Convolutioinal Neural Networks for Classification of Canopy Coverage from Earth Observation Data of Urban Areas

Student: Niamh Donnelly (17204509), Supervisor: Dr. Brian Mac Namee,

University College Dublin, Dublin 4, Ireland
niamh.donnelly1@ucdconnect.ie

Abstract. The availability of high resolution satellite imagery offers a compelling opportunity for the utilisation of state-of-the-art deep learning techniques in the area of remote sensing. This research investigates the application of Convolutional Neural Networks (CNNs) for pixel-wise classification of canopy coverage from earth observation image data of urban areas. The primary aim of this study was to develop a CNN model capable of segmenting aerial imagery into canopy or non-canopy to a high degree of accuracy. In the first phase of the research, a study to evaluate the performance of both patch-based and encoder-decoder CNN assemblies for canopy coverage segmentation was performed. Using an aerial image dataset of the city of Vaihingen in Germany, obtained from the International Society for Photogrammetry and Remote Sensing (ISPRS)[1], two established patch-based CNN architectures (LeNet, VGG16) for pixel-wise classification of canopy coverage were developed and tested. The performance of this approach was then compared with results obtained using two encoder-decoder architectures (UNet, custom 4-layer encoder-decoder). In the second phase of the research, a bespoke smaller image dataset covering the US city of Denver, Colorado was created using aerial image data from Google Earth Engine. Both patch-based approaches used in phase 1 were applied to this dataset and their performance was assessed. For tests performed using the Vaihingen dataset, patch-based approaches achieved F1-scores above 0.89, which significantly out-performed encoder-decoder models, where the highest F1-score achieved only 0.66. For the Denver dataset, the LeNet greatly under-performs, achieving a 0.48 F1-score value in comparison to the pre-trained VGG16 networks F1-score of 0.79. While the CNNs developed in this work provide satisfactory performance in almost all cases, it is postulated these results could be improved further with use of post-processing steps.

Keywords: Convolutional Neural Network · Remote Sensing · Deep Learning · Canopy Coverage · Google Earth Engine.

[1] Access to the ISPRS dataset of Vaihingen is provided at http://www2.isprs.org/commissions/comm3/wg4/2d-sem-label-vaihingen.html

1 Introduction

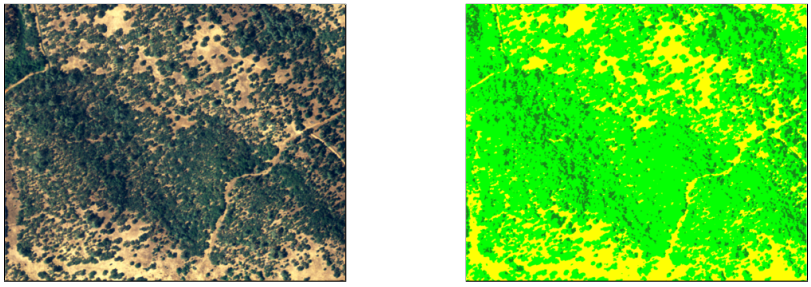
Remote sensing of tree canopy coverage in urban regions is vital to the task of monitoring environmental resources (i.e. soil and air quality, wildlife habitats, levels of CO2 emissions) and for large scale civic planning. Early techniques of identifying the quantity of canopy coverage required human surveyors to manually annotate sample areas on a map. Sample annotations were then extrapolated to provide an estimate of the total tree coverage for an urban area. This approach was slow, resource intensive, and was especially prone to problems of consistency. In recent years, the area of remote sensing has witnessed major advancements owing to improvements in satellite technology, public availability of high resolution aerial imagery, and advances in machine learning techniques. In particular, image classification techniques in remote sensing have benefited from the successes of Convolutional Neural Networks (CNNs) previously witnessed in the broader field of image processing and computer vision.

The overarching aim of this study is to produce a machine learning model to accurately detect the pixel coverage pertaining to canopy from an aerial image. Based on this aim, five research objectives were set, which included: (1) a review of the literature to determine appropriate approaches and identifying an earth observation dataset appropriate for the task of canopy coverage detection, (2) development, evaluation and comparison of two patch-based CNN architectures, (3) development, evaluation and comparison of two two encoder-decoder CNN architectures, (4) creation of a bespoke dataset using the Google Earth Engine software [1], (5) evaluation and comparison of CNN architectures on the new developed dataset.

In section 2, an in-depth review of the literature surrounding pixel-wise classification in the area of remote sensing is presented. Section 3 outlines the development of CNN models, testing metrics, and experimental design. Results of several experiments are presented in section 4 and discusses in section 5. Finally, a conclusion and personal reflection is outlined in section 6.

2 Related Work

The purpose of a pixelwise classification model in a canopy coverage application is to determine a class label, pertaining to either canopy or non-canopy, for every individual pixel in the image. This process results in an image representation similar to that represented in figure 1 where the color-coded image reconstruction represents canopy labels as green pixels.



(a) Aerial image (b) Color-coded reconstruction

Fig. 1: An example of (a) an aerial image and (b) its corresponding reconstructed ground truth representation [2].

Three distinct classification approaches currently exist in the area of remote sensing; pixel-based (involves the classification of each pixel in an image), object-based (the detection and classification of whole objects in an image), and scene classification (the assignment of an entire scene to a thematic class). The most appropriate approach to use varies by application, nature of the dataset, and available computational resources. Specifically the segmentation of canopy coverage requires a pixel-based method, as this task requires the assignment of a label to every pixel in an image. The following sections present a brief discussion of this approach and identify relevant prior studies involving its use in remote sensing applications.

2.1 Pixel-wise Classification

Pixel-based approaches are considered one of the more fine-grained methods of image classification in the remote sensing and geoscience fields. Unlike traditional classification, the approach involves not only the detection of an object in an image, but subsequently the assignment of each pixel to an object class [3][4]. Traditionally, pixel-based approaches concern the use of spectral features (i.e. information from single pixels) with shallow learning models (e.g. SVMs, KNNs). However, it is advantageous to include spatial information surrounding individual pixels. Recent work has made use of CNN architectures to obtain rich spectral and spatial information from image data. CNNs are a particular type of neural network designed to detect visual patterns directly from image pixels [5]. Two predominant approaches to pixel-wise classification using CNNs exist: Patch-based approaches and Encoder-decoder approaches.

2.1.1 Patch-based CNNs Issues arise when pixel-wise classification frameworks are developed to label a single pixel without considering its surrounding context. A pixel label may pertain to short or even long range information. For example, classification of a green pixel to a either a tree, a meadow, or other

vegetation requires a wider contextual consideration. Pixel-based methods often suffer from what is known as the "salt and pepper effect" [6], whereby an area of the classification contains instances where single isolated pixels are labeled incorrectly and differently to their surroundings, resulting in the segmented mapping appearing visually speckled. Consideration of surrounding pixels could potentially influence these isolated pixels to an accurate classification and result in stronger segmentation between spatial regions. Inclusion of spatial features is commonly achieved using a pixel wise patch-based approach [7] . This is implemented by extracting a patch of pixels of fixed width and length and labeling the center pixels based on its surrounding neighbors contained within the patch area. This procedure can be efficiently implemented via CNN architectures whereby the CNN takes as input a patch of an image, and generates a label associated with the center pixel of a patch.

