labelled.

Figure 4.16 helps illustrate the depth of the network. By the final convolutional block, we have two sequential stacks of 128 feature maps. Each feature map in each block represents a different pattern in the patch. We can also see how the size of the pattern is kept constant throughout.

This architectural style of having repeating convolutional blocks was first established in VGG-16 [73, 56], a renowned image classification CNN. The creators coined the double-convolutional and pooling block pattern, with a 3x3 filter size and doubling the feature-map count at each block. Largely inspired by this approach, our model has some unique features. Our image inputs were much smaller compared to the VGG-16 inputs, at 120x120 pixels. The patch inputs to our model were too small to be shrunk further, meaning our network removed any pooling operations. As shown by Sharma et al., who implemented a similar network for an approach training with 5x5 patches, this architecture is still feasible without pooling layers.

Similar to the previous method, there was also a range of parameters we wished to explore. As well as the patch-sizes previously mentioned, we would be interested in the effect of patch-count. Additionally, as is a recurring theme with neural networks, there was a wide range of parameters to configure the network by. These were the learning rate, which controls the degree to which the neurons are updated after backpropagation, the batch-size, the amount of patches considered at each network pass, and the fully-connected node count.

### 4.3.4 Training and Validation

Our training approach was similar to the segmentation method. We had constructed a dataframe of patch arrays and their corresponding labels. Using similar train/-

validation/test divisions, we divided our dataset into proportions of 60/20/20. To reiterate, throughout training, the model would be detecting the patterns and features within the patch arrays, and how they relate to it's **centre** pixel. This is one of the key features of the methodology, and it allows us to predict at a level of 10m resolution.

Throughout training we experimented with a range of patch-counts. As before, we'll examine the effect of this, and also a range of model parameters, over the next chapter.

### 4.3.5 Prediction

For prediction, we were aiming to both predict over the entire AOI, and also to provide a direct comparison to the available GSI metrics. Predicting every pixel in our AOI would involve partitioning the scene into patches. These patches would have to match the input size, i.e. the patches, which our final model architecture trained on. The model would predict along each row of the AOI, one patch at a time, and moving along the row pixel by pixel. To predict every pixel, including the border pixels, the AOI would need to be padded with empty pixels along the boundary of the scene. This was due to neural networks requiring a fixed size input. Without padding, we would not be able to form a complete patch around the border pixels. We can visualise this procedure below.

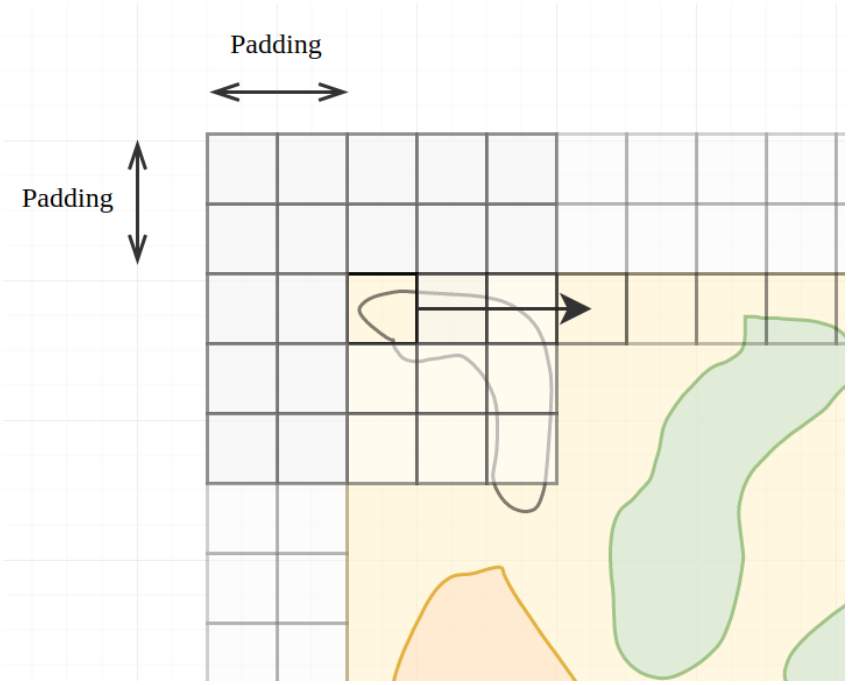



**Figure 4.16:** Padding according to the patch dimensions around the AOI. The highlighted pixels along the top row of the AOI indicate the path of travel for the patch, representing the centre pixel sliding along the row with a stride of 1, and are distinct from the external padding pixels.

We can clearly visualise the sliding patch mechanism above. When it slides along the border pixels, the patch hovers over the empty padding pixels. These allow the input to remain constant, however predicting these pixels could be more difficult, due to the null pixel values not adding any information on the centre pixel. However, this could also be viewed as a clear indicator of a border pixel, if there were a correlation between border pixels and a certain land-type.

After training our model on the extracted patches, discussed previously, we could use the above method to predict a CORINE label for every pixel in the AOI. By storing these predictions in raster format, we would be able to visualise the predictions spatially, with the true-colour scene or ground-truth below. This would be the final stage of our patch prediction approach. However, an issue with this approach was that we were extracting our training patches from the same AOI that we would be predicting over. Although the training patch count was small versus the number of AOI patches we'd be predicting, around 4 million, this could still be viewed as an unreliable approach. A key indicator of a machine learning model's performance is its ability to generalize onto completely unseen data. Therefore, this gave us clear rationale to explore external training sets. We'll discuss this approach in the next section.

### 4.3.6 Transfer Learning

As discussed in the literature review, transfer learning can be a powerful tool in machine learning. As a general rule of thumb, the more data a model is exposed to, the better the training performance. Using a model that has been pre-trained on a separate dataset means we can utilise the knowledge gained from that dataset and apply it to our own.

In this context, we aimed to train a model using an external patch-based dataset, and use that model to predict over our AOI. The results of this would give a good indication of the feasibility of a patch-based approach, especially as these patches came from new and unseen imagery. It would provide a different angle on the approach, and potentially alert GSI to new data archive to train future models on.

**BigEarthNet**

We implemented our transfer-learning patch approach by exploring the recently compiled BigEarthNet dataset [63]. BigEarthNet, referred to as BigEarth from now on, was a solution to the previous lack of any large-scale annotated remote sensing archives. Previous transfer learning projects in the remote sensing realm have generally used ImageNet, or aerial and drone image sets. Compared to high-resolution satellite imagery, there are vast differences in the content, detail and spectral variety of these datasets and satellite imagery. We can see some common remote sensing datasets below in Figure 4.17.

| Archive Name | Image Type | Annotation Type | Number of Images |
|---|---|---|---|
| UC Merced | Aerial RGB | Single Label [2] | 2,100 |
| | | Multi-Label [11] | 2,100 |
| WHU-RS19 [3] | Aerial RGB | Single Label | 1,005 |
| RSSCN7 [4] | Aerial RGB | Single Label | 2,800 |
| SIRI-WHU [5] | Aerial RGB | Single Label | 2,400 |
| AID [6] | Aerial RGB | Single Label | 10,000 |
| NWPU-RESISC45 [7] | Aerial RGB | Single Label | 31,500 |
| RSI-CB [8] | Aerial RGB | Single Label | 36,707 |
| EuroSat [9] | Satellite Multispectral | Single Label | 27,000 |
| PatternNet [10] | Aerial RGB | Single Label | 30,400 |

**Figure 4.17:** Commonly-used remote sensing image archives. [63]

As we can see above, the majority of the archives are RGB, with the exception of EuroSat. However, as with the rest, EuroSat has a relatively small number of images. For effective classification with deep learning, large training sets are usually required.

BigEarth consists of 590,326 Sentinel-2 image patches, taken between June 2017 and May 2018. These patches were taken from 125 Sentinel-2 tiles. Sentinel-2 tiles represents an individual image captured, which are 10,000 $km^2$ in area, and projected in WGS84 projection. These patches cover 10 different countries across Europe, showing a diverse range of terrain and seasons. We can see a selection of BigEarth patches below.

**Figure 4.18:** Sentinel-2 Bands 2-7, 11 and a true colour image, from top left to bottom right.

A notable feature, and one which makes it very attractive to this project, is that BigEarth is labelled using the CORINE Land-Classification system. This means its images are annotated with the same labelling system as used for the ground-truth of our AOI. However, a single BigEarth patch is associated with multiple labels. According to Sumbul et al., the BigEarth creators, the number of labels per patch ranges from between 1 and 12. A machine learning model training on BigEarth would be required to predict multiple labels per patch, making this a multi-label classification problem. Multi-label classification has distinct requirements on model configuration and parameter settings. Our project was concerned with multi-class classification, where each pixel only had one label. Additionally, the range of labels present in BigEarth represented the entire 45 CORINE Land-Classsification list, whereas our AOI only contained 17. Therefore if we were to attempt to utilise BigEarth for use with our project, the archive had to be augmented.

### Dataset Pruning

After downloading BigEarth, the dataset had to be altered. The data arrived with two separate files, listing images which had poor visibility due to cloud and snow coverage. These were removed from the archive. Next, we wanted to eliminate those patches with labels belonging to any of the 45 CORINE labels that were not included in the AOI class representations. This would significantly reduce the dataset but was necessary for our project. Additionally, we wanted to limit the labelling per patch to one label. This would also drastically reduce the amount of information available from BigEarth. However, it was deemed necessary to allow experimentation with transfer learning to occur.

### Data Representation

After pruning the dataset, we were left with 294684 patches, each belonging to 1 of the 17 classes in our AOI. Finally, we wanted to standardise the number of bands for all of our approaches, so only bands 2,3,4 and 8 were used for each patch. After merging these bands for each patch, and storing the merged image as a three-dimensional array, we could construct our training set. This was made up of 294684 120 by 120 by 4 patches, associated with a single label. The patches were also scaled to values between 1 and 255.

