

**Integrating Convolutional
Neural Networks and
Conditional Random Field
Inference for Land-Cover
Classification of Multi-Scale
Remote-Sensing Images**

AMAN GUPTA

February, 2018

SUPERVISORS:

Dr. Claudio Persello

Dr. Valentyn Tolpekin



Integrating Convolutional Neural Networks and Conditional Random Field Inference for Land-Cover Classification of Multi-Scale Remote-Sensing Images

AMAN GUPTA

Enschede, The Netherlands, [February, 2018]

Thesis submitted to the Faculty of Geo-Information Science and Earth
Observation of the University of Twente in partial fulfilment of the
requirements for the degree of Master of Science in Geo-information
Science and Earth Observation

Specialization: Geoinformatics [GFM]

SUPERVISORS:

Dr. Claudio Persello (1st Supervisor)

Dr. Valentyn Tolpekin (2nd Supervisor)

THESIS ASSESSMENT BOARD:

Dr. Alfred Stein (Chair)

Dr. Begüm Demir (External Examiner, University of Trento)

ABSTRACT

The abundant availability of multi-scale very high-resolution (VHR) remote-sensing imagery precipitated the need for an image classification method to automatically learn relevant features from raw images and make land-cover class predictions in an end-to-end framework. For this, deep learning methodologies were adapted, specifically in the form of convolutional neural networks or CNNs. In this research, we investigate the ability of CNN structures to dynamically fuse multi-scale VHR satellite images, taken by the WorldViewIII sensor over parts of Quezon City, Mexico, to produce high-accuracy land-cover classification maps. We do so by comparing the classification results of a CNN that is designed to fuse discrete sets of multi-scale images, with the results of a CNN that was trained on images that were pan-sharpened using an independent state-of-the-art technique. In addition, this research applies the mean-field approximation to the dense-CRF inference which models label and spectral compatibility in the pixel-neighbourhood effecting a more locally, smooth segmentation. Finally, a recurrent neural network or RNN is also implemented in order to enhance the stand-alone fusion network by accounting for local class-label dependencies within its end-to-end structural framework by concatenating the class-probability scores of the first FCN with the raw input of the following FCN instance. We compare the dense-CRF optimized classification results with those obtained by applying the trained RNN classifier. Classification results indicate that our designed classification FCN models have performed favourably. The fusion network consistently produces high-accuracy land-cover classification maps effectively demonstrating the ability of FCN structures to dynamically fuse multi-scale images. Furthermore, the RNN results show an improvement on the singular fusion network performance. This illustrates the capacity of FCN architectures to account for local spatial and spectral dependencies in an image space, when implemented as a RNN.

Keywords: *Image classification, convolutional neural networks, conditional random fields, very high resolution satellite imagery*

ACKNOWLEDGMENTS

I want to thank my supervisors Dr. Claudio Persello and Dr. Valentyn Tolpekin for their guidance and support through this process of research and documentation. I also want to acknowledge John Ray Bergado for his feedback and help throughout this research period.

I want to particularly thank the GeoInformatics department for sharing all the knowledge and expertise and making these 18 months a challenging intellectual endeavour. I want to also thank my friends for their company and support throughout this process. Before last I would also like to thank the network of colleagues and the larger body of staff at ITC (and at the ITC Hotel) for having provided comfortable, dependable-like-clockwork conditions that were suited for work and study.

Finally, I want to thank Dr. Tsegaye Nega, Dr. Shashikant Gupta, Dr. Laveesh Bhandari and Tisha Sehdev for encouraging my interest in the study of GIS and remote-sensing based information systems.

TABLE OF CONTENTS

1.	Introduction.....	1
1.1.	Research Motivation.....	1
1.2.	Research Identification.....	3
1.3.	Research Objectives.....	3
1.4.	Research Questions	3
2.	Literature Review	5
2.1.	Related Work	5
2.2.	A Brief Overview of Convolutional Neural Networks	7
2.3.	A Brief Overview of Markov and Conditional Random Fields.....	10
3.	Methodology.....	13
3.1.	Training and Configuring – SimpleNet (SNN)	15
3.2.	Training and Configuring - FuseNet	18
3.3.	Dense-CRF – Parameter Optimization.....	22
3.4.	Training and Configuring – ReUseNet	23
4.	Data.....	25
4.1.	Data Preparation	26
4.2.	Images For Testing Classification Accuracy	27
5.	Parameter Sensitivity Analysis	28
5.1.	FCN Architectural Parameters	28
5.2.	FCN Hyper-Parameters	29
5.3.	Dense-CRF Parameters.....	29
6.	Results.....	32
6.1.	Results I	32
6.2.	Results II.....	36
7.	Discussion and Conclusion.....	41

LIST OF FIGURES

FIGURE 1: SCHEMATIC REPRESENTATION OF AN ANN-NEURON	7
FIGURE 2: SCHEMATIC REPRESENTATION OF A LINEARLY CONNECTED ANN	7
FIGURE 3: 1ST AND 2ND ORDER NEIGHBOURHOOD CLIQUE ARRANGEMENTS SOURCE: LI (2009)	11
FIGURE 4: RESEARCH METHODOLOGY FLOWCHART	14
FIGURE 5: SCHEMATIC REPRESENTATION FOR THE BASELINE FCN ARCHITECTURE	15
FIGURE 6: SCHEMATIC REPRESENTATION OF THE OPTIMIZED SIMPLE NEURAL NETWORK ARCHITECTURE	17
FIGURE 7: FUSENET ARCHITECTURE	18
FIGURE 8: A GRAPHICAL REPRESENTATION OF REUSENET	23
FIGURE 9: STUDY AREA AND LAYOUT OF TRAINING, VALIDATION AND TESTING TILES	25
FIGURE 10: A SAMPLE OF THE PANCHROMATIC (L), MULTISPECTRAL AND GROUND REFERENCE DATA	26
FIGURE 11: TRUE COLOUR COMPOSITE (R, G, B) OF TESTING TILES	27
FIGURE 12: PLOT (A): OA ON TILE 45 AND θ_A ; PLOT (B) OA ON TILE105 AND θ_A	30
FIGURE 13: PLOT (A): OA ON TILE 45 AND θ_B ; PLOT (B) OA ON TILE105 AND θ_B	31
FIGURE 14: SIMPLENET AND FUSENET CLASSIFICATIONS MAPS FOR TILE 32	34
FIGURE 15: SIMPLENET+DENSE-CRF AND REUSENET CLASSIFICATION MAPS FOR TILE 32	35
FIGURE 16: SIMPLENET AND FUSENET CLASSIFICATIONS MAPS FOR TILE 32 (RESULT II)	38
FIGURE 17: SIMPLENET+DENSE-CRF AND REUSENET CLASSIFICATION MAPS FOR TILE 32 (RESULT II)	39
FIGURE 18: VISUAL COMPARISON BETWEEN DENSE-CRF AND REUSENET	44

LIST OF TABLES

TABLE 1: BASELINE ARCHITECTURE AND HYPER-PARAMETER VALUES	15
TABLE 2: SIMPLENET EXPERIMENTS – PATCH SIZE AND LAYER-DEPTH	16
TABLE 3: SIMPLENET EXPERIMENTS – ARCHITECTURAL PARAMETERS	16
TABLE 4: SIMPLENET EXPERIMENTS – TRAINING EPOCHS, LEARNING RATE & WEIGHT DECAY	17
TABLE 5: FUSENET EXPERIMENTS - DS, FEMS & BN BLOCKS (FOR FEATURE SIZE = 32)	19
TABLE 6: FUSENET EXPERIMENTS - BN (SMALLEST FEATURE SIZE = 16).....	20
TABLE 7: EXPERIMENTS - BN (SMALLEST FEATURE SIZE = 8).....	20
TABLE 8: FUSENET EXPERIMENTS – US	21
TABLE 9: FUSENET ARCHITECTURE.....	21
TABLE 10: DENSE-CRF - BATCH 1 EXPERIMENTS.....	22
TABLE 11: DENSE-CRF - BATCH 2 EXPERIMENTS.....	22
TABLE 12: REUSENET EXPERIMENTS WITH ADDITIONAL LOSS	24
TABLE 13: RAW DATA - IMAGE CHARACTERISTICS.....	25
TABLE 14: NUMBER OF EXPERIMENTS PER PARAMETER-VALUE PAIRS IN BEST45 AND BEST105	29
TABLE 15: OVERALL ACCURACY (%) - TILE 32 AND TILE 103	32
TABLE 16: PERCENTAGE OF TOTAL PIXELS PER CLASS-LABEL PER CLASSIFICATION METHOD	32
TABLE 17: PRODUCER AND USER ACCURACY METRICS FOR TILE 32	33
TABLE 18: PRODUCER AND USER ACCURACY METRICS FOR TILE 103	36
TABLE 19: OVERALL ACCURACY (%) - TILE 32 AND TILE 103 (RESULT II)	36
TABLE 20: PERCENTAGE OF TOTAL PIXELS PER CLASS-LABEL PER CLASSIFICATION METHOD (RESULT II)	37
TABLE 21: PRODUCER AND USER ACCURACY METRICS FOR TILE 32 (RESULT II).....	37
TABLE 22: PRODUCER AND USER ACCURACY METRICS FOR TILE 103 (RESULT II)	40
TABLE 23: AVERAGE TRAINING TIME PER FCN.....	46

1. INTRODUCTION

1.1. RESEARCH MOTIVATION

The commercial availability of very high-resolution (VHR) satellite imagery raised expectations of achieving improved land-cover classification maps. However, the results obtained from applying standard non-contextual supervised classifiers, like Gaussian Maximum Likelihood, were less than satisfactory (Moser, Serpico & Benediktsson, 2013). This failing was largely attributed to the increased spatial resolution and the limited spectral resolution of VHR satellite sensors. In effect, the spectral variability within thematic land-cover classes increased while ‘class-separability’ decreased (Carleer, Debeir & Wolff, 2005). For VHR remote-sensing images there exists a high-degree of spatial correlation between neighbouring pixel values; and the colour and intensity of an individual pixel, irrespective of its context, is insufficient to determine its true class label. An established approach to account for this spectral complexity is to include hand-crafted features, like texture statistics, oriented gradients, and morphological profiles which tend to capture the local spectral variation around a given pixel (Volpi & Tuia, 2017). The nature of these descriptors is, however, controlled by a user-defined set of parameters; and their selection is usually guided by user-experience or by a trial and error procedure. These approaches are inherently suboptimal and non-exhaustive (Volpi & Tuia, 2017). Also, while hand-crafted features improve classification accuracy, their unbounded inclusion into the feature-set makes the model susceptible to the curse of dimensionality or Hughes’ phenomenon (Hughes, 1968). This is a significant limitation for both parametric and non-parametric supervised image classification methodologies; and accordingly justifies the need for a classifier that could automatically learn relevant features from a remote sensing data set.

Accordingly, deep learning methodologies were adopted - specifically in the form of Convolutional Neural Networks (CNNs) – for image classification. CNNs were initially conceived and adapted for pattern recognition tasks. LeCun et al. pioneered their application for handwritten character recognition, formalizing the multi-tiered design (dubbed LeNet-5) that has served as a prototype for all subsequent architectures (LeCun et al., 1989). Since then, CNNs have demonstrated outstanding performance in tasks as diverse as face recognition, natural language processing, audio recognition and bioinformatics (Fu, Liu, Zhou, Sun, & Zhang, 2017). Their application to image classification, however, was hampered for a number of years by the following two factors: (i) the unavailability of large semantically labelled training data, and (ii) the high computational costs involved in training the network (Zeiler & Fergus, 2014). In recent years, advances in GPU technology have made the training of CNNs more computationally feasible. In 2012, Alex Krizhevsky and his team crafted a CNN (since known as AlexNet) that demonstrated state-of-the-art performance at the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Krizhevsky, Sutskever & Hinton, 2012). This event proved a watershed for the application of CNNs for image classification.

It is worth emphasizing that image classification, as understood in the domain of *pattern recognition* and *computer vision*, is essentially image categorization, i.e., assigning a single class label to an entire image. On the other hand, in the field of remote-sensing image classification implies assigning a single class label to each pixel of the image. To obtain such dense ‘pixel-label’ outputs, CNN architectures have typically employed a ‘patch-based’ approach. This involves decomposing the original image into small patches of uniform size and using the CNN to predict a class label for the pixel centred on each patch. These patch label predictions are later re-joined to produce the desired ‘pixel-label’ map at the original image resolution. A significant shortcoming of this approach is the redundancy involved in processing overlapping patches which leads to high computational costs. Furthermore, the segmentation of the larger

image leads to a loss of useful information contained at the patch boundaries. In 2015, Long et al. addressed this shortcoming and proposed the Fully Convolutional Network (FCN), replacing the fully connected layers in a standard CNN with convolution layers (Long, Shelhamer, & Darrell, 2015). The FCN helped mitigate the aforementioned limitations of the ‘patch-based’ approach by (i) reducing the number of trainable parameters without resulting in a loss of generality; (ii) enabling feature learning over the entire image instead of independently conducted patch-wise inference, and (iii) allowing arbitrary sized images to be accepted as input, and doing away with the complexity of decomposing and rearranging the image and the output respectively.

The advantages of FCN notwithstanding, substantial challenges remain in trying to adapt them for land-cover classification of VHR remote-sensing images. Due to considerations of computational feasibility, convolutional filters are kept small in size in order to reduce the total number of trainable parameters. This constraint, when combined with repeated max-pooling operations, progressively coarsens the feature representations produced as the output of each convolution layer. In recent years numerous modifications have been proposed to enhance the classification accuracies of CNNs, such as, the use of dilated convolution filters and up-sampling ‘deconvolution’ operations. Despite these developments, CNNs are susceptible to reduced localization accuracy due to their inherent invariance to spatial transformations (Chen, Papandreou, Kokkinos, Murphy, & Yuille, 2015). Consequently, the inferred result at the original image resolution suffers from non-sharp boundaries and blob-like segmentations (Zheng et al., 2015).

In this context, Markov Random Fields (MRF) and its variant Conditional Random Fields (CRF) have been readily used to refine coarse pixel-level predictions and to produce sharp, fine-grained classification maps (Zheng et al., 2015). These are a class of probabilistic graphical models, usually applied as a post-processing step, which transform the semantic-labelling problem into a Bayesian maximum a-posteriori (MAP) inference, conditioned on a random field (group of pixels), constituting the Markovian neighbourhood of a given pixel (Krähenbühl & Koltun, 2011). While MRF models the spatial dependency between pixels considering only class labels, the CRF takes into account both class labels and spectral values (Li, 2009). The efficacy of CRF models is predicated on their ability to incorporate contextual information (such as the presence of edges, homogeneous image regions and texture) as a linear combination of pairwise energy potentials (Moser et al., 2013). This encourages class label agreement between spectrally similar valued pixels resulting in a smooth and homogenous output.

In applying CRF modelling to classify VHR imagery, one needs to account for long-range dependencies between pixel values in the image. In order to do so CRFs need to consider higher-order neighbourhoods, and go beyond the adjacency structure of commonly used pixel models (Krähenbühl & Koltun, 2011). Defining CRF models over the complete set of pixels in an image, however, exponentially increases the combination of pairs to consider and renders the procedure computationally prohibitive. In a breakthrough, a mean-field approximation to the fully-connected dense-CRF inference was developed with striking computation efficiency. This algorithm replaced the conventional posterior probability with a product of independent marginal distributions for each pixel in the entire image space; and further utilizes a contrast-sensitive two kernel potential and high-dimensional filtering to optimize the implementation of the dense-CRF inference (Krähenbühl & Koltun, 2011).

This research integrates various applied methods from across remote-sensing image-classification and RGB-oriented image-segmentation literature into a methodological framework to assess the efficacy of CNN structures to dynamically fuse multi-scale VHR images for generating high-accuracy land-cover classification maps. More specifically, this work combines the receptive-field-enlargement capacity of dilated convolutional filters and implements a ‘downsampling-upsampling’ architecture that allows for the dynamic fusion of MS and PAN image data in order to generate high-accuracy land-cover classification

maps. In addition, to further refine the classification maps this research applies the fully-connected dense-CRF inference algorithm as developed by Krähenbühl et al. (2011). This study also investigates the potential of using a recurrent network architecture or RNN to incorporate contextual neighbourhood information by co-joining two instances of an FCN into an end-to-end block of trainable parametric weights. In doing so, the research argues, the FCN-RNN implementation accounts for the spatial auto-correlation between image pixels and simulates the pairwise compatibility modelling of the CRF inference. All classification methodologies proposed herein are compared on the basis of overall classification accuracy and the computational cost of their implementation.

1.2. RESEARCH IDENTIFICATION

This research work builds on previous efforts to use deep fully convolutional architectures to detect informal settlements in VHR remote-sensing images as proposed by Persello and Stein (2017). The motivation stems from the tractability of the CNN modelling paradigm to allow for the construction of complex non-linear networks that can dynamically downsample and upsample image features within the network. The novelty of this research lies in the design and implementation of a FCN that can dynamically fuse multi-scale images for generating high-accuracy and large-scale land-cover classification maps. We further implement a recurrent network architecture that explicitly incorporates the class-probability scores of a FCN as an additional, explicit, input into a second instance of the same network and trains both network parameters simultaneously. In doing so, the first network learns class-label and contextual dependencies between image pixels via the parameter weights of the convolutional filters and the second network uses this additional information to predict higher accuracy classification maps. In this research, we have also applied the fully-connected dense-CRF inference, which was originally developed for segmenting ground-perspective RGB images for the land-cover classification of large-scale remotely-sensed images (Krähenbühl & Koltun, 2011). A list of research objectives along with specific research questions pertaining to each objective are provided below.

1.3. RESEARCH OBJECTIVES

1. Design and train a FCN that dynamically fuses *multi-scale* (multispectral - MS and panchromatic - PAN) VHR remote-sensing images.
2. Design and train a FCN-RNN that concatenates *two instances* of a FCN, allowing both network parameters to be trained simultaneously in an end-to-end block of learnable parametric weights.
3. Apply and assess the ability of fully-connected dense-CRF inference for refining the pixel-label classification of non-RGB remote-sensing VHR satellite images (Krähenbühl & Koltun, 2011).
4. Compare classification methodologies based on overall accuracy and computational feasibility.

1.4. RESEARCH QUESTIONS

Objective 1:

- How is the dynamic fusion of multi-scale images achieved within the FCN network design? What architectural parameters and network training hyper-parameters influence the success of applying FCN architectures for image fusion?

- What parameters enable the pixel-level classification of VHR remotely-sensed satellite imagery?

Objective 2:

- How is the FCN-RNN developed?
- What aspects of the FCN-RNN architecture motivate the comparison with the MRF/CRF inference?

Objective 3:

- How is the fully-connected dense-CRF implementation applied to the classification problem?
- What are the relevant inputs and parameters required for specifying the dense-CRF model? What is their impact on the classification results?

Objective 4:

- What is the difference in classification accuracy between a FCN trained on pan-sharpened images and a FCN that dynamically fuses discrete multi-scale images?
- What is the difference in classification accuracy between FCN+CRF and RNN?
- What is the difference in computational requirements between the proposed methods?

2. LITERATURE REVIEW

This chapter begins with a literature review of research applications of CNNs for image segmentation and image classification and focuses on studies that have used CNNs, MRFs and CRFs particularly for land-cover classification of VHR remote-sensing satellite images. The following sections present an overview of the design and mechanism of CNNs and MRF/CRF models and motivates the need for applying the fully-connected dense-CRF inference for refining CNN generated class-probability scores.

2.1. RELATED WORK

Convolutional Neural Networks (CNNs) were first conceived and developed for computer vision and pattern recognition tasks. One of their first applications was for the automatic recognition of handwritten characters (LeCun et al., 1989; Bengio et al., 1998). But after AlexNet (the CNN crafted by Krizshensky et al.) outperformed all other state-of-the-art classifiers at the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) of 2012, the application of CNNs for image classification and segmentation has become more widespread.

The AlexNet architecture, which was originally conceived for object recognition, consisted of 5 convolutional and 3 fully connected layers. Other notable design features included: (i) the utilization of the ‘drop-out’ strategy - to mitigate the tendency of CNNs to over-fit the training data; and (ii) the substitution of the sigmoidal nonlinear activation function with the rectified linear unit (or ReLU), which, being unsaturated, enables the efficient propagation of gradient descent through the network (Krizhevsky et al., 2012). Over the years, numerous studies have introduced prominent modifications with the aim of improving computational efficiency and increasing the discriminative power of CNNs. In 2014, Zeiler and Fergus demonstrated improved performance by reducing the stride and the receptive field of the first convolutional layer of the AlexNet architecture (Zeiler & Fergus, 2014). Soon after, the ‘VGG network’ was developed that consisted of 16 convolutional and 3 fully connected layers, illustrating the potential of layer depth in crafting more robust and accurate classifiers (Simonyan & Zisserman, 2014). In the same year, GoogleNet (also known as SermaNet) upended the traditional stacking of neural layers with its multi-scale architecture and demonstrated significant improvement in computational efficiency (Szegedy, Liu, Jia, & Sermanet, 2014). The following year, Long et al. developed the Fully Convolutional Neural Network (FCN) (Long et al., 2015). This was a milestone in overcoming the challenges of the ‘patch-based’ classification approach and led to substantial improvements in the overall efficiency of the network training procedure.

It is worth emphasizing that the afore-mentioned CNNs were used for the purpose of RGB-image categorization and segmentation. Several studies have, however, already applied CNNs to classify remotely-sensed images. In one of its first applications, CNNs were used for detecting vehicles in high resolution satellite imagery (Chen, Xiang, Liu, & Pan, 2014). Later, Romero et al. (2015) utilized a sparse unsupervised convolutional network for classifying remotely-sensed imagery. Also, Langkvist et al. (2016) investigated the effect of multi-scale architectures on land-cover classification accuracy. In the same year, Cheng et al. (2016) used CNNs for object detection in very high-resolution remote-sensing images. More recently, Maggiori et al. (2017) implemented a two-step FCN training procedure for the land-cover classification of large-scale remotely-sensed image data.

Using CNNs for the purpose of land-cover classification of VHR satellite imagery entails producing dense ‘pixel-label’ output maps. To this end, a large number of parameters are required to be learned during the network training procedure. This represents a significant challenge. In recent years, a number of

innovative strategies have been proposed to increase the discriminatory powers of CNNs while reducing the overall computational cost of training the network. For instance, Chen et al. (2015) have proposed to replace downsampling operations from the last few max-pooling layers with dilated convolutional filter. This technique essentially enlarges the receptive field of the convolutional filter without increasing the number of its trainable parameters. The receptive field of a convolutional filter determines the portion of the image that is linearly transformed using parametric kernel weights. Another approach that is gaining tractability is including upsampling ‘deconvolutional’ layers into the network architecture that are capable of refining coarse feature representations. To this end, Badrinarayanan et al. (2017) devised a deep convolutional encoder-decoder architecture that used the ‘VGG network’ as a base model, but replaced its fully-connected layers with a sequence of decoders – one corresponding to each encoder. In this manner, each decoder uses max-pooling indices from its matching encoder and maps low-resolution feature representations to full input resolutions - thus producing denser, high-resolution classified maps. This ‘encoder-decoder’ structure effectively eliminates the training required for upsampling coarse resolution feature maps. Other works have also investigated the possibility of refining pre-trained networks using data augmentation and transfer learning strategies to compensate for the lack of labelled remote-sensing data (Nogueira, Penatti, & Jefersson, 2017; Scott et al., 2017)

In the domains of image classification and image segmentation Markov Random Field (MRF) and its variant Conditional Random Field (CRF) have also been extensively applied to refine patchy and non-homogenous segmentation maps. Extensive work has been done on modelling contextual information using Markov Random Fields for predicting land-cover categories using high-resolution satellite imagery (Moser & Serpico, 2009; Moser et al., 2013). In other applications, a pairwise CRF inference has been applied to incorporate texture and other contextual information for object-class segmentation (Shotton, Winn, & Rother, 2006). Hierarchical CRF modelling has also been used for object class segmentation (Ladicky, Russell, Kohli, & Torr, 2009). CRF modelling has also been used for detecting built-up area in optical and synthetic aperture radar images (Kenduiwo, Tolpekin, & Stein, 2014). More recent studies have combined the Bayesian framework of CRF modelling with the automatic feature learning capacities of CNNs for semantic labelling of aerial and satellite images (Paisitkriangkrai, Sherrah, Janney, & Hengel, 2016). Other closely related works have also used dilated convolutional filters for processing multi-scale images and further applied CRF modelling to refine class-region boundaries (L. Chen et al., 2015; Fu et al., 2017).