By implementing a patch-based structure, Mnih and Hinton [7] observed high accuracies and a reduction in the salt-and-pepper appearance/noise for classification of aerial imagery using CNNs. A study by Hu et al. [8] developed a patch-based CNN architecture using the Theano python programming library to classify landmass objects (e.g. trees, gravel, meadows) at a pixel level. This study compared the proposed 5 layer CNN with a SVM model and four CNNs with architectures of varying complexity, including the well established LeNet [9]. Comparison of their custom CNNs with other established architectures of a higher complexity provided lower accuracies. Authors postulate this was due to an inadequate number of training samples, which is a common occurrence when developing CNNs. In comparison to the SVM, the proposed model showed an average of 2% accuracy improvements over all three datasets. Similarly a study by Zhao et al. [10], employed a patch-based CNN approach for classification of earth observation data and observed higher accuracy rates in comparison to engineered features and a SVM classifier.

When constructing a CNN (i.e. determining number of convolutional and dense layers, convolutional sizes, etc.), it is common to imitate a previously existing and well established architecture. A number of CNN assemblies have previously shown successes in the past, among the most notable range from long established architectures like LeNet [9] and AlexNet [11], to newly discovered architectures like VGG16 [12] , ResNet [13], and Inception[14]. For the purpose of this study, both the LeNet and VGG16 models are developed:

LeNet The LeNet is considered a pioneer in terms of CNN architectures, showing promising results early-on in the area of image classification. The architecture, presented in figure 2, was first introduced by LeCun et. al. in the seminal paper [9], which produced impressive results for the task of document recognition. In comparison to later CNN architectures, the LeNet is considered to be a simple architecture (i.e. has fewer convolutional layers). While this has the advantage of requiring fewer computational resources and less training time, the architecture is more shallow than modern deep architectures which may effect expressive power and overall accuracy.

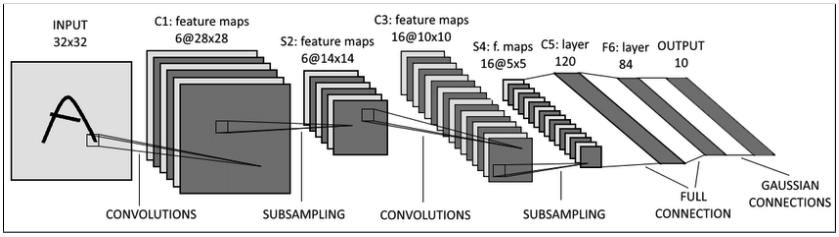


Fig. 2: LeNet CNN architecture [9].

VGG16 The VGG16 model is characterised by its significant layer depth and reduced complexity (i.e. few parameters). The architecture is displayed in figure 3. The success of the architecture is due to the combination of its 16 layer depth, and use of small kernels which keep the number of parameters low; these properties provide a deep hierarchical structure that have been shown to learn visual features efficiently. It was first recognized for its impressive accuracy scores in the ImageNet challenge [15]. It is common, when adapting this model, to make use of what is referred to in the literature as pre-trained version of the VGG16 model [16]. Early convolutional layers, up to the last layer, of the CNN are composed of weights that are pre-trained on an arbitrary image dataset, while the latter dense layers are fine-tunned to the intended data. In the area of remote sensing, pre-trained networks have shown encouraging results in comparison to fully trained networks, particularly when the dataset contains limited samples [17].

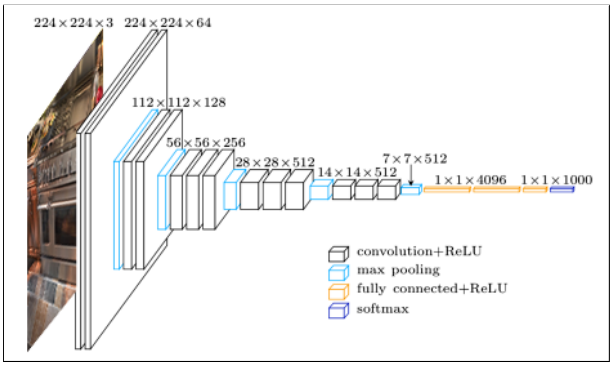


Fig. 3: VGG16 CNN architecture [18].

2.1.2 Encoder-decoder CNNs In contrast to patch-based approaches, encoder-decoder techniques produce a segmented image representation of an input rather than a single class label. The pooling layers of the encoder gradually down-

sample and decrease the size of the input. Subsequent to this step, the decoder network replaces pooling layers with upsample layers to gradually increase the convolutional layer size back to the scale of the original image (see figure 4).

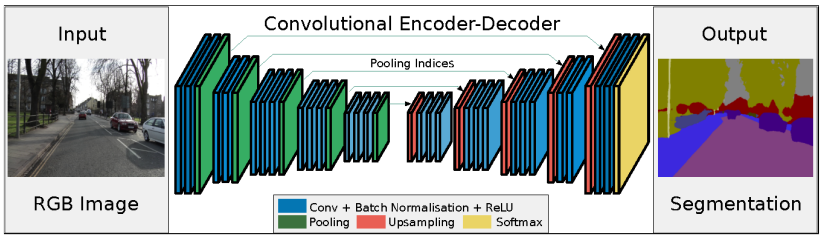


Fig. 4: Encoder-decoder CNN architecture [19].

Training encoder-decoders requires that the ground truth of a dataset consists of semantically segmented image masks whereby each pixel is color coded to its class label. In comparison to patch-based techniques, this requires a rigorous labeling technique. Patch-based methods involve labeling a single pixel and extracting the surrounding patch whereas the encoder-decoder technique requires detailed segmentation of the entire patch by detailing the borders of all objects in an image. A number of established encoder-decoder architectures exist in the literature, this study makes use of the preexisting UNet [20] additionally a custom 4-layer encoder-decoder was developed, with a similar layout to that displayed in figure 4.

UNet The UNet architecture presented in figure 5 was first developed for the purpose of biomedical image segmentation. In contrast to a simple encoder-decoder, the UNet includes an added nuance of skip connections between coinciding encoder and decoder layers. These connections, represented in grey in figure 5, are used to skip features from the the encoder path to the decoder path in order to recover spatial information lost during downsampling.

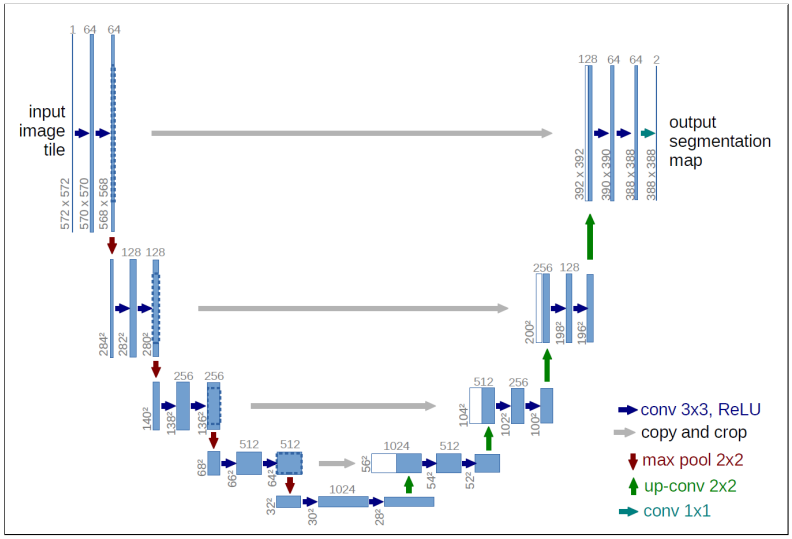


Fig. 5: UNet CNN architecture [20].

