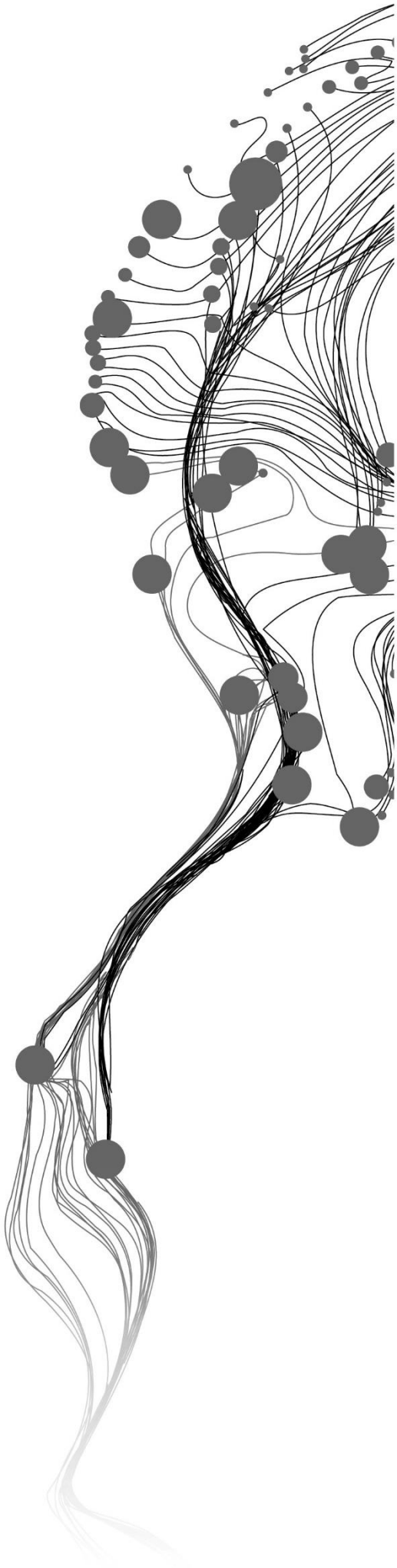
AUTOMATIC DELINEATION OF SMALL HOLDER AGRICULTURAL FIELD BOUNDARIES USING FULLY CONVOLUTIONAL NETWORKS

GIDEON MUTUA MUSYOKA
March, 2018

SUPERVISORS:
Dr. V.A. Tolpekin
Dr. C. Persello



AUTOMATIC DELINEATION OF SMALL HOLDER AGRICULTURAL FIELD BOUNDARIES USING FULLY CONVOLUTIONAL NETWORKS

GIDEON MUTUA MUSYOKA
Enschede, The Netherlands, March, 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

SUPERVISORS:
Dr. V.A. Tolpekin
Dr. C. Persello

THESIS ASSESSMENT BOARD:
Prof. Dr. Ir. A. Stein
Dr. M.N. Koeva (External Examiner, University of Twente, ITC-PGM)

ABSTRACT

Accurate information on agricultural field boundaries is important for precision agriculture and can serve as a basis for establishing cadastral information in countries which do not yet have. Though previous works have shown promising results, automatic extraction of agricultural field boundaries remains a nontrivial task, especially in the case of smallholder farms in Africa. The field boundaries are often irregularly shaped and have a poor spectral contrast between internal and external parts of the fields. The internal parts of most smallholder farms are heterogeneous with mixed crops and trees. In this research, we investigate a deep feature learning approach based on Fully Convolutional Networks(FCNs) for the detection of agricultural field boundaries in Kofa region, located in Kano state in the northern parts of Nigeria. FCNs are powerful visual models that learn a hierarchy of spatial-contextual features and have proven to be very successful in characterising complex patterns. To this aim, we optimised an FCN architecture and trained it to detect visible boundaries from VHR satellite imagery. The obtained results were compared against state-of-the-art methods which included the globalized probability of a boundary (gPb) detection, multi-resolution segmentation in eCognition and Canny detector. Experimental results show that the proposed method outperforms the other contenders in all the performance metrics considered apart from computational time and complexity. We conclude that our FCNs were able to effectively learn spatial-contextual features for accurate discrimination of the boundary class from a very complex dataset.

Keywords: convolutional networks, agricultural boundary, deep learning, very high resolution satellite imagery.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my supervisors Dr. V.A. Tolpekin and Dr. C. Persello for the valuable suggestions and constructive criticism throughout the entire research period. I couldn't have wished for better, friendlier supervisors than you, you were always a team.

Secondly, I would like to thank the Netherlands Fellowship Program for providing me with the scholarship to study in ITC. It was a dream come true.

I thank my employer for granting me a study leave. Thank you for believing in me. I'll forever be grateful.

Special thanks to my GFM colleagues who I interacted with on a daily basis. The new friends I made in class. You made the learning experience fun. You'll forever be cherished.

Last but not the least I would like to thank my family, friends and relatives back home in Kenya. The messages, phone calls and WhatsApp calls kept me going.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Motivation and problem statement

Alleviating food insecurity is a top global agenda which can be achieved by improving the productivity of smallholder farming (FAO, 2016). Smallholding is characterised by farmers owning small pieces of land on which subsistence crops and sometimes a few varieties of cash crops are grown relying almost entirely on family labour (Lowder et al., 2016). Studies have shown that proper boundary information records can lead to improved food security (Rockson et al., 2013).