In the next sections, we briefly describe the mathematical and theoretical foundations of CNNs and MRF/CRF model(s) as applied for land-cover classification of VHR remote-sensing images.

2.2. A BRIEF OVERVIEW OF CONVOLUTIONAL NEURAL NETWORKS

In machine learning, artificial neural networks or ANNs consist of a system of interconnected neurons such that each neuron takes an input vector $\mathbf{x} = \{x_1, x_2, \dots, x_d\}$ and performs a dot-product operation with a vector of weights $\mathbf{w} = \{w_1, w_2, \dots, w_d\}$. Here d is the number of inputs to the given neuron. A bias term b is appended to this product and the output is thereafter passed through an activation function. Figure 1 depicts a schematic representation of an artificial neuron. A mathematical definition of a neuron layer is given as follows:

$$Y = \sigma(\mathbf{w} \cdot \mathbf{x} + b) \quad (1)$$

In Equation (1), σ is called the activation function. A vector of weights, along with the bias term constitute the parameters controlling the value of output Y .

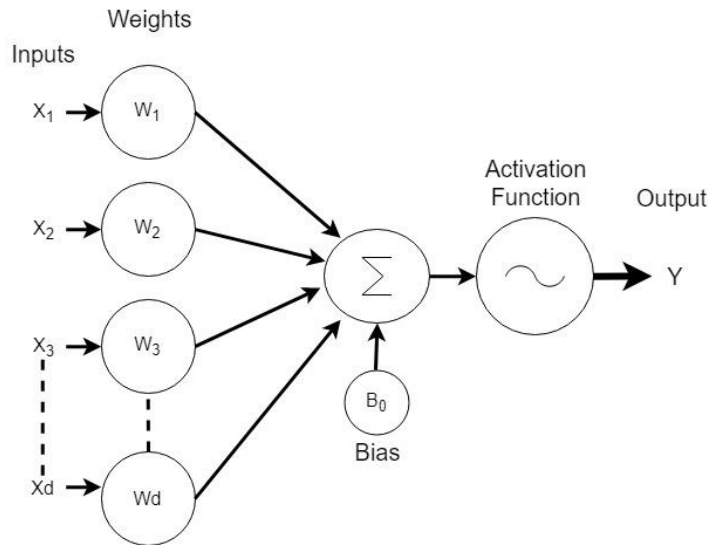


Figure 1: Schematic Representation of an ANN-Neuron

When these neurons are arranged in a linear sequence of processing layers they are said to form a linearly-connected ANN as shown in Figure 2. When network layers are linked in a non-linear sequence, where inputs can be shared by multiple layers, the network is said to have a skip-layer architecture (Fu et al., 2017).

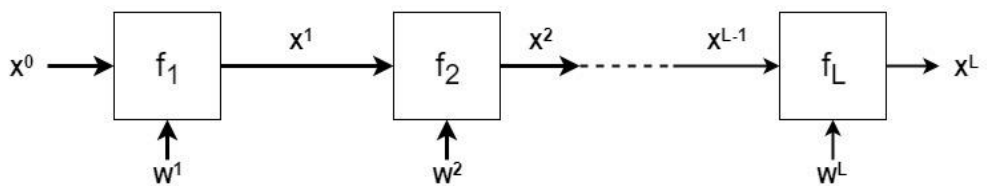


Figure 2: Schematic Representation of a Linearly Connected ANN

In the domain of image classification, the artificial neuron is modelled in the form of a convolutional layer that convolves the input image using kernel weights, the neural network is referred to as a CNN. Each layer of the CNN sequentially implements convolution, non-linear activation and region-based pooling on the input image patch. A deep CNN can be crafted by stacking many such CNN layers, such that low-level features are combined with features that display a higher level of abstraction (Längkvist et al., 2016). To represent the CNN mathematically, it can be understood as a function \mathbf{g} mapping image-data \mathbf{x} to an output vector \mathbf{y} of class-label predictions. Let this be defined as follows:

$$\mathbf{y} = \mathbf{g}(\mathbf{x}) \quad (2)$$

In Equation (2), the image data \mathbf{x} has a 4-dimensional structure such that $\mathbf{x} \in \mathbb{R}^{H \times W \times F \times N}$. Here H, W and F represent the height, width and the number of feature channels in the input image while N represents the number of such 3-dimensional tensors that together make up the contents of a single input batch. The output vector $\mathbf{y} \in \mathbb{R}^{\mathcal{C}}$, assuming that we have \mathcal{C} user-defined land-cover classes. The function \mathbf{g} is the composition of a sequence of functions, such that $\mathbf{g} = \mathbf{f}_1 \circ \dots \circ \mathbf{f}_L$. Here L signifies the number of layers in the network and $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^L$ represent the outputs of each network layer respectively. Meanwhile \mathbf{x}^0 denotes the network input. Each intermediate output \mathbf{x}^L is found by applying the function \mathbf{f}_L with parameters \mathbf{w}^L on the output of the previous layer \mathbf{x}^{L-1} such that:

$$\mathbf{x}^L = \mathbf{f}_L(\mathbf{x}^{L-1}; \mathbf{w}^L) \quad (3)$$

The aim of training a CNN is to optimize the values of the weights and biases so that the overall loss between the predicted and the ground truth labels is minimized. The loss function ℓ quantifies the error in classification and is defined as follows:

$$\ell(\mathbf{y}, \mathbf{c}) = \sum_{i=1}^N \ell(y_i, c_i) \quad (4)$$

In Equation (4), $\mathbf{y} \in \mathbb{R}^{H \times W \times \mathcal{C} \times N}$ and $\mathbf{c} \in \mathbb{Z}^{H \times W \times 1 \times N}$. Here, N corresponds with the number of discrete pixels across all input patches for which there exists a corresponding ground reference label. Consequently, y_i represents a vector of \mathcal{C} class-probability scores for all pixels of the input image batch. Similarly, c_i represents a vector of ground reference labels. One of the most frequently used loss functions is the softmax log-loss which is defined as follows:

$$\ell(x, c) = -x_c + \log \left(\sum_{k=1}^{\mathcal{C}} e^{x_k} \right) \quad (5)$$

Once the loss function has been specified, the learning of the weights happens by computing the partial derivatives of the cumulative loss with respect to each parameter weight and by recursively propagating the gradient backwards through the network using the chain rule. A user-specified learning rate λ determines the proportion of the gradient loss to be used in adjusting the weights. This weight update procedure is represented as follows:

$$w_i \leftarrow w_i + \lambda \frac{\partial \ell}{\partial w_i} \quad (6)$$

We now briefly describe the salient constitutive elements of a CNN, and their functionality with respect to the image classification procedure.

Convolutions: A convolutional layer in a CNN consists of a square shaped filter of size $M \times M$ with learnable parameter weights \mathbf{w} . Each filter is applied by sliding it over the input image, implementing a dot-product operation between corresponding filter weights and image spectral values. The repeated application of this procedure generates unique values for the output feature map (at the spatial coordinates found at the centre of the filter application). Eventually, a learned bias term is appended to each feature map as well. For instance, to generate K' feature maps from a K dimensional input image would require $((M \times M \times K') + K')$ number of learnable parameters in the convolutional layer. The size of the output feature map is determined by two other factors, namely: **stride** and **padding**. Stride is the interval difference between successive filter applications as it slides over the input image, while padding controls the number of 0 valued rows and columns appended to the borders of the input image (Volpi & Tuia, 2017). **Dilation**, on the other hand, allows convolutional filters to effectively increase the size of their receptive fields without increasing the number of its learnable parameters. This is implemented by inserting zeros between the weighted elements of a convolutional filter. Thus, a dilation factor of d applied to a filter of size M , effectively increases its kernel size to $d \times (M - 1) + 1$ (Vedaldi & Lenc, 2014).

Non-Linear Activations: The neural network community has, over the years, experimented with various forms of non-linear activation functions. Examples of saturating non-linear activation functions include the exponential and the hyperbolic tangent. The drawback of using these activation functions is that their gradients tend to zero when the input magnitude is large leading to negligibly small updates to the weight associated with each parameter (Volpi & Tuia, 2017). This is popularly termed as the ‘vanishing gradient’ problem and it slows down the training of the network. On the other hand, non-saturating activation functions like the Rectified Linear Unit (ReLU) train several times faster as they are not susceptible to the problem of vanishing gradients (Krizhevsky et al., 2012). In recent years, a variant of the ReLU called the ‘Leaky ReLU’ has been readily adopted as it allows for gradient propagation in neurons that would, in other circumstances, be considered ‘dead’ or ‘deactivated.’

Maximum or Average Pooling: This layer within the neural network has the function of aggregating the input signal information over user-defined window sizes. The two standard pooling strategies include maximum and average pooling. While the former returns the maximum value from amongst the input cells, the latter returns the average of the input values. The use of this layer makes the CNN more robust in allowing the network to recognize objects independently of their spatial location within the input image (Volpi & Tuia, 2017).

Dropout: This is now a widely applied technique to mitigate overfitting of neurons within the network. It involves coercing the output of a neuron to zero and, by doing so, preventing neurons in a particular layer from becoming co-dependent on each other (also referred to as co-adaptation). The use of the dropout layer within the network significantly increases the convergence time of a neural network (Krizhevsky et al., 2012). However, it also results in making the network more robust to input variance and increases the generalization capability of the trained model (Volpi & Tuia, 2017).

2.3. A BRIEF OVERVIEW OF MARKOV AND CONDITIONAL RANDOM FIELDS

Markov Random Field (MRF) and its variant Conditional Random Field (CRF) are a class of probabilistic graphical models that have found wide application in the fields of image restoration and image classification (Li, 2009). The efficacy of MRF/CRF models is predicated on their ability to incorporate contextual information as a linear combination of pairwise energy potentials (Moser et al., 2013). This encourages class label agreement between spectrally similar valued pixels, resulting in a smooth and homogenous segmentation/classification output. While MRF models the spatial dependency considering only class labels, the CRF takes into account both class labels and spectral values (Li, 2009).

A formal definition of a MRF is as follows:

Let a set of random variables $f = \{f_1, f_2, \dots, f_m\}$ be defined on the set I containing m sites in which each random variable y_i ($1 \leq i \leq m$) takes a label from set L . The family f , in such a case, is called a random field. In the context of an image classification task, f is equivalent to the set of pixel DN values. I is equivalent to an image containing m pixels; and the label set L corresponds to the user-defined set of semantic (land-cover) classes, e.g., $L = \{\text{water, built-up, impervious surface, etc.}\}$.

For a given pixel-label configuration r (where $r \in I$ and $r_i \in L$) to be a MRF, the probability density function applied at all locations m should satisfy the following properties:

- **Positivity:** $P(r_i) > 0$ for all sites i
- **Markovianity:** $P(r_i | r_{m-i}) = P(r_i | r_{N_i})$
- **Homogeneity:** $P(r_i | r_{N_i})$ is the same for all sites i

Here, $m - i$ indicates the set difference (i.e. set of all pixels excluding i). r_{m-i} denotes the set of labels at sites in set $m - i$ and N_i denotes the set of neighbours of pixel i (Tso & Mather, 2009).

A global optimization for the arrangement of pixel-labels is provided by the Gibbs Random Field (GRF) probability distribution and is defined as following:

$$P(r) = \frac{1}{Z} \exp \left[-\frac{U(r)}{T} \right] \quad (7)$$

In Equation (7), $U(r)$ is called an energy function that is defined locally in a constrained neighbourhood or random field. Also T is a constant, referred to as temperature. Z is normalizing factor also referred to as a partition function and is defined as the sum of all possible pixel-label configurations.

While an MRF models the local dependencies of a given pixel within a specified image region (or quantization level), the GRF describes the global properties of an image in terms of the joint distribution

of classes for all pixels (Tso & Mather, 2009). Given the constraints specified by the Gibbs probability distribution in Equation (7), it can be argued that maximizing $P(\mathbf{r})$ is equivalent to minimizing the energy function $U(\mathbf{r})$, which is defined as following:

$$U(\mathbf{r}) = \sum_{\mathbf{c} \in \mathcal{C}} V_{\mathbf{c}}(\mathbf{r}) \quad (8)$$

In Equation (8), clique \mathcal{C} is a collection of all possible cliques such that $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3 \cup \dots$, and $V_{\mathbf{c}}(\mathbf{r})$ is called the potential function defined for clique type \mathbf{c} . Consequently, clique \mathcal{C} is a subset of \mathcal{N}_i comprising of adjacent or nearby pixels that specify the neighbourhood of pixel i . This can be a single site, or a neighbouring pair, or even a more complex arrangement of neighbouring pixels, as shown in Figure 3. The most general form of the energy function, most readily deployed for image classification tasks, and based upon the pairwise clique potential function, is usually defined as follows:

$$U(\mathbf{r}) = \sum_{i \in \mathcal{I}, \{i\} \in \mathcal{C}_1} V_1(r_i) + \sum_{i \in \mathcal{I}, \{i, i'\} \in \mathcal{C}_2} V_1(r_i, r_{i'}) \quad (9)$$

In Equation (9), the energy function is expressed as a linear combination of two components. The first component can be understood as the unary energy or the class-likelihood of a given pixel. The second component, however, accounts for the spatial context of each pixel, and is computed by testing the label compatibility of a given pixel with all other pixels in its clique.

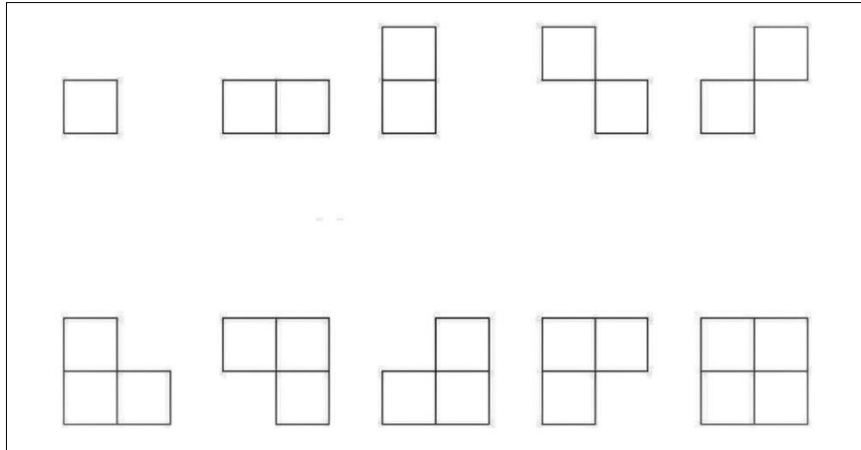


Figure 3: 1st and 2nd order neighbourhood clique arrangements
Source: Li (2009)

From Equation (9) and Figure 3, one can see that as the clique-size increases and accommodates a higher order of neighbourhood pixels, the computation of the pairwise potential becomes increasingly complex. This context illuminates the significance of the fully-connected dense-CRF inference algorithm as proposed by Krähenbühl and Koltun (2011).

The mean-field approximation to the fully-connected dense-CRF inference replaces the conventionally used posterior distribution $P(\mathbf{x})$ with a distribution $Q(\mathbf{x})$ that is defined as a product of independent marginal distributions $Q_i(\mathbf{x}_i)$ over all random variables – pixels – of the input image. Krähenbühl and Koltun (2011) define $Q_i(\mathbf{x}_i)$ by the following Gibbs energy distribution:

$$Q_i = \frac{1}{Z_i} \cdot \exp \left\{ -\Psi_u(\mathbf{x}_i) - \left(\sum_{l' \in \mathcal{L}} \mu(l, l') \sum_{m=1}^K w^{(m)} \sum_{j \neq i} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) Q_j(l') \right) \right\} \quad (10)$$

In Equation (10), Ψ_u refers to the unary potential energy of pixel \mathbf{x}_i . This is the likelihood of pixel \mathbf{x}_i to belong to class-label l . Further, $w^{(m)}$ are linear combination weights where m - an index that goes from 1 to K - reflects the number of dimensions of the input image (in our case, $K = 3$). Also, μ is a label compatibility function, commonly referred to as the Potts model. The pairwise energy potential is formulated by aggregating the label compatibility function over all-other class-labels l' , across all dimensions of the input image, and for every pixel j in the neighbourhood of i captured by the Gaussian kernels. Finally, $k(\mathbf{f}_i, \mathbf{f}_j)$ is expressed as a linear combination of two Gaussian kernels and is defined as follows:

$$k(\mathbf{f}_i, \mathbf{f}_j) = w^{(1)} \cdot \exp \left(-\frac{|\mathbf{p}_i - \mathbf{p}_j|^2}{2\theta_\alpha^2} - \frac{|\mathbf{I}_i - \mathbf{I}_j|^2}{2\theta_\beta^2} \right) + w^{(2)} \cdot \exp \left(-\frac{|\mathbf{p}_i - \mathbf{p}_j|^2}{2\theta_\gamma^2} \right) \quad (11)$$

In Equation (11), feature vector \mathbf{f} is expressed as a concatenation of vectors \mathbf{p} and \mathbf{I} such that $\mathbf{f} = \mathbf{p} \parallel \mathbf{I}$. Here, \mathbf{p}_i and \mathbf{p}_j are the position vectors and \mathbf{I}_i and \mathbf{I}_j are spectral DN-value vectors for pixels i and j respectively. The parameters θ_α , θ_β and θ_γ are equivalent to measures of variance that control the effective size of the Gaussian kernels. Further, $w^{(1)}$ and $w^{(2)}$ are weight measures that determine the proportion of influence that each kernel has in the computation of the pairwise potential. For effectively applying the dense-CRF inference we need to optimize for the value of these parameters. This process has been described in Section 3.3.

3. METHODOLOGY

This chapter is structured to begin with a description of the method used to train and refine the architectural design and the network training hyper-parameters of the SimpleNet and the FuseNet networks. Thereafter, we describe the process of optimizing the kernel parameters of the mean-field approximation to the dense-CRF inference algorithm. Finally, we present the method used to train and reconfigure the ReUseNet.

In summary, the methodology highlights the flexibility of CNN architectures that allows them to simulate common image enhancement procedures within its trainable block of parametric weights. In particular, we assess their ability to: (i) dynamically fuse multi-scale images for land-cover classification, and (ii) capture contextual information via the learned convolutional filter weights when implemented in the form of a recurring neural network. In such a RNN, two instances of the same FCN (FuseNet) are joined in a manner where the contextual information captured via the learned convolutional filter weights (of FuseNet1) are explicitly incorporated as an additional input into the succeeding (independently initialized) instance of the FCN (FuseNet2). In this manner the ReUseNet can be considered similar to the MRF/CRF models.

In order to meet the research objectives we first designed and fine-tuned the architectural parameters and the network training hyper-parameters of the following two CNN architectures:

1. **SimpleNet** - a FCN design in which the network layers are *linearly connected* to one another – with the output of the first layer being used as an input to the second layer and so on. In this network the dimensions of the input-image is not altered as the patch is processed through the network. The SimpleNet is trained on images that were pan-sharpened using the Gram-Schmidt’s spectral sharpening technique. The experimental method used to determine the design and parameter configuration for SimpleNet is described in Section 3.1. Once the SimpleNet design is *fixed* we train it on two different pan-sharpened image databases for generating the final classification results. These consist of (i) 4000 patch samples per training images (pan-sharpened using the Gram-Schmidt spectral sharpening technique) and (ii) 2000 patch samples per training image (pan-sharpened using the Subtractive Resolution Merge technique).
2. **FuseNet** – a FCN that takes as input discrete sets of MS and PAN image blocks. This network uses a skip-architecture and employs convolutional and de-convolutional operations to *dynamically fuse* multi-scale data in order to generate land-cover classification maps at the PAN resolution. The experimental method used to determine the design and parameter configuration of the FuseNet is described in Section 3.2. For the final classification results, in order to draw a meaningful comparison with the SimpleNet case, the final FuseNet design is also trained on two different image databases, comprising of 4000 and 2000 samples each collected randomly from both the MS and PAN training image sets.

In addition, the mean-field approximation to the fully-connected **dense-CRF** inference (**DCRF**) is applied to further refine the accuracy of the land-cover classification maps (Krähenbühl & Koltun, 2011). The method for optimizing the five pairwise potential kernel parameters is described in Section 3.3.

Finally the **ReUseNet (RNN)** is designed and trained. For this, two instances of a FCN (FuseNet) are joined such that the class-probability scores of the first FCN are concatenated with the raw input to the *succeeding instance* of the same network. The ReUseNet, in this manner, implements a recurring network

architecture that enables the combined FCN structure to simultaneously train two instances of the same network in an end-to-end trainable block of parametric weights. The experimental method for formalizing the parametric design of the ReUseNet is described in Section 3.4. A sensitivity analysis of the FCN and dense-CRF parameters is provided in Chapter 5. The final classification results for all methods are presented in Chapter 6. A flowchart depicting the methodology is presented in Figure 4.

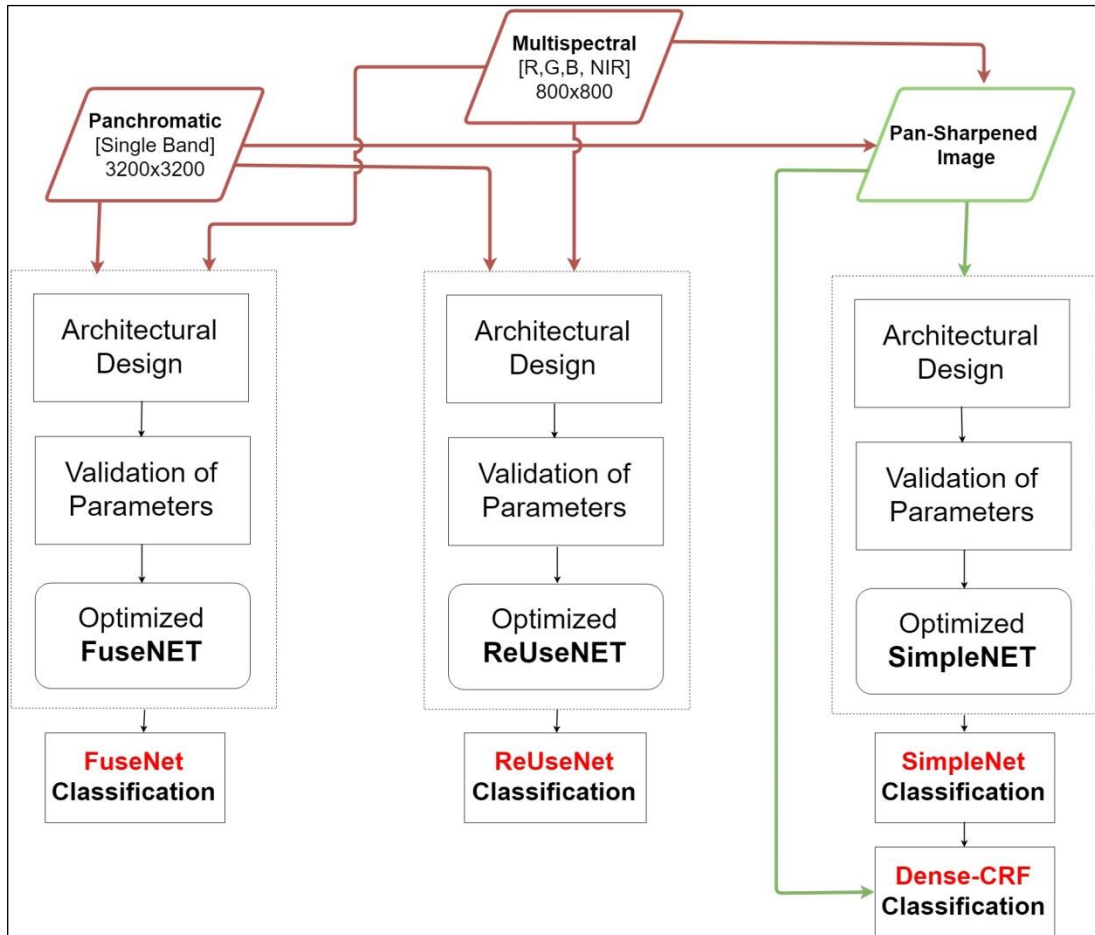


Figure 4: Research Methodology Flowchart

3.1. TRAINING AND CONFIGURING – SIMPLENET (SNN)

The method to refine the SimpleNet design involves the iterative re-configuration of its architectural elements and network training hyper-parameters. For this we utilized a training image database consisting of 2000 samples of image-patches of size $65 \times 65 \times 4$. These were collected randomly from the set of training images that were pan-sharpened using the Gram-Schmidt spectral sharpening technique. A summary of the spatial and spectral characteristics of the dataset and the experimental layout of the training, validation and test tiles is provided in detail in Chapter 4. Figure 5 shows a graphical representation of the baseline network architecture that was used as a reference in order to methodically assess the sensitivity of the network to incremental changes in hyper-parameter values.