2.2 Canopy Coverage

Forest canopy cover is identified as the percentage cover of tree canopy in a give area, including only trees/shrubs and ignoring any other forms of vegetation [21]. It encompasses the vertical projection of the tree outline to the earth surface. Prior to accessibility of remote sensing data, the detection process of canopy coverage was achieved by ground-based techniques, whereby human surveyors were required to manually identify areas of tree coverage using hand-held devices. Another commonly employed traditional technique made use of surveyors to manually mark areas of tree coverage from satellite images. Government agencies in the UK previously implemented the latter technique through a crowd-sourcing methodology to roughly estimate the percentage of tree canopy cover in urban areas [22]. The initiative procured the use of voluntary human participants to manually label trees for a sample of earth observation data of the London city area. The percentage cover of the these areas was then extrapolated to provide an approximate calculation for other areas of the city not included in the sample subset.

The task of estimating the area of tree cover is one that benefits hugely from machine learning capabilities, in terms of resource requirements and accuracy. Stojanova et al. [23] explored a broad set of machine learning models to predict canopy coverage from land satellite imagery. Models implemented in the study include decision trees, random forests, bagging techniques, and tree-based ensembles. Random forest models outperform in terms of root-mean-square deviation (RMSE) for canopy cover, with most models showing competitive performance rates on a benchmark dataset.

While remote sensing using object-based, pixel-based, and scene classification approaches have experienced high performance rates spurred by advancements in deep learning architectures, canopy coverage evaluation using these techniques is a relatively undiscovered area with only a few relevant available published studies. Most notably, Guirado et al. [24] drew a comparative study between the use of traditional classifiers (e.g. SVM, KNN) and a CNN classification approach to determine the presence of tree shrubs in extracted patches of satellite image obtained via Google Earth. The authors obtained a 100% accuracy score on test data with the use of a ResNet CNN architecture for object based detection, outperforming object-based image analysis with SVM (88%), KNN (88%) and random forest (91.78%) classifiers.

While some studies identify land types containing tree coverage through scene classification of forestry vegetation (for example, see [25]), few focus on optimising classifiers for the task of identifying percentage of canopy coverage. As such, pixel-wise segmentation of tree cover remains a relatively uncharted area of remote sensing. Although CNNs have been used in the specific domain of canopy coverage, the application has predominantly used a object-based approach to classify individual trees [26,27,28]. The use of CNN architectures associated with pixel-wise segmentation strategies is still a largely unexplored area. Pixel-wise segmentation CNN architectures (i.e. patch-based, encoder-decoder) have been favored for land-use classification competitions, including the renowned DeepGlobe competition, with variations of UNet and Segnet holding top positions on the current leader-boards [29]. This provides a strong indication that the task of canopy segmentation could potentially also benefit from similar approaches.

3 Experimental Methods

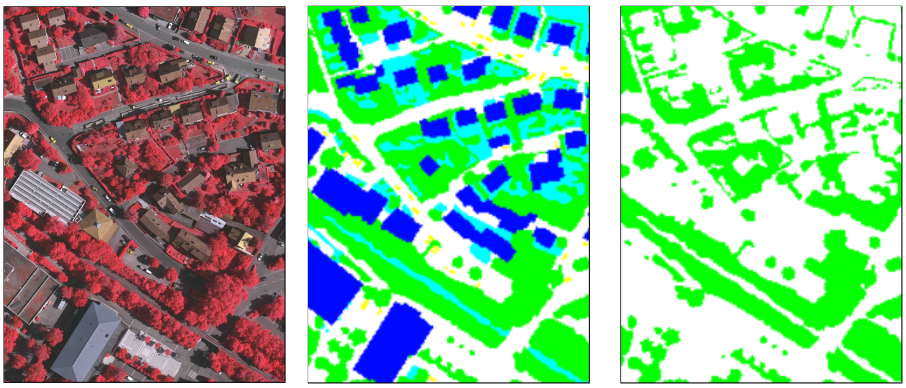
To explore the use of CNNs for the application of canopy coverage classification, four model architectures were developed for evaluation: LeNet, VGG16, UNet and a 4-layer encoder-decoder. Experiments were implemented using two datasets. The following sections discuss both datasets in detail, and outline experimental procedures for developing and testing model architectures.

3.1 Remote Sensing Datasets

A number of established earth observation benchmark datasets exist in the field of remote sensing. This study makes use of a pre-existing dataset developed by the International Society for Photogrammetry and Remote Sensing (ISPRS) which contains images of the city of Vaihingen in Germany. In addition to making use of preexisting data, a bespoke dataset was generated for the purpose of this research [2]. Data was obtained using the Google Earth Engine (GEE) platform [1] and covers the US city of Denver, Colorado.

[2] This dataset was developed in collaboration with Breadboard Labs, an Irish environmental technology company.

3.1.1 ISPRS Dataset of Vaihingen The ISPRS dataset contains a densely populated town with sufficient areas of tree coverage to act as a ideal candidate for the task of testing CNNs for canopy coverage detection. The dataset contains 33 tiles of varying sizes, however, most contain approx. 2000 x 2000 pixels. For each tile area, a RGB image with a resolution of 0.2 m is provided along with the corresponding ground truth image. The ground truth images are color coded with RGB values to represent six classes (impervious surfaces, building, low vegetation, tree, car, clutter/background). For the specific task of tree canopy coverage, the Vaihingen dataset was converted to binary format containing the original canopy label and converting all additional classes to non-canopy. A example of an Vaihingen tile before and after this conversion is represented in figure 6.



(a) (b) (c)

Fig. 6: Images from the Vaihingen dataset: (a) aerial satellite image, (b) ground truth including all original classes, (c) ground truth cleaned for only canopy coverage.

Upon examination, the high resolution aerial image tiles (2000x2000 pixels) of the Vaihingen dataset were deemed too large to be processed by the CNN architectures on single pass. Loading a dataset of this size is restricted by GPU memory limitations, therefore, the original Vaihingen tiles were segmented into smaller images patches using a sliding window approach. Data preparation involved extracting 33x33 pixel image patches from each of the Vaihingen dataset aerial image tiles. A sample of these patches are presented in figure 7.

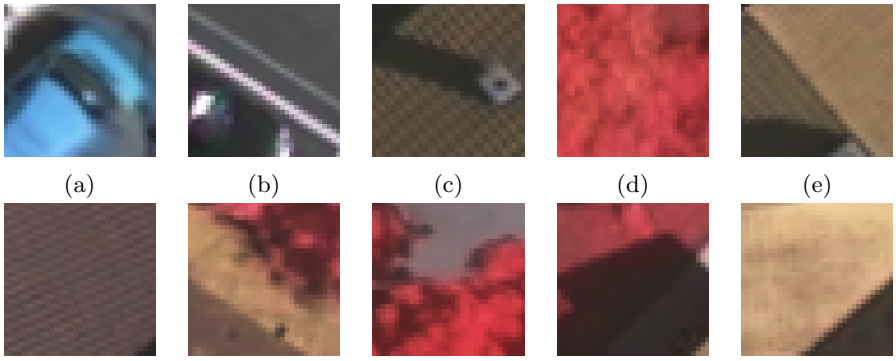


Fig. 7: Sample of 33x33 pixel patches extracted from the Vaihingen dataset.

A total of 25 of the original 33 Vaihingen tiles were identified for extracting training images. An overlap (or stride) of 18 pixels was used for this process, resulting in 372884 training images.