### Model Selection

For our model selection, it seemed logical to use a similar model for our previous patch-based approach. However, the BigEarth patches were 120 pixels by 120 pixels. Comparing these to the patch sizes extracted from our AOI, these were much larger. Therefore this gave the opportunity to experiment with the pooling layers mentioned previously. Our model architecture can be seen below.

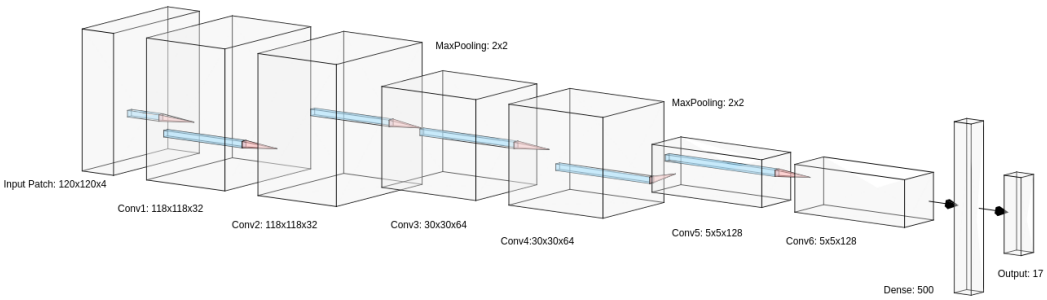



**Figure 4.19:** BigEarth CNN architecture. Similar to previous CNN, with the addition of MaxPooling layers after each convolution double. Notably, the final two convolutional blocks are of similar dimensions to the previous architecture.

### Training and Validation

We could now begin training our second CNN on the modified BigEarth dataset. Using similar train, testing and validation subsets as before (60/20/20 split), we partitioned our dataset.

### Prediction

Prediction was also carried out in the same manner: we implemented the sliding window mechanism to feed our AOI into the BigEarth CNN as rows of patches. Predicting on each patch attributed that prediction the centre pixel of the given patch. This time it would be more computationally expensive. More padding was required, and the prediction run time increased. Having to consider so much more information, 120x120 pixels worth, compared to the smaller patches of the first

CNN, meant the model took longer making each prediction. We'll have a look at the results of these predictions and the other methods next, in Chapter 5.

# Chapter 5

# Model Performance

To recount, we've introduced and explained both of our approaches, patch-based and segment-based, with details given on the workings of each, the model used, and training, validation and prediction methods. This method will display the results of each approach, aiming to illuminate the performance variance between the two approaches, our baseline and GSI-supplied metrics.

## 5.1 Segmentation Approach

### 5.1.1 Training Performance

After segmenting our scene, constructing the segment hierarchy and defining our MLP model, we could begin training. We wanted to determine the optimum scene segmentation. First, we had to determine our controlled parameters, on which to compare the segmentation parameters to. We fixed the pixel count at 10,000 pixels, and kept our MLP architecture constant.

We can examine the effect segmentation scale had on the training and validation accuracy of our MLP model below. These were the accuracies after training for 100 epochs, with a batch-size of 64 pixels and a validation split of 20%, as mentioned before.

**Table 5.1:** Training our MLP for 100 epochs on 10,000 segment-tagged pixels.

| Test Run | Scale | MinSize | Valid Acc (%) | Test Acc (%) |
|---|---|---|---|---|
| 1 | 50 | 50 | 45 | 43 |
| 2 | 150 | 100 | 48 | 46 |
| 3 | 300 | 100 | 54 | 52 |
| 4 | 750 | 100 | 47 | 42 |
| 5 | 1000 | 100 | 40 | 39 |

We can see that Test Run 3 had the best accuracies. We could now fix our segmentation parameters to these settings. Next, aiming to improve the overall accuracy, we could increase the number of pixels. Before this, we'll visualise some of the above runs to gain a wider picture of the training process.

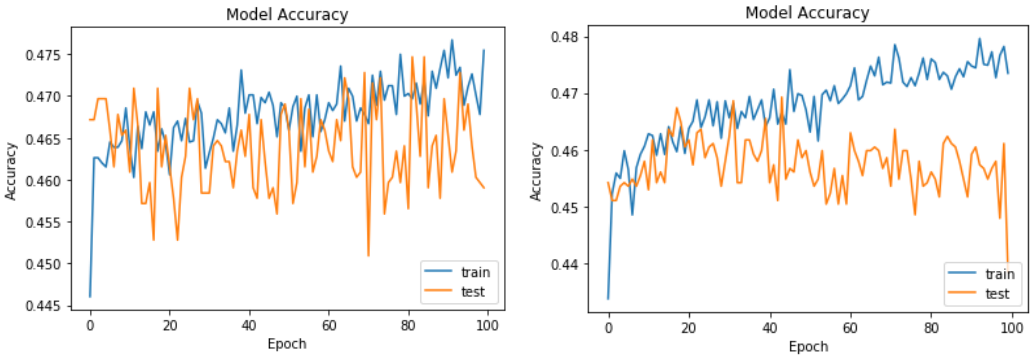



**Figure 5.1:** Overfitting during Test Runs 2 and 4, from left to right.

As we can see, there is a severe degree of overfitting taking place. This is shown by the extreme noise in the curves, as well as the deviation between the training and testing curves in Test Run 4. This tells us that the model may benefit from some structural changes before training on more pixels. This led us to implement drop-out regularization between layers, which cancels out a given ratio of inputs moving between layers. This is a computationally cheap tool, which forces the network to determine weight changes which are directly improving the model accuracy. We can see the effect of this in conjunction with an increase in pixel count next.

Using our final segmentation parameters of 300 and 100 scaling and size, we updated our MLP to include 20% drop-out between each of the dense layers. Let's see the performance of the new network when training on 100,000 pixels.

We can see the immediate affect of regularization in negating overfitting. This is shown by the proximity of the training and testing curves. We can see by the sharp descent of the loss, and the plateau of the accuracy, that the model is struggling to fully understand our dataset. Furthermore, the testing accuracy actually remains

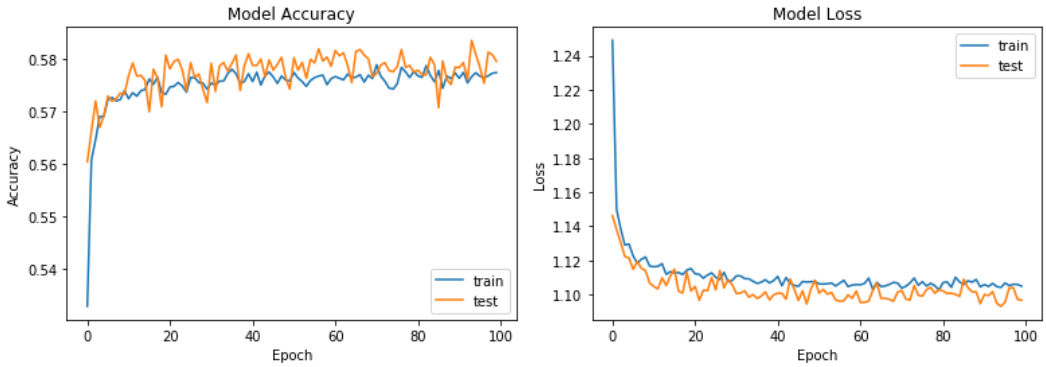



**Figure 5.2:** Results from 20% drop-out regularization and training set increase to 100,000 pixels.

above the training accuracy for a large portion of the run. It is clear the model is struggling to properly understand the data. The choice of optimizer, Adam, may be confusing learning, as it alters the learning rate between batches. Let's try using Stochastic Gradient Descent, which keeps the rate of learning constant, yet still encourages dynamic learning through exposing the model to randomly selected samples.

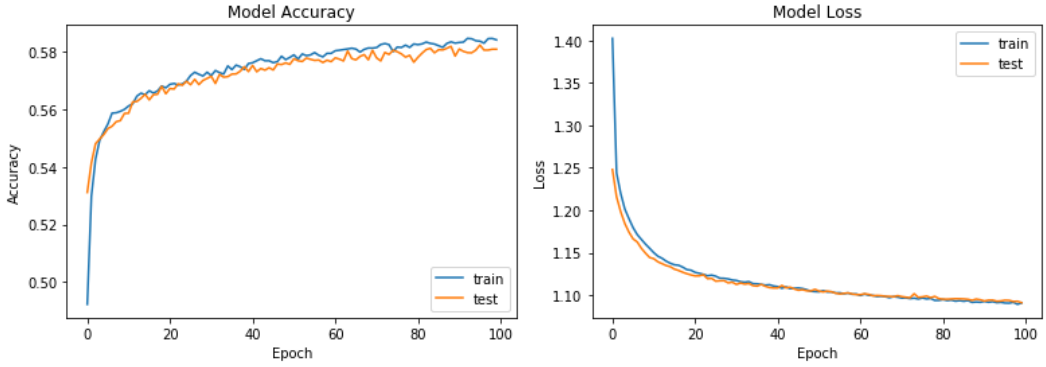



**Figure 5.3:** Optimizer changed to Stochastic Gradient Descent with a learning rate of 0.01.

We can see the constant learning rate smoothing out the curves. Also, the testing accuracy is no longer above the training. However our accuracy is not higher overall. This could be due to our training set size. However, after a dataset increase to 250,000, we only reported only a 1% increase, to 59.3%. This tells us the model may not benefit from further increases.

### 5.1.2 Validation Performance

We can evaluate our MLP model performance based on it's predictions made over the unseen testing subset. This was 20% of our total training size. The final model trained on 250,000 pixels, meaning we had 50,000 pixels to test on. We can compare this with our baseline model, the Random Forest Regressor, which classified the same training set for each run, minus the *segment-rank* variable. We also trained our MLP model on the same dataset, to prevent the model choice outweighing the *segment-rank* variable influence. We can compare validation performance for all three models below, where the testing accuracy represents the performance over the unseen 20% subset of their training data.