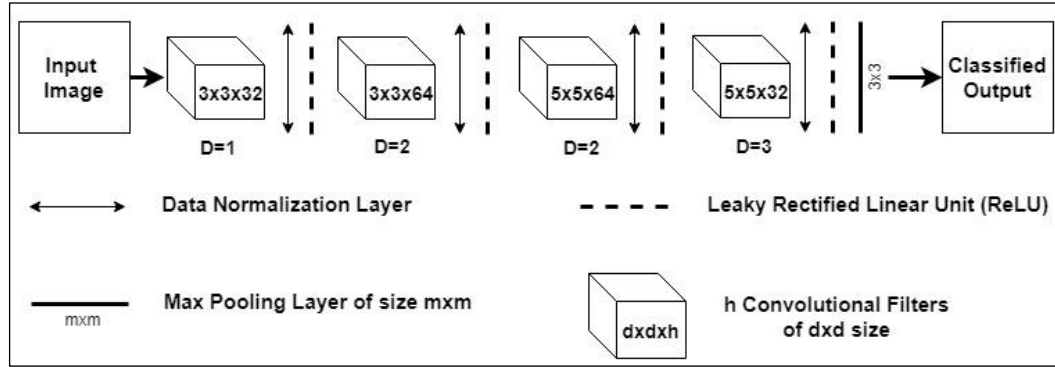


Figure 5: Schematic Representation for the Baseline FCN Architecture

The baseline network consists of four convolution layers, each followed by a data normalization layer and a leaky ReLU layer. The first layer of 32 convolution filters has a kernel size of 3×3 and a dilation factor of 1 (default). The following two layers comprise of 64 convolution filters each. Their kernel sizes are 3×3 and 5×5 respectively, with a dilation factor of 2. The final layer consists of 32 convolution filters with a kernel size of 5×5 with a dilation factor of 3. The last convolutional layer has an added max-pooling layer with window-size 3×3 . In addition, a dropout layer (with probability 0.5) was used before classifying the output. A complete list of hyper-parameter values used in the baseline experiment is presented in Table 1. Further, all experiments were carried out using two different learning rates with each training step lasting for 100 epochs each.

Table 1: Baseline Architecture and Hyper-Parameter Values

Hyper-Parameter ^a	Value(s)
Layers	I-C1-B1-A-C2-B2-A-C3-B3-A-C4-B4-A-MP4-D1-O
Patch Size	45
Kernel Depth	32, 64, 64, 32
Kernel Size (Dilation)	3(1), 3(2), 5(2), 5(3)
Pooling Size	NA, NA, NA, 3
Learning Rate	10^{-5} , 10^{-6}
Weight Decay	5×10^{-4}
Momentum	0.9
Number of Epochs	100, 100

^a Layer Notation: I = Input, C = Convolution, B = Batch Normalization, A = ReLU Activation (Leak Rate = 0.1), MP = Max-Pooling, D = Dropout (Rate = 0.5), O = Output. The network parameters were initialized using the normalization technique (Glorot & Bengio, 2010). The convolution and pooling strides was set to 1. A fixed mini-batch size of 32 was used in all experiments.

The method to optimize the SimpleNet design is divided into three steps. We first investigate the effect of varying the patch-size of the input training image. Further we also assess the effect of altering the depth of the network. In Table 2 each row represents a modification to the specified parameter value as compared with the baseline state described in Table 1.

Table 2: SimpleNet Experiments – Patch Size and Layer-Depth

Modified Parameter^a	Value(s)
Patch Size	65
Patch Size	85
Patch Size	125
Layer Depth (Removing)	I-C1-B1-A-C2-B2-A-C3-B3-A-MP3-D1-O
Layer Depth (Adding)	I-C1-B1-A-C2-B2-A-C3-B3-A-C4-B4-A-C5-B5-A-MP5-D1-O

^a Layer Notation: I = Input, C = Convolution, B = Batch Normalization, A = ReLU Activation (Leak Rate = 0.1), MP = Max-Pooling, D = Dropout (Rate = 0.5), O = Output. The network parameters were initialized using the normalization technique (Glorot & Bengio, 2010). The convolution and pooling strides was set to 1. A fixed mini-batch size of 32 was used in all experiments.

Subsequently, we focused on optimizing the architectural parameters of the SimpleNet. To this end, we experimented with the effect of including additional max-pooling layers into the baseline network architecture. We also investigated the effect of increasing the number of filters per convolutional layer and tested different kernel sizes with varying dilation factors. The list of parameter configurations that were iteratively modified in order to refine the SimpleNet architecture are presented in Table 3.

Table 3: SimpleNet Experiments – Architectural Parameters

Modified Architectural Parameter^a	Value(s)
Layers (Adding MP3)	I-C1-B1-A-C2-B2-A-C3-B3-A-MP3-C4-B4-A-MP4-D1-O
Layers (Adding MP2/3)	I-C1-B1-A-C2-B2-A-MP2-C3-B3-A-MP3-C4-B4-A-MP4-D1-O
Layers (Adding MP1/2/3)	I-C1-B1-A-MP1-C2-B2-A-MP2-C3-B3-A-MP3-C4-B4-A-MP4-D1-O
Kernel Depth	32, 32, 32, 32
Kernel Depth	64, 64, 64, 64
Kernel Size (Dilation)	3(1), 3(1), 3(1), 3(1)
Kernel Size (Dilation)	3(1), 3(1), 3(2), 3(2)
Kernel Size (Dilation)	3(1), 3(1), 5(2), 5(2)
Kernel Size (Dilation)	3(1), 5(1), 3(2), 5(2)
Kernel Size (Dilation)	3(1), 5(1), 5(2), 5(3)
Kernel Size (Dilation)	3(1), 3(2), 3(2), 5(2)
Kernel Size (Dilation)	3(1), 3(2), 5(2), 5(2)
Kernel Size (Dilation)	5(1), 3(2), 5(2), 5(3)
Pooling Size	3,3,3,3
Pooling Size	5,5,5,5

^a Layer Notation: I = Input, C = Convolution, B = Batch Normalization, A = ReLU Activation (Leak Rate = 0.1), MP = Max-Pooling, D = Dropout (Rate = 0.5), O = Output. The network parameters were initialized using the normalization technique (Glorot & Bengio, 2010). The convolution and pooling strides was set to 1. A fixed mini-batch size of 32 was used in all experiments.

Finally, we also varied the hyper-parameters that collectively determine the procedure for training the network. These include the number of training epochs, the learning rate and the proportion of ‘decay’ for each weight update. Table 4 provides a list of the hyper-parameter value combinations that were explored in order to locate a set of parameter configurations that would optimize the performance of the SimpleNet.

Table 4: SimpleNet Experiments – Training Epochs, Learning Rate & Weight Decay

Modified Hyper-Parameter	Value(s)
Learning Rate	$(10^{-3}, 10^{-4}), (10^{-4}, 10^{-5}), (10^{-6}, 10^{-7})$
Weight Decay	10^{-4}
Weight Decay	10^{-3}
Number of Epochs	200, 200

In this manner, the SimpleNet design was incrementally refined. A schematic representation of the final SimpleNet architecture is shown in Figure 6.

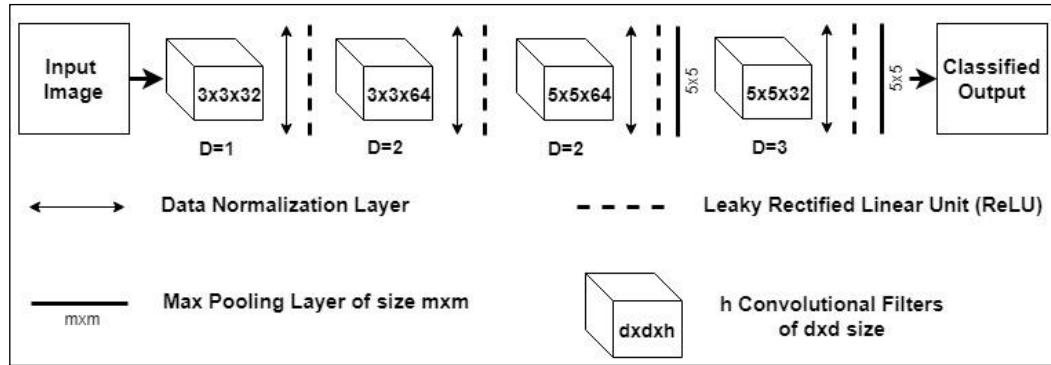


Figure 6: Schematic Representation of the Optimized Simple Neural Network Architecture

This SimpleNet architecture shown in Figure 6 is finally trained on two different training image databases for generating the final classification results. In the first case, we train the SimpleNet using 4000 samples of image-patches of size $45 \times 45 \times 4$ collected from each of the three training tiles that were pan-sharpened using the Gram-Schmidt spectral sharpening technique. In the second case, the SimpleNet is trained using 2000 samples collected randomly from the set of images that were pan-sharpened using the Subtractive Resolution Merge technique. The final SimpleNet is trained using two different learning rates 10^{-5} and 10^{-6} , with each training step lasting for 100 epochs each. The value for the weight decay parameter is held constant at 5×10^{-4} . The final classification results are presented in Chapter 6.

3.2. TRAINING AND CONFIGURING - FUSENET

The FuseNet is designed to determine an optimal FCN architecture that can dynamically fuse multi-scale remote-sensing images for generating high-accuracy land-cover classification maps. To reiterate, the objective of designing the FuseNet is to draw a comparison with the classification performance of the SimpleNet model that is trained using a set of pan-sharpened images.

In order to refine the FuseNet architecture and its network hyper-parameters a training image database is created that comprises of 2000 random samples taken from 3×2 (MS, PAN) tiles each. These image-patches were collected in the following patch-size combinations $[(32 \times 32), (128 \times 128)]$ from the sets of MS and PAN input images respectively. The salient blocks of the FuseNet architecture (presented in the sequence as processed within the network) are as follows:

1. The Downsampling Block (**DS**): in this block the input PAN image is downsampled to match the dimensions of the MS input image. Experiments investigate the step-wise implementation of the downsampling procedure (so for example, downsampling by a factor of two + downsampling by a factor of two vs. downsampling by a factor of four).
2. Concatenation Layer (**CNCT**): Concatenates the downsampled PAN features and the MS feature input along the third dimension.
3. Bottleneck (**BN**): A block of convolutional layers sharing and successively transforming feature representations with a string of 3×3 and 5×5 filters with varying factors of dilation. The features, at this point within the network, are their smallest size (usually downsampled four times or even 16 times from the input patch sizes of 32×32 or 128×128) after which they are upsampled to make final class-label predictions.
4. Upsampling Block (**US**): Deconvolution filters with adjustable parameters are trained to upsample image features within the FCN. Similar to the downsampling block the effect of step-wise up-scaling of the image was investigated for this block.

A graphical representation of the FuseNet architecture is given below in Figure 7.

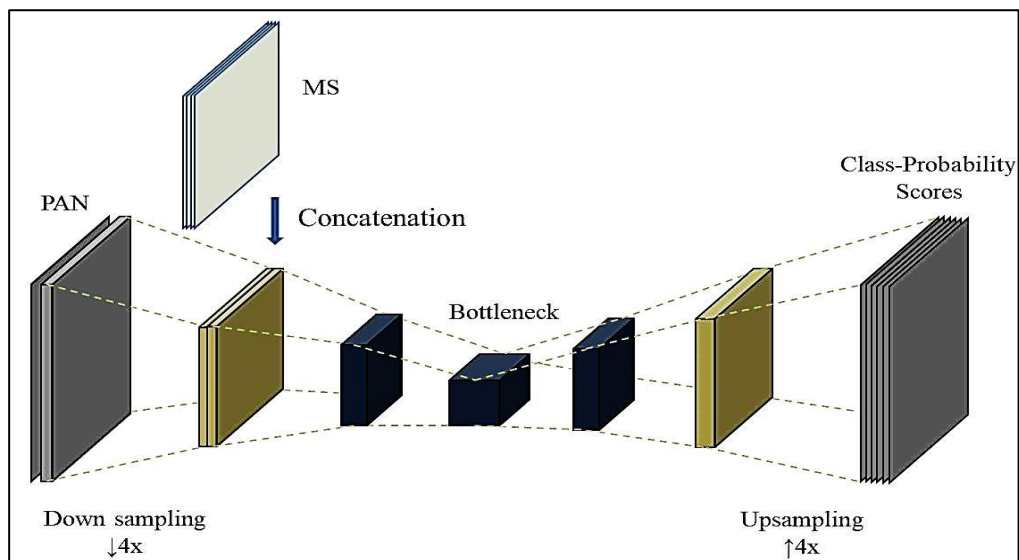


Figure 7: FuseNet Architecture

Additionally, we also investigated the effects of enhancing the number of features (for example, increasing 4 MS features to 16) of the input MS image (FEMS). For this, we apply unit-filter single-stride convolutions to multiply the input features and generate fully-connected representations at the same input dimensions. These investigations were motivated by the observation that the volume of the concatenated spatio-spectral information (both downsampled PAN and input MS features), should be in relative proportion to their occurrence in the raw-data representations. The FuseNet architectural design (shown in Figure 7) specifies the order of the experimental method as presented below. The method employed to refine the architecture and parameter configuration of the FuseNet model involved incrementally optimizing the parameters block-by-block.

Table 5 summarizes the method to refine the structure of the DS, FEMS and BN blocks. Here we determine whether the FuseNet architecture is more amenable to downsampling in one step or two steps, the worth of increasing the number of features of the MS input patch and the ideal arrangement of convolutional blocks within the bottleneck layer. For this the size of the smallest feature within the network is maintained at 32x32. Here the notation [D,P,S] – shown in the tables below as a part of each convolutional filter block – indicates the factors for dilation, padding and stride associated with the every convolutional layer.

Table 5: FuseNet Experiments - DS, FEMS & BN Blocks (for feature size = 32)

Fuse Net Block	DS	FEMS	BN			US
	[D,P,S]	[D,P,S]	1 st Cnv [D,P,S]	2 nd Cnv [D,P,S]	3 rd Cnv [D,P,S]	
DS	1 step [4x4]	3x3 (32) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	4x4 (6)
	2 step [2x2, 2x2]	3x3 (32) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	4x4 (6)
FEMS	-	3x3 (16) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	4x4 (6)
	-	3x3 (32) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	4x4 (6)
	-	NA/Raw (4)	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	4x4 (6)
BN (32)	-	-	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	4x4 (6)
	-	-	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	4x4 (6)
	-	-	3x3 (64) [1,1,1]	5x5 (64) [1,2,1]	3x3 (64) [2,2,1]	4x4 (6)
	-	-	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	3x3 (64) [2,2,1]	4x4 (6)
	-	-	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	4x4 (6)
	-	-	5x5 (64) [1,2,1]	5x5 (64) [1,2,1]	3x3 (64) [2,2,1]	4x4 (6)
	-	-				

LR(Epochs): 10^{-4} (80), 10^{-5} (40); Weight Decay: 5×10^{-4}
[D,P,S] = [Dilation, Pad, Stride]

Next, we investigate the effects of including an additional downsampling step (DS1) (post-concatenation of MS and PAN features) into the network design. The size of the smallest feature in the bottleneck, consequently, becomes equal to 16x16. The different architectural arrangements for the network in this case are summarized in Table 6.

Table 6: FuseNet Experiments - BN (smallest feature size = 16)

DS [D,P,S]	FEMS [D,P,S]	DS1 [D,P,S]	BN			US
			1 st C _{Nv} [D,P,S]	2 nd C _{Nv} [D,P,S]	3 rd C _{Nv} [D,P,S]	
-	-	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	-	4x4 (6)
-	-	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	5x5 (64) [1,2,1]	-	4x4 (6)
-	-	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	4x4 (6)
-	-	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	5x5 (64) [1,2,1]	3x3 (64) [2,2,1]	4x4 (6)
-	-	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	3x3 (64) [2,2,1]	4x4 (6)

LR(Epochs): 10^{-4} (80), 10^{-5} (40); Weight Decay: 5×10^{-4}
[D,P,S] = [Dilation, Pad, Stride]

Thereafter, we investigate the inclusion of another downsampling step (DS2) into the network. This reduces the size of the smallest feature within the network to 8x8. These different architectural configurations are listed in Table 7.

Table 7: Experiments - BN (smallest feature size = 8)

DS [D,P,S]	FEMS [D,P,S]	DS1 [D,P,S]	DS2 [D,P,S]	BN		US
				1 st C _{Nv} [D,P,S]	2 nd C _{Nv} [D,P,S]	
-	-	3x3 (64) [1,1,2]	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	4x4 (6)
-	-	3x3 (64) [1,1,2]	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	5x5 (64) [1,2,1]	4x4 (6)
-	-	3x3 (64) [1,1,2]	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	4x4 (6)
-	-	3x3 (64) [1,1,2]	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	4x4 (6)
-	-	3x3 (64) [1,1,2]	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	4x4 (6)

LR(Epochs): 10^{-4} (80), 10^{-5} (40); Weight Decay: 5×10^{-4}
[D,P,S] = [Dilation, Pad, Stride]

Similar to the investigations made on the downsampling or DS block (Table 5), we intend to ascertain the effect of step-wise upsampling on the accuracy of the obtained land-cover classification maps. The combination of upsampling strategies explored for the US block are listed in Table 8.

Table 8: FuseNet Experiments – US

DS [D,P,S]	FEMS [D,P,S]	BN			US
	CNCT	1 st Cnv [D,P,S]	2 nd Cnv [D,P,S]	3 rd Cnv [D,P,S]	
-	-	-	-	-	1 step [4x4]
-	-	-	-	-	2 step [2x2, 2x2]
Upsampling Experiments (with size of smallest feature = 8)					
-	-	-	-	-	4 Step [2x2; 2x2; 2x2; 2x2]
-	-	-	-	-	2 step [4x4, 4x4]

LR(Epochs): 10^{-4} (80), 10^{-5} (40); Weight Decay: 5×10^{-4}
[D,P,S] = [Dilation, Pad, Stride]

Finally, we also investigate: (i) the effect of including another loss function into the network (at the point in the network, post the BN block, when the size of the feature map becomes equal to the size of the multispectral training input) and (ii) the appropriate learning rate for the network. In this manner, proceeding block-by-block, we refine the architecture and the network training hyper-parameters of the FuseNet. Table 9 shows the best parameter configuration for each block that together defines the final FuseNet architecture.

Table 9: FuseNet Architecture

DS [D,P,S]	FEMS [D,P,S]	BN			US
	CNCT	1 st Cnv [D,P,S]	2 nd Cnv [D,P,S]	3 rd Cnv [D,P,S]	
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,1]	5x5 (64) [1,2,1]	3x3 (64) [2,2,1]	4x4 (6)
3x3 (32) [1,1,2]	[32+16 = 48 feat.]				

The FuseNet model specified in Table 9 is finally trained using two different image databases for generating the final classification results. For the first set of results, we use an image database consisting of 4000 samples of training patch-sizes 32x32x4 and 128x128x1 collected from each of the three training tiles from the sets of MS and PAN images respectively. In the second case, we use an image database of 2000 samples collected from each of the three training tiles from the sets of MS and PAN images respectively. Both networks are trained using two different learning rates, 10^{-4} and 10^{-5} , with each training step lasting for 100 epochs each. The value for the weight decay parameter is held constant at 5×10^{-4} . The trained network is used to predict classification maps for Tile 32 and Tile 103 respectively. The final classification results are presented in Chapter 6.

3.3. DENSE-CRF – PARAMETER OPTIMIZATION

The mean-field approximation to the fully-connected dense-CRF inference reformulates the pairwise energy potential as a linear combination of two Gaussian kernels (see Equation (11) in Section 3.3).

This two kernel formulation yields five model parameters: θ_α , θ_β , $w^{(1)}$, θ_γ and $w^{(2)}$. The first kernel accommodates both spatial and spectral differences between pixels i and j . The parameters θ_α and θ_β control the ‘width’ of the kernel - effectively filtering an acceptable deviation of values. Here $w^{(1)}$ controls the proportion of the kernel weight. The second kernel, on the other hand, only takes into account the spatial position of pixel-labels and its variance is controlled by parameter θ_γ while $w^{(2)}$ has the same interpretation as its first-kernel counterpart. The implementation of the algorithm requires two inputs from the user: (i) raw image with spectral DN values and (ii) class-probability scores generated using the SimpleNet of Section 3.1. Both data are rescaled to an 8-bit radiometric resolution prior to being used in the model. In addition, an epsilon value is added to the rescaled class-probability scores in order to meet the positivity criterion for defining a Markov Random Field mentioned in Section 2.3.

To optimize the kernel parameters we recursively narrowed the search range for each parameter. In this we were guided by relevant academic literature, especially the suggestion of maintaining parameter value $\theta_\gamma = 1$ (Krähenbühl & Koltun, 2011). The parameter search space for the first batch of experiments (in total 144 per validation tile) to optimize the dense-CRF inference is shown in Table 10.

Table 10: Dense-CRF - Batch 1 Experiments

DCRF Parameters	Initial Parameter Search Ranges	Number of Experiments
$w^{(1)}$	1,3,10,15,30,50	24
θ_α	1,3	72
θ_β	1,5,10,15	36
θ_γ	1	144
$w^{(2)}$	10^{-4} , 10^{-2} , 1	48

For the next batch of experiments, we target one parameter at a time - only changing its value while holding all other parameters constant at the following default values: $w^{(1)} = 50$, $\theta_\alpha = 3$, $\theta_\beta = 10$, $w^{(2)} = 0.01$ and $\theta_\gamma = 1$. By readjusting the parameter search ranges to centre on these newly identified intervals we recursively optimize the classification performance of the dense-CRF model.

Table 11 summarizes the second batch of experiments. Here, we simultaneously tested configurations for parameters θ_α and θ_β (a total of 32 experiments). Additionally, lengthier search ranges were tested for parameters $w^{(1)}$, $w^{(2)}$ and θ_γ as well. Experiments where two variables were tested simultaneously are indicated with a \times - implying all combinations between the sets of parameter values.

Table 11: Dense-CRF - Batch 2 Experiments

DCRF Parameters	Value(s)	Number of Experiments
$\theta_\alpha \times \theta_\beta$	(3,5,7) \times (4,6,10,15)	12
$\theta_\alpha \times \theta_\beta$	(4,5,6,7) \times (3,4,5,6,7)	20
θ_γ	1,3,5,7,9	5
$\theta_\gamma \times w^{(2)}$	(1,3,5) \times (1,5,20)	9
$w^{(1)}$	1,10,20,50,80,100,150,200,500,1000	10

Upon analysing the experimental results, we pick parameter configurations that are stable and consistently high-performing across both validation tiles. Once the parameter values are fixed we applied the dense-CRF inference on the SimpleNet generated classification scores. The classification results as observed on Tile 32 and Tile 103 are presented in Chapter 6.

3.4. TRAINING AND CONFIGURING – REUSENET

The motivation for designing the ReUseNet was to simulate the underlying mechanism of MRF/CRF models that apply context-sensitive label-similarity statistical modelling to refine land-cover classification maps. This research posits that the RNN structure explicitly incorporates pixel-context information, via the functionality of the receptive fields of convolutional filters. Receptive fields specify the portion of the input feature image that is encoded via convolutional filter weights to produce a singular output for the subsequent feature map. As the filter slides over the input image, or feature map, depending on its receptive field, it captures and compresses neighbourhood spectral information around a given pixel into a single cell value for the subsequent feature map. The training of these cascading filter weights is what determines the final classification scores. In Figure 8 a simple graphical representation of the ReUseNet architecture is shown. The architecture of the ReUseNet consists of two *instances* of the FuseNet network designed in Section 3.2. This is also referred to as a RNN. A simple graphical representation of the ReUseNet architecture is shown in Figure 8.