For extracting testing image patches, two adjoining tiles from the Vaihingen dataset were identified, composing of a approximately a 2500x4000 pixel area. In order to obtain a prediction for every pixel in test tiles, different extraction techniques were used for patch-based and encoder-decoder techniques. For patch-based approaches, a stride of 1 was used when extracting the 33x33 image patches, resulting in a patch for every pixel in the image and 9358992 test images. For encoder-decoder architectures, the test tiles segmented into 33x33 patches without any overlap, therefor a stride of 32 is used, resulting in 9322 test images.

3.1.2 Development of Denver City Dataset For the purpose of this study, a bespoke dataset was created using the Google Earth Engine. This involved several researchers manually labelling a sample of points dispersed over a Google Earth aerial map of Denver, Colorado. An interface was developed using GEE's web-based code editor and the Java script programming language. This allowed participants to log into GEE with Google credentials and apply one of three labels : tree/non-tree/unsure to a number of pre-defined points in the city. After all participants completed the labeling process, a second script extracted 33x33 image patches surrounding each labeled point. Labels for these patches were accessed through the python API in the main code file and any patches with a label of 'unsure' were removed. A total of 6895 image patches were extracted. Unlike the Vaihingen dataset, the ground truth of the Denver consists of a single label for each patch rather than semantically segmented label encoded images, therefore the Denver dataset is unsuited for encoder-decoder techniques. A sample of the Denver dataset is presented in figure 8.

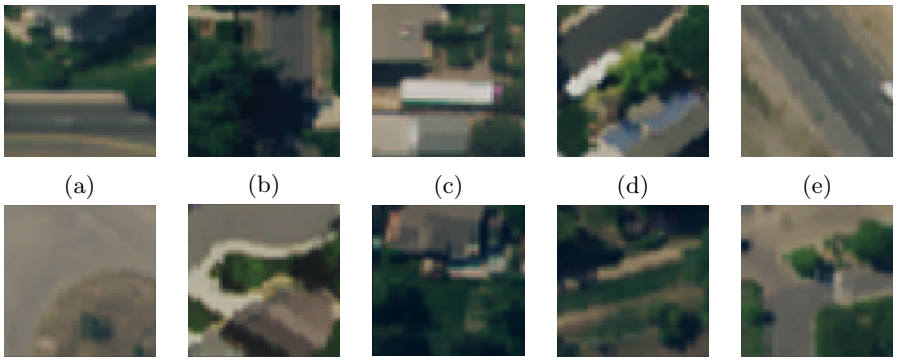


Fig. 8: Sample of images extracted from the Denver dataset.

3.2 Software tools

Development of the project code base involved the use of a number of software tools and libraries:

Google Earth Engine The Denver dataset was generated using the Google Earth Engine platform, a cloud-based software developed for geospatial data processing and analysis. The platform provides access to a web-based code editor where users can access data using the Java programming language. In addition to a Java tool, Google provide a python API to access functionality of the platform. For development of the Denver dataset, the code editor was used for the labeling process and creating and extracting image patches while python API was also used to access the label values for each patch.

Python Libraries The Python programming language (ver 3.6) was used to develop the project code base. All models architectures were developed using a combination of the Tensoflow [30] and Keras [31] libraries. Keras is a high-level python API which was used to build and train CNNs with Tensorflow running in the back-end carrying out low level tasks. All image processing made use of the OpenCV python library. Lastly the web based Jupyter notebook code editor was used to run and develop code.

Computational Resources CNNs require a high level of processing power particularly when working with a large dataset. All models were trained on a shared virtual server (model: Intel(R) Xeon(R) CPU E5-2695 v4 @ 2.10GHz) with 72 CPU cores and approx. 500G of RAM.

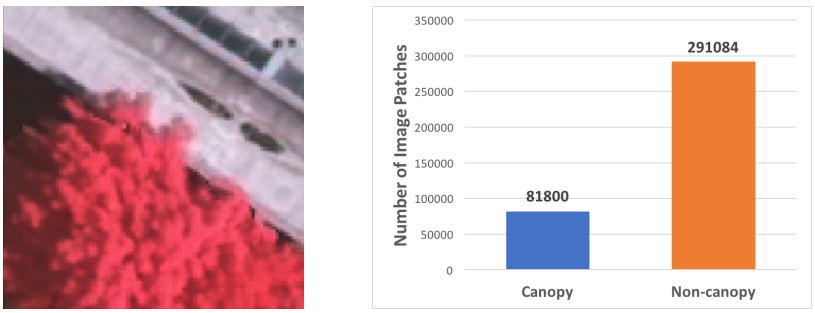
3.3 Experimental Design

A total of three experimental procedures are outlined in the following sections. Hyper-parameters were chosen for all experiments based on guidance from previ-

ous studies in the literature. Ideally, the process of identifying the most efficient hyper-parameters would have involved performing grid searches, however, this was not practically feasible given the time and computational resources required. Similarly, a decision to refrain from the use of cross-validation was influenced by the training times and computational resources required for the deep architectures of the chosen CNN models and the large quantity of data. Rather, a percentage (30% across all models) of training samples were withheld for validation at each training epoch.

3.3.1 Experiment 1: Patch-based CNNs for Vaihingen Dataset

Data pre-processing Each 33x33 image patch was loaded using the openCV python library. The key data pre-processing step for the patch-based approach was to identify the the class label associated with a training image. This was achieved by calculating the co-ordinates of the centre pixel and identifying the label of this pixel from the ground truth. If this pixel was green, the label assign to the patch was canopy; if the pixel was any other color, the label assigned was non-canopy. The class label balance for the training set of images is presented in figure 9b, and an example of an training image patch with the label canopy is displayed in figure 9a.



(a) (b)

Fig. 9: (a) Training example with canopy label for patch-based approach, (b) Class label balance of Vaihingen training data

LeNet model The LeNet CNN was coded manually using the Keras sequential model library. The model replicated the original LeNet with one alteration to the size of the input layer. The input layer was reduced to 33x33 to match the input size of image patches.

Pre-trained VGG16 model The VGG16 model implemented for this study made use of model weights that were pre-trained on the Imagenet dataset. These were

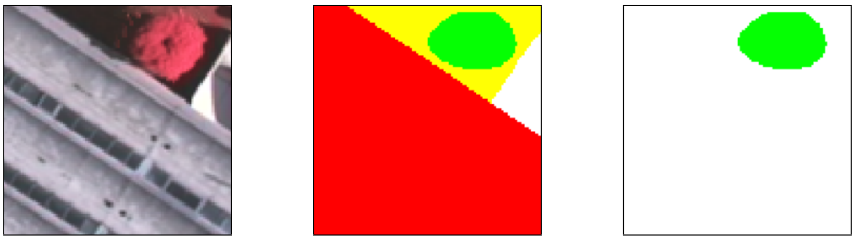
accessible through the Keras python library along with a pre-built VGG16 CNN architecture. Due to a size restriction for the pre-built VGG16 Keras model, the training image patches were expanded to a resolution of 48x48 pixels. The VGG16 training parameters are displayed in table 1. The optimizer used was Stochastic gradient descent (SDG).

Table 1: Training parameters for patch-based models.

Model	Learning Rate	Loss Function	Optimizer	Batch Size
LeNet	.000001	Binary crossentropy	Adam	100
Pretrained VGG16	.0001	Binary crossentropy	SGD	128

3.3.2 Experiment 2: Encoder-decoder CNNs for Vaihingen dataset

Data pre-processing For encoder-decoder CNN architectures, the dataset contained the same 33x33 images as used previously for the patch-based implementation. However, for this segmentation technique, the label for a training example consisted of color coded images representative of the corresponding image patch. This differs from patch-based methods where a single binary label was assigned to each image. Another notable difference for encoder-decoder training methods was that the conversion to binary labels process was carried out after training. Therefore, the model was trained with all of the original classes of the Vaihingen dataset (buildings, canopy, etc.), and the conversion to binary labels was carried out for the testing dataset only. This is illustrated in figure 10, whereby canopy is represented by green pixels and all other classes represented by white pixels.