**Table 5.2:** Run to run performance comparison for our MLP, MLP-Pixel and Random Forest Classifier baseline.

| Classifier | Training Set | Valid Acc (%) | Test Acc (%) |
|---|---|---|---|
| **MLP** | 10000 | 54.53 | 51.84 |
| | 100000 | 58.5 | 56.19 |
| | 250000 | 59.21 | 58.3 |
| **MLP-Pixel** | 10000 | 55.64 | 51.75 |
| | 100000 | 56.32 | 55.53 |
| | 250000 | 57.23 | 56.87 |
| **Random Forest** | 10000 | 24.25 | 21.52 |
| | 100000 | 47.85 | 44.56 |
| | 250000 | 51.36 | 47.52 |

Our model clearly outperforms the baseline measure. MLP-Pixel also outperforms the baseline, but is still lower than the MLP trained on segment information. We can take a closer look at our highest-performing model with a classification report below.

**Table 5.3:** Class breakdown for our best-performing MLP model, tested on our test subset of 50,000 pixels from the segment-rank dataset.

| Land-Type | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| *Discontinuous urban* | 0.54 | 0.53 | 0.51 | 5202 |
| *Non-irrigated arable land* | 0.62 | 0.46 | 0.53 | 16691 |
| *Pastures* | 0.60 | 0.80 | 0.68 | 22931 |
| *Industrial/commercial* | 0.56 | 0.47 | 0.51 | 1365 |
| *Transitional woodland-shrub* | 0 | 0 | 0 | 34 |
| *Sport and leisure facilities* | 0 | 0 | 0 | 1246 |
| *Mixed forest* | 0.35 | 0.28 | 0.31 | 338 |
| *Broad-leaved forest* | 0.44 | 0.29 | 0.35 | 1203 |
| *Green urban areas* | 0 | 0 | 0 | 368 |
| *Agriculture and vegetation* | 0 | 0 | 0 | 49 |
| *Dump sites* | 0 | 0 | 0 | 48 |
| *Airports* | 0 | 0 | 0 | 258 |
| *Continuous urban* | 0 | 0 | 0 | 86 |
| *Water bodies* | 0.68 | 0.27 | 0.38 | 49 |
| *Construction sites* | 0 | 0 | 0 | 56 |
| *Road and rail networks* | 0 | 0 | 0 | 25 |

**Table 5.4:** MLP performance on testing subset of 50,000 pixels.

| | Correct Pred. | Incorrect Pred. | Overall Accuracy |
|---|---|---|---|
| **MLP** | 29950 | 20050 | 59.10% |

We can see the moderate performance of the model when classifying the commonly found classes such as pastures, discontinuous urban and arable land, with moderately high precision levels. Precision represents the ratio of true positives to false positives. Out all of positive predictions made for water bodies, for example, 68% were correct. The precision rate here is contrasted by the low recall. Recall represents the ratio of true positives to false negatives, or how many negative predictions were made for that land-type. Although precision was 68%, out of all water body pixels existing in the AOI, only 27% were classified correctly. F1-score represents an aggregate of precision and recall. Support represents the count of each land-type. For the land-types with low support, our model did not perform well, missing almost every land-type below a support of 1000. We'll examine performance of our MLP predictions over a larger dataset next, where class representations may not be as skewed.

### 5.1.3 Prediction Results

Thus far, our MLP had been trained and validated over 250,000 pixels extracted from our AOI. Our knowledge of it's performance was known in comparison only

to our own MLP variant and Random Forest baseline. The overall goal of the project was to develop an alternative approach to the actual GSI method. To make a reliable estimate of the quality of our devised methods, they had to be directly compared to GSI's. This meant testing both under the same conditions. GSI had previously supplied us with performance metrics for their actual Random Forest Regressor, based on predictions made on 400,000 pixels from the same AOI our project is using. We can see these below.

**Table 5.5:** Prediction performance for the GSI Random Forest (RF) classifier on 400,000 extracted pixels from our AOI.

| | **Land-Type** | **Precision** | **Recall** | **F1-Score** | **Support** |
|---|---|---|---|---|---|
| **GSI-RF** | *continuous urban* | 0.05 | 0.92 | 0.09 | 4042 |
| | *discontinuous urban* | 0.42 | 0.52 | 0.46 | 39915 |
| | *industrial/commercial* | 0.64 | 0.94 | 0.76 | 75389 |
| | *road and rail networks* | 0.00 | 0.00 | 0.00 | 6459 |
| | *airports* | 0.21 | 1.00 | 0.35 | 51498 |
| | *green urban areas* | 0.00 | 0.00 | 0.00 | 0 |
| | *sport/leisure facilities* | 0.00 | 0.00 | 0.00 | 1540 |
| | *non-irrigated arable land* | 0.22 | 0.24 | 0.23 | 51650 |
| | *pastures* | 0.98 | 0.14 | 0.25 | 28582 |
| | *broad-leaved forest* | 0.39 | 0.45 | 0.42 | 51063 |
| | *mixed forest* | 0.08 | 0.85 | 0.15 | 78961 |

**Table 5.6:** Overall metrics for the GSI Random Forest Regressor.

| | **Correct Pred.** | **Incorrect Pred.** | **Overall Accuracy** |
|---|---|---|---|
| **GSI-RF** | 145479 | 254521 | 37.00% |

We can see relatively low performance across all land-types in Table 5.13. However, there is considerably high recall across a number of classes, such as continuous urban, mixed forest and industrial/commercial. The GSI-RF fails to predict any road and rail, green urban areas or sport/leisure facilities. With an overall accuracy of 37%, it is a fairly low performing approach. Let's compare our MLP model's performance over a set of 400,000 pixels. Due to us not having access to the actual GSI-RF model, we could not arrange the exact same pixel dataset. Therefore there may be some class representation discrepancies between our prediction set and GSI's. However, due to the nature of the random pixel extraction used in both methods, we can assume enough similarity to make a fair comparison.

After extracting 400,000 pixels, we calculated their *segment-rank* variable and formed our prediction set. Instead of subsetting, as done in training and validation, we'd be predicting the classification of the entire dataset. We can see the results of this

below.

**Table 5.7:** MLP prediction performance over 400,000 pixels.

| Land-Type | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| continuous urban fabric | 0.04 | 0.05 | 0.04 | 97 |
| discontinuous urban fabric | 0.63 | 0.40 | 0.49 | 567 |
| industrial or commercial units | 0.59 | 0.32 | 0.41 | 42588 |
| road and rail networks land | 0.01 | 0.00 | 0.00 | 10741 |
| airports | 0.28 | 0.02 | 0.04 | 2000 |
| dump sites | 0.00 | 0.00 | 0.00 | 429 |
| construction sites | 0.00 | 0.00 | 0.00 | 422 |
| green urban areas | 0.10 | 0.00 | 0.00 | 2968 |
| sport and leisure facilities | 0.02 | 0.00 | 0.00 | 9025 |
| non-irrigated arable land | 0.63 | 0.07 | 0.12 | 134621 |
| pastures | 0.73 | 0.75 | 0.74 | 181995 |
| agriculture/vegetation | 0.00 | 0.00 | 0.00 | 514 |
| broad-leaved forest | 0.47 | 0.27 | 0.34 | 51063 |
| mixed forest | 0.57 | 0.29 | 0.38 | 9102 |
| transitional woodland-shrub | 0.00 | 0.00 | 0.00 | 183 |
| water bodies | 0.61 | 0.16 | 0.26 | 454 |

**Table 5.8:** Prediction metrics for the MLP.

| | Correct Pred. | Incorrect Pred. | Overall Accuracy |
|---|---|---|---|
| **MLP** | 116999 | 283001 | 29.25% |

We can see that the MLP approach fails to improve on the GSI methodology for this prediction set. Largely following its performance on the validation set, it is hindered by the class representation difference between the 50,000 pixel validation set and the 400,000 prediction set. It does predict well for the classes it has since the most of during training, however the prediction set also contained classes it had not seen before. This made the model unable to predict these classes correctly, having never trained on their features before. Therefore this performance could be expected.

We will discuss the implications of the results shown in this section in the following chapter. First, we exmaine the results of the patch-based implementation.

## 5.2 Patch-based Performance

We'll describe the performance of our patch-based approach in this section.

### 5.2.1 Training Performance

Our model would be training on a variety of patch sizes. As illustrated in the previous chapter, the patch-size could be a key influencer in the quality of predictions, due to the level of pixel information contained within. We can visualize this below in Table **??**

**Table 5.9:** Range of patches considered, with dimensions, pixel contents and label number per patch.

| Patch Dimensions | Pixel Count | Label Count |
|---|---|---|
| 3x3x4 | 36 | 9 |
| 5x5x4 | 100 | 25 |
| 7x7x4 | 196 | 49 |
| 11x11x4 | 484 | 121 |
| 21x21x4 | 1764 | 441 |
| 61x61x4 | 14884 | 3721 |

As we can see, the pixel count increases quickly as we expand our patch sizes. We can examine the effect this had on our training process. By fixing patch count to 10,000 patches, we could explore the effect of patch-size on training accuracy.

Using the same train/test splits used as before, we began training on 10,000 patches for each size in Table 5.9. We trained for 20 epochs on each patch, validating our training after each epoch by predicting the centre pixel label of 2,000 new pixels. We were using Adam adaptive learning rate, with a starting value of 0.01. We can see the variation in validation accuracies for each patch size below.