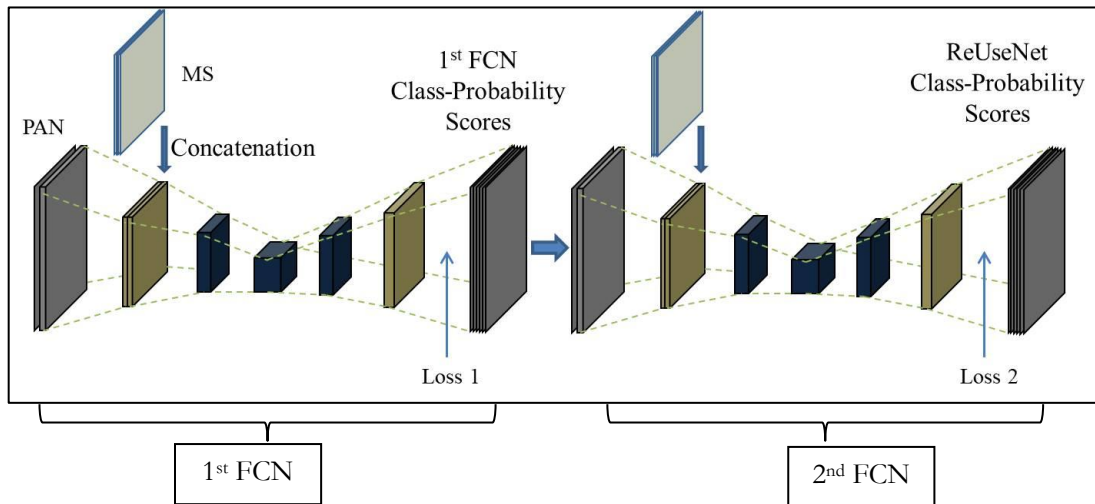


Figure 8: A graphical representation of ReUseNet

In Figure 8, both networks are initialized with different parameters but are joined by concatenating the class-probability scores of the first network with the PAN input of the succeeding network. The second network is an exact copy of the first. In this manner, the ReUseNet implements a recurring FCN architecture where the parameters of both networks are trained simultaneously in an end-to-end manner. In formalizing its design we investigated the influence of including another loss function into the network (at the end of FCN1; as also shown in Figure 8). In the case of a dual-loss network, the user needs to further specify the weight proportion to be used for balancing the two objective loss functions within the network. Table 12 presents a list of ReUseNet architectures that were investigated in order to determine whether including an additional loss is beneficial for increasing the overall accuracy of the classification. Furthermore, in this case of a two-loss network, we investigated varying the proportion of the weights for each objective loss function.

Table 12: ReUseNet Experiments with Additional Loss

Number of Loss Functions in ReUseNet	Weight of Objective Loss 2
Single Loss	1
Double Loss	0.8
Double Loss	0.7
Double Loss	0.5
Double Loss	0.3

Further, as a *control* for the ReUseNet model, we extracted class-prediction scores (using a singular instance of a FuseNet) of the images in the training set. These class-prediction scores are later concatenated with the raw-PAN features of the respective training images. This new set of 7 dimensional PAN resolution features, along with the raw-MS images, are thereafter used to generate a new image database using the same training patch locations as the ones used in the training of the singular FuseNet model. At this point a fresh, newly initialized, FuseNet model is trained on this new image database.

The final ReUseNet architecture consists of two loss functions, with the weight of the second objective loss function (shown as Loss2 in Figure 8) equal to 0.7. For generating the final classification results, the ReUseNet is trained using an image database of 4000 samples of training patch-sizes $32 \times 32 \times 4$ and $128 \times 128 \times 1$ collected from each of the three training tiles from the sets of MS and PAN images respectively. Furthermore, we also train the ReUseNet network on an image database in which 2000 samples were collected from each of the three MS and PAN sets of training tiles. The ReUseNet(s) are trained using two different learning rates, 10^{-4} and 10^{-5} , with each training step lasting for 100 and 50 epochs each. The value for the weight decay parameter is held constant at 5×10^{-4} . The trained network is used to predict classification maps for Tile 32 and Tile 103 respectively. The final results are presented in Chapter 6.

4. DATA

The image data used in this research was acquired by the WorldView III (DigitalGlobe) VHR satellite sensor. In all 7 tile-sets of panchromatic (PAN) and multispectral (MS) images were used. These depict semi-urban to highly urbanized areas of Quezon City – a densely populated city in the Philippines. The salient spatial and spectral characteristics of the dataset are summarized in Table 13.

Table 13: Raw Data - Image Characteristics

Image	Number of Band(s)	Image Dim [row,col]	GSD (meters)
MS	4 - Red, Green, Blue and NIR	800 x 800	1.24
PAN	Single	3200 x 3200	0.31

In all experiments that were carried out as part of this research method a consistent division of the image data set is maintained in the following manner: Tile 78, Tile 82 and Tile 100 are used only for *training* the proposed FCN classification model(s), Tile 45 and Tile 105 are used only for *validating* the reconfiguration of the FCN networks and the DCRF parameters and Tile 32 and Tile 103 are exclusively used for *testing* and comparing the final classification accuracy of the chosen network models and methods. The study area along with the layout of the training, validation and testing sets of images is shown in Figure 9.

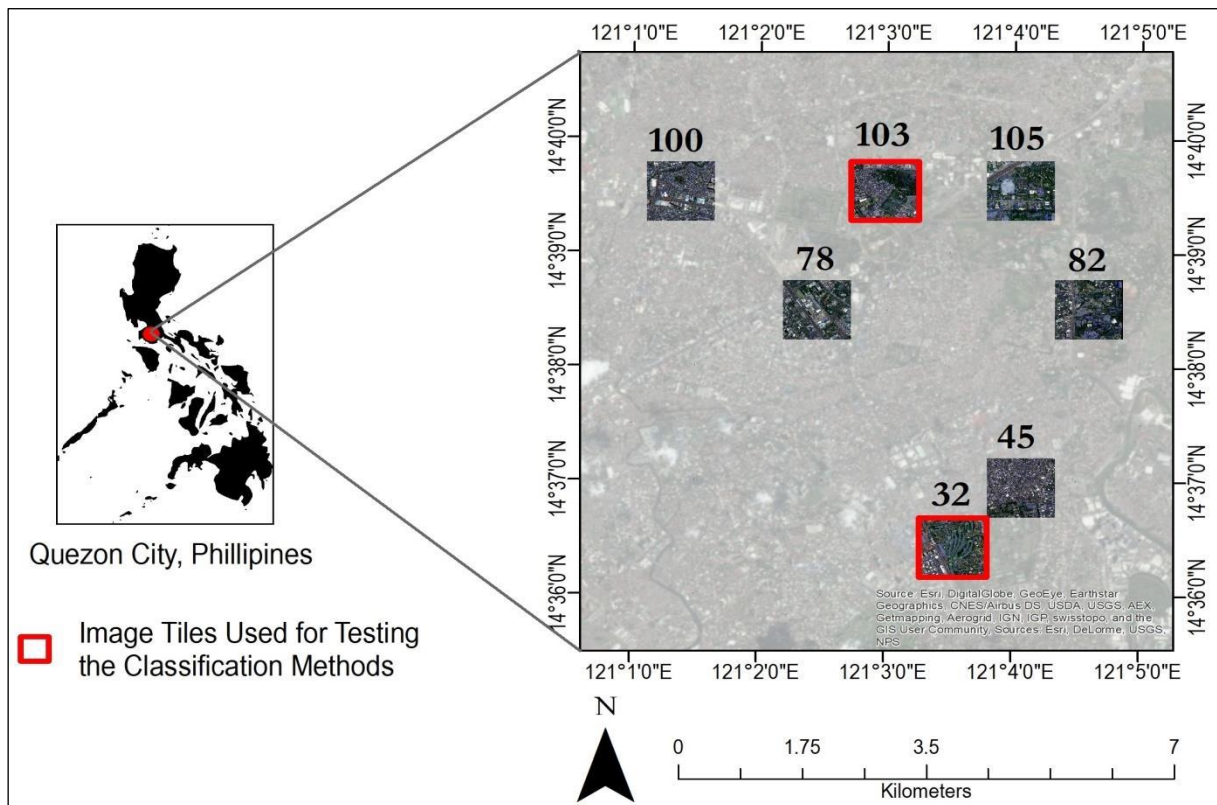


Figure 9: Study Area and Layout of Training, Validation and Testing Tiles

4.1. DATA PREPARATION

Ground Reference

In order to ensure the availability of adequate ground reference data for the training, validation and testing of the FCNs, the image-tiles were manually annotated and classified into the following six land-cover categories: (i) Built-Up, (ii) Low Vegetation, (iii) Tree, (iv) Cars, (v) Water and (vi) Impervious Surface.

A sample of both MS and PAN images, along with the annotated ground reference that was prepared, is shown in Figure 10. For an account of the number of ground truth pixels collected per class per image tile, see Appendix E.

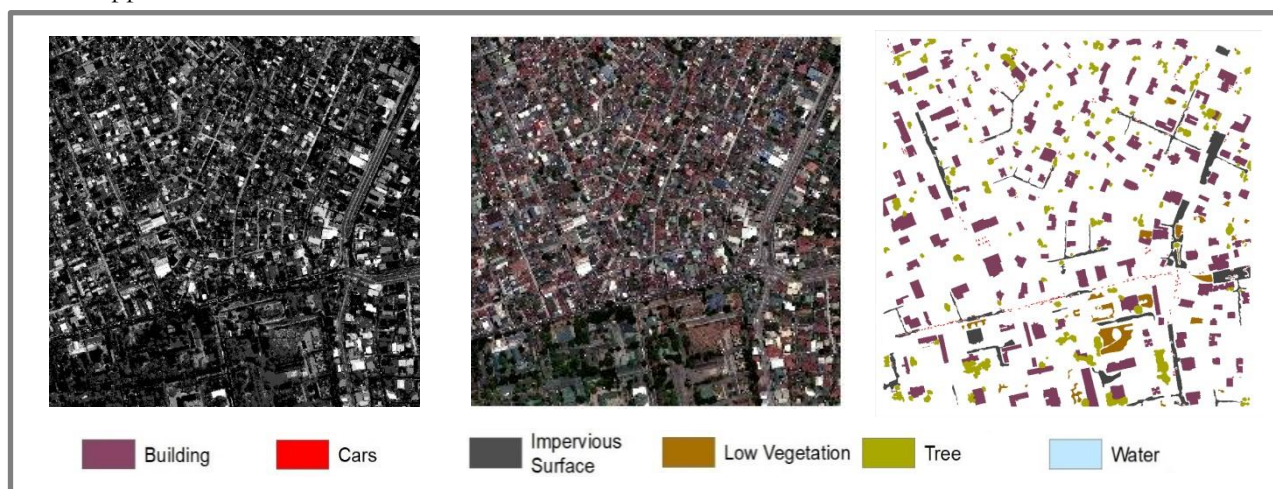


Figure 10: A Sample of the Panchromatic (L), Multispectral and Ground Reference Data

Data Augmentation

Due to the large number of parametric weights in even modestly designed CNN architectures, large amounts of annotated data are required to adequately train the network. To compensate for the volumetric demands of semantically-labelled data, image transposition has been suggested as a means to augment the amount of available training data (Scott et al., 2017). In our experiments we implemented this using a MATLAB function (*padarray*) that pads an input image with a mirror reflection of itself - effectively implementing image transposition (or mirroring) along the borders of each training patch.

Pan-Sharpening

One of the objectives of this research is to assess the ability of CNNs to fuse multi-scale VHR images. In order to draw a meaningful comparison with the FuseNet approach, the SimpleNet was trained and reconfigured using images that were pan-sharpened using a third-party commercially licensed software. To pan-sharpen the raw MS and PAN images we used the Gram-Schmidt spectral sharpening technique available in ESRI's ArcMap. Further, we also used the Subtractive Resolution Merge technique from ERDAS Imagine. We use these to two sets of pan-sharpened images to construct two different databases for testing the final classification results of the SimpleNet. These results are presented in Chapter 6.

4.2. IMAGES FOR TESTING CLASSIFICATION ACCURACY

As part of the research method, we use two tiles – Tile 32 and Tile 103 – for testing the final classification accuracy of our proposed approaches. Figure 11 shows the true-colour composite image of the two tiles that were used for testing the land-cover classification performance of the four classification methodologies proposed in this research.

The coordinate reference system is WGS 1984 UTM Zone 51. The map projection used is Transverse Mercator. Further, all classification maps in Chapter 6 are also presented at the same scale as shown in Figure 11.

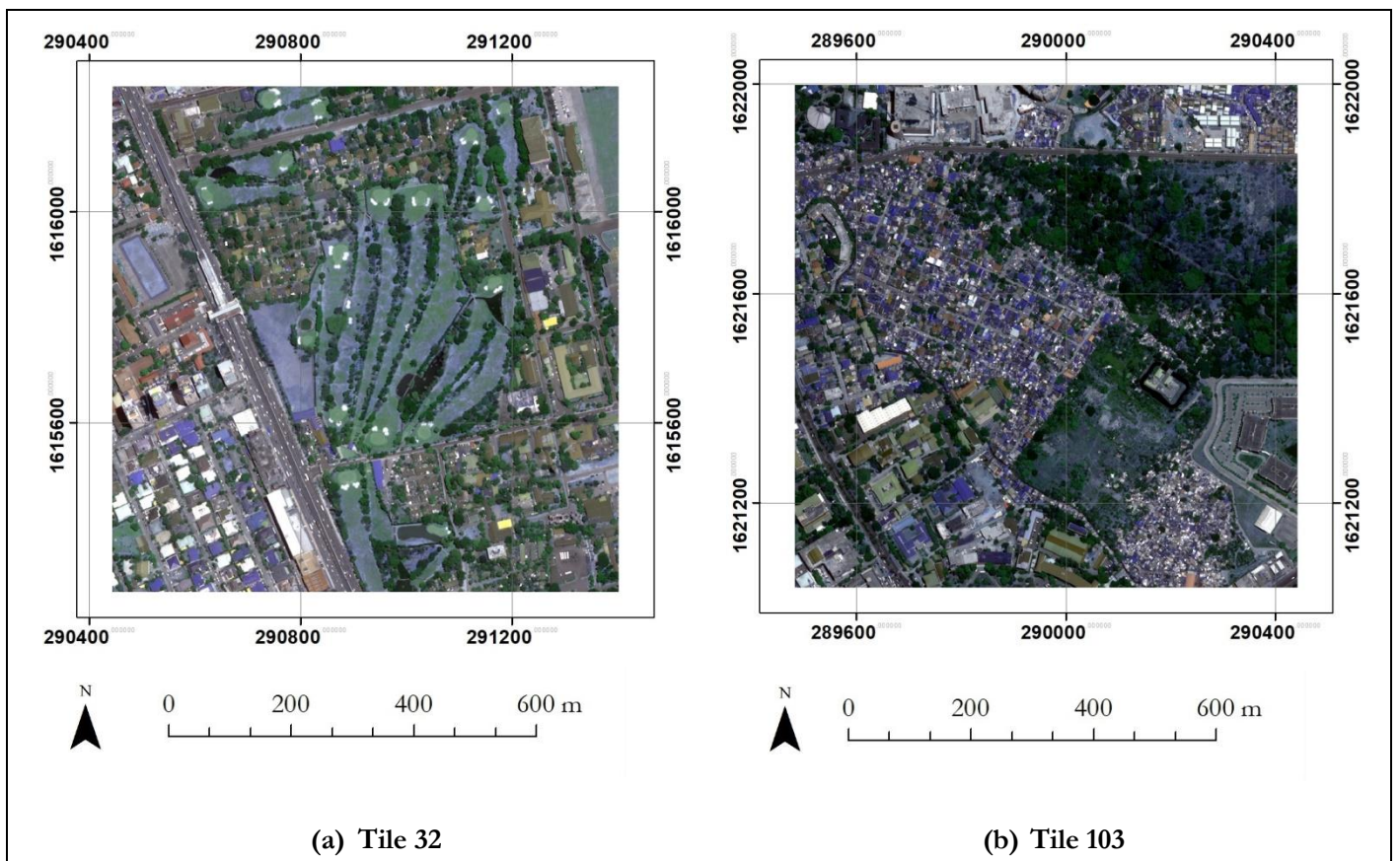


Figure 11: True Colour Composite (R, G, B) of Testing Tiles

5. PARAMETER SENSITIVITY ANALYSIS

In this chapter we first present a sensitivity analysis of the parameters used in the FCN design and assess their impact on the overall classification accuracy. This analysis is based on the classification results of Tile 45 and Tile 105. These are provided in Appendix A and B at the end of this report. For FCN parameters, we further sub-divided them into two categories, namely architectural parameters and the network training hyper-parameters. The architectural parameters are patch-size, layer depth, kernel depth, kernel size, number of max-pooling layers and the size of the pooling windows. The hyper-parameters are learning rate, weight decay and the number of training epochs. In the following text we first analyse the influence of each of these parameters on the overall accuracy of the classification results. Herein, the experimental results are interpreted in relation to the objectives of the study, as specified in Section 1.3. Thereafter, we also analyse the sensitivity of the overall classification accuracy to changes in the value of the kernel parameters of the mean-field approximation to the dense-CRF inference algorithm.

5.1. FCN ARCHITECTURAL PARAMETERS

Training Image Patch Size: In the SimpleNet experiments listed in Appendix A, we can see that increasing the patch-size of the training image does not necessarily translate into an increased overall accuracy score. This is because the ability of a CNN to exhaustively capture relevant features from an input training patch depends on the architecture of the network. In the case of our experiments, the network form is static, which explains its fluctuating classification performance as we increase the input patch-size. This highlights the necessity of adapting network designs for different scales and sizes of the input training image patch. The disadvantage of using a large input training patch-size is however more readily apparent as the computational time drastically increases with an increase in the patch-size.

Layer Depth: While the baseline SimpleNet architecture consisted of four convolutional blocks, we also tested architectures with three and five convolutional blocks of layers. Our experiments with layer depth revealed, quite expectedly, that overall accuracy and layer depth are positively correlated. However, increased layer-depth comes at an increased computational cost. And further even smaller finely-tuned architectures can outperform more complex FCNs that have not been adequately optimized. This is shown in the experimental results provided in Appendix A, where the best performing 4-block network was even better than the 5-block network that was tested.

Kernel Depth: Increasing the number of convolutional filters per block increases the overall accuracy of the classification but only by a marginal amount. Furthermore, the computation costs increases linearly with an increase in the kernel depth per convolutional layer. See Appendix A for metrics on classification accuracy and network training time.

Kernel Size: For producing high accuracy land-cover classification maps it is extremely important to optimize for the appropriate sequence of convolutional filters (with varying factors of dilation), given a fixed layer depth and training image patch-size. This is demonstrated by the list of experiments in Appendix A and B respectively. Further, larger kernel sizes lead to higher computation cost during the training of the network.

Max Pooling Layer: This layer makes the network more robust to object translation and rotation within the image (Längkvist et al., 2016). From the SimpleNet results we can see that their iterative addition into the baseline architecture increases the overall accuracy of the classification. Furthermore, larger pooling windows of size 5x5 seem more suited for input features than windows of size 3x3.

5.2. FCN HYPER-PARAMETERS

Learning Rate: We trained our FCNs using a two-step training procedure with differential learning rates. As can be seen in the results of Appendix A and B, optimizing for the appropriate learning rate is important for the overall accuracy of the classification results. Interestingly, in the case of the SimpleNet experiments, where the network was trained on image samples taken from the set of pan-sharpened images of size $45 \times 45 \times 4$, the appropriate learning rate combination was found to be 10^{-5} and 10^{-6} ; while in the case of the FuseNet experiments where image samples were collected in sizes of $32 \times 32 \times 4$ and $128 \times 128 \times 1$ from the set of MS and PAN images the appropriate learning rate combination was found to be 10^{-4} and 10^{-5} .

Weight Decay: Changing the value of this parameter had a minimal impact on the overall classification accuracy. This can be seen in results provided in Appendix A.

Number of Training Epochs: The number of epochs required to adequately train an FCN depends on the amount of training data being fed to the network, along with the cumulative number of parameters that are required to be learned during the training procedure. Over-training the network can lead to loss in the generalization capabilities of the network and usually the worth of increasing the number of training epochs can be gauged by assessing the network error (observe the flattening of the error curve) during the training process. In the cases tested for this research, we find that increasing the number of training epochs beyond a certain threshold does not lead to increase in the overall accuracy of the classification.

5.3. DENSE-CRF PARAMETERS

In order to optimize the classification performance of the dense-CRF model, the search-space or the range of parameter values were iteratively refined. The method for optimizing the parameter values of the dense-CRF model involved the following:

The experimental results of Table 10 of Section 3.3, are subset to contain only the observations that exceed the 85th percentile of the distribution of overall accuracy (OA) of the classification of Tile 45 and Tile 105 respectively. These subsets are referred to as Best45 and Best105 respectively and contain 22 and 21 experimental observations each. These subsets are further grouped by each parameter value and analysed. Table 14 lists the number of experiments in the Best45 and Best105 sets for each parameter-value configuration.

Table 14: Number of Experiments per Parameter-Value Pairs in Best45 and Best105

	Values	Experiments in Best45	Experiments in Best105
$w^{(2)}$	10^{-4}	7	6
	10^{-2}	7	7
	1	8	8
θ_α	1	-	-
	3	22	21
θ_β	1	-	7
	5	12	14
	10	10	-
	15	-	-

$w^{(1)}$	1	-	2
	3	-	5
	10	4	4
	15	6	4
	30	6	3
	50	6	3

In Table 14 we can see that upon grouping the Best45 and Best105 experiments by each parameter-value pair, a pattern is clearly reflected in the distribution of experiments for parameters θ_α and θ_β . The distribution for parameter θ_α shows a clear predomination for the value 3 suggesting an optimal value that is equivalent or higher. On the other hand, the optimum for parameter θ_β , as suggested by its uneven distribution, should lie closer to a range centred on value 5. For parameters $w^{(1)}$ and $w^{(2)}$ this analysis proves inconclusive. All tested values for these parameters were equally present in the Best45 and Best105 sets. However, the distribution of $w^{(1)}$ provides an indication to test for higher values of the parameter.

Figure 12 and 13 depict scatter plots showing the effect of parameters θ_α and θ_β on the overall classification accuracy.