(a) (b) (c)

Fig. 10: (a) Image patch from Vaihingen dataset, (b) ground truth if patch in (a) was training data, (c) ground truth if patch in (a) was testing data.

Encoder-decoder architectures A simple autoencoder with four convolutional layers and a replica of the UNet model was developed for segmentation of the Vaihingen dataset. Image patches were resized to 32x32 for both model inputs. The

up-sampling layers of the encoder-decoder architectures required input values of even numbers in order to reconstruct an output of the same dimensions using the UNets convolutional kernel sizes. Training Parameters for encoder-decoder methods are presented in table 2.

Table 2: Training parameters for encoder-decoder models.

Model	Learning Rate	Loss Function	Optimizer	Batch Size
4-layer encoder-decoder	NA	Mean squared error	Adadelta	120
UNet	.0002	Mean squared error	Adam	112

3.3.3 Experiment 3: Patch-based CNNs for Denver Dataset

Data preparation It was observed during data collection that the dataset contained significantly more cases of 'non-canopy' data. This imbalance is presented in figure 11. To counteract effects caused by this imbalance, a technique called Synthetic Minority Over-sampling Technique (SMOTE) was used [32]. This technique generates synthetic examples of the under represented class, resulting in an equal number of class samples in the dataset. Figure 12 shows examples of synthetic up-sampled images with canopy labels in comparison to original samples. This technique was used to augment the dataset for all model architectures.

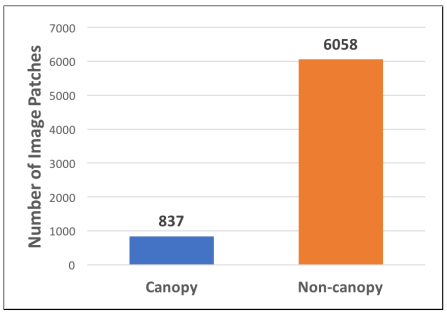


Fig. 11: Class label balance of Denver dataset

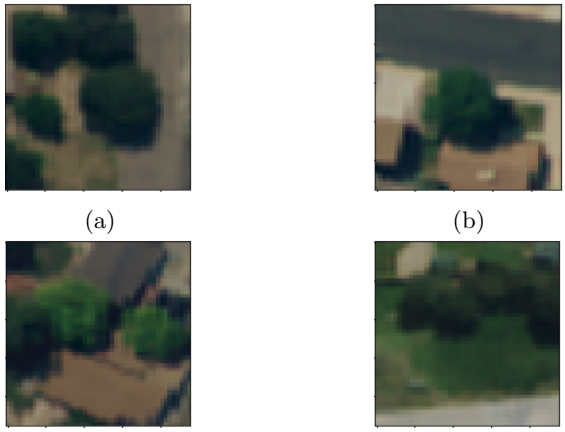


Fig. 12: Application of SMOTE to improve balance on test image data: (a,b) original images from Denver image dataset, (c,d) Synthetic images for the Denver dataset using SMOTE

Training Parameters For the Denver dataset models, a test/train split of .75/.25 percent respectively was used for both patch-based models. Experimentation for the Denver dataset involved the use of the pre-trained VGG16 and LeNet CNN architectures previously implemented on the Vaihingen dataset. Training parameters for each model are identical to that of patch-based models for experiment 1, displayed in table 1.

3.4 Evaluation Techniques

For each experiment, precision, recall and F1-scores were computed [33]. These were used to calculate the macro average; this value takes an average over both canopy and non-canopy class labels, and helps to remove the bias towards the higher represented classes (i.e. non-canopy) in the training set.

Precision Precision represents the proportion of examples that are correctly classified for a class against the total correctly classified examples. High precision values indicate a low false positive rate.

Recall Recall, also referred to as 'sensitivity' is the ratio of the number of true positive predictions of a class to the actual number of positive examples of that class. Recall represents the sensitivity of the model i.e. out of all the actual positives how many were caught by the model.

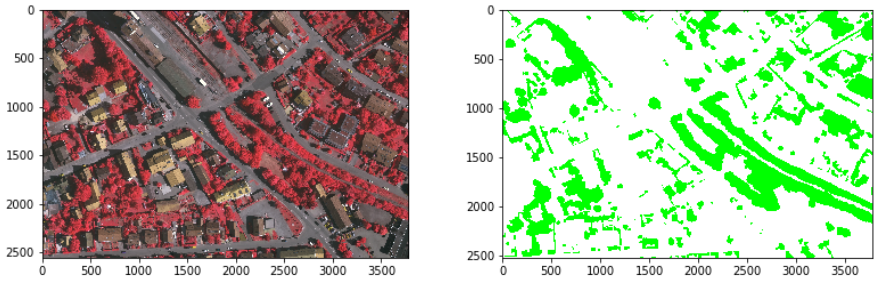
F1-Score The F1-score is simply a weighted average of recall and precision.

4 Results

The following sections outline results of the varying model architectures on various dataset, including performance metrics (Precision, Recall and F1-Score) and color-coded image reconstructions of the Vaihingen test tiles. The code for the experiments can be viewed and accesses on the github platform at the repository [3]

4.1 Vaihingen Dataset

Two tiles spanning the center area of the city were used as the test area for both patch-based and encoder-decoder methods. This area was chosen as it contained representative elements of many aspects of the wider city area. Additionally, visual inspection identified substantial dispersion of canopy (see figure 13), which further indicated that the area would provide a challenging test case. Performance results and confusion matrices for classification are presented in tables 3 and 4 respectively. Reconstructions of the test tiles were produced from the predictions of each of the trained model. These representations are presented in figure 14).



(a) Aerial image (b) Ground truth

Fig. 13: Aerial image and ground truth for test area.

[3] Access to the code is provided at the following Github repository

Table 3: Performance results on the Vaihingen test dataset .

Model	Precision	Recall	F1-Score
LeNet	0.9100	0.9017	0.9057
Pre-trained VGG16	0.8844	0.8989	0.8913
4-layer encoder-decoder	0.8716	0.6398	0.6631
UNet	0.8714	0.6007	0.6078

Table 4: Confusion Matrices for Vaihingen dataset

		Predicted	
		Non-canopy	Canopy
True	Non-canopy	6689190	295981
	Canopy	368562	2020403

(a) LeNet

		Predicted	
		Non-canopy	Canopy
True	Non-canopy	6416847	459193
	Canopy	319203	2039213

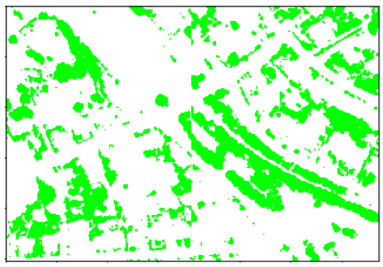
(b) Pre-trained VGG16

		Predicted	
		Non-canopy	Canopy
True	Non-canopy	696639	1741136
	Canopy	43743	7064210

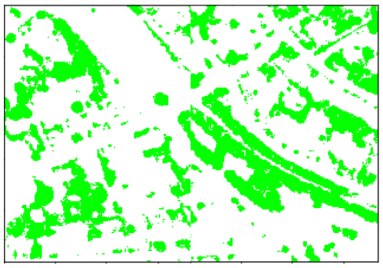
(c) 4-layer encoder-decoder

		Predicted	
		Non-canopy	Canopy
True	Non-canopy	420858	1939099
	Canopy	22082	7085871