**Table 5.10:** Patch Size vs Validation Accuracy

| Patch Size | Validation Acc (%) |
|---|---|
| 3x3 | 58.32 |
| 5x5 | 59.66 |
| 7x7 | 60.27 |
| 11x11 | 49.06 |
| 21x21 | 35.78 |
| 61x61 | 41.32 |

As we can see, the smaller patches generally performed better, although the worse performing was the second largest at 21x21. As before, we wanted to take a closer look at the training performance before making any model changes. Moving forward, we took the top two performing patches, 3x3 and 7x7, abandoning the other sizes. We can see the training history for these patches below in Figure 5.4

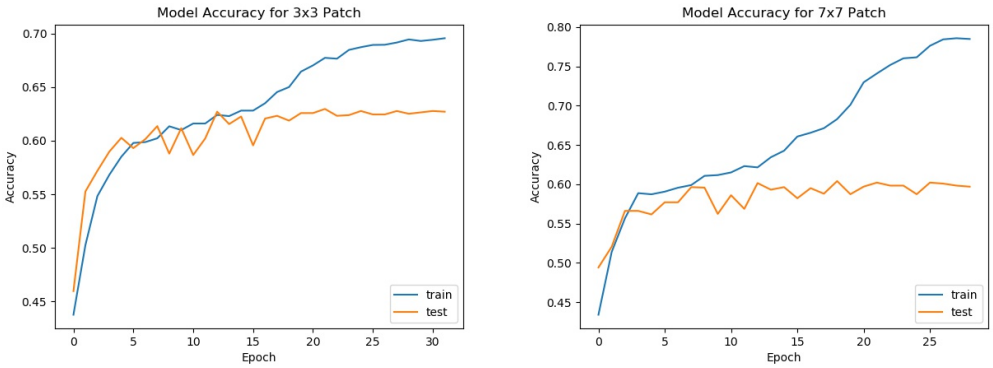



**Figure 5.4:** Training performance on 3x3 and 7x7 patches, from left to right. Note: epoch difference due to an early-stopper callback. As the validation accuracy began to plateau, training was stopped. This was a useful way of optimising the training process.

We can observe clear overfitting occurring. Let's try and implement some similar regularization techniques as for our segmentation approach. Experimenting with drop-out paralysed training, causing no learning to take place, with accuracy and loss not changing between epochs. Therefore instead, we decreased our learning rate, removing early stoppping callbacks to give the model more time to learn. A less common practice to improve learning is to increase the batch-size. Traditionally, one would decay the learning rate first, however a paper by Smith et al. suggest that increasing the batch-size may have more of an effect [60]. We decided to do both, doubling batch-size from 256 to 512, and can observe the effect of this below in Figure 5.14.

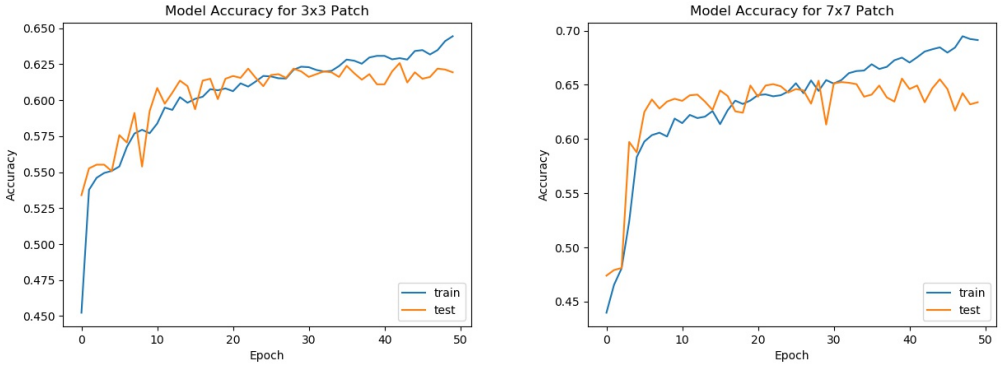



**Figure 5.5:** Adjusted training performance, with a learning rate decrease to 0.001 and batch size increase to 512.

We can see these measures had the intended affect, as both train and test curves are much more aligned. The 7x7 patch also achieves a new highest validation accuracy, of 64%. We can still see the early signs of overfitting, however, as the training curve begins to tail upwards. The curves also look slightly erratic, leading us to experiment

with Stochastic Gradient Descent. This resulted in a similar effect to implementing drop-out: paralysing the learning process. Therefore, our model maintained an Adam adaptive rate, with a new setting of 0.001.

In Figure 5.14, both the validation and testing accuracy seem to be plateauing after 30 epochs, indicating that they could benefit from training set increases. After experiencing some small gains by increasing to 50,000, we can show the final training set performance below, for 500,000 patches. Due to the long training times experienced with a training set this size, it was logical to reactivate our early-stopping callback. We can see the performance of our models training on 500,000 patches below.

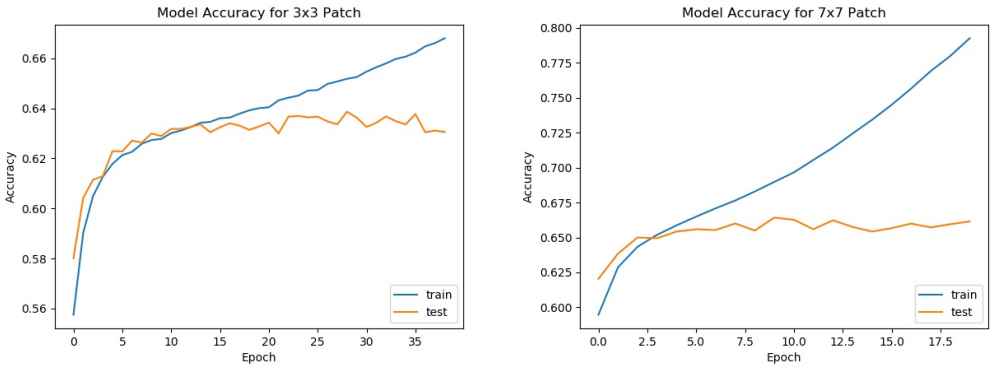



**Figure 5.6:** Training on 500,000 patches for 3x3 and 7x7 patch models.

We managed to achieve a slight increasing in final validation accuracy after the size increase. However, such a marginal increase indicates the model has reached its learning capacity. We also failed to prevent overfitting, which begins occurring around 10 epochs in for the 3x3 patch, and as early as 4 epochs for the 7x7 patch model. With the 7x7 trained CNN reaching the highest validation accuracy, we'll choose this model to examine further in the next section.

### 5.2.2 Validation Performance

We can evaluate our best CNN model by examining its performance on the unseen testing subset of 100,000 patches. We can the classification breakdown below.

**Table 5.11:** Class breakdown for our best-performing CNN model, tested on our testing subset of 100,000 patches.

| Land-Type | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| *Discontinuous urban* | 0.69 | 0.74 | 0.72 | 10523 |
| *Non-irrigated arable land* | 0.65 | 0.81 | 0.72 | 45438 |
| *Pastures* | 0.71 | 0.54 | 0.61 | 33675 |
| *Industrial/commercial* | 0.62 | 0.34 | 0.44 | 685 |
| *Transitional woodland-shrub* | 0.36 | 0.14 | 0.20 | 2462 |
| *Sport and leisure facilities* | 0.49 | 0.41 | 0.45 | 2244 |
| *Mixed forest* | 0.56 | 0.53 | 0.54 | 2694 |
| *Broad-leaved forest* | 0.50 | 0.01 | 0.02 | 119 |
| *Green urban areas* | 1.00 | 0.05 | 0.09 | 107 |
| *Agriculture and vegetation* | 0.71 | 0.42 | 0.53 | 511 |
| *Dump sites* | 0.28 | 0.06 | 0.11 | 790 |
| *Airports* | 0.36 | 0.09 | 0.15 | 106 |
| *Continuous urban* | 0.37 | 0.18 | 0.24 | 146 |
| *Water bodies* | 0.42 | 0.15 | 0.22 | 33 |
| *Construction sites* | 0.50 | 0.15 | 0.25 | 41 |
| *Road and rail networks* | 0.61 | 0.37 | 0.46 | 105 |

**Table 5.12:** Overall CNN performance on our testing set of 100,000 pixels.

| | Correct Pred. | Incorrect Pred. | Overall Accuracy |
|---|---|---|---|
| **CNN-7x7** | 66140 | 33860 | 66.14% |

Our CNN model has trained relatively well, based on the the overall accuracy in Table 5.12. In Table 5.11, we can the strong performance for classes of both a high and mid-level of support. We see some divergences in precision and recall, notably for broad-leaved forest and green urban areas. Overall the model has provided a good set of predictions. We'll examine the model performance over a larger test set in the next section.

### 5.2.3 Prediction Results

Using the metrics supplied by GSI, we could make directly compare the predictive performance of our model against the current GSI-RF methodology. By sampling 400,000 patches from our AOI, and making pixel-wise predictions, we'd be able to compare our approach to theirs. We chose our highest-performing model, the 7x7-patch trained CNN, to make the predictions. We can visualise these, along with the original GSI metrics below.

**Table 5.13:** Prediction performance for the GSI Random Forest (RF) classifier on 400,000 extracted pixels from our AOI.

| | Land-Type | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|---|
| **GSI-RF** | *continuous urban* | 0.05 | 0.92 | 0.09 | 4042 |
| | *discontinuous urban* | 0.42 | 0.52 | 0.46 | 39915 |
| | *industrial/commercial* | 0.64 | 0.94 | 0.76 | 75389 |
| | *road and rail networks* | 0.00 | 0.00 | 0.00 | 6459 |
| | *airports* | 0.21 | 1.00 | 0.35 | 51498 |
| | *green urban areas* | 0.00 | 0.00 | 0.00 | 0 |
| | *sport/leisure facilities* | 0.00 | 0.00 | 0.00 | 1540 |
| | *non-irrigated arable land* | 0.22 | 0.24 | 0.23 | 51650 |
| | *pastures* | 0.98 | 0.14 | 0.25 | 28582 |
| | *broad-leaved forest* | 0.39 | 0.45 | 0.42 | 51063 |
| | *mixed forest* | 0.08 | 0.85 | 0.15 | 78961 |

**Table 5.14:** Overall metrics for the GSI Random Forest Regressor.

| | Correct Pred. | Incorrect Pred. | Overall Accuracy |
|---|---|---|---|
| **GSI-RF** | 145479 | 254521 | 37.00% |