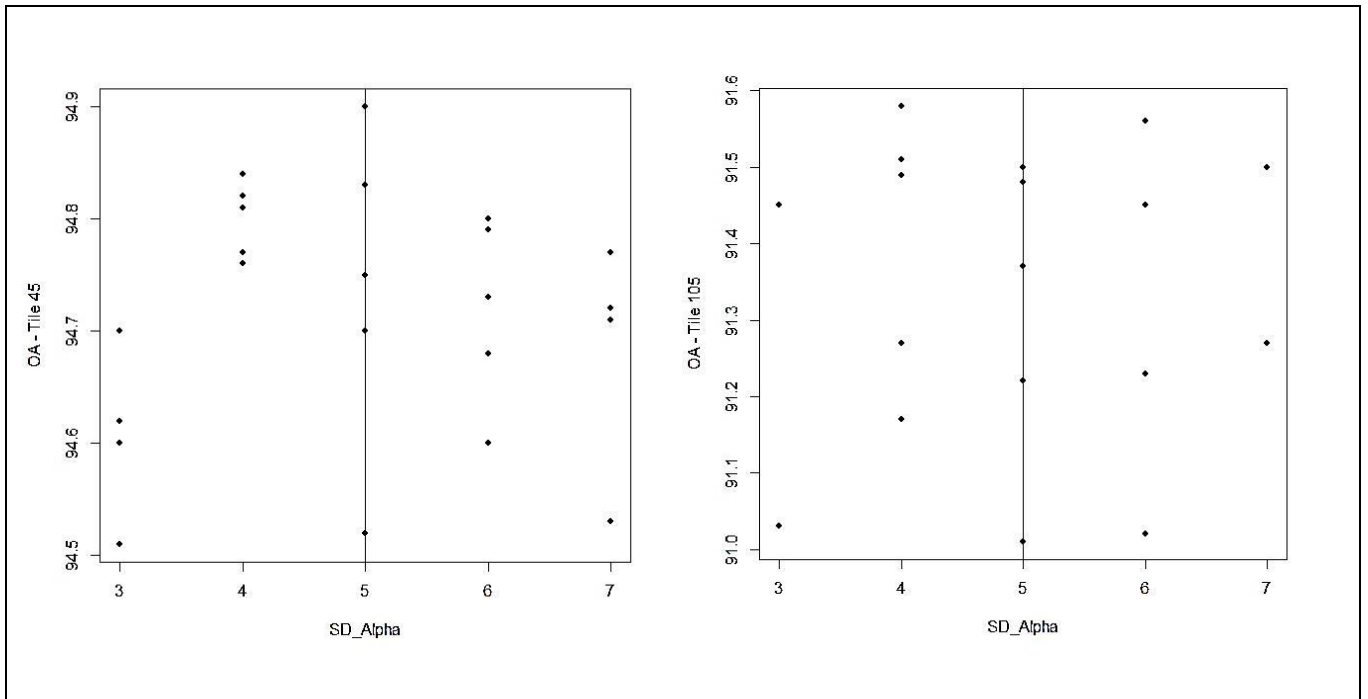


Figure 12: Plot (a): OA on Tile 45 and θ_α ; Plot (b) OA on Tile105 and θ_α

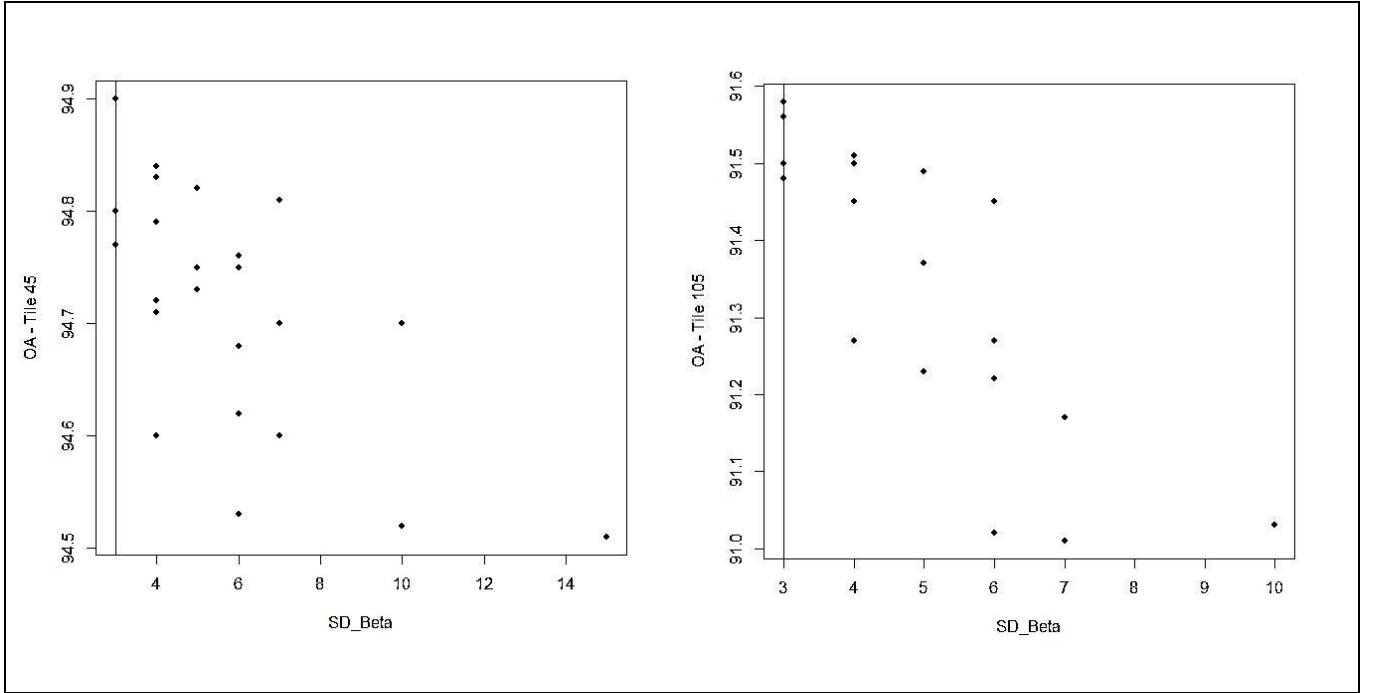


Figure 13: Plot (a): OA on Tile 45 and θ_β ; Plot (b) OA on Tile105 and θ_β

The experiments shown in Figures 12 and 13 are those given in Table 11 of Section 3.3. However, for the purpose of scaling the display of the scatter plots, the results (shown in Appendix C) were subset to contain only experiments that had an overall accuracy of over 94 percent and over 91 percent as measured on Tile 45 and Tile 105 respectively. While the scatter plots in Figures 12 and 13 contradict the intent to come up with a single set of kernel parameters that would be suitable for a general classification case, they at the least indicate θ_α and θ_β configurations that are more stable and consistently high-performing. In the case of parameter θ_α , as shown in Figure 12, the model’s performance on Tile 45 indicates an optimal value close to 5. In the case of Tile 105 the optimum seems closer to 4. Between the two we settle on the value of 5, as is indicated by a vertical line in the plots in Figure 12. For parameter θ_β , on the other hand, the value of 3 seems outstandingly high-performing as observed on the results of both validation tiles. This has also been indicated with a vertical line in the plots of Figure 13.

As a note for the reader, the first batch of dense-CRF experiments (shown in Table 10 of Section 3.3) were too large to be included in the Appendix of this document; but would be included in the ‘Supplementary Data’ submitted for archival purposes at the end of the thesis period to the research supervisor. All experimental results for the second batch of dense-CRF experiments are provided in Appendix C.

In analysing the experimental results of the other three parameters we found that they had a minimal impact on the overall classification accuracy. The optimum value for each parameter is decided as the following: $\theta_\alpha = 5$, $\theta_\beta = 3$, $w^{(1)} = 50$, $w^{(2)} = 0.01$, and $\theta_\gamma = 1$. Once the parameter values are fixed we test the final dense-CRF model on Tile 32 and Tile 103.

6. RESULTS

Having designed the three FCN networks, and optimized for the values of the dense-CRF parameters, we finally present **two** sets of classification results for the four methodologies – SimpleNet (SNN), FuseNet (FNN), SimpleNet+Dense-CRF (DCRF) and ReUseNet (RNN) - as observed on Tile 32 and Tile 103.

1. Result I - In the first set of results, the three FCNs were trained on an image database created by collecting 4000 random samples from each of the 3 training tiles. Furthermore, the image-set used for training the SimpleNet, for the first set of results, was pan-sharpened using the Gram-Schmidt spectral sharpening technique.
2. Results II - For the second set of results, the FCNs were trained on a database of 2000 samples collected from each training tile. Also in this case, the pan-sharpening of the images, used for training the SimpleNet, was done using the Subtractive Resolution Merge technique.

6.1. RESULTS I

The SimpleNet derived in Section 3.1, the FuseNet designed in Section 3.2, the dense-CRF model as specified in Section 3.3 and the ReUseNet designed in Section 3.4 were used to predict land-cover classification maps for Tile 32 and Tile 103 respectively. In this case, the SimpleNet was trained on 4000 samples that were collected from images that were pan-sharpened using the Gram-Schmidt spectral-sharpening technique. The FuseNet and the ReUseNet were trained on 4000 image samples collected from both sets of MS and PAN training image tiles. Table 15 depicts a summary of the overall accuracies of all four classification methodologies as observed on Tile 32 and Tile 103 respectively.

Table 15: Overall Accuracy (%) - Tile 32 and Tile 103

	SNN	FNN	DCRF	RNN
OA(%) – Tile 32	97.53	97.05	98.20	97.21
OA(%) – Tile 103	93.35	92.88	94.53	93.61

An overall assessment of Table 15 shows that the SimpleNet classification performs marginally better than the FuseNet classification results by an average of 0.48%. Further, on applying the dense-CRF inference on the SimpleNet generated scores the overall accuracy of the classification increases by an average of 0.93%. Also, the ReUseNet results marginally improve on the FuseNet performance by 0.45% as measured by an average of the overall accuracy of the two results.

We now compare the cumulative proportion of pixels per land-cover class in each of the 8 (4x2) classification maps. Table 16 shows the percentage of pixels per class-label for each classification map.

Table 16: Percentage of Total Pixels per Class-Label per Classification Method

Tile 32	SNN	FNN	DCRF	RNN	Tile 103	SNN	FNN	DCRF	RNN
Built-Up	27.92	27.74	27.56	27.59	Built-Up	43.74	41.49	43.77	41.82
Low-Veg	23.03	24.60	22.81	26.14	Low-Veg	9.95	10.94	9.77	11.69
Tree	30.74	29.23	30.70	27.84	Tree	35.64	34.97	35.54	34.00
Cars	0.85	1.43	0.57	1.74	Cars	0.89	1.51	0.58	1.82

Water	1.14	0.86	1.12	0.00	Water	0.43	0.29	0.37	0.00
Imp. Surf	16.33	16.15	17.24	16.69	Imp. Surf	9.34	10.79	9.97	10.66

Table 16 provides a nuanced picture of the change affected by each methodology on the proportion of pixels per class-label as seen on the final classification maps. In this table, observing the changing proportions of pixels classified as class ‘Cars’, we can see how the FuseNet and ReUseNet methods consistently predict a significantly higher proportion of pixels as ‘Cars’ than do the other two methodologies. Further, the decrease in the proportion of pixels classified as class ‘Cars’, as effected by the dense-CRF on the SimpleNet classification scores, is also important to note. In Table 14, it is interesting to observe the column for class ‘Water’. In this, we again notice a significant difference between the performance of the FuseNet and the ReUseNet methods as compared to the other two methodologies. Further, it is remarkable that while the FuseNet manages to predict ‘Water’ class pixels, the ReUseNet is completely unable to do so. Finally, it is also worth observing that in the classifications results for Tile 32, for class ‘Low-Veg’, the FuseNet and the ReUseNet methods predict a higher percentage of pixels as compared with the other two classification methods. Further, for class ‘Tree’ this trend is reversed, where now the FuseNet and the ReUseNet methods predict a smaller proportion of pixels as compared with the results of SimpleNet and SimpleNet+dense-CRF methodologies.

We now present the producer and user accuracy metrics and the classification maps for all four methods as observed on Tile 32. Thereafter, we also present the producer and accuracy metrics for the classification results as observed on Tile 103. The classification maps of Tile 103, however, can be seen in Appendix F.

Tile 32

Table 17 presents a summary of the producer and the user accuracy statistics for all four classifications results as observed on Tile 32.

Table 17: Producer and User Accuracy Metrics for Tile 32

	SNN (%)		FNN(%)		DCRF(%)		RNN(%)	
	UA	PA	UA	PA	UA	PA	UA	PA
Built-Up	94.02	98.11	93.02	98.20	95.84	98.53	94.77	97.79
Low-Veg	99.19	99.57	99.12	99.54	99.31	99.81	97.91	99.93
Tree	99.59	99.62	99.52	99.51	99.61	99.84	99.38	98.67
Cars	95.07	63.03	82.76	64.06	97.80	51.72	82.02	70.57
Water	84.83	62.69	68.69	30.42	88.90	91.23	0.00	0.00
Imp. Surf	97.47	90.50	97.00	88.26	98.19	93.02	97.32	91.67

In Table 17, we can observe the difference between the proposed classification methods by observing their performance for class ‘Cars’ and class ‘Water’. We see that for class ‘Cars’ the SimpleNet user accuracy is significantly higher than that of the FuseNet model. It is also interesting to observe that the producer accuracy of class ‘Cars’ drops significantly after applying the dense-CRF inference on the SimpleNet classification results. Further, the ReUseNet shows improved performance in detecting class ‘Cars’ as can be observed by the increase in the producer accuracy for the class as compared with the FuseNet classification results. For class ‘Water’ one can observe that the SimpleNet and the SimpleNet+dense-CRF methods perform significantly better than the FuseNet and the ReUseNet methods. The consistent inability of the ReUseNet to predict pixels of class ‘Water’ is a striking result and of noticeable concern.

In Figure 14, we present the classification maps for the SimpleNet and the FuseNet methods as observed on Tile 32. Here, we can visually corroborate the patchy classification of class ‘Water’ by the FuseNet method and compare it with the smoother segmentation results of the SimpleNet classification map. Further, one can also observe the over-estimation of class ‘Cars’ in the FuseNet map as compared with the SimpleNet results.

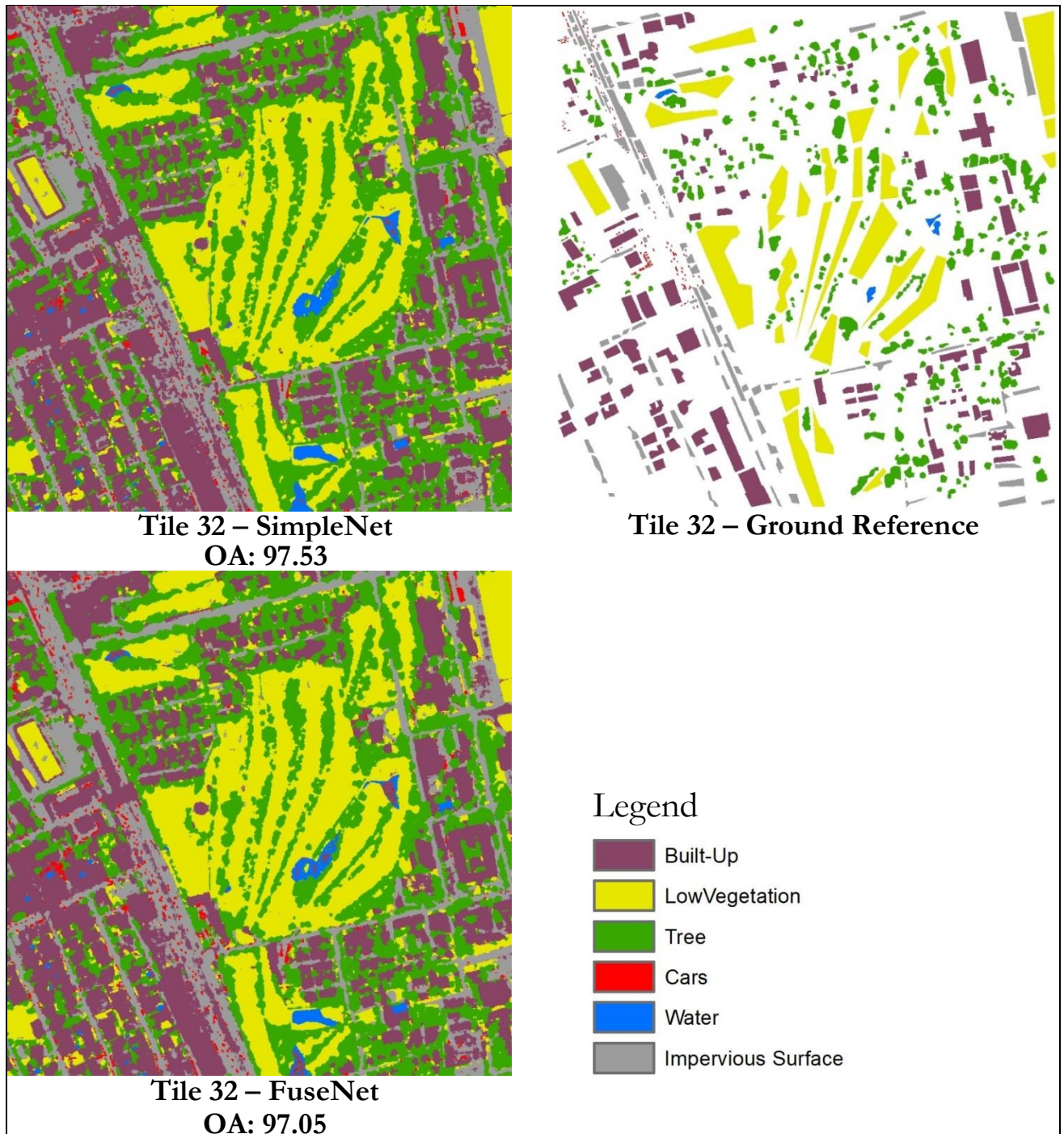
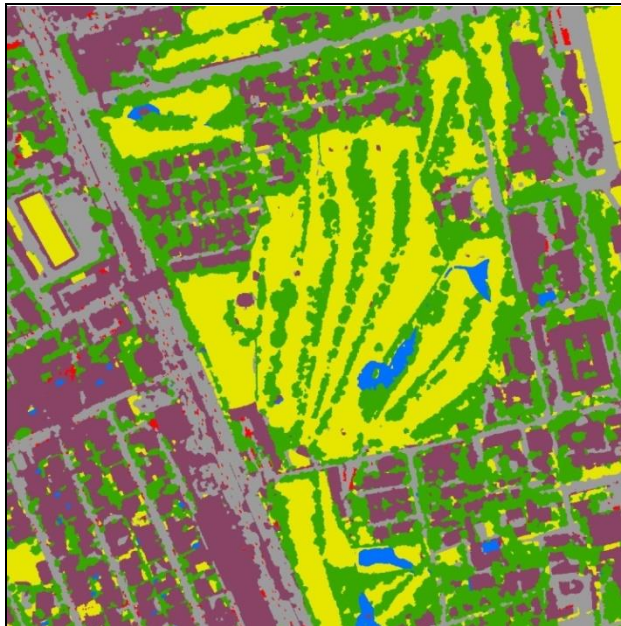
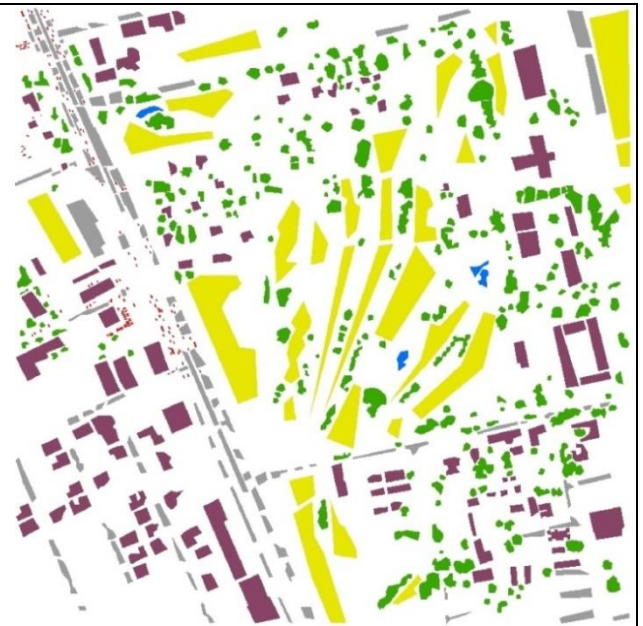


Figure 14: SimpleNet and FuseNet Classifications Maps for Tile 32

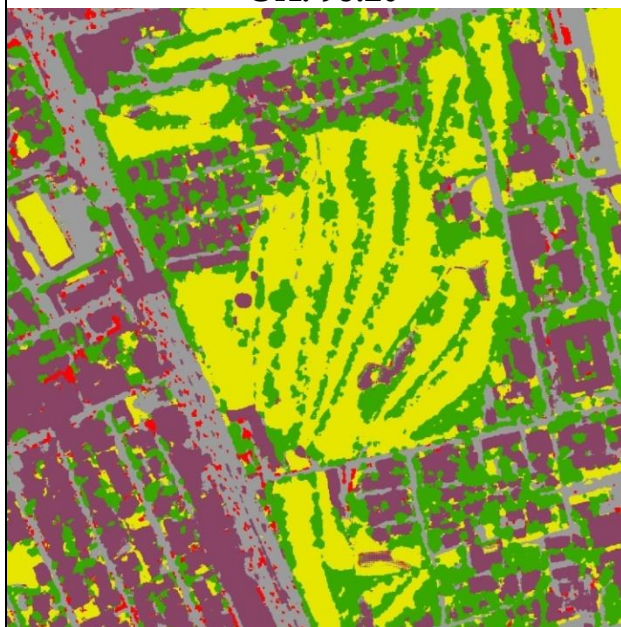
In Figure 15, we present the classification maps of the SimpleNet+dense-CRF and the ReUseNet methodologies.



Tile 32 – SimpleNet+Dense-CRF
OA: 98.20



Tile 32 – Ground Reference



Tile 32 – ReUseNet
OA: 97.21

Legend



Figure 15: SimpleNet+Dense-CRF and ReUseNet Classification Maps for Tile 32

In comparing the classification maps of the SimpleNet+dense-CRF and the ReUseNet methods, one can clearly see the performance of the dense-CRF inference in refining the results of the SimpleNet classification. In Figure 15 we can see the inability of the ReUseNet to predict pixels of class 'Water' and compare that with the smooth, classification output of the SimpleNet+dense-CRF method. Further, observing the ReUseNet classification map, one can see its over-estimation of class 'Cars' as compared with the SimpleNet+dense-CRF classification map.

Tile 103

Table 18 presents a summary of the producer and the user accuracy statistics for all four classifications results as observed on Tile 103.

Table 18: Producer and User Accuracy Metrics for Tile 103

	SNN(%)		FNN(%)		DCRF(%)		RNN(%)	
	UA	PA	UA	PA	UA	PA	UA	PA
Built-Up	95.18	96.13	97.45	93.22	95.83	97.95	98.10	93.75
Low-Veg	98.77	77.37	98.87	82.18	99.34	77.55	97.90	86.31
Tree	98.23	99.66	97.15	99.67	98.42	99.75	97.91	99.66
Cars	66.83	65.73	57.60	65.99	78.04	59.27	44.40	65.78
Water	88.97	62.09	98.17	51.20	98.63	55.30	0.00	0.00
Imp. Surf	62.97	89.69	56.22	94.27	68.81	94.61	59.81	97.76

In Table 18, we see that the producer accuracy for class ‘Low-Veg’, for the SimpleNet and the SimpleNet+dense-CRF methods, is considerably less compared with its counterpart in the case of the FuseNet and the ReUseNet methods. For class ‘Impervious Surface’ it is striking to note its consistently low user accuracy across the four methods. However, the producer accuracies for the same class are reasonably high, with the FuseNet and ReUseNet performing better than the other two methods. In addition, for class ‘Cars’ one can observe how the dense-CRF inference improves on the user accuracy of the SimpleNet method, but shows decreased performance as measured by their producer accuracy scores. Further, the low user-accuracy scores for class ‘Cars’ in the case of the FuseNet and ReUseNet classification results, is explained by the increased prediction of class ‘Cars’ by the two methods (as also shown in Table 16). The classification maps for Tile 103, due to considerations of length, are provided in Appendix F. The confusion matrices for all classifications maps in Results I are given in Appendix H.

6.2. RESULTS II

For this set of results, the SimpleNet was trained on 2000 samples that were collected from each training image that was pan-sharpened using the Subtractive Resolution Merge technique. The FuseNet and the ReUseNet were trained on 2000 image samples collected from each training image in MS and PAN sets. Table 19 depicts a summary of the overall accuracies of all four classification methodologies as observed on Tile 32 and Tile 103 respectively.

Table 19: Overall Accuracy (%) - Tile 32 and Tile 103 (Result II)

	SNN	FNN	DCRF	RNN
OA(%) – Tile 32	89.06	96.07	89.63	96.93
OA(%) – Tile 103	85.62	93.24	86.10	93.29

An overall assessment of Table 19 shows that the FuseNet and the ReUseNet methods have outperformed their counterparts by a significant margin. While the FuseNet classification improves on the overall accuracy of the SimpleNet classification by an average of 7.32%, the ReUseNet classification improves on the overall accuracy of the SimpleNet+Dense-CRF classification by an average of 7.25%. Furthermore, the dense-CRF inference certainly helps in refining the SimpleNet generated classification maps with an average improvement of 0.53% as measured on the overall accuracy of the two

classifications. Also, the ReUseNet improves on the overall accuracy of the FuseNet method by an average of 0.46%.

We now compare the cumulative proportion of pixels per land-cover class in each of the 8 (4×2) classification maps. Table 20 shows the percentage of pixels per class-label for each classification map.

Table 20: Percentage of Total Pixels per Class-Label per Classification Method (Result II)

Tile 32	SNN	FNN	DCRF	RNN	Tile 103	SNN	FNN	DCRF	RNN
Built-Up	29.66	29.91	32.43	27.13	Built-Up	40.88	42.91	41.44	39.81
Low-Veg	24.16	23.19	23.70	24.84	Low-Veg	6.09	10.59	5.58	12.73
Tree	35.17	30.85	33.67	29.20	Tree	42.30	35.53	41.55	33.99
Cars	0.77	1.44	0.52	1.79	Cars	0.57	1.43	0.35	2.21
Water	2.69	0.00	1.98	1.30	Water	0.84	0.00	0.73	0.49
Imp. Surf	7.55	14.62	7.69	15.73	Imp. Surf	9.32	9.54	10.34	10.78

In Table 20 we can see that for class ‘Tree’ the FuseNet and the ReUseNet predict a significantly smaller proportion of pixels as compared with the SimpleNet and the SimpleNet+dense-CRF methods. This trend is reversed, for the classification results of Tile 32 for class ‘Impervious Surface’ and for the classification results of Tile 103 for class ‘Low-Veg’. Furthermore, it is interesting to observe that the FuseNet and the ReUseNet methods consistently over-estimate the presence of class ‘Cars’ as compared with the SimpleNet and the SimpleNet+dense-CRF methods. Also, for class ‘Water’ we observe that while the FuseNet is unable to predict any pixels as being of class ‘Water’, the ReUseNet improves on the singular FuseNet architecture and rectifies the classification accuracy for this particular class. This result is in stark contrast to the first set of classification results provided in Table 16. We now present the producer and user accuracy metrics and the classification maps for all four methods as observed on Tile 32. Thereafter, we also present the producer and accuracy metrics for the classification results as observed on Tile 103. The classification maps of Tile 103, for the second set of results, can be seen in Appendix G.