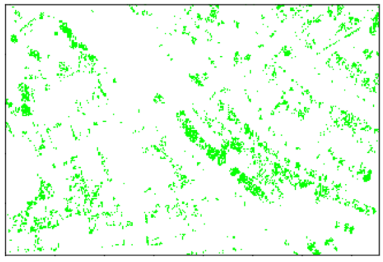
(d) UNet



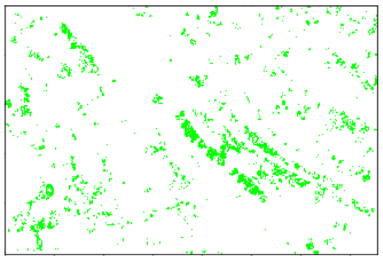
(a) LeNet



(b) Pre-trained VGG16



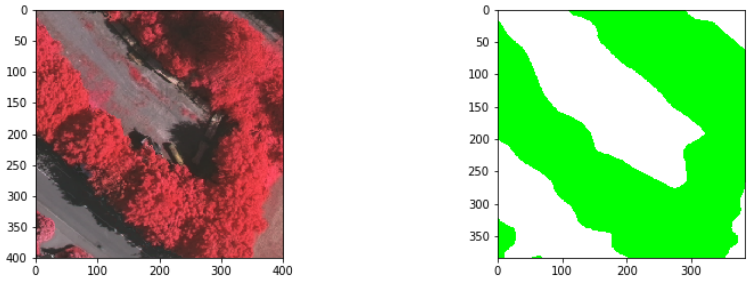
(c) 4-layer encoder-decoder



(d) UNet

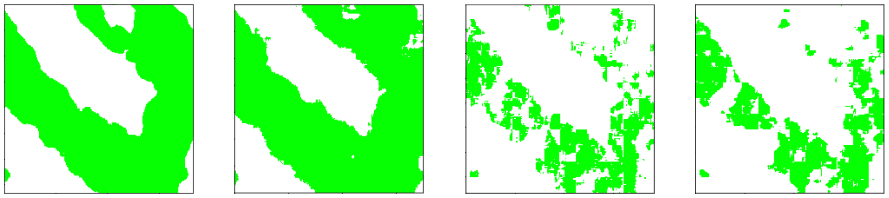
Fig. 14: Segmentation results of models test area of the Vaihingen data.

In order to visually inspect segmentation results at a higher resolution, a small area of the test tile encompassing 384x384 pixels was identified and expanded. The sample area presented in figure 15. Reconstructions produced by each model for the sample area are displayed in figure 16. From these images, it is clear that the patch-based approaches produced a more accurate representation to the original ground truth of the sample. Numerical performance results further indicate this (see table 5)



(a) Aerial image of sample area (b) Ground truth of sample area

Fig. 15: Image and ground truth for sample patch.



(a) LeNet (b) VGG16 (c) 4-layer encoder (d) UNet

Fig. 16: Segmentation results of models for sample area.

Table 5: Performance results for sample patch.

Model	Precision	Recall	F1-Score
LeNet	0.9402	0.9431	0.9415
Pre-trained VGG16	0.9359	0.9163	0.9241
4-layer encoder-decoder	0.7611	0.7066	0.6453
UNet	0.7677	0.7135	0.6530

4.2 Denver Dataset

Resulting model test metrics for the Denver dataset are presented in tables 6 and 7 respectively. The VGG16 architecture produced the highest scores overall across all metrics.

Table 6: Test results of patch-based models on the Denver dataset.

Model	Precision	Recall	F1-Score
LeNet	0.5823	0.5031	0.4763
Pre-trained VGG16	0.7840	0.7940	0.7889

Table 7: Confusion Matrices for Denver dataset

		Predicted	
		Non-canopy	Canopy
True	Non-canopy	1535	6
	Canopy	180	3

(a) LeNet

		Predicted	
		Non-canopy	Canopy
True	Non-canopy	1421	86
	Canopy	77	140

(b) Pre-trained VGG16

5 Discussion

It is clear from observation of the performance results for the Vaihingen dataset (see table 3), CNNs show promising potential for the detection of canopy in urban areas. The maximum F1-score, produced by the LeNet (0.91) on the Vaihingen dataset, exceeds that of traditional methods which are typically in the range 0.85 to 0.9 [24], observed previously in the literature for the general task of canopy detection. Results demonstrated that CNNs have the capability of performing the task of pixel-wise classification of canopy coverage to a high degree of accuracy, comparable with ground truth values in many instances. Despite achieving high levels of performance on the Vaihingen dataset, the LeNets model accuracy was greatly reduced when trained and evaluated on the Denver data. This is most likely due to a reduced number of examples in the dataset. Improved performance on the Denver dataset was observed using the pre-trained VGG16; this suggests that pre-training can help improve performance for small datasets. The proceeding sections discuss effects of implementing differing architecture approaches, advantages of pre-trained models, and examine model errors.

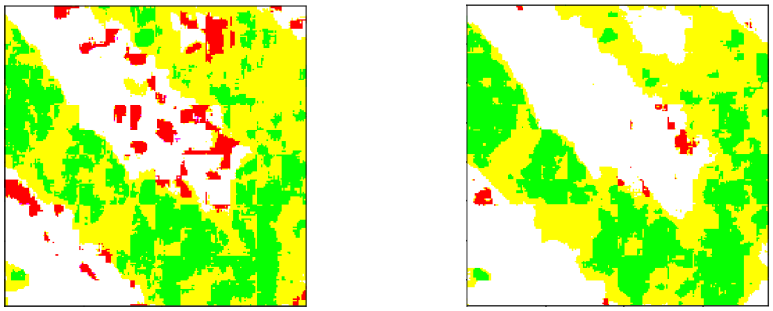
5.1 Patch-based Vs. Encoder-decoder Approaches

The task of canopy coverage measurement at a fine-grained pixel level provides a useful avenue for comparison of the differing architectures of patch-based and encoder-decoder CNNs. The effectiveness of both methods are compared and evaluated under by several headings below.

5.1.1 Performance High prediction and recall scores indicate predictions are correct and successfully identify canopy/non-canopy pixels. Performance results on the Vaihingen test data (see table 3) indicate patch-based approaches achieved greater accuracy than the encoder-decoder approach. While both approaches fared similarly in precision ($\mu = 0.88 \pm 0.018$), the performance difference was notable for recall scores, where LeNet (0.9017) and VGG (0.8989) was significantly higher than UNet (0.6007) and the 4-layer encoder-decoder (0.6398). This implies that for encoder-decoder approaches, predictions of canopy are exact, however, the coverage of predictions is lower than the patch-based models. It is perhaps more beneficial to compare both approaches in terms of F1-score, as this value takes into account precision and recall, both of which are important metrics for this classification task. Patch-based approaches again out perform encoder-decoder approaches with LeNet and VGG16 producing F1-Scores of 0.9057 and 0.8913 respectively and the 4-layer encoder-decoder and UNet obtaining 0.6631 and 0.6077 respectively.