**Table 5.15:** CNN-7x7 performance over the prediction set of 400,000 unseen pixels.

| Land-Type | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| *continuous urban fabric* | 0.33 | 0.05 | 0.09 | 97 |
| *discontinuous urban fabric* | 0.53 | 0.70 | 0.66 | 567 |
| *industrial or commercial units* | 0.58 | 0.32 | 0.41 | 42588 |
| *road and rail networks land* | 0.55 | 0.38 | 0.00 | 10741 |
| *airports* | 0.30 | 0.10 | 0.15 | 2000 |
| *dump sites* | 0.22 | 0.04 | 0.00 | 429 |
| *construction sites* | 0.44 | 0.11 | 0.00 | 422 |
| *green urban areas* | 0.58 | 0.10 | 0.00 | 2968 |
| *sport and leisure facilities* | 0.47 | 0.39 | 0.00 | 9025 |
| *non-irrigated arable land* | 0.55 | 0.70 | 0.62 | 134621 |
| *pastures* | 0.75 | 0.55 | 0.63 | 181995 |
| *agriculture/vegetation* | 0.42 | 0.00 | 0.00 | 514 |
| *broad-leaved forest* | 0.44 | 0.27 | 0.33 | 51063 |
| *mixed forest* | 0.57 | 0.29 | 0.38 | 9102 |
| *transitional woodland-shrub* | 0.33 | 0.00 | 0.00 | 183 |
| *water bodies* | 0.41 | 0.15 | 0.22 | 454 |

**Table 5.16:** Overall metrics for the GSI Random Forest Regressor.

| | Correct Pred. | Incorrect Pred. | Overall Accuracy |
|---|---|---|---|
| **CNN-7x7** | 187520 | 212480 | 43.94% |

We manage to slightly increase the current GSI methodology. Similar to the MLP, our model predicts well those classes that it has trained on. Unlike the MLP, the CNN-7x7 had been exposed to all 17 classes prior to predicting. This is likely due to the increased pixel content contained in the patches dataset, versus the individual pixels in the MLP training set. We'll examine the implications of these results in the next chapter.

### 5.2.4 Scene Prediction

Using the sliding window mechanism, illustrated in the previous chapter, we could use our patch-trained CNN models to make predictions onto our AOI. To provide an accurate measure of our approach, this was done with two models, each trained on one of the two optimal patch size and count configurations, examined in the previous section. We can see the predictions each model made over our entire AOI below.

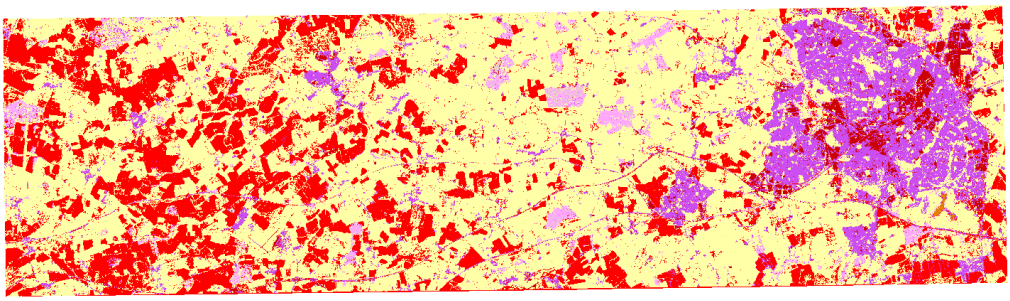


**Figure 5.7:** CNN-3x3 label predictions, corresponding to the CORINE Land-Type colour legend shown earlier.

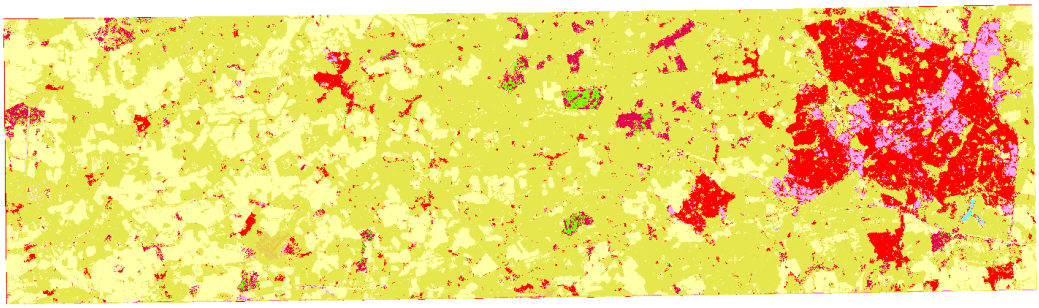


**Figure 5.8:** CNN-7x7 label predictions, also corresponding to CORINE colour scheme.

We can see the scene-wide predictions made by both CNN models above. Notably, the models have different classifications for the Swindon urban area, to the right of

the image. However, both models capture the shape of the town and surrounding outskirts well. We'll examine these plots in detail in the next section.

**Figure 5.9:** CNN-3x3 prediction probabilities, with black representing low confidence and white representing high confidence.

**Figure 5.10:** CNN-7x7 prediction probabilities, similar colour-scale.

We can see above how confident both of the models were in making their label predictions. Interestingly, the 7x7 model was highly confident predicting Swindon, shown by the white colouring across the area, compared to the grey shading present in the 3x3 predictions. The 3x3 patch was more confident predicting fields and arable land, present more to the left-hand side of the scene. We'll investigate the implications of these results in the next chapter.

## 5.3 Transfer Learning

### 5.3.1 Training and Validation Results

Training on BigEarth was a challenge due to it's large size. When loaded into memory, the entire dataset required 4.4GB of RAM. This produced limitations on the practicality of training. After splitting our dataset of 294684 images into the

60/20/20 split, we could begin training. Using our CNN discussed earlier, we can visualise the performance below.

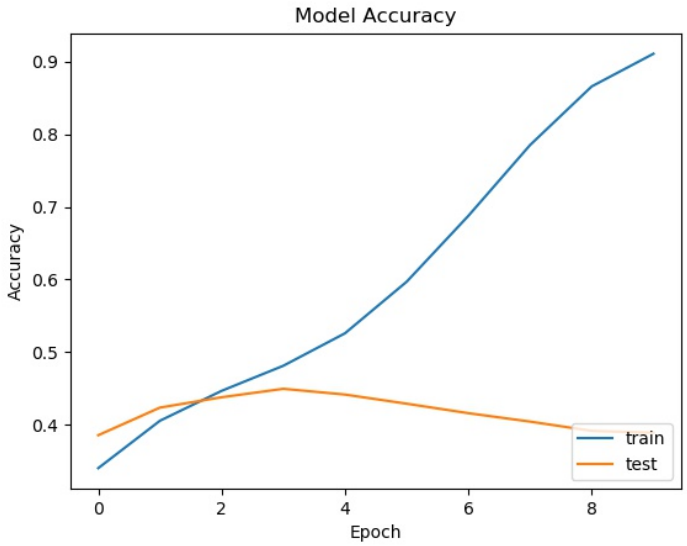



**Figure 5.11:** Initial BigEarth training performance.

The model overfit BigEarth quite severely. We attempted to slow down training by implementing the regularization techniques discussed earlier, allowing for them to take effect by increasing the epoch count. However, due to excessive training times, we were limited by how far we could increase this. Nonetheless, we can visualise the effect of implementing drop-out between our convolutional blocks below.

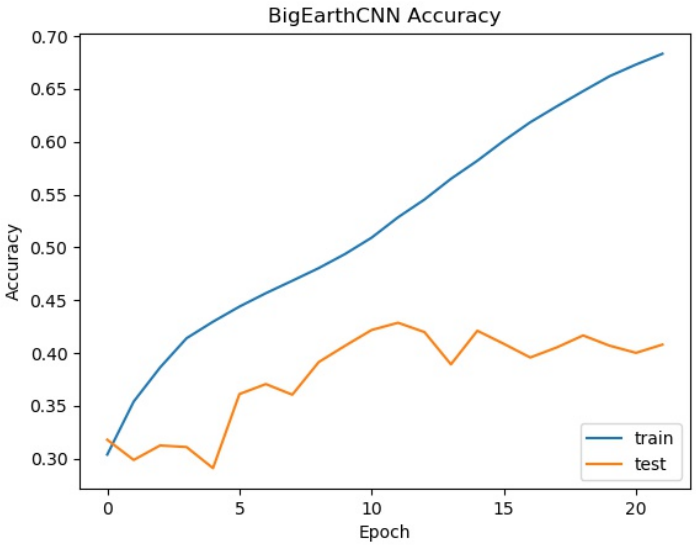



**Figure 5.12:** BigEarth training after drop-out regularization.

We can see slowing down the training rate of learning has allowed the testing accuracy to track the training for the first 10 epochs. However, it subsequently plateaus at around 40%, as the model begins to overfit. Next, we can see the effect of altering our model architecture, to try and reduce the number of parameters that are updated after each epoch. We reduced our six convolution blocks to just one, made up of two convolutional layers and a pooling layer. Let's visualize the effect of this.

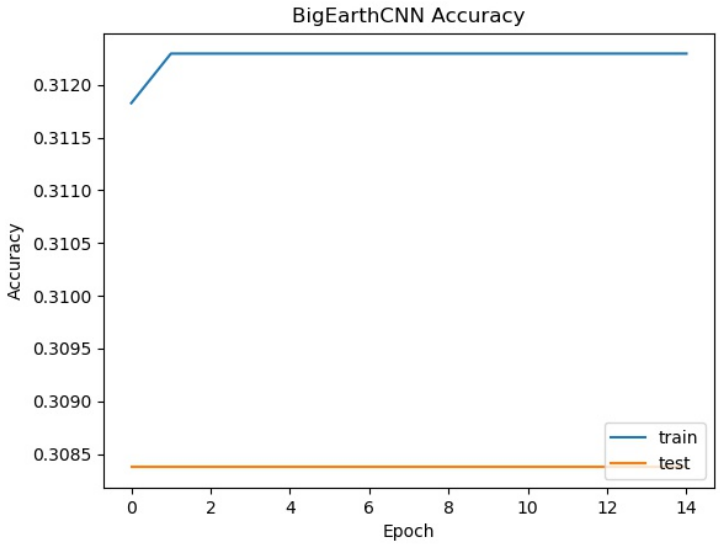



**Figure 5.13:** Reduced BigEarth-CNN architecture performance.

The model has flatlined. Without the repeating convolutional blocks, it can't detect any pattern whatsoever within the 120x120 patches. Therefore, we restored our original model architecture. Whilst not achieving significant validation accuracies, with the highest at 40%, it was clear the model was making some informed predictions. Therefore, we decided to test our CNN-BigEarth model over our entire scene. This was deemed more likely to produce interesting results, as opposed to comparing it directly to the GSI approach, as in the previous two methods.