Tile 32

Table 21 presents a summary of the producer and the user accuracy statistics for all four classifications results as observed on Tile 32.

Table 21: Producer and User Accuracy Metrics for Tile 32 (Result II)

	SNN(%)		FNN(%)		DCRF(%)		RNN(%)	
	UA	PA	UA	PA	UA	PA	UA	PA
Built-Up	75.97	95.16	89.37	98.33	75.10	96.69	92.87	98.07
Low-Veg	97.89	97.51	99.53	98.54	98.52	98.61	99.04	99.16
Tree	96.71	99.98	99.44	99.78	98.55	99.94	99.50	98.99
Cars	93.09	58.90	90.43	59.00	96.72	49.77	81.61	68.98
Water	45.71	94.94	0.00	0.00	79.69	100.00	89.12	92.08
Imp. Surf	89.65	37.61	96.46	84.30	88.40	36.22	96.56	86.86

In Table 21, the low producer accuracy of the class ‘Impervious Surface’ in the SimpleNet and the SimpleNet+dense-CRF classification results is a striking result. Further, class ‘Cars’ is a challenging class to accurately predict as is shown by its low producer accuracies across all method results. It is also interesting to observe that the ‘Water’ class, which was missing from the FuseNet classification has been

accurately captured by the ReUseNet model. This reflects the self-learning capability of the ReUseNet architecture and is in stark contrast with its performance as reflected in Results I. In Figure 16, we present the classification maps for the SimpleNet and the FuseNet methods as observed on Tile 32.

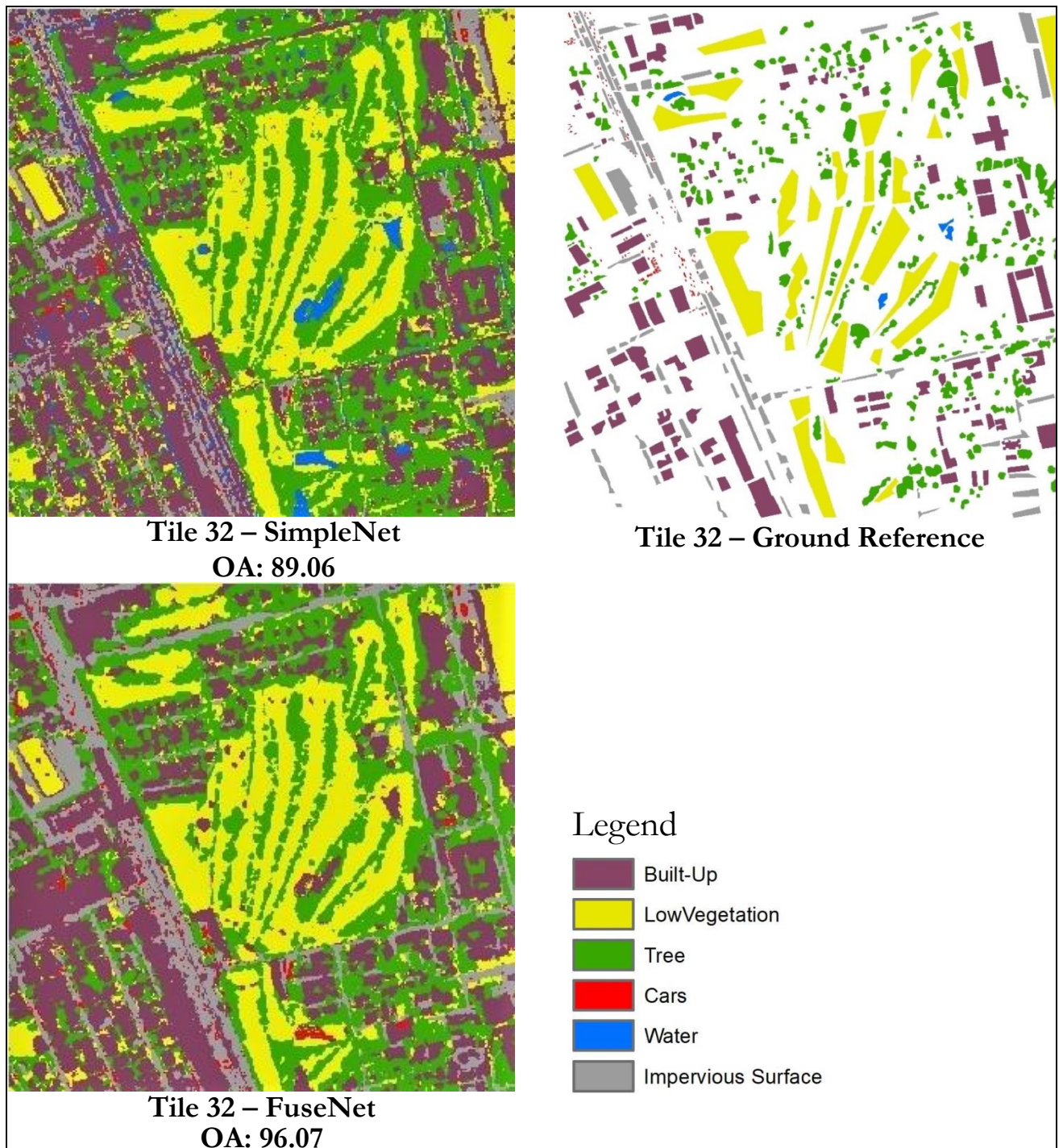


Figure 16: SimpleNet and FuseNet Classifications Maps for Tile 32 (Result II)

In comparing the classification maps, with a particular focus on the road network that runs - from centre-bottom to the top-left corner - we can discern a more accurate classification for class 'Impervious Surface' by the FuseNet method. This visual hypothesis is corroborated by observing the producer accuracy for this class in the two classification methods presented in Table 21. Here, one can also observe the over-

estimation of class ‘Cars’ by FuseNet method as compared with the SimpleNet results. Finally, FuseNet’s inability to accurately predict any pixels as belonging to class ‘Water’ is noteworthy. In Figure 17, we present the classification maps of the SimpleNet+dense-CRF and the ReUseNet methodologies.

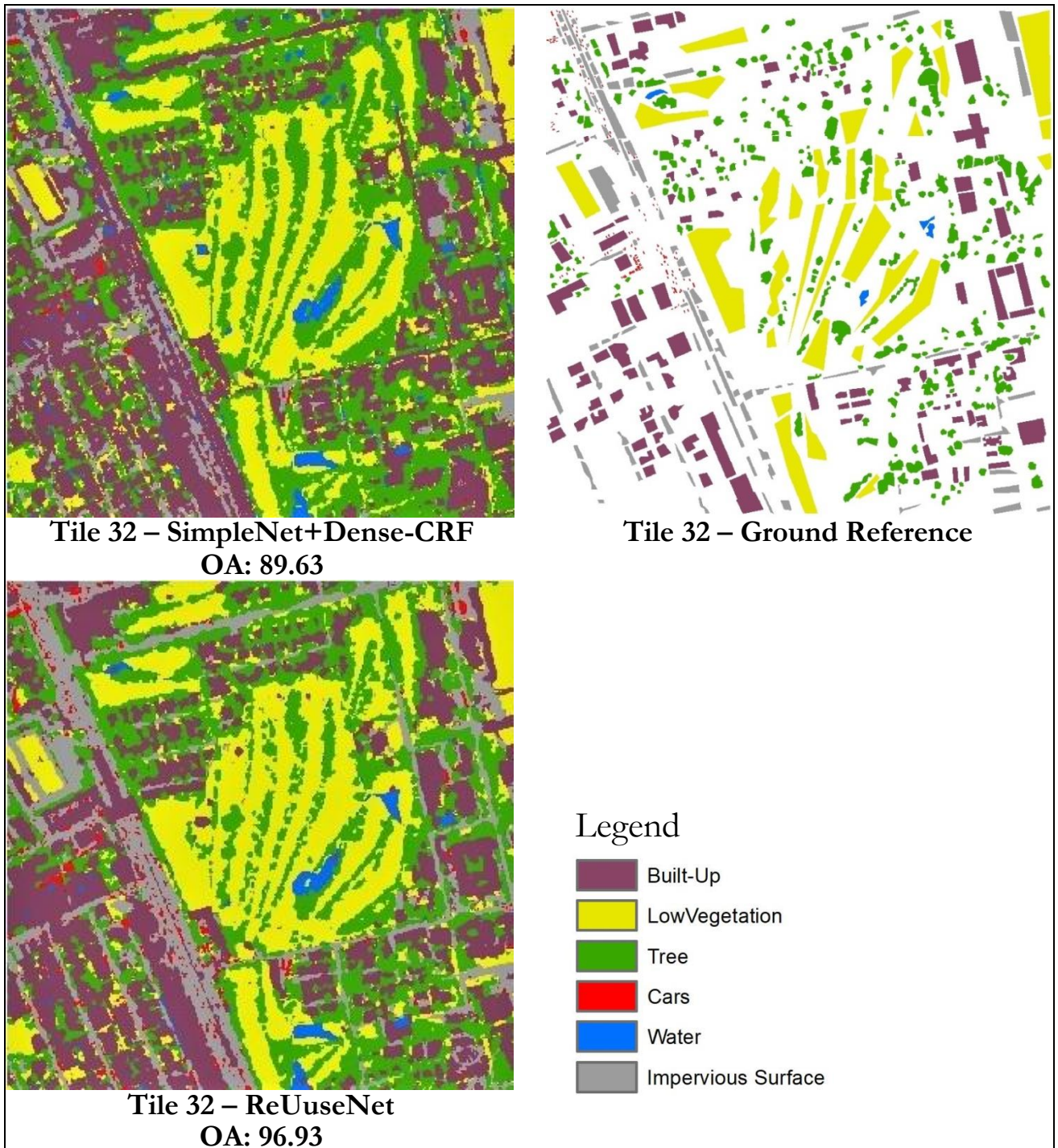


Figure 17: SimpleNet+Dense-CRF and ReUseNet Classification Maps for ‘Tile 32 (Result II)

The classification maps in Figure 17 reveal how the SimpleNet+dense-CRF classification frequently overestimates the presence of the ‘Built-Up’ class at the expense of the ‘Impervious Surface’ class. Further, it is interesting to note how the ReUseNet improves on the classification of class ‘Water’ as compared with the results of the FuseNet classification shown in Figure 16. Finally, from inspecting the two

classification maps one can discern how the ReUseNet tends to over-predict pixels as belonging to class ‘Cars’.

Tile 103

Table 22 presents a summary of the producer and the user accuracy statistics for all four classifications results as observed on Tile 103.

Table 22: Producer and User Accuracy Metrics for Tile 103 (Result II)

	SNN(%)		FNN(%)		DCRF(%)		RNN(%)	
	UA	PA	UA	PA	UA	PA	UA	PA
Built-Up	93.55	89.14	93.90	94.50	94.16	89.63	96.87	91.47
Low-Veg	98.75	49.93	99.12	83.70	99.78	50.56	98.23	87.81
Tree	84.82	99.99	98.00	99.71	85.18	99.98	98.44	99.40
Cars	81.78	62.30	47.81	59.98	89.76	50.48	51.99	62.61
Water	67.40	95.24	0.00	0.00	70.29	97.42	92.87	93.76
Imp. Surf	47.98	84.52	68.48	92.91	48.37	87.34	59.44	96.56

In Table 22, we see that the producer accuracy for class ‘Low-Veg’, for the SimpleNet and the SimpleNet+dense-CRF methods, is considerably less compared with its counterpart in the case of the FuseNet and the ReUseNet methods. For class ‘Impervious Surface’ it is striking to note its consistently low user accuracy across the four methods. However, the producer accuracies for the same class, shows the reverse trend. For class ‘Impervious Surface’ the FuseNet and ReUseNet methods perform better than the SimpleNet and SimpleNet+dense-CRF methods. In addition, for class ‘Cars’ one can observe how the dense-CRF inference improves on the user accuracy of the SimpleNet method, but shows decreased performance as measured by their producer accuracy scores. The classification maps for Tile 103, due to considerations of length, are provided in Appendix G. The confusion matrices for all classifications maps in Results II are given in Appendix I.

7. DISCUSSION AND CONCLUSION

The FCN(s) designed as part of this methodology, along with the dense-CRF implementation, were able to address the need outlined by the objectives of this research. The FuseNet demonstrated the capability of FCN structures to alter image resolution and dynamically fuse multi-scale images for generating high-accuracy land cover classification maps. While Results I indicate the superior performance of the SimpleNet model over the FuseNet approach, as measured by the overall classification accuracy, the overall accuracy of the FuseNet model is nevertheless high. Results II, however, clearly demonstrate the consistent performance of the FuseNet as compared to the SimpleNet classification results. Both sets of results provide compelling evidence for the ability of FCN structures to combine multi-scale images with high-accuracy. Also, in terms of computational cost and tractability the FuseNet model is more convenient and overall less of a burden on system and man-power resources. An overall assessment of the classification results indicates that while the FuseNet is almost as accurate as the SimpleNet model, it utilizes only half as much of the systems' resources. This behaviour can be gauged by observing the average training time per experiment as shown in Table 22. This is possible because the FuseNet accommodates input from discrete image datasets and creates an end-to-end framework that fuses and classifies multi-scale raw-image pixels simultaneously. The ReUseNet, on the other hand, demonstrates the self-learning capability of the FCN-RNN structure by its improved performance over the singular FuseNet classification model (as observed across all classification results provided in Chapter 6). The ReUseNet also shows the FCNs ability to capture pixel-context through convolutional filter weights and further demonstrates the flexibility of FCN structures to implement non-linear architectures. That our two main FCN networks, that were primarily crafted in order to fulfil the first two objectives of this research, have managed to achieve their functional utility, serves to underscore the strength of this study. Also, the dense-CRF inference is successfully applied to the classification scores of the SimpleNet and improves the overall accuracy of the classification.

We now consolidate our findings from this research and address the questions associated with each of the four research objectives, in the order specified in Section 1.4.

Research Questions - Objective 1

How is the dynamic fusion of multi-scale images achieved within the network design? What architectural parameters and network training hyper-parameters influence the success of applying CNN architectures for image fusion?

The dynamic fusion of multi-scale images is achieved by utilizing convolutional and de-convolutional filters within the FCN architecture. The latter are essentially upsampling filters with learnable parameter weights. In order to implement the FuseNet we had to employ a skip-architecture that allowed for the non-linear arrangement of layers within the network. The sequence of steps followed within the network for implementing a dynamic fusion network are: (i) downsampling the PAN input to match the dimensions of the MS input, (ii) process concatenated features through a block of convolutional layers with a sequence of 3×3 and 5×5 convolutional filters with varying factors of dilation, and (iii) upsampling bottleneck features to PAN dimensions for producing final class-probability scores. For a detailed description of the method used to derive the FuseNet please see Section 3.2.

The classification accuracy of the fusion network depends on a number of aspects such as, the factor used for downsampling image features, the point of concatenation of the MS and PAN features, the ratio of the number of MS features to the number of PAN features, the number of convolutional blocks in the bottleneck layer along with the size of the filters and their respective dilation factors in each block, and

finally the factor used for upsampling the network features to the dimensions of the output image. From the FuseNet experimental results presented in Appendix B we arrive at the following conclusions: (i) Step-wise downsampling is preferable to downsampling in a single step, (ii) enhancing the number of MS features prior to concatenation with the PAN features increases the overall accuracy of the FuseNet classification, (iii) the upsampling of network features in a single-step is better than upsampling in multiple-steps. Overall the performance of the FuseNet demonstrates the utility of the FCN classification paradigm to enable the dynamic fusion of multi-scale images for producing high-accuracy land-cover classification maps in an end-to-end framework. Furthermore, the significantly shorter training time for the FuseNet, as compared with the SimpleNet, is an added advantage for the method.

What network parameters enable the pixel-level classification of VHR remotely-sensed satellite imagery?

In assessing the impact of the network parameters on the overall classification accuracy of the validation set of Tile 45 and Tile 105, we made a broad distinction between architectural parameters and network training hyper-parameters. The latter group includes values pertaining to the number of training epochs, the learning rate of the neural network, and the weight decay associated with each weight update. In refining the SimpleNet and the FuseNet, the methods for which are described in Section 3.1 and Section 3.2 respectively, we separately optimized the architecture and the network hyper-parameters.

In the case of the SimpleNet experiments, changing the patch-size of the input training image was found to have a marginal impact on the overall accuracy of the classification. We can see this from the experimental results listed in Appendix A. However, the computational cost required for training the network increases quite substantially. Further, the introduction of max-pooling layers into the baseline architecture leads to an increase in the overall accuracy of the classification. We also conclude that the size, dilation and sequence of the convolutional filters are very significant for generating high-accuracy land-cover classification maps. The size of the convolutional filter determines the receptive field of the kernel, that determines the portion of the image ‘seen’ by the convolutional filter (Maggiori et al., 2017). Having large filter sizes means having a large number of trainable parameters, which effectively slows down the time required for training the network. Furthermore, while it may seem desirable to increase the number of filters per convolutional block, our experimental results indicate this to not have a very significant impact on the overall classification accuracy.

In addition to these architectural parameters, the design of the FCN(s) also entails optimizing for hyper-parameter values that control the training of the network. These are namely, learning rate, weight decay and the number of training epochs. Through our experiments we indicate the importance of choosing the right learning rate for training the FCN(s). The parameter for weight decay and the number of training epochs, out of the values tested and shown in Appendix A and B, have shown to have a marginal effect on the overall accuracy of the classification results. See Section 5.1 and Section 5.2 for a more detailed discussion of the same.

Research Questions - Objective 2

How is the FCN-RNN developed?

First, the FuseNet is refined over many experimental trials. The method, described in Section 3.1 and the experimental results given in Appendix B, illustrate the various permutations of the architectural parameters and the network training hyper-parameters that were incrementally tested in order to optimize the architecture of the FuseNet. Once this is achieved, we bring together two copies of the FuseNet and join them in a manner such that the class-probability scores predicted by the 1st FCN are concatenated

with the raw-PAN input of the 2nd FCN. At this juncture we additionally experiment with introducing a second loss layer into the network, and varying the respective weights of the two objective loss functions. The results for these experiments are provided in Appendix D. The final ReUseNet architecture is trained on an image database of 4000 (and 2000) samples of training image-sizes of $32 \times 32 \times 4$ and $128 \times 128 \times 1$ taken from the set of MS and PAN images respectively. The ReUseNet is trained using two different learning rates 10^{-4} and 10^{-5} , with each training step lasting for 100 and 50 epochs respectively. The value for the weight decay parameter is held constant at 5×10^{-4} . The trained network is used to predict land-cover classification maps for Tile 32 and Tile 103 respectively. The resulting metrics and maps are presented in Chapter 6.

What aspects of the FCN-RNN architecture motivate the comparison with the MRF/CRF inference?

The MRF/CRF inference models contextual relationship by aggregating a measure of similarity based on class-labels and spectral DN values between pairs of pixels defined in a Markovian neighbourhood. This pairwise potential energy in linear combination with the unary potential energy, derived from a FCN generated class-probability scores, is used to generate a posterior probability map for the distribution of the random variable.

On the other hand, a recurrent neural network structure enables the combined FCN model to retain information about past inputs, thereby allowing it to discover contextual dependencies between pixels that might not be in close proximity (Pascanu, Mikolov, & Bengio, 2013). It does so via the use of convolutional filter weights, which are the basic building blocks of all FCNs. The receptive field of a convolutional filter is a function of the kernel size; and, in effect, regulates the size of the input image or feature that is linearly transformed by its parameter weights. The receptive field, in this manner, can be interpreted as a measure of the size of the neighbouring context of an image pixel. As image features are cascaded through the network and acted upon by additional convolutional filters, the effective receptive field size in relation to the input image drastically increases. For instance, in a two layer network (consisting of two 3×3 convolutional filters arranged one after the other) the effective receptive field of the 2nd convolutional filter in relation to the input image would be 9×9 . It is for this reason that increasing the depth of a neural network does not necessarily lead to a substantial increase in the overall accuracy of the classification. See the SimpleNet results in Appendix A for experiments relating the impact of layer depth on the overall accuracy of the classification. Through this cascading procedure, however, CNNs can learn to predict context-sensitive class-probability scores.

In the case of the recurrent neural network or the RNN, two instances of a high-performing FCN are joined such that their network parameters can be trained in an end-to-end block of parametric weights. Specifically, the network concatenates the class-probability scores of the 1st FCN with the PAN input of the 2nd FCN. The RNN, in this manner, explicitly incorporates pixel-context in the form of class-probability scores, as an additional input into the 2nd instance of the FuseNet. By simultaneously training the parametric weights of both networks, the RNN, it can be argued, simulates the underlying mechanism of the CRF inference, where class-probability scores in addition to spectral DN values are used for refining classification maps.

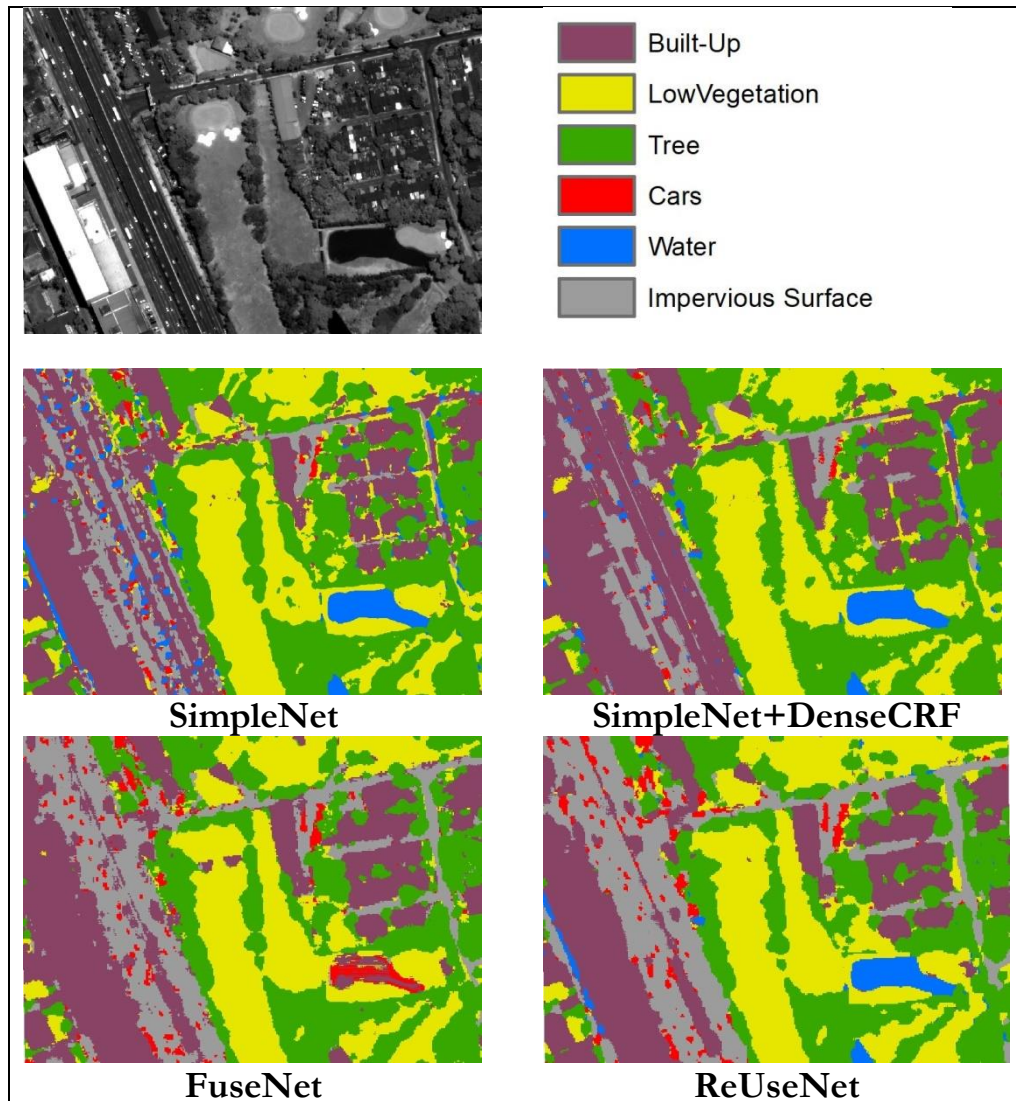


Figure 18: Visual Comparison between Dense-CRF and ReUseNet

Figure 18 shows the classification maps (from Results II) of the four methodologies developed for this research on a small portion of Tile 32. The aim of the graphic is to juxtapose the classification performance of the SimpleNet+Dense-CRF and the ReUseNet methodologies. CRFs have long been applied to refine coarse pixel-level label predictions to produce sharp boundaries and fine-grained segmentation maps (Zheng et al., 2015). By enforcing spectral and class-label homogeneity, the CRF inference favours the generation of smoothly connected regions of the same land-cover class. While this behaviour can be prominently seen when comparing the two classification maps of the SimpleNet and the SimpleNet+Dense-CRF methodologies, we argue that a similar effect can be observed in comparing the FuseNet and the ReUseNet classification maps as well. The ReUseNet implementation, in this manner, demonstrates the ability of FCN architectures to account for spatial autocorrelation between image pixels in a VHR multispectral remote-sensing image.