Further insights into the differences between these methods can be gained by observation of the 400x400 sample patch presented in figure 16. Examination of the reconstructed images shows a definitive contrast between the two approaches. While patch-based approaches closely mirror the ground truth presented in figure 15, the encoder-decoder models produce a more speckled, less defined representation. This decrease in performance for the sample patch is further demonstrated by results in table 5), where precision, recall and F1-scores are lower for both encoder-decoder models in comparison to patch-based. Further investigation of this disparity indicates that the reduced performance is a result of models miss-classifying areas of low vegetation for the canopy class. Figure 17 shows encoder-decoder predictions inclusive of classes originally present in the ground truth images of the Vaihingen dataset. The 'low vegetation' class is color coded as yellow and can be seen, for both models, to predominantly consume pixels where the true value should identify canopy. It is intuitive that the canopy class is misclassified as low vegetation as both objects are visually similar. This misclassification is common amongst tree identifying classifiers [34]. It is possible this misclassification rate could be reduced if the models were trained without including the original labels, like 'low vegetation' of the Vaihingen dataset.

5.1.2 Computational Cost In addition to performance, an important factor in identifying an appropriate model is the time and computation cost associated with each method. This metric is especially important in the field of remote sensing, which usually involves manipulation of data in high volumes. The encoder-decoder predicts all pixels in a 32x32 patch in one pass through the network, so



(a) 4-layer encoder-decoder (b) UNet

Fig. 17: Encoder-decoder predictions including all original classes.

the total passes for the 400x400 sample patch equate to 144 (excluding padding). However, implementation of a patch-based method requires one pass to predict just one pixel, requiring a total of 147,456 (excluding padding). This results in a large disparity of training time and computational resources consumed by the two model approaches with patch-based models requiring a higher capacity. Furthermore, training time (in seconds) per epoch for patch-based approaches was 500 and 1600 for LeNet and VGG16 models respectively, and approximately 250 for both the UNet and 4-layer encoder-decoder.

5.1.3 Labelling Efforts An important difference between designing an architecture for segmentation is the labeling process involved to obtain a training dataset. First of all, in order to open the design process to include encoder-decoder architectures it is necessary for a ground truth label to consist of image patch where every pixel in the image pertains to a class, this is referred to as a segmented ground truth. Recall the encoder-decoder methods were not applied to the Denver data, as the ground truth was not present as a segmented image. In contrast, patch-based methods exclusively require the label of the center pixel of a patch to represent its class. Producing a dataset of segmented ground truth is a more rigorous and time consuming task, as labelers must precisely outline borders of every object in an image. Alternatively, producing labels for the Denver dataset required labelling single points, which was less intensive and produced examples at a quicker rate. For the Denver dataset, in most cases, participants produced 500 labeled points in under an hour.

5.2 Pre-training Performance

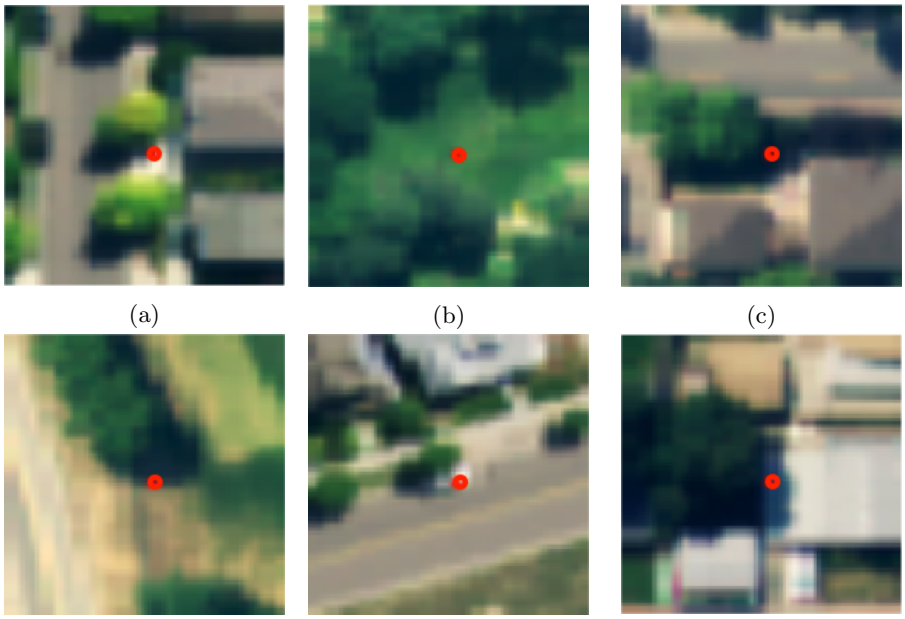
5.2.1 Vaihingen dataset Previous research suggests that pre-trained networks can improve performance, particularly when datasets are small [17]. However, in the case of the Vaihingen dataset, this effect was not observed. The LeNet model produced slightly higher score across all three performance metrics in comparison to the pre-trained VGG16. It is important to note that grid

searches or cross-validation techniques were not implemented for either methods, which may explain why the pre-trained VGG16 demonstrated lower performance in comparison to the fully trained LeNet. Although the LeNet had a higher success rate in terms of performance metrics examining the reconstructions of the sample patch in figure 16, the VGG16 is the only model to identify an area of tree covered by shadow. This indicates it is possible, with some post-processing techniques, the VGG16 model could obtain a higher final F1-score.

5.2.2 Denver dataset The total number of training examples for the Denver dataset was initially 5171, which was then increased to 9088 through a SMOTE upsampling process. In comparison to the Vaihingen dataset and machine learning domain standards, this is considered small and could explain some of the low performance metrics demonstrated by the LeNet model (see table 6) . In contrast, the pre-trained VGG16 displayed substantially higher scores across all performance metrics. It appears the pre-trained weights of early layers alleviate the effects associated with low training examples, and resulted in a model capable of reasonably impressive predictive power.

5.3 Prediction Errors

Development of the Denver dataset required human annotators to apply a label to a point from a Google earth engine electronic map. Although an option of 'Unsure' was provided to participants, it is possible some labels suffered from response bias, whereby ambiguous labels were recorded as either canopy/non-canopy resulting in the dataset suffered from labeling errors. To identify if this anomaly occurred and effected performance scores, misclassified test examples of both models were studied. Figure 18 shows a sample of some ambiguous points that were incorrectly classified as non-canopy by our model yet participants labeled as canopy. This analysis revealed common sources of error to be if a point lies on the border of canopy and non-canopy (see figure 18a, 18d , 18e), within an area of shadow (see figure 18c, 18f), or over areas of other vegetation (see figure 18b).



(a) (b) (c)

(d) (e) (f)

Fig. 18: Examples from Denver test dataset misclassified as non-canopy by LeNet and VGG16 models

It is probable that these points should be identified as 'unsure' and consequently have effected the final classification scores. A possible solution would be to audit the labeling instructions to warn participants of these issues and have a sample of data annotated by a number of participants to confirm labels are correct.

6 Conclusion

This research highlights the potential for improved automation in the field of remote sensing. It addresses the need for accurate classification of canopy coverage in urban areas, which has important applications in large-scale urban planing and environmental monitoring. The research pioneered the use of two distinct CNN approaches: patch-based architectures and encoder-decoder architectures. Experiments performed two variants of both methods to the problem of canopy coverage for a dataset containing aerial images of the German town of Vaihingen. It was found that the patch-based approaches outperformed the encoder-decoder approaches by a significant margin, with an LeNet model demonstrating the highest performance values overall in terms of F1-score (0.91). A follow on study using a smaller dataset was investigated. This dataset was constructed, using the Google Earth Engine platform, exclusively for this study and focused on the town of Denver, Colorado. Analysis of patch-based approaches on this dataset found that a pre-trained VGG16 produces substantially higher performance levels than a fully trained LeNet.