### 5.3.2 Prediction Performance

Whilst failing to achieve significant validation accuracies, it seemed logical to still make predictions with our trained model. Results of this would be a useful indicator for any future training on BigEarth, as well as establish the use of transfer learning as a possible approach for GSI to consider.

By imitating the same sliding patch-window approach as used in our original patch method, we could feed our AOI into the BigEarth-CNN. The model would then

make predictions row by row and produce scene-wide pixel predictions. We can visualise these predictions below.

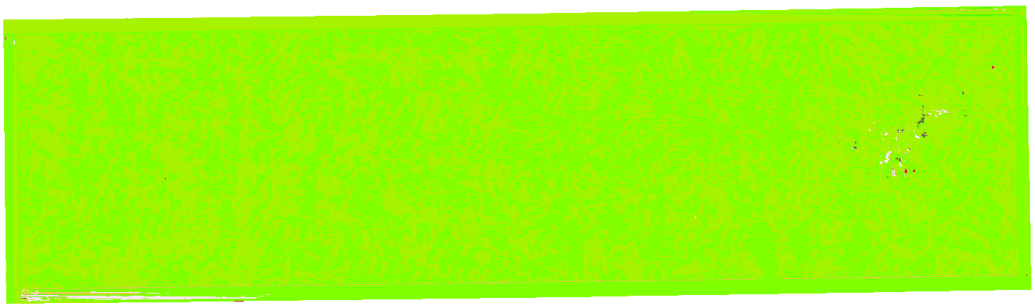


**Figure 5.14:** BigEarth-CNN scene-wide predictions.

As we can see, the BigEarth-CNN essentially makes the same two predictions across the entire AOI, with the exception of a cluster of classes centred within Swindon. After the level of overfitting observed during training, this was largely expected. We'll discuss the implications this has on the wider patch-based approach in the following chapter.

# Chapter 6

# Discussion

## 6.1 What is the impact of considering local regions of pixels when performing image classification activities?

### 6.1.1 Segmentation Approach

Our segmentation approach highlighted the effect of considering local image regions for classification. By analysing the results of our MLP model, versus the results of both a pixel-only variant and the GSI current methodology, we can see that there is a clear potential for segmentation to have an impact on classification accuracies.

In Section 5.1.2 we can see the performance of the MLP classifier, demonstrating superior results when compared to a pixel-only MLP and Random Forest classifier. However, when compared with the actual GSI methodology, we fail to surpass the current accuracy measure. Although performing well on a subset of 50,000 pixels, when generalising over 400,000, we fail to sustain an accuracy level above the comparison. However, on some classes we do outperform the GSI approach. For discontinuous urban fabric, arable land and mixed-forest we achieve high precision levels. We could attribute this to the prevalence of these classes in our AOI, as well as the characteristics of these land-types. These land-types are frequently found in concentrations of high density over large swathes of the AOI, therefore would be more easily segmented, as opposed to green urban areas or water bodies, for example. Therefore this could be an indication of the success of the segmentation approach, if only for certain land-types. However considering overall accuracy only, compared to the GSI metrics, we did not improve performance.

However, after analysing the differences between the MLP and GSI prediction sets, it's clear that the varying levels of support between them may have had an impact. The GSI prediction set had a significantly more even distribution of classes. Whereas the data our MLP was predicting had support levels ranging from 181,000 to just 97, for single classes. Therefore, assessing the quality of the MLP performance solely based on this comparison would not be a fair measure to take. We would require identical training and prediction sets for this, which was not possible in the practicalities of this project. Therefore we can answer this question more fully by comparing our MLP to our own methods. The increase in accuracy over MLP-Pixel and our baseline Random Forest tell us that use of local region knowledge, in this case the inclusion of our *segment-rank*, has had a clear positive impact on image classification.

### 6.1.2 Patch Approach

The patch approach can be used to answer this question using a range of points.

Firstly, our results show that patch-size, or size of local region size, can determine prediction accuracies for image classification. We highlighted in Section 5.2 the effect of altering our patch size on validation accuracy. We showed that altering the extent of the local region considered can have a positive effect on validation accuracy. But let's examine the effect of this when compared to a non-local region methodology.

This is shown in Section 5.2.3, where we compare against the current GSI methodology. Although similarly contrasting levels of support existed between the prediction sets, our method still managed to outperform the GSI method. With a 7x7 sized patch, we achieved almost a 7% increase in accuracy. Pastures, green-urban areas and industrial/commercial units were all classed with high precision. The patch-based training seems to have helped the model detect the key shapes and features which distinguish these land-types, i.e. the tiled shaping of pastures, the irregular shaping of green urban areas, and the block-like shapes present in industrial areas. This argument could be countered by mentioning the poor performance of the patch-trained model when classifying water bodies, airports and dump sites, all land-types with highly unique shapes. However, these classes had very low support in both the CNN's training and predicting sets. The model had rarely been exposed to patches containing pixels belonging to these classes, therefore low prediction performance would be expected.

In Section 5.2.4, we produce scene-wide predictions for the AOI. This will help us

further illuminate the effect of local image regions, in a more visual context.

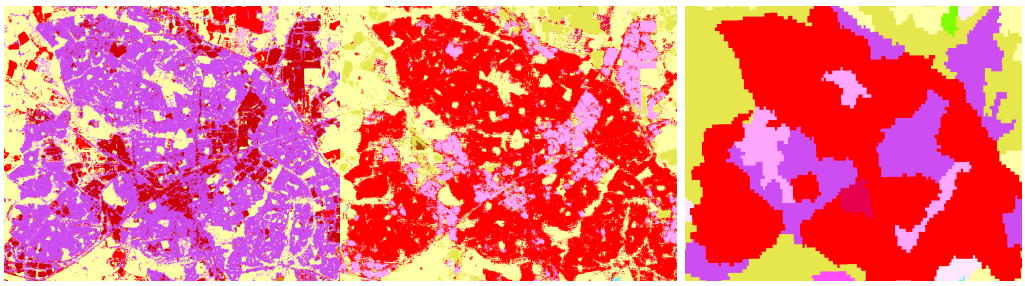


**Figure 6.1:** CNN-3x3, CNN-7x7 and ground-truth over Swindon.

We can see the effect of altering our local patch size above. The larger patch area correctly classifies Swindon as continuous urban (red), whereas the 3x3 patch classifies as industrial (purple). This is likely due to the fact that the 3x3 would be more sensitive to the distinct colour of industrial warehouses, mistaking urban fabric as industrial due to the colour similarities between urban and industrial areas. Whereas the 7x7 patch would be able to detect the defining features of urban areas, such as the grid-like layout of street blocks, leading to an accurate classification of the whole Swindon area. It does misclassify the industrial areas as sports and leisure (light-purple), however. Let's take a look at another region of the predicted AOI.

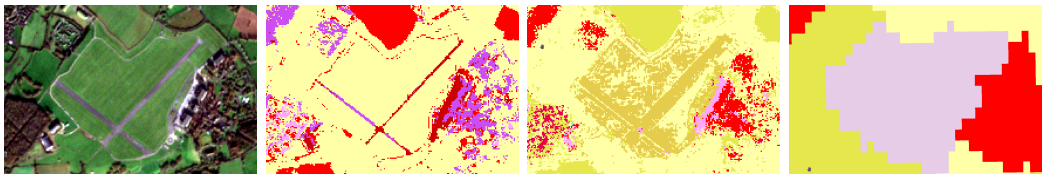


**Figure 6.2:** True colour image, CNN-3x3, CNN-7x7 and ground-truth over a small airport.

Figure 6.2 demonstrates the proficient shape detection of our patches. In contrast to the Swindon scene, the 3x3 patch almost exactly detects the shape of the runway, compared to the 7x7 patch. This is referenced by the actual true-colour image, as the resolution of the ground-truth fails to display the airport runway. We can see here that local image region consideration, if variable, can provide even greater classification results. We can see this above, where the thin runways have been detected by our 3x3 patch, and thus producing fine predictions, due to the small patch size. This is reinforced by the more blurry predictions by the 7x7 patch. We can take another perspective on this by analysing road detection next.

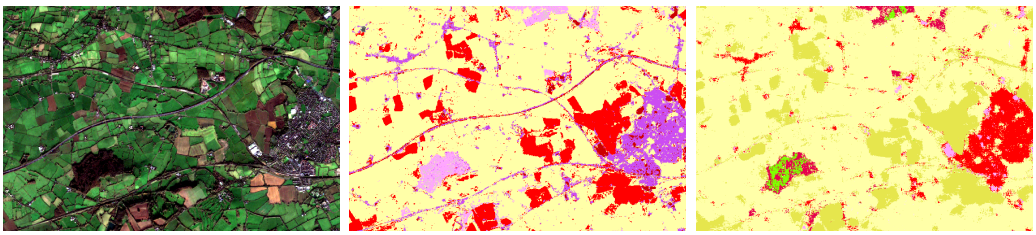


**Figure 6.3:** CNN-3x3, CNN-7x7 predictions and the true-colour image over a region of roads.

Again, the 3x3 shows optimal performance classifying this small network of roads. The CNN-7x7 completely misses them altogether, and classifies the entire area around the roads as fields. It is likely the surrounding fields confused the 7x7 model, as it would have detected a greater proportion of fields than roads in each patch. This reinforces our suggestion that small patches are best detecting small features, whereas larger patches are best at detect larger features. Therefore, this adds valuable evidence to our answer that local image regions can have a positive effect on accuracy. To demonstrate some counter evidence, we can examine a cropping of the BigEarth AOI prediction below.