Research Questions - Objective 3

How is the fully-connected dense-CRF inference applied to the classification problem?

The mean-field approximation to the dense-CRF inference is formulated as a linear combination of two Gaussian kernels (see Equation (10) in Section 2.3) and is implemented by optimizing for the five parameters that together define the pairwise potential energy of the dense-CRF approximation. In this equation, the first kernel considers both spatial and spectral distances between two pixels, while the second kernel considers only spatial proximity. The five parameters either control the weight of the respective kernels in the overall pairwise-energy computation, or are factors controlling the ‘width’ of the kernel or its permissible deviation of values from that of pixel i . The parameter values of the dense-CRF model are optimized, by iteratively refining the search range of the values and by selecting final parameter configurations that consistently generated high-accuracy classification maps. For a detailed discussion of the method see Section 3.3. For a discussion of the sensitivity of the overall classification accuracy to changes in kernel parameter values see Section 5.3.

What are the relevant inputs and parameters required for specifying the dense-CRF model? What is their impact on the classification results?

The implementation of the mean-field approximation to the dense-CRF inference requires two user inputs: (i) raw-image with spectral DN values and (ii) class probability scores for each image pixel.

Equation (10) listed in Section 2.3 shows the formulation of the pairwise energy potential as a linear combination of two Gaussian kernels. This equation is determined by the following five parameters: θ_α , θ_β , $w^{(1)}$, θ_γ and $w^{(2)}$. The two kernels are respectively called the ‘appearance kernel’ and ‘smoothness kernel’. The ‘appearance kernel’ takes into account both ‘colour’, i.e. a measure of spectral reflectance captured by the raw-DN value, and spatial proximity. The ‘smoothness kernel’ is said to remove small isolated regions from the classification maps (Krähenbühl & Koltun, 2011). For this reason, the value of the θ_γ parameter is kept small. In the experimental results listed in Appendix C, we found the overall classification accuracy as observed on Tile 45 and Tile 105 to be most sensitive to changes in the values of parameters θ_α and θ_β . These parameters control the spatial and the spectral similarity between pixel-pairs allowed for in the first Gaussian kernel of Equation (10). We also tested the other three parameters over longer search ranges and finally decided on the five parameter value combinations that consistently produced higher classification maps. The experimental method explained in Section 3.3, the results reported in Appendix C, and the sensitivity analysis provided in Section 5.3, illustrate the impact of making incremental changes to kernel parameter values on the overall classification accuracy.

Research Questions – Objective 4

What is the difference in classification accuracy between a FCN trained on pan-sharpened images and FCN that dynamically fuses discrete multi-scale images?

In Results I in Chapter 6, we see that the SimpleNet trained on the set of images that were pan-sharpened using the Gram-Schmidt spectral sharpening technique performs better as compared to the FuseNet classification showing an average improvement in overall accuracy of 0.43% as observed on the classification maps of Tile 32 and Tile 103 respectively. Upon inspecting the class-specific producer and user accuracy metrics for the classification of Tile 32 (as provided in Table 17) we see that the FuseNet model has difficulty in accurately predicting pixels of class ‘Water’ and class ‘Cars’. Further, on inspecting the classification results of Tile 103 (as shown in Table 18) we see that the user-accuracy of the class ‘Impervious Surface’ is low for the FuseNet classification results. In summary, Results I indicate the superior performance of the SimpleNet FCN that was trained using images that were pan-sharpened using the Gram-Schmidt spectral sharpening technique. Nevertheless, the high-accuracy displayed by the FuseNet model also illustrates the ability of FCN architectures to dynamically fuse multi-scale images.

Results II however indicate the consistently high-performance of the FuseNet model in comparison with the SimpleNet. In Results II, we see that the FuseNet method outperforms the SimpleNet method by an average of 7.32% as measured by the overall classification accuracy of Tile 32 and Tile 103 respectively. Upon inspecting the class-specific producer and user accuracy metrics for the classification of Tile 32 (as provided in Table 21), and the confusion matrices provided in Appendix I, we see that the SimpleNet finds it difficult to distinguish between ‘Built-Up’ and ‘Impervious Surface’ class pixels. Also, the inability of the FuseNet to detect pixels of class ‘Water’ is noteworthy. In the classification results of Tile 103 (as shown in Table 22), we observe SimpleNet’s persisting troubles with the class ‘Built-Up’. Further, reflecting on the breakdown of producer and user accuracies it seems that the class ‘Cars’ is a challenging class for both the SimpleNet and the FuseNet methods respectively. In summary the Results II highlight the superior performance of the FuseNet as compared with the SimpleNet classification results, effectively demonstrating the FCNs ability to dynamically fuse multi-scale images and to produce consistently high accuracy land-cover classification maps.

What is the difference in classification accuracy between FCN+CRF and RNN?

In Results I, upon comparing the classification accuracy of the SimpleNet+dense-CRF and the ReUseNet methods, we see that the method using the dense-CRF inference outperforms the classification of the ReUseNet method by 0.96% as measured by the average of the overall accuracy of the two classification methods respectively.

In Results II, we observe that the ReUseNet classification clearly outperforms the SimpleNet+dense-CRF method by an average of 7.25% as measured by the overall accuracy of the two classification methods respectively.

What is the difference in computational requirements between the proposed methods?

The average training time per epoch for training the three FCNs, as derived from the experiments given in Appendix A, B and D respectively, is given in Table 23.

Table 23: Average Training Time per FCN

	Average Training Time per Experiment (sec)	Average Training Time per Epoch (sec)
SimpleNet	11897	59.49
FuseNet	7497	62.48
ReUseNet	22199	147.99

Here, we can see that the ReUseNet takes the maximum amount of time to train. This is quite expected since the ReUseNet has over twice the amount of parameters contained within a singular FuseNet network. We see from Table 22 that the average training time per experiment in the FuseNet case is the minimum amongst the three values. Further, it bears mentioning that the average training time per epoch is a cumulative function of the size of the image database, along with the number of network parameters that are required to be learned during the training procedure.

Concluding Remarks

In this study, we designed the FuseNet that can dynamically fuse multi-scale VHR remote-sensing images for generating high-accuracy land-cover classification maps in an end-to-end framework with striking computational efficiency. We also implemented a recurrent network architecture in the form of the ReUseNet that was able to improve on the classification of the standalone fusion network. The RNN

effectively displayed the ability of FCN structures to (i) capture pixel-context through the weights of the convolutional filter, where receptive fields determine the neighbourhood of the pixel, and (ii) to account for these local spatial and spectral dependencies by incorporating the class probability scores of a fusion network as an explicit input into the succeeding instance of the network. The RNN, in this manner, highlights the extended context learning capability of the FCN classification paradigm.

However, there remain significant limitations to the proposed approach. Firstly, there is an inherent randomness that is incorporated at the stage of preparing an image database for training the FCN(s). The image database is created by randomly sampling 2000 (or 4000) row and column numbers from a set of ground-referenced pixel locations from each training image. For this reason, it is unfortunately difficult to reproduce the results of this research, but comparable results can nevertheless be achieved by following the research methodology provided herein. In addition, the limited availability of densely labelled ground reference data was a concern for this study.

For future research efforts, the inherent richness of possible parameter variations in even modestly sized FCN structures offers large scope for increased experimentation with architectural and hyper-parameter configurations. While the architectural elements and the network training hyper-parameters for the SimpleNet and the FuseNet models were extensively experimented with, only a limited number of experiments were carried out on the ReUseNet model. Further methodical exploration of the RNN structure could perhaps reveal a more suitable architecture for fusing multi-scale VHR remote-sensing images. In other recommendations, the FCN networks that were designed as part of this research method could be augmented with the use of a customized loss function - one that was sensitive to the 'proximity' between class-labels. This means that, given a true class-label of 'Built-Up', the loss function induced penalty for a predicted label 'Impervious Surface' should be less than if the prediction were 'Water'. The underlying assumption in the construction of such a customized loss-function is that certain land-cover classes have a higher probability of being alike, or in 'proximity' to each other, than other land-cover classes. Further, given the challenge of integrating multi-scale image data, it would be useful to investigate the effect of different image scales and image resolutions on the design and configuration of network parameters. With regards to the implementation of the mean-field approximation to the dense-CRF inference, further research can explore incorporating higher-order neighbourhoods and advanced pairwise energy formulations that integrate relationships between land-cover classes, their spectral sub-classes, and edges or zones of sharp spectral transition (Moser et al., 2013). Also, as it stands now, the dense-CRF implementation of Krahenbuhl et al. accepts only a three-band image input. This is because the algorithm was crafted for the image segmentation of RGB-images. In order to adapt it for use on multispectral remote-sensing images, the dense-CRF inference can be appropriately altered to accommodate a larger number of image bands. This could potentially increase its effectiveness for the land-cover classification of VHR remote-sensing images. Lastly, the research done by Zheng et al., of incorporating the implementation of the mean-field approximation of the dense-CRF inference as a trainable layer within the FCN structure, is promising and worth exploring (Zheng et al., 2015).

LIST OF REFERENCES

- Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet : A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99), 1–1.
- Carleer, A. P., Debeir, O., & Wolff, E. (2005). Assessment of very high spatial resolution satellite image segmentations. *Photogrammetric Engineering and Remote Sensing*, 71(11), 1285–1294. <https://doi.org/10.1117/12.511027>
- Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2015). Semantic Image Segmentation with Deep Convolutional Nets , Atrous Convolution , and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–14.
- Chen, X., Xiang, S., Liu, C., & Pan, C. (2014). Vehicle Detection in Satellite Images by Hybrid Deep Convolutional Neural Networks. *IEEE Geoscience and Remote Sensing Letters*, 11(10), 1797–1801. <https://doi.org/10.1109/LGRS.2014.2309695>
- Cheng, G., Zhou, P., & Han, J. (2016). Learning Rotation-Invariant Convolutional Neural Networks for Object Detection in VHR Optical Remote Sensing Images, 54(12), 1–11. <https://doi.org/10.1109/TGRS.2016.2601622>
- Fu, G., Liu, C., Zhou, R., Sun, T., & Zhang, Q. (2017). Classification for High Resolution Remote Sensing Imagery Using a Fully Convolutional Network, 1–21. <https://doi.org/10.3390/rs9050498>
- Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh & M. Titterton (Eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Vol. 9, pp. 249–256). Sardinia, Italy: PMLR. Retrieved from <http://proceedings.mlr.press/v9/glorot10a.html>
- Hughes, G. (1968). On the Mean Accuracy of Statistical Pattern Recognizers. *IEEE Transactions on Information Theory*, 14(1), 55–63. <https://doi.org/10.1109/TVT.1968.1054102>
- Kenduiwo, B. K., Tolpekin, V. A., & Stein, A. (2014). Detection of built-up area in optical and synthetic aperture radar images using conditional random fields. *Journal of Applied Remote Sensing*, 8. <https://doi.org/10.1117/1.JRS.8.083672>
- Krähenbühl, P., & Koltun, V. (2011). Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. *Advances in Neural Information Processing Systems*, 24, 109–117.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25 (NIPS2012)*, 1–9. <https://doi.org/10.1109/5.726791>
- Ladicky, L., Russell, C., Kohli, P., & Torr, P. H. S. (2009). Associative Hierarchical CRFs for Object Class Image Segmentation. In *IEEE 12th International Conference on Computer Vision*.
- Långkvist, M., Kiselev, A., Alirezaie, M., & Loutfi, A. (2016). Classification and Segmentation of Satellite Orthoimagery Using Convolutional Neural Networks. <https://doi.org/10.3390/rs8040329>
- LeCun, Y., Jackel, L., Boser, B., Denker, J., Graf, H., Guyon, I., ... Hubbard, W. (1989). Handwritten Digit Recognition: Applications. *IEEE Communications Magazine*, 27(November), 41–46.
- Li, S. Z. (2009). *Markov Random Field Modeling in Image Analysis*. (S. Singh, Ed.) (3rd Editio). London: Springer.
- Long, J., Shelhamer, E., & Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation, 3431–3440.
- Maggiori, E., Tarabalka, Y., Charpiat, G., & Alliez, P. (2017). Convolutional Neural Networks for Large-Scale Remote-Sensing Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2), 645–657. <https://doi.org/10.1109/TGRS.2016.2612821>

- Moser, G., & Serpico, S. B. (2009). Edge-Preserving Classification of High-Resolution Remote-Sensing Images by Markovian Data Fusion. In *IEEE International Geoscience and Remote Sensing Symposium* (pp. 765–768). Cape Town, SOUTH AFRICA.
- Moser, G., Serpico, S. B., & Benediktsson, J. A. (2013). Land-Cover Mapping by Markov Modeling of Spatial – Contextual Information in Very-High-Resolution Remote Sensing Images. *Proceedings of the IEEE*, *101*(3), 631–651. <https://doi.org/10.1109/JPROC.2012.2211551>
- Nogueira, K., Penatti, O. A. B., & Jefersson, A. (2017). Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*, *61*(SI), 539–556. <https://doi.org/10.1016/j.patcog.2016.07.001>
- Paisitkriangkrai, S., Sherrah, J., Janney, P., & Hengel, A. Van Den. (2016). Semantic Labeling of Aerial and Satellite Imagery, *9*(7), 2868–2881.
- Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the Difficulty of Training Recurrent Neural Networks. In S. Dasgupta & D. McAllester (Eds.), *ICML'13 Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28* (p. Pages III-1310-III-1318). Atlanta, GA, USA: JMLR.org ©2013.
- Persello, C., & Stein, A. (2017). Deep Fully Convolutional Networks for the Detection of Informal Settlements in VHR Images. *IEEE Geoscience and Remote Sensing Letters*, *14*(12), 2325–2329. <https://doi.org/10.1109/LGRS.2017.2763738>
- Romero, A., Gatta, C., & Camps-Valls, G. (2015). Unsupervised Deep Feature Extraction for Remote Sensing Image Classification. *IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING*, *54*(3), 1–14. <https://doi.org/10.1109/TGRS.2015.2478379>.
- Scott, G. J., England, M. R., Starms, W. A., Member, S., Marcum, R. A., & Davis, C. H. (2017). Training Deep Convolutional Neural Networks for Land – Cover Classification of High-Resolution Imagery. *IEEE Geoscience and Remote Sensing Letters*, *14*(4), 549–553.
- Shotton, J., Winn, J., & Rother, C. (2006). TextonBoost for Image Understanding : Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture , Layout , and Context. In A. Leonardis, H. Bischof, & A. Pinz (Eds.), *9th European Conference on Computer Vision* (pp. 1–15). Graz, Austria. <https://doi.org/10.1007/s11263-007-0109-1>
- Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition, 1–14. <https://doi.org/10.1016/j.infsof.2008.09.005>
- Szegedy, C., Liu, W., Jia, Y., & Sermanet, P. (2014). Going deeper with convolutions. *arXiv Preprint arXiv: 1409.4842*, 1–9. <https://doi.org/10.1109/CVPR.2015.7298594>
- Tso, B., & Mather, P. M. (2009). *Classification Methods for Remotely Sensed Data* (2nd ed.). Boca Raton, FL, USA: CRC Press.
- Vedaldi, A., & Lenc, K. (2014). MatConvNet Convolutional Neural Networks for MATLAB. *Eprint arXiv:1412.4564*.
- Volpi, M., & Tuia, D. (2017). Dense Semantic Labeling of Subdecimeter Resolution Images With Convolutional Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing*, *55*(2), 881–893. <https://doi.org/10.1109/TGRS.2016.2616585>
- Zeiler, M. D., & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks arXiv:1311.2901v3 [cs.CV] 28 Nov 2013. *Computer Vision–ECCV 2014*, *8689*, 818–833. https://doi.org/10.1007/978-3-319-10590-1_53
- Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., ... Torr, P. H. S. (2015). Conditional Random Fields as Recurrent Neural Networks. In *2015 IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV)* (pp. 1529–1537). Santiago, CHILE. <https://doi.org/10.1109/ICCV.2015.179>

APPENDIX

Appendix A: SimpleNet Experimental Results - Tile 45 and Tile 105

Hyper-Parameter Modified	Value(s)	OA_45	OA_105	Training Time (sec)
Baseline	NA	93.84	88.14	11432
Patch Size	65x65	93.77	87.97	22049
Patch Size	85x85	94.41	89.27	36728
Patch Size	125x125	93.52	88.33	81527
Layer Depth (Removing)	I-C1-B1-A-C2-B2-A-C3-B3-A-MP3-D1-O	91.67	85.88	7312
Layer Depth (Adding)	I-C1-B1-A-C2-B2-A-C3-B3-A-C4-B4-A-C5-B5-A-MP5-D1-O	94.23	89.30	13674
Layers (Adding MP3)	I-C1-B1-A-C2-B2-A-C3-B3-A-MP3-C4-B4-A-MP4-D1-O	94.10	89.83	11710
Layers (Adding MP2 & MP3)	I-C1-B1-A-C2-B2-A-MP2-C3-B3-A-MP3-C4-B4-A-MP4-D1-O	94.39	90.04	12028
Layers (Adding MP1, MP2 & MP3)	I-C1-B1-A-MP1-C2-B2-A-MP2-C3-B3-A-MP3-C4-B4-A-MP4-D1-O	94.33	90.13	12178
Pooling Size	3,3,3,3	94.33	90.13	12178
Pooling Size	5,5,5,5	94.86	90.94	12808
Kernel Depth	32, 32, 32, 32	93.38	87.67	9974
Kernel Depth	64, 64, 64, 64	93.41	88.19	16735
Kernel Size (Dilation)	3(1), 3(1), 3(1), 3(1)	91.09	85.30	9687
Kernel Size (Dilation)	3(1), 3(1), 3(2), 3(2)	92.04	85.54	9594
Kernel Size (Dilation)	3(1), 3(1), 5(2), 5(2)	93.07	87.44	13677
Kernel Size (Dilation)	3(1), 5(1), 3(2), 5(2)	92.67	85.74	12227
Kernel Size (Dilation)	3(1), 5(1), 5(2), 5(3)	93.79	88.11	12217
Kernel Size (Dilation)	3(1), 3(2), 3(2), 5(2)	92.82	87.11	9413
Kernel Size (Dilation)	3(1), 3(2), 5(2), 5(2)	93.78	87.55	11307

Kernel Size (Dilation)	5(1), 3(2), 5(2), 5(3)	93.63	87.69	11740
Learning Rate	0.001, 0.0001	92.62	86.76	12221
Learning Rate	0.0001, 0.00001	92.96	87.51	12177
Learning Rate	0.000001, 0.0000001	93.64	87.89	12217
Weight Decay	0.0001	93.58	87.43	12159
Weight Decay	0.001	93.66	87.78	12170
Number of Epochs	200, 200	93.93	88.33	27808

Appendix B: FuseNet Experimental Results - Tile 45 and Tile 105

Downsampling Layer Experiments								
Patch Size [128,32]; LR(Epochs): 10^{-4} (80), 10^{-5} (40)								
DS [D,P,S]	FEMS [D,P,S] & CNCT	1 st Cnv [D,P,S]	2 nd Cnv [D,P,S]	3 rd Cnv [D,P,S]	US	OA Tile 45	OA Tile 105	Train Time (sec)
3x3 (16) [1,1,2]	3x3 (32) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	4x4 (6)	93.33	86.51	7465
3x3 (32) [1,1,2]	[32+32 = 64 feat.]							
3x3 (16) [1,1,1]	3x3 (32) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	4x4 (6)	92.85	86.77	9131
9x9 (32) [1,4,4]	[32+32 = 64 feat.]							
9x9 (32) [1,4,4]	3x3 (32) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	4x4 (6)	91.90	86.40	7204
NA	[32+32 = 64 feat.]							

FEMS Layer Experiments								
Patch Size [128,32]; LR(Epochs): 10^{-4} (80), 10^{-5} (40)								
DS [D,P,S]	FEMS [D,P,S] & CNCT	1 st Cnv [D,P,S]	2 nd Cnv [D,P,S]	3 rd Cnv [D,P,S]	US	OA Tile 45	OA Tile 105	Train Time (sec)
3x3 (16) [1,1,2]	NA/ Raw Band Input	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	4x4 (6)	92.96	87.00	7126
3x3 (32) [1,1,2]	[32+4 = 36 feat.]							

3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	4x4 (6)	93.52	87.44	7495
3x3 (32) [1,1,2]	[32+16 = 48 feat.]							
3x3 (16) [1,1,2]	3x3 (32) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	4x4 (6)	93.33	86.51	7465
3x3 (32) [1,1,2]	[32+32 = 64 feat.]							

Bottleneck Experiments (with size of smallest feature = 32)

Patch Size [128,32]; LR(Epochs): 10^{-4} (80), 10^{-5} (40)

DS [D,P,S]	FEMS [D,P,S] CNCT	BN			OA Tile 45	OA Tile 105	Train Time (sec)	
		1 st Cnv [D,P,S]	2 nd Cnv [D,P,S]	3 rd Cnv [D,P,S]				
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	4x4 (6)	92.46	86.06	6879
3x3 (32) [1,1,2]	[32+16 = 48 feat.]							
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	4x4 (6)	93.40	86.49	6858
3x3 (32) [1,1,2]	[32+16 = 48 feat.]							
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,1]	5x5 (64) [1,2,1]	3x3 (64) [2,2,1]	4x4 (6)	93.58	87.65	7563
3x3 (32) [1,1,2]	[32+16 = 48 feat.]							
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	3x3 (64) [2,2,1]	4x4 (6)	93.56	87.34	6900
3x3 (32) [1,1,2]	[32+16 = 48 feat.]							
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	4x4 (6)	93.52	87.44	7495
3x3 (32) [1,1,2]	[32+16 = 48 feat.]							
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	5x5 (64) [1,2,1]	5x5 (64) [1,2,1]	3x3 (64) [2,2,1]	4x4 (6)	92.98	86.36	7989
3x3 (32) [1,1,2]	[32+16 = 48 feat.]							