These findings provide strong evidence of the high performance potential of CNN architectures, although they also suggest the need for further research. Due to time and computational constraints, it was not feasible to perform grid searches on any of the models. Model performance could potentially improve if optimal model parameters were identified through these searches. Another avenue for improvements could be obtained through post processing techniques. Dilation and erosion techniques are commonly use to add definition to object boundaries by filling in holes/gaps in images and separating objects that overlap. These techniques often increase model accuracies of final predictions. Lastly, it is common known that performance of CNNs is sensitive to the size of the dataset. Potentially increasing the number of examples in the dataset could add further improvement. It is hoped that results and techniques are used for future study in the area of deep learning for remote sensing applications.

Reflection Working on this project has generally been a highly enjoyable and rewarding experience. Looking back, not only has it afforded me the opportunity to work on a meaningful research project for the first time, but it also provided the chance to develop a different type of problem-solving skills than were customary during structured project modules. Furthermore, I was able to further develop both communication skills, as well as technical skills in machine learning. Going forward, I am confident that these latter two aspects will be particularly beneficial to my professional career. In the remainder of this section, I will elaborate on

three important lessons I have learnt over the course of my research. First, the process of identifying, assessing, and preparing the dataset is a critically important task and one that should not be overlooked. I gained this insight relatively early in the project, when there were some difficulties in attaining the datasets to use in this research. This ultimately led to some some stress, uncertainty, and time loss which may have been avoided with better preparation.

Second, even with use of a state-of-the-art computer infrastructure, training CNN models in some cases took up to 2 or 3 days. This made performing tests a time consuming process, and forced me to take additional care in preparing the experiment. Also, as computer was a shared resource, I was conscious of other researchers who may need to use it. If this was not the case, it may have been possible to perform more experiments, extend the dataset, or to conduct more thorough training. Even still, it was enjoyable to work in a professional research environment, and I am very grateful for the opportunity to make use of this resource for my project.

Finally, the months since beginning the project have elapsed very quickly and I am conscious of how short the time really has been to familiarise myself with the research topic, up-skill in the necessary technical areas, conduct the research, and write the thesis among other things. I feel that much of the progress that I was able to make was attributable to good planning, working consistently, and regular meetings with my supervisor. I am very grateful to my supervisor for always being able to find time to meet with me, and for the support they consistently showed throughout the project.

7 Acknowledgments

The author thanks Dr. Brian Mac Namee who provided insight and expertise that greatly assisted the research, Dr. Conor Nugent and Paul Hickey from Breadboard Labs who provided the motivation behind the project and the development of the Google Earth Engine applications for labeling and exporting the Denver dataset and Tanushree Biswas from The Nature Conservancy who provided insight and resources throughout the project and lastly to all participants who helped label the Denver dataset.

References

1. N. Gorelick, M. Hancher, M. Dixon, S. Ilyushchenko, D. Thau, and R. Moore, "Google earth engine: Planetary-scale geospatial analysis for everyone," *Remote Sensing of Environment*, 2017.
2. "Canopy quantification via image classification."
3. H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1520–1528, Dec 2015.
4. E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 640–651, April 2017.

5. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.
6. C. Liu, L. Hong, S. Chu, and J. Chen, "A svm ensemble approach combining pixel-based and object-based features for the classification of high resolution remotely sensed imagery," in *2014 Third International Workshop on Earth Observation and Remote Sensing Applications (EORSA)*, pp. 140–144, June 2014.
7. V. Mnih and G. E. Hinton, "Learning to label aerial images from noisy data," in *Proceedings of the 29th International conference on machine learning (ICML-12)*, pp. 567–574, 2012.
8. W. Hu, Y. Huang, W. Li, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *Journal of Sensors*, 2015.
9. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.
10. J. Zhao, W. Guo, S. Cui, Z. Zhang, and W. Yu, "Convolutional neural network for sar image classification at patch level," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 945–948, July 2016.
11. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, pp. 84–90, May 2017.
12. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.
13. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016.
14. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, June 2015.
15. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
16. J. Y. Choi, T. K. Yoo, J. G. Seo, J. Kwak, T. T. Um, and T. H. Rim, "Multi-categorical deep learning neural network to classify retinal images: A pilot study employing small database," *PLoS One*, vol. 12, 11 2017. Copyright - 2017 Choi et al. This is an open access article distributed under the terms of the Creative Commons Attribution License: http://creativecommons.org/licenses/by/4.0/ (the License), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2017-11-03.
17. K. Nogueira, O. A. Penatti, and J. A. dos Santos, "Towards better exploiting convolutional neural networks for remote sensing scene classification," *Pattern Recognition*, vol. 61, pp. 539 – 556, 2017.
18. T. Sugata and C. Yang, "Leaf app: Leaf recognition with deep convolutional neural networks," in *IOP Conference Series: Materials Science and Engineering*, vol. 273, p. 012004, IOP Publishing, 2017.
19. A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. G. Rodríguez, "A review on deep learning techniques applied to semantic segmentation," *CoRR*, vol. abs/1704.06857, 2017.

20. O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.
21. S. Jennings, N. Brown, and D. Sheil, "Assessing forest canopies and understorey illumination: canopy closure, canopy cover and other measures," *Forestry: An International Journal of Forest Research*, vol. 72, no. 1, pp. 59–74, 1999.
22. "Measuring the tree canopy cover in london: An analysis using aerial imagery," Tech. Rep. MSU-CSE-06-2, London SE1 2AA, September 2015.
23. D. Stojanova, P. Panov, V. Gjorgjioski, A. Kobler, and S. Deroski, "Estimating vegetation height and canopy cover from remotely sensed data with machine learning," *Ecological Informatics*, vol. 5, no. 4, pp. 256 – 266, 2010.
24. E. Guirado, S. Tabik, D. Alcaraz-Segura, J. Cabello, and F. Herrera, "Deep-learning versus obia for scattered shrub detection with google earth imagery: Ziziphus lotus as case study," *Remote Sensing*, vol. 9, no. 12, p. 1220, 2017. Copyright - Copyright MDPI AG 2017; Last updated - 2018-01-18; SubjectsTermNotLitGenreText - Ziziphus lotus.
25. S. Basu, S. Ganguly, S. Mukhopadhyay, R. DiBiano, M. Karki, and R. Nemani, "Deepsat: A learning framework for satellite imagery," in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '15, (New York, NY, USA), pp. 37:1–37:10, ACM, 2015.
26. X. Chen, S. Xiang, C. L. Liu, and C. H. Pan, "Vehicle detection in satellite images by hybrid deep convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, pp. 1797–1801, Oct 2014.
27. E. Okafor, P. Pawara, F. Karaaba, O. Surinta, V. Codreanu, L. Schomaker, and M. Wiering, "Comparative study between deep learning and bag of visual words for wild-animal recognition," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, Dec 2016.
28. G. Cheng, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, pp. 7405–7415, Dec 2016.
29. I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raskar, "Deepglobe 2018: A challenge to parse the earth through satellite images," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.
30. M. Abadi and A. A. and, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.
31. F. Chollet *et al.*, "Keras." `https://github.com/fchollet/keras`, 2015.
32. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.
33. J. D. Kelleher, B. M. Namee, and A. D'Arcy, *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. The MIT Press, 2015.
34. S. Paisitkriangkrai, J. Sherrah, P. Janney, and A. Van-Den Hengel, "Effective semantic pixel labelling with convolutional networks and conditional random fields," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2015 IEEE Conference on*, pp. 36–43, IEEE, 2015.