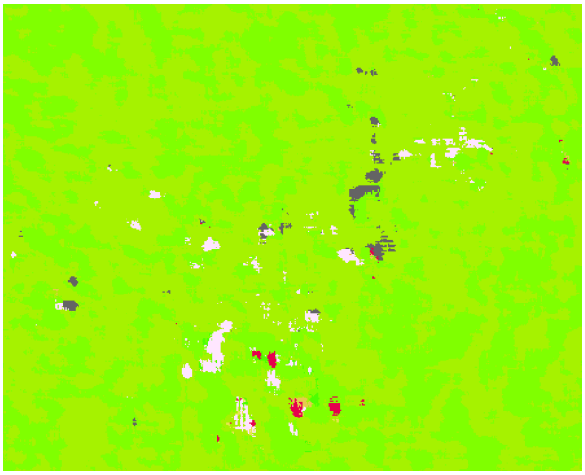


**Figure 6.4:** CNN-BigEarth predictions for the Swindon area.

Evidently, BigEarth was unable to provide any meaningful predictions. At first glance, this would negate our argument suggesting the positive impact of local region consideration. However, the BigEarth patch sizes were 120x120 in dimension. A 120x120 region of 10m resolution imagery covers over $1km^2$, which in the context of our AOI, would be hard to consider as a local region. In fact, we can see this in Figure 6.4 by the continuous urban predictions, coloured red, in the centre of Swindon town centre. The only correct classifications in the entire scene, the surrounding patch would have been entirely made up of urban fabric. Therefore, the BigEarth model

evidently needs more than a local region to be able to effectively classify. Therefore, the poor performance of BigEarth highlights the superior performance of our local-region classifiers, the two CNN models.

These results clearly show that considering local regions of an image can positively impact image classification activities. Although it was difficult to fully compare to the GSI current methodology, we were able to cross-examine both approaches to GSI performance metrics. Whilst only one of the two methods outperformed the GSI approach, both had unique performance aspects, such as accurate shape detection and effective classification for highly homogeneous land-types, that could be of interest to GSI. We'll discuss the feasibility of GSI adopting these methods below.

## 6.2 What would be the feasibility of GSI adopting these methods into the current methodology?

In order to assess the feasibility of these methods, further training and experimentation would be required. This was due to the similarities between the content of our training and testing data. Although we mentioned class representation differences, the actual pixel values were both taken from the same image. Therefore, the level of content, such as terrain type and seasonality, exposed to the model are constant. To attain a more realistic evaluation, GSI would have to experiment by exposing the models to a greater variety of training data.

Considering the data the models have been trained on, however, has shown that expanding classification training to consider local region variability, as opposed to aggregating to single pixel averages, greatly increases the complexity of the problem. Satellite imagery is already inherently non-linear, and when expanding the training region of an image from pixel to patches, there are clear design considerations which GSI should consider to assess the method feasibilities. The three-dimensional nature of multispectral imagery also increases the complexity of local region consideration, as the pixel count quickly skyrockets with a larger patch size, as shown. Additionally, greater thought must be taken for algorithm choice, due to the wide array of possible CNN architectures, of which we have only discussed a small fraction of in this paper.

However, these extra considerations also come with the huge potential increase in performance that comes with local region classification. Although not fully explored in this project, there is clear indication that, if tested further on a wide variety of data, these methods could achieve greater increases in accuracy.

Therefore, with some more development and work on the training of these methods, it could be said that it is fairly feasible that GSI would benefit from adopting these methods into their methodology.

## 6.3 Future Work

There are clear extensions of the work this project has produced. We'll give some recommendations on how these could be carried out below.

**Increasing Spectral Content**

Our project only utilised four bands out of the potential 13 on offer from Sentinel-2 data. Determining a method to merge the differing resolutions into a 13 band raster could greatly improve training accuracies. The CNNs our project trained and tuned were optimised for use on 4-band deep imagery. Increasing image depth would allow experimentation with interesting methodologies, such as the three-dimensional convolutions discussed by Hamida et al. [4].

**Semantic Segmentation**

The patch-based approach can definitely outperform a pixel-only classifier. However, as shown, it still does not produces significant accuracies. Exploring semantic segmentation, where an entire image is convolved over and predicted at once, to produce a classification heat-map, could be worthwhile. Incorporating the hourglass architecture discussed by Liu et al. could be combined with our sliding patch approach [36]. Instead of classifying the central pixel, the entire patch would be segmented as a heat-map. This could then be aggregated with all other patches to gain a deeper scene understanding.

**Transfer Learning**

We only partially managed to explore BigEarthNet, having to conform it's class and multispectral content to fit the constraints of this project. A very recently compiled dataset, the potential of achieving a low-loss trained model could be high for predicting Sentinel-2 imagery. Exploring it further could therefore be worthwhile. Additionally, the SEN12MS dataset, another large-scale Sentinel-2 dataset compiled in June of this year, would also be worth exploring [51]. Numbering in the 100,000s, it could also be a valuable training tool. Training the bulk of a model on BigEarth and finetuning the upper layers on SEN12MS could produce a powerful predictive model.

**Conditional Random Fields for Postprocessing**

Conditional random fields could be an interesting tool to explore. They are a method of modelling spatial-contextual information in an image, directly examining a pixel's colour, spectral content and spatial relations. More information for their use in remote sensing and CNN classification can be found in work done by Liu et al. [38].

**Ensemble Learning**

As shown by Farabet et al., using multiple CNN models in parallel can be beneficial for image classification [20]. They implement a CNN ensemble which predicts an image at three different scalings. It could be interesting to experiment with this for our image patches. We've clearly demonstrated the variable performance for different patch sizes, with a range of performance characteristics. For instance, the larger patch sizes explored in this project, whilst generally performing worse, informed on some spatial content which was missed by the smaller patches. By sliding three different patch sizes over our AOI and feeding each into a CNN, we could achieve a greater accuracy by aggregating each of the outputs, versus using a single patch size.

## 6.4 Conclusion

Our project has produced two classification methodologies, involving patch formation and segmentation, as the basis of an alternative solution to the current GSI classification methodology. Both solutions incorporate different spatial and textural elements of an area of interest, and use this wider range of information to make pixel-wise predictions. Whilst only one solution, the patch-based approach, outperformed the GSI methodology, both patches and segments had different performance aspects which could be of interest to GSI. Our highest performing solution, the patch based approach, achieved 66.14% validation accuracy, and 43.94% testing accuracy, an 8% increase on the GSI method. Whilst our segmentation approach under-performed by 7%, it demonstrated some useful performance characteristics, such as effective classification of land-types of larger, more homogeneous features. Both methods demonstrated the potential of deep neural networks when compared with the project baseline, and the GSI Random Forest. Therefore, this project has successfully demonstrated the potential of deep learning methodologies for use in GSI's land-classification methodology.

# Bibliography


[1] Anderson, J. R. *A land use and land cover classification system for use with remote sensor data*, vol. 964. US Government Printing Office, 1976.

[2] Arel, I., Rose, D. C., Karnowski, T. P., et al. Deep machine learning-a new frontier in artificial intelligence research. *IEEE computational intelligence magazine 5*, 4 (2010), 13–18.

[3] Belgiu, M., and Drăguţ, L. Random forest in remote sensing: A review of applications and future directions. *ISPRS Journal of Photogrammetry and Remote Sensing 114* (2016), 24–31.

[4] Ben Hamida, A., Benoit, A., Lambert, P., and Ben Amar, C. Three dimensional deep learning approach for remote sensing image classification, 06 2018.

[5] Bioucas-Dias, J. M. A variable splitting augmented lagrangian approach to linear spectral unmixing. In *2009 First workshop on hyperspectral image and signal processing: Evolution in remote sensing* (2009), IEEE, pp. 1–4.

[6] Boryan, C., Yang, Z., Mueller, R., and Craig, M. Monitoring us agriculture: the us department of agriculture, national agricultural statistics service, cropland data layer program. *Geocarto International 26*, 5 (2011), 341–358.

[7] Castelluccio, M., Poggi, G., Sansone, C., and Verdoliva, L. Land use classification in remote sensing images by convolutional neural networks. *arXiv preprint arXiv:1508.00092* (2015).

[8] Castillejo-González, I. L., López-Granados, F., García-Ferrer, A., Peña-Barragán, J. M., Jurado-Expósito, M., de la Orden, M. S., and González-Audicana, M. Object-and pixel-based analysis for mapping crops and their agro-environmental associated measures using quickbird imagery. *Computers and Electronics in Agriculture 68*, 2 (2009), 207–215.

[9] Chehata, N., Guo, L., and Mallet, C. Airborne lidar feature selection for urban classification using random forests. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 38*, Part 3 (2009), W8.

[10] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence 40*, 4 (2017), 834–848.

[11] Cheng, G., Zhou, P., and Han, J. Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing 54*, 12 (2016), 7405–7415.

[12] Craig, M. D. Minimum-volume transforms for remotely sensed data. *IEEE Transactions on Geoscience and Remote Sensing 32*, 3 (1994), 542–552.

[13] Curran, P. J. Remote sensing: Using the spatial domain. *Environmental and Ecological Statistics 8*, 4 (2001), 331–344.

[14] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (2009), Ieee, pp. 248–255.

[15] Deng, L., Yu, D., et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing 7*, 3–4 (2014), 197–387.

[16] Deng, Z., Sun, H., Zhou, S., Zhao, J., Lei, L., and Zou, H. Multi-scale object detection in remote sensing imagery with convolutional neural networks. *ISPRS journal of photogrammetry and remote sensing 145* (2018), 3–22.