Bottleneck Experiments (with size of smallest feature = 16)

Patch Size [128,32]; LR(Epochs): 10^{-4} (80), 10^{-5} (40)

DS [D,P,S]	FEMS [D,P,S]	DS	BN	US	OA Tile 45	OA Tile 105	Train Time
---------------	-----------------	----	----	----	---------------	----------------	---------------

	CNCT	1 st /2 nd C _{nv}	2 nd /3 rd C _{nv}	3 rd /4 th C _{nv}				(sec)
		[D,P,S]	[D,P,S]	[D,P,S]				
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	2x2 (6)	92.72	86.04	6252
3x3 (32) [1,1,2]					4x4 (6)			
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	5x5 (64) [1,2,1]	2x2 (6)	92.52	85.56	6237
3x3 (32) [1,1,2]					4x4 (6)			
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	2x2 (6)	92.97	86.59	6384
3x3 (32) [1,1,2]		3x3 (64) [1,1,1]			4x4 (6)			
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,2]	5x5 (64) [1,2,1]	3x3 (64) [2,2,1]	2x2 (6)	93.09	85.83	6757
3x3 (32) [1,1,2]		3x3 (64) [1,1,1]			4x4 (6)			
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,2]	3x3 (64) [2,2,1]	3x3 (64) [2,2,1]	2x2 (6)	92.27	86.18	6386
3x3 (32) [1,1,2]		3x3 (64) [1,1,1]			4x4 (6)			
Bottleneck Experiments (with size of smallest feature = 8)								
Patch Size [128,32]; LR(Epochs): 10^{-4} (80), 10^{-5} (40)								
DS	FEMS	BN			US	OA	OA Tile	Train
[D,P,S]	[D,P,S]	1 st C _{nv}	2 nd C _{nv}	3 rd C _{nv}		Tile 45	105	Time [TT]
	CNCT	[D,P,S]	[D,P,S]	[D,P,S]				
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	4x4 (6)	92.10	84.29	5977
3x3 (32) [1,1,2]		3x3 (64) [1,1,2]			4x4 (6)			
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	5x5 (64) [1,2,1]	4x4 (6)	90.68	82.85	6150
3x3 (32) [1,1,2]		3x3 (64) [1,1,2]			4x4 (6)			
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,2]	3x3 (64) [1,1,1]	3x3 (64) [1,1,1]	4x4 (64)	92.15	84.69	6193
3x3 (32) [1,1,2]		3x3 (64) [1,1,2]			4x4 (6)			
Upsampling Experiments								
Patch Size [128,32]; LR(Epochs): 10^{-4} (80), 10^{-5} (40)								
DS	CNCT	BN			US	OA	OA Tile	Train
[D,P,S]						Tile 45	105	Time

		1 st Cnv	2 nd Cnv	3 rd Cnv	[TT]			
		[D,P,S]	[D,P,S]	[D,P,S]				
3x3 (16)	3x3 (16)	3x3 (64)	3x3 (64)	3x3 (64)	4x4	92.15	84.69	6193
[1,1,2]	[1,1,1]	[1,1,2]	[1,1,1]	[1,1,1]	(64)			
3x3 (32)		3x3 (64)			4x4			
[1,1,2]		[1,1,2]			(6)			
3x3 (16)	3x3 (16)	3x3 (64)	3x3 (64)	3x3 (64)	4x4	92.10	84.29	5977
[1,1,2]	[1,1,1]	[1,1,2]	[1,1,1]	[1,1,1]	(6)			
3x3 (32)	[32+16 =	3x3 (64)			4x4			
[1,1,2]	48 feat.]	[1,1,2]			(6)			
3x3 (16)	3x3 (16)	3x3 (64)	3x3 (64)	3x3 (64)	2x2	91.07	84.58	6770
[1,1,2]	[1,1,1]	[1,1,2]	[1,1,1]	[1,1,1]	(6)			
					2x2			
					(6)			
3x3 (32)	[32+16 =	3x3 (64)			2x2			
[1,1,2]	48 feat.]	[1,1,2]			(6)			
					2x2			
					(6)			
3x3 (16)	3x3 (16)	3x3 (64)	3x3 (64)	3x3 (64)	2x2	91.47	85.13	6523
[1,1,2]	[1,1,1]	[1,1,2]	[1,1,1]	[1,1,1]	(32)			
					2x2			
					(32)			
3x3 (32)	[32+16 =	3x3 (64)			2x2			
[1,1,2]	48 feat.]	[1,1,2]			(6)			
					2x2			
					(6)			

Experiments with Additional Loss

Patch Size [128,32]; LR(Epochs): 10^{-4} (80), 10^{-5} (40)

DS	CNCT	BN			US	OA	OA	Train
[D,P,S]		1 st Cnv	2 nd Cnv	3 rd Cnv		Tile 45	Tile 105	Time (sec)
		[D,P,S]	[D,P,S]	[D,P,S]				
3x3 (16)	NA/ Raw	3x3 (64)	3x3 (64)	5x5 (64)	Loss	92.81	87.59	7358
[1,1,2]	Band Input	[1,1,2]	[1,1,2]	[2,4,1]	+			[loss2 wt. = 0.7]
3x3 (32)					4x4			
[1,1,2]	[32+4 =				(6)			
	36 feat.]							
3x3 (16)	NA/ Raw	3x3 (64)	3x3 (64)	5x5 (64)	Loss	92.29	86.59	7368
[1,1,2]	Band Input	[1,1,2]	[1,1,2]	[2,4,1]	+			[loss2 wt. = 0.5]
3x3 (32)					4x4			
[1,1,2]	[32+4 =				(6)			
	36 feat.]							
3x3 (16)	NA/ Raw	3x3 (64)	3x3 (64)	5x5 (64)	4x4	92.96	87.00	7126
[1,1,2]	Band Input	[1,1,1]	[2,2,1]	[2,4,1]	(6)			
3x3 (32)	[32+4 =							
[1,1,2]	36 feat.]							

3x3 (16) [1,1,2]	3x3 (32) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	Loss +	93.48	87.07	8123 [loss2 wt. = 0.6]
					4x4 (6)			
3x3 (32) [1,1,2]	[32+32 = 64 feat.]							
3x3 (16) [1,1,2]	3x3 (32) [1,1,1]	3x3 (64) [1,1,1]	3x3 (64) [2,2,1]	5x5 (64) [2,4,1]	4x4 (6)	93.33	86.51	7465
3x3 (32) [1,1,2]	[32+32 = 64 feat.]							

Experiments with Learning Rate;

Patch Size [128,32]; LR(Epochs): 10^{-4} (80), 10^{-5} (40)								
DS [D,P,S]	CNCT	BN			US	OA Tile 45	OA Tile 105	Train Time (sec)
		1 st Cnv [D,P,S]	2 nd Cnv [D,P,S]	3 rd Cnv [D,P,S]				
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,1]	5x5 (64) [1,2,1]	3x3 (64) [2,2,1]	4x4 (6)	93.58	87.65	7563
3x3 (32) [1,1,2]	[32+16 = 48 feat.]							

LR(Epochs): 10^{-5} (80), 10^{-6} (40)

3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,1]	5x5 (64) [1,2,1]	3x3 (64) [2,2,1]	4x4 (6)	92.44	86.00	7345
3x3 (32) [1,1,2]	[32+16 = 48 feat.]							

LR(Epochs): 10^{-3} (80), 10^{-4} (40)

3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,1]	5x5 (64) [1,2,1]	3x3 (64) [2,2,1]	4x4 (6)	93.07	86.58	7349
3x3 (32) [1,1,2]	[32+16 = 48 feat.]							

Experiments with Weight Decay;

Patch Size [128,32]; LR(Epochs): 10^{-4} (80), 10^{-5} (40) WD = 0.0001								
DS [D,P,S]	CNCT	BN			US	OA Tile 45	OA Tile 105	Train Time (sec)
		1 st Cnv [D,P,S]	2 nd Cnv [D,P,S]	3 rd Cnv [D,P,S]				
3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,1]	5x5 (64) [1,2,1]	3x3 (64) [2,2,1]	4x4 (6)	93.46	86.72	8045
3x3 (32) [1,1,2]	[32+16 = 48 feat.]							

WD = 0.001

3x3 (16) [1,1,2]	3x3 (16) [1,1,1]	3x3 (64) [1,1,1]	5x5 (64) [1,2,1]	3x3 (64) [2,2,1]	4x4 (6)	92.95	86.97	8266
3x3 (32) [1,1,2]	[32+16 = 48 feat.]							

Appendix C: Dense-CRF Experiments (2nd Batch) – Tile 45 and Tile 105

θ_α	θ_β	OA – Tile 45	OA – Tile 105
3	4	94.60	91.45
3	6	94.62	91.45
3	10	94.70	91.03
3	15	94.51	90.41
5	4	94.83	91.50
5	6	94.75	91.22
5	10	94.52	90.11
5	15	93.78	88.37
7	4	94.72	91.27
7	6	94.53	90.86
7	10	93.98	89.40
7	15	92.95	86.21
4	3	94.77	91.58
4	4	94.84	91.51
4	5	94.82	91.49
4	6	94.76	91.27
4	7	94.81	91.17
5	3	94.90	91.48
5	4	94.83	91.50
5	5	94.75	91.37
5	6	94.75	91.22
5	7	94.70	91.01
6	3	94.80	91.56
6	4	94.79	91.45
6	5	94.73	91.23
6	6	94.68	91.02
6	7	94.60	90.67
7	3	94.77	91.50
7	4	94.71	91.27
7	5	94.65	91.04
7	6	94.53	90.86
7	7	94.50	90.60

θ_γ	OA – Tile 45	OA – Tile 105
1	94.70	91.03
3	94.70	91.03
5	94.70	91.03
7	94.71	91.02
9	94.72	91.01

$w^{(2)}$	θ_γ	OA – Tile 45	OA – Tile 105
1	1	94.69	91.02
1	3	94.68	90.96
1	5	94.40	90.54
5	1	94.65	91.06
5	3	94.41	90.72
5	5	93.62	88.90
20	1	94.66	91.15
20	3	93.94	89.83

20

5

87.49

82.89

$w^{(4)}$	OA – Tile 45	OA – Tile 105
1	94.42	90.70
10	94.63	90.95
20	94.67	90.96
50	94.71	91.03
80	94.69	91.03
100	94.70	91.02
150	94.66	91.02
200	94.67	91.02
500	94.65	91.07
1000	94.66	91.06

Appendix D: ReUseNet Experiments – Tile 45 and Tile 105

Number of Loss Function in ReUseNet	Weight of Objective 1	TT (sec)	OA- Tile 45	OA – Tile 105
Single Loss	1	24534	92.66	86.29
Double Loss	0.8	28711	95.06	88.29
Double Loss	0.7	28920	95.01	88.57
Double Loss	0.5	-	94.68	88.18
Double Loss	0.3	28831	93.90	87.53

Control Experiment	TT (sec)	OA45	OA105
FuseNet_Scores+FuseNet2	15816	94.23	89.79

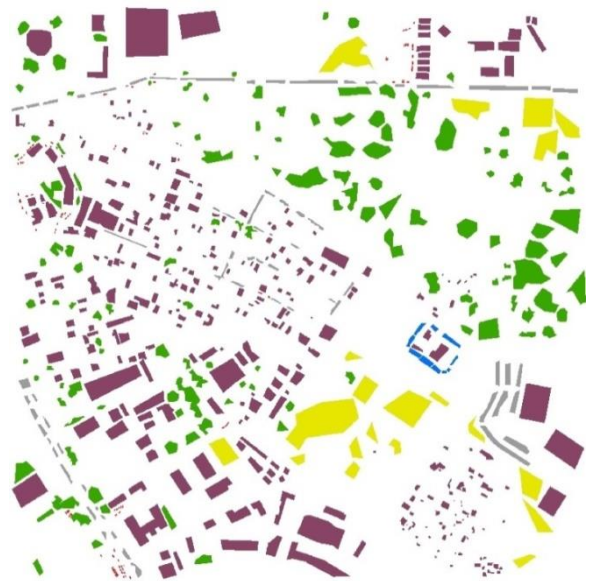
Appendix E: Ground Reference Pixels Annotated per Class per Tile

	Tile 32	Tile 45	Tile 78	Tile 82	Tile 100	Tile 103	Tile 105
Built-Up	694810	1186802	1153967	890312	1499922	857494	844186
Low Veg	881970	128933	197838	316407	54792	272643	484311
Tree	601717	374808	266812	289433	214261	479736	449219
Cars	15711	25331	28502	35557	37291	8075	31167
Water	13671	292	10516	8092	22436	13630	4349
Imp. Surface	347165	347805	319701	422154	350066	112563	358949
Total	2555044	2063971	1977336	1961955	2178768	1744141	2172181

Appendix F: Classification Maps for Tile 103 (Result I)



Tile 103 – SimpleNet
OA: 93.35







Tile 103 – Ground Reference



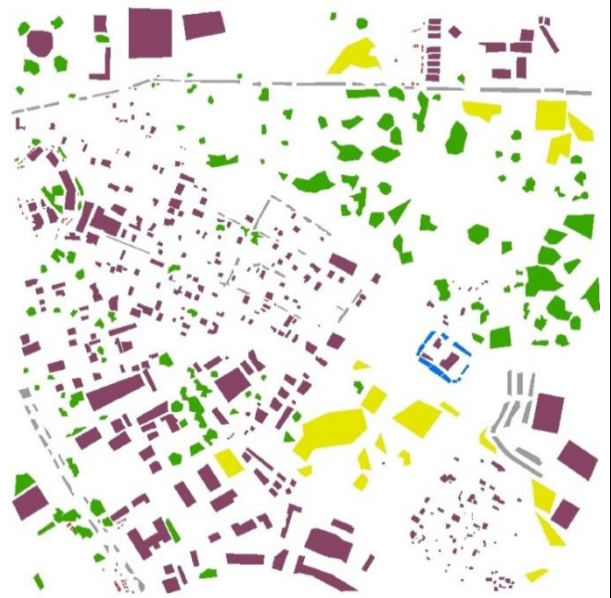
Tile 103 – FuseNet
OA: 92.88

Legend

-  Built-Up
-  LowVegetation
-  Tree
-  Cars
-  Water
-  Impervious Surface



Tile 103 – SimpleNet+DenseCRF
OA: 94.53





Tile 103 – Ground Reference



Tile 103 – ReUseNet
OA: 93.61

Legend

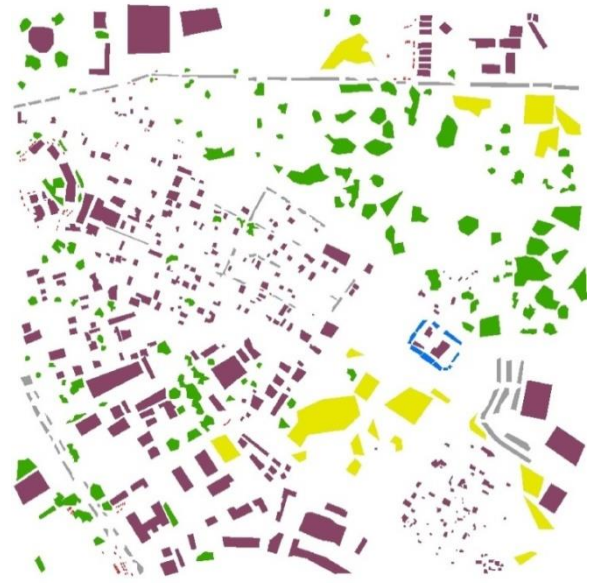
-  Built-Up
-  LowVegetation
-  Tree
-  Cars
-  Water
-  Impervious Surface

Appendix G: Classification Maps for Tile 103 (Result II)



Tile 103 – SimpleNet

OA: 85.62




Tile 103 – Ground Reference



Tile 103 – FuseNet

OA: 93.24

Legend

-  Built-Up
-  LowVegetation
-  Tree
-  Cars
-  Water
-  Impervious Surface



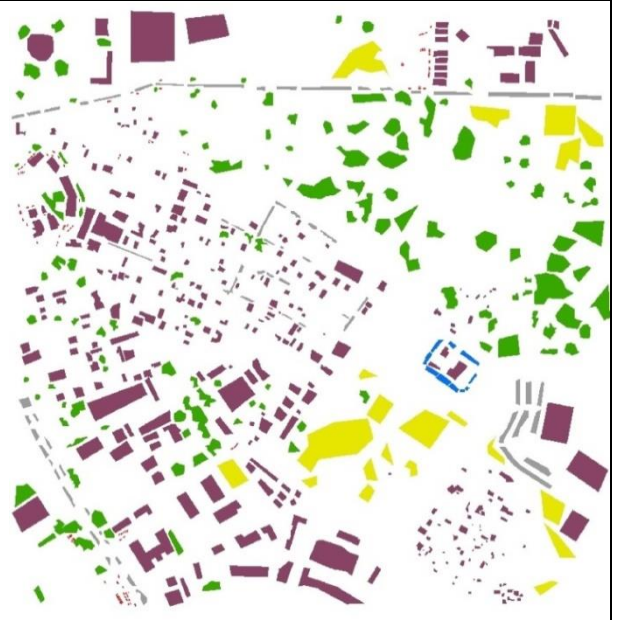
Tile 103 – SimpleNet+DenseCRF

OA: 86.10




Tile 103 – ReUseNet

OA: 93.29



Tile 103 – Ground Reference

Legend

-  Built-Up
-  LowVegetation
-  Tree
-  Cars
-  Water
-  Impervious Surface

Appendix H: Confusion Matrices (Results I)

SimpleNet Classification - Tile 32							
	Built-Up	Low_Veg	Tree	Cars	Water	Imp. Surf	UA
Built-Up	681683	1695	302	4440	5101	31840	94.02
Low_Veg	4818	878183	1912	7	0	393	99.20
Tree	1473	346	599407	253	0	377	99.59
Cars	119	0	0	9902	0	395	95.07
Water	1446	3	64	1	8570	19	84.83
Imp. Surf	5271	1743	32	1108	0	314177	97.47
PA	98.11	99.57	99.62	63.03	62.69	90.50	

FuseNet Classification – Tile 32							
	Built-Up	Low_Veg	Tree	Cars	Water	Imp. Surf	UA
Built-Up	682277	606	285	3919	9105	37314	93.02
Low_Veg	4462	877948	2598	1	0	725	99.12
Tree	1308	230	598786	85	61	1236	99.52
Cars	409	0	0	10065	346	1342	82.76
Water	1759	0	3	0	4159	134	68.69
Imp. Surf	4595	3186	45	1641	0	306414	97.00
PA	98.20	99.54	99.51	64.06	30.42	88.26	

SimpleNet + DCRF - Tile 32							
	Built-Up	Low_Veg	Tree	Cars	Water	Imp. Surf	UA
Built-Up	684588	1094	48	4349	1199	23020	95.84
Low_Veg	4797	880287	885	10	0	447	99.31
Tree	985	540	600733	283	0	538	99.61
Cars	40	0	0	8125	0	143	97.80
Water	1471	0	5	0	12472	81	88.90
Imp. Surf	2929	49	46	2944	0	322936	98.19
PA	98.53	99.81	99.84	51.72	91.23	93.02	

ReUseNet – Tile 32							
	Built-Up	Low_Veg	Tree	Cars	Water	Imp. Surf	UA
Built-Up	679477	34	632	2898	11213	22711	94.77
Low_Veg	8126	881359	6860	28	387	3387	97.91
Tree	1354	142	593689	82	544	1602	99.38
Cars	755	0	154	11087	286	1235	82.02
Water	0	0	0	0	0	0	-
Imp. Surf	5098	435	382	1616	1241	318230	97.32
PA	97.79	99.93	98.67	70.57	0.00	91.67	

SimpleNet Classification - Tile 103							
	Built-Up	Low_Veg	Tree	Cars	Water	Imp. Surf	UA
Built-Up	824305	23993	104	2243	5167	10238	95.18
Low_Veg	991	210947	1386	32	0	224	98.77
Tree	2092	5895	478105	182	0	464	98.23
Cars	1912	54	13	5308	0	655	66.84
Water	896	0	124	4	8463	25	88.97
Imp. Surf	27298	31754	4	306	0	100957	62.97
PA	96.13	77.37	99.66	65.73	62.09	89.69	

FuseNet – Tile 103							
	Built-Up	Low_Veg	Tree	Cars	Water	Imp. Surf	UA
Built-Up	799314	7056	308	1983	6597	4943	97.45
Low_Veg	1256	224058	1214	49	0	46	98.87
Tree	2932	10412	478159	264	55	374	97.15
Cars	2220	589	31	5329	0	1083	57.60

Water	129	0	0	0	6978	1	98.17
Imp. Surf	51643	30528	24	450	0	106116	56.22
PA	93.22	82.18	99.67	65.99	51.20	94.27	

SimepleNet+DCRF - Tile 103

	Built-Up	Low_Veg	Tree	Cars	Water	Imp. Surf	UA
Built-Up	839906	22713	129	2382	6093	5276	95.83
Low_Veg	429	211423	929	18	0	37	99.34
Tree	1911	5140	478556	176	0	481	98.42
Cars	1054	23	7	4786	0	263	78.04
Water	20	0	72	3	7537	10	98.63
Imp. Surf	14174	33344	43	710	0	106496	68.81
PA	97.95	77.55	99.75	59.27	55.30	94.61	

ReUseNet – Tile 103

	Built-Up	Low_Veg	Tree	Cars	Water	Imp. Surf	UA
Built-Up	803927	4542	273	1966	7525	1292	98.10
Low_Veg	1709	235310	1275	44	1827	204	97.90
Tree	2931	5795	478082	404	749	329	97.91
Cars	3953	495	15	5312	1491	698	44.40
Water	0	0	0	0	0	0	-
Imp. Surf	44974	26501	91	349	2038	110040	59.81
PA	93.75	86.31	99.66	65.78	0.00	97.76	

Appendix I: Confusion Matrices (Results II)

SimpleNet Classification - Tile 32

	Built-Up	Low_Veg	Tree	Cars	Water	Surf	UA
Built-Up	661186	9806	24	4649	692	194005	75.97
Low_Veg	6370	860039	29	374	0	11800	97.89
Tree	7172	11473	601624	57	0	1782	96.71
Cars	415	0	1	9253	0	271	93.09
Water	5854	18	39	752	12979	8754	45.71
Imp. Surf	13813	634	0	626	0	130553	89.65
PA	95.16	97.51	99.98	58.90	94.94	37.61	

FuseNet Classification – Tile 32

	Built-Up	Low_Veg	Tree	Cars	Water	Surf	UA
Built-Up	683210	10772	513	5042	13562	51398	89.37
Low_Veg	2303	869057	807	21	3	947	99.53
Tree	1065	685	600388	101	65	1459	99.44
Cars	266	0	0	9269	2	9713	90.43
Water	0	0	0	0	0	0	-
Imp. Surf	7966	1456	9	1278	39	292648	96.46
PA	98.33	98.54	99.78	59.00	0	84.30	

SimpleNet + DCRF - Tile 32

	Built-Up	Low_Veg	Tree	Cars	Water	Surf	UA
Built-Up	671785	5449	29	4935	0	212374	75.10
Low_Veg	5362	869668	235	243	0	7273	98.52
Tree	2240	5789	601349	50	0	788	98.55
Cars	166	0	0	7820	0	99	96.72
Water	1889	0	104	599	13671	893	79.69
Imp. Surf	13368	1064	0	2064	0	125738	88.40
PA	96.69	98.61	99.94	49.77	100.00	36.22	

ReUseNet – Tile 32

	Built-Up	Low_Veg	Tree	Cars	Water	Surf	UA
Built-Up	681373	6159	383	2632	1083	42038	92.87
Low_Veg	2740	874546	5410	16	0	343	99.04
Tree	1378	394	595637	87	0	1145	99.50
Cars	462	0	0	10837	0	1980	81.61
Water	1166	17	241	0	12588	113	89.12
Imp. Surf	7691	854	46	2139	0	301546	96.56
PA	98.07	99.16	98.99	68.98	92.08	86.86	

SimpleNet Classification - Tile 103

	Built-Up	Low_Veg	Tree	Cars	Water	Surf	UA
Built-Up	764327	35188	5	2095	635	14777	93.55
Low_Veg	1315	136143	13	186	8	196	98.75
Tree	8574	76479	479697	323	6	492	84.82
Cars	887	0	11	5031	0	223	81.78
Water	4500	0	0	39	12981	1739	67.40
Imp. Surf	77891	24833	10	401	0	95136	47.98
PA	89.14	49.93	99.99	62.30	95.24	84.52	

FuseNet – Tile 103

	Built-Up	Low_Veg	Tree	Cars	Water	Surf	UA
Built-Up	810290	33251	187	2218	10438	6585	93.90
Low_Veg	622	228193	1175	87	21	116	99.12
Tree	2206	5619	478346	384	1107	460	98.00
Cars	2325	159	21	4843	1958	824	47.81
Water	0	0	0	0	0	0	-
Imp. Surf	42051	5421	7	543	106	104578	68.48
PA	94.50	83.70	99.71	59.98	0	92.91	

SimepleNet+DCRF - Tile 103

	Built-Up	Low_Veg	Tree	Cars	Water	Surf	UA
Built-Up	768565	32136	9	2675	352	12516	94.16
Low_Veg	39	137836	10	199	0	51	99.78
Tree	6592	76072	479655	328	0	438	85.18
Cars	350	0	4	4076	0	111	89.76
Water	4411	0	39	30	13278	1132	70.29
Imp. Surf	77537	26599	19	767	0	98315	48.37
PA	89.63	50.56	99.98	50.48	97.42	87.34	

ReUseNet – Tile 103

	Built-Up	Low_Veg	Tree	Cars	Water	Surf	UA
Built-Up	784326	20407	32	1915	738	2263	96.87
Low_Veg	1584	239411	2536	13	0	187	98.23
Tree	3213	3336	476837	398	113	489	98.44
Cars	3422	306	113	5056	0	828	51.99
Water	662	0	212	1	12779	106	92.87
Imp. Surf	64287	9183	6	692	0	108690	59.44
PA	91.47	87.81	99.40	62.61	93.76	96.56	