[17] Duro, D. C., Franklin, S. E., and Dubé, M. G. A comparison of pixel-based and object-based image analysis with selected machine learning algorithms for the classification of agricultural landscapes using spot-5 hrg imagery. *Remote sensing of environment 118* (2012), 259–272.

[18] ESA. Copernicus sentinel-1 maps norway in motion, 2019.

[19] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International journal of computer vision 88*, 2 (2010), 303–338.

[20] Farabet, C., Couprie, C., Najman, L., and LeCun, Y. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence 35*, 8 (2012), 1915–1929.

[21] Feng, Q., Liu, J., and Gong, J. Uav remote sensing for urban vegetation mapping using random forest and texture analysis. *Remote sensing 7*, 1 (2015), 1074–1094.

[22] Gilbert, J. Key trends in the geospatial industry, 2019.

[23] Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

[24] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680.

[25] Gould, S., Fulton, R., and Koller, D. Decomposing a scene into geometric and semantically consistent regions. In *2009 IEEE 12th international conference on computer vision* (2009), IEEE, pp. 1–8.

[26] Han, W., Feng, R., Wang, L., and Cheng, Y. A semi-supervised generative framework with deep learning features for high-resolution remote sensing image scene classification. *ISPRS Journal of Photogrammetry and Remote Sensing 145* (2018), 23–43.

[27] Helber, P., Bischke, B., Dengel, A., and Borth, D. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2019).

[28] Hoiem, D., Efros, A. A., and Hebert, M. Recovering surface layout from an image. *International Journal of Computer Vision 75*, 1 (2007), 151–172.

[29] Huang, W., Xiao, L., Wei, Z., Liu, H., and Tang, S. A new pan-sharpening method with deep neural networks. *IEEE Geoscience and Remote Sensing Letters 12*, 5 (2015), 1037–1041.

[30] Jensen, J. R., and Lulla, K. Introductory digital image processing: a remote sensing perspective.

[31] Khandait, S. P., Thool, R. C., and Khandait, P. Automatic facial feature extraction and expression recognition based on neural network. *arXiv preprint arXiv:1204.2073* (2012).

[32] Kunkel, K. E., Karl, T. R., Brooks, H., Kossin, J., Lawrimore, J. H., Arndt, D., Bosart, L., Changnon, D., Cutter, S. L., Doesken, N., et al. Monitoring and understanding trends in extreme storms: State of

knowledge. *Bulletin of the American Meteorological Society 94*, 4 (2013), 499–514.

[33] Lasaponara, R., and Masini, N. Detection of archaeological crop marks by using satellite quickbird multispectral imagery. *Journal of archaeological science 34*, 2 (2007), 214–221.

[34] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature 521*, 7553 (2015), 436.

[35] Li, T. L., Chan, A. B., and Chun, A. H. Automatic musical pattern feature extraction using convolutional neural network. *Genre 10* (2010), 1x1.

[36] Liu, C.-C., Zhang, Y.-C., Chen, P.-Y., Lai, C.-C., Chen, Y.-H., Cheng, J.-H., and Ko, M.-H. Clouds classification from sentinel-2 imagery with deep residual learning and semantic image segmentation. *Remote Sensing 11*, 2 (2019), 119.

[37] Liu, F., Lin, G., and Shen, C. Crf learning with cnn features for image segmentation. *Pattern Recognition 48*, 10 (2015), 2983–2992.

[38] Liu, Y., Piramanayagam, S., Monteiro, S. T., and Saber, E. Semantic segmentation of multisensor remote sensing imagery with deep convnets and higher-order conditional random fields. *Journal of Applied Remote Sensing 13*, 1 (2019), 016501.

[39] Lu, D., Hetrick, S., and Moran, E. Land cover classification in a complex urban-rural landscape with quickbird imagery. *Photogrammetric Engineering & Remote Sensing 76*, 10 (2010), 1159–1168.

[40] Luus, F. P., Salmon, B. P., Van den Bergh, F., and Maharaj, B. T. J. Multiview deep learning for land-use classification. *IEEE Geoscience and Remote Sensing Letters 12*, 12 (2015), 2448–2452.

[41] Ma, L., Liu, Y., Zhang, X., Ye, Y., Yin, G., and Johnson, B. A. Deep learning in remote sensing applications: A meta-analysis and review. *ISPRS journal of photogrammetry and remote sensing 152* (2019), 166–177.

[42] Marcello, J., Rodriguez-Esparragon, D., and Moreno, D. Comparison of land cover maps using high resolution multispectral and hyperspectral imagery. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium* (2018), IEEE, pp. 7312–7315.

[43] Marmanis, D., Datcu, M., Esch, T., and Stilla, U. Deep learning earth observation classification using imagenet pretrained networks. *IEEE Geoscience and Remote Sensing Letters 13*, 1 (2015), 105–109.

[44] Marschner, F. J. Major land uses in the united states, 1950.

[45] Maul, G. A., and Gordon, H. R. On the use of the earth resources technology satellite (landsat-1) in optical oceanography. *Remote Sensing of Environment 4* (1975), 95–128.

[46] Munoz, D., Bagnell, J. A., and Hebert, M. Stacked hierarchical labeling. In *European Conference on Computer Vision* (2010), Springer, pp. 57–70.

[47] Pal, M. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing 26*, 1 (2005), 217–222.

[48] Reut, A. B., and Hara, T. Remote monitoring of military assets using commercial leo satellites. In *Proceedings of MILCOM'95* (1995), vol. 2, IEEE, pp. 869–873.

[49] Rezatec. Rezatec compass project, 2016.

[50] Schmidhuber, J. Deep learning in neural networks: An overview. *Neural networks 61* (2015), 85–117.

[51] Schmitt, M., Hughes, L. H., Qiu, C., and Zhu, X. X. Sen12ms–a curated dataset of georeferenced multi-spectral sentinel-1/2 imagery for deep learning and data fusion. *arXiv preprint arXiv:1906.07789* (2019).

[52] Schmullius, P. Earth system monitoring and modelling, 2016.

[53] Sebastiani, F. Machine learning in automated text categorization. *ACM computing surveys (CSUR) 34*, 1 (2002), 1–47.

[54] Seeber, G. *Satellite geodesy*. Walter de gruyter, 2008.

[55] Self, J. *Artificial intelligence and human learning: intelligent computer-aided instruction*. Chapman and Hall London, 1988.

[56] Sengupta, A., Ye, Y., Wang, R., Liu, C., and Roy, K. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience 13* (2019).

[57] Shao, W., Yang, W., Liu, G., and Liu, J. Car detection from high-resolution aerial imagery using multiple features. In *2012 IEEE International Geoscience and Remote Sensing Symposium* (2012), IEEE, pp. 4379–4382.

[58] Sharma, A., Liu, X., Yang, X., and Shi, D. A patch-based convolutional neural network for remote sensing image classification. *Neural Networks 95* (2017), 19–28.

[59] Smith, J. H., Stehman, S. V., Wickham, J. D., and Yang, L. Effects of landscape characteristics on land-cover class accuracy. *Remote Sensing of Environment 84*, 3 (2003), 342–349.

[60] Smith, S. L., Kindermans, P.-J., Ying, C., and Le, Q. V. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489* (2017).

[61] Socher, R., Lin, C. C., Manning, C., and Ng, A. Y. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)* (2011), pp. 129–136.

[62] Song, H., Kim, Y., and Kim, Y. A patch-based light convolutional neural network for land-cover mapping using landsat-8 images. *Remote Sensing 11*, 2 (2019), 114.

[63] Sumbul, G., Charfuelan, M., Demir, B., and Markl, V. Bigearthnet: A large-scale benchmark archive for remote sensing image understanding. *arXiv preprint arXiv:1902.06148* (2019).

[64] Thrun, S., and Pratt, L. *Learning to learn.* Springer Science & Business Media, 2012.

[65] Tong, X.-Y., Xia, G.-S., Lu, Q., Shen, H., Li, S., You, S., and Zhang, L. Learning transferable deep models for land-use classification with high-resolution remote sensing images. *arXiv preprint arXiv:1807.05713* (2018).

[66] UNOSSA. Sentinel online, 2018.

[67] UNOSSA. United nations register of objects launched into outer space, 2019.

[68] Wen, Y., Zhang, K., Li, Z., and Qiao, Y. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision* (2016), Springer, pp. 499–515.

[69] Wertheimer, M. Laws of organization in perceptual forms.

[70] Wu, Q., Zhou, C., and Wang, C. Feature extraction and automatic recognition of plant leaf using artificial neural network. *Advances in Artificial Intelligence 3* (2006), 5–12.

[71] Wurm, M., Stark, T., Zhu, X. X., Weigand, M., and Taubenböck, H. Semantic segmentation of slums in satellite images using transfer learning on fully convolutional neural networks. *ISPRS journal of photogrammetry and remote sensing 150* (2019), 59–69.

[72] Yang, Y., and Newsam, S. Bag-of-visual-words and spatial extensions for land-use classification. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems* (2010), ACM, pp. 270–279.

[73] Yu, W., Yang, K., Bai, Y., Xiao, T., Yao, H., and Rui, Y. Visualizing and comparing alexnet and vgg using deconvolutional layers. In *Proceedings of the 33 rd International Conference on Machine Learning* (2016).

[74] Yu, X., Wu, X., Luo, C., and Ren, P. Deep learning in remote sensing scene classification: a data augmentation enhanced convolutional neural network framework. *GIScience & Remote Sensing 54*, 5 (2017), 741–758.

[75] Zeng, K., Yu, J., Wang, R., Li, C., and Tao, D. Coupled deep autoencoder for single image super-resolution. *IEEE transactions on cybernetics 47*, 1 (2015), 27–37.

[76] Zhu, L., Chen, Y., Ghamisi, P., and Benediktsson, J. A. Generative adversarial networks for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing 56*, 9 (2018), 5046–5063.

[77] Zou, Q., Ni, L., Zhang, T., and Wang, Q. Deep learning based feature selection for remote sensing scene classification. *IEEE Geoscience and Remote Sensing Letters 12*, 11 (2015), 2321–2325.