Delineation of agricultural farms has been addressed using different approaches for a long time. Land surveyors have been measuring boundaries and recording the information on cadastral maps for many centuries (Swetz, 2008). Modern surveying involves carrying out field measurements using theodolites, total stations and RTK GPS receivers followed by calculations to determine the precise location of boundaries. This process is however time-consuming, and the costs involved are quite high especially if the area to be surveyed is large. Manual digitization is another popular approach for boundary information acquisition from aerial photographs and satellite images. Although the approach is faster than direct field surveys, it is still time-consuming and dependent on the operator's subjectivity.

Availability of very high resolution (VHR) satellite images and UAV images have provided a cheaper and faster alternative for carrying out surveys for large areas. Satellite images have also presented other opportunities like crop monitoring. Utilization of VHR remote sensing data is however still low in Africa, especially in smallholding applications. Apart from the highly heterogeneous fields and landscape which make it difficult to extract information from the VHR remotely sensed images; the cost of acquisition of these images is quite high for the farmers. To overcome these challenges, there is a need for accurate, low-cost innovative methods for information extraction to help farmers in these regions. To this effect, there has been research on initiatives in the last couple of years focusing on the exploitation of VHR remote sensing technology in improving crop production and consequently improving the livelihoods of farmers. One of those initiatives is "Spurring a Transformation for Agriculture through Remote Sensing", the STARS project (de By & R.A., 2015).

Earlier automatic and semi-automatic techniques for boundary delineation were based on edge detection techniques (Mathias & Lemmens, 1996). Edge detectors (e.g., Roberts detector, Sobel edge detector, Zero crossing, Laplacian of Gaussian, Canny and Gaussian) extract edges by calculating gradients of local brightness (Bowyer et al., 1999). The Canny detector outperforms the rest with regards to simplicity, accuracy and its capability to reduce noise (Canny, 1986). A more recent agricultural boundary detection technique used by Alemu (2016) is the line segment detection algorithm (LSD), aimed at detecting straight contours in images (Grompone von Gioi et al., 2012). These methods perform reasonably well in regularly shaped agricultural fields but fail when presented with heterogeneous datasets as is the case of most farms in Sub-Saharan Africa. Another shortcoming is their inability to detect boundaries contextually, leading to many false detections.

Researchers have also proposed approaches based on image segmentation. Segmentation involves partitioning an image into segments which share similar attributes like texture, intensity, etc. with the aim of extracting the object of interest in the background. Mueller et al. (2004) proposed an object-oriented

segmentation approach for extracting large, man-made objects, especially agricultural fields, from high resolution satellite imagery. This approach involved integrating region- and edge-based segmentation techniques to extract straight edges. Image segmentation algorithms, however, are generally sensitive to intra-class variability which leads to over-segmentation. Smallholder farms mostly contain multiple crop types, therefore, have a high within-field variability. Secondly, image segmentation methods are highly dependent on a correct parameter selection, meaning they require a prior understanding of the scene or one may perform parameter tuning through trial and error (García-Pedrero et al., 2017).

Machine learning approaches have in recent years gained popularity in detection, segmentation and recognition of objects and regions in images (LeCun et al., 2015). These approaches include the conventional hand engineered feature extracting approaches which consume a considerable amount of time and the general-purpose learning frameworks like deep learning approaches. Deep learning approaches learn a hierarchy of features by automatically constructing high-level features from low-level ones (Nogueira et al., 2015).

In deep learning approaches like Convolutional Neural Networks(CNN), agricultural boundary detection can be defined as a classification problem which requires the definition of classes with a higher level of semantic abstraction. A boundary class takes different orientations, scale and spectral properties (bare soil, stones, trees, vegetation). This is a complex task which cannot rely on spectral signature alone but needs better features to discriminate classes (Scott et al., 2017). In deep learning, feature learning refers to a set of techniques that enable a framework to automatically find representations required for feature detection or classification from input data (Bengio et al., 2013).

A feature can be loosely defined as an "interesting" part or property of an image like edges, corners or blobs. Features can be local or global. Global features can be interpreted as a particular property of an image involving all pixels e.g. colour histograms, texture, edges etc. Local features on the other hand are patterns or a distinct structure associated with an image patch that differs from its surroundings by texture, colour or intensity (Salahat & Qasaimeh, 2017). The term "feature" in the context of agricultural boundary detection has different meanings in literature. This term has been used by some authors to mean physical objects which can be identified in an image like fences, walls, ditches etc. In this thesis, the term feature refers to both the local and global features which are automatically learnt by CNNs to enable them perform classification.

1.2. Research Identification

This research will focus on the design, analysis and evaluation of a per-pixel feature learning approach, based on Fully Convolutional Networks (FCNs), that can automatically detect and extract visible boundaries of agricultural fields in a Sub-Saharan Africa set-up. Most fields in Sub-Saharan Africa are small, irregular and with mixed crops hence high within field variability is inherent. The effect of varying the model's hyperparameters to the classification results will be examined in order to come up with an optimal FCN.

1.3. Research objectives

The main objective is to formulate a methodology for detection of visible agricultural field boundaries from VHR satellite images using Fully Convolutional Networks (FCN).

The main objective is achieved through the following specific objectives:

1. To review and evaluate state-of-the-art CNN/FCN architectures
2. To develop a methodology for detection of visible agricultural field boundaries from VHR images.
3. To compare the performance of the method against other state-of-the-art boundary detection methods.

1.3.1. Research questions

The following questions will be answered for the specific-objectives above.

Specific-objective 1:

1. Which CNN/FCN architectures have been proposed in literature and how do they work?
2. Which CNN/FCN architecture is appropriate for detection and extraction of agricultural field boundaries?

Specific-objective 2:

1. How should the classes be defined?
2. What are the optimal dimensions of the network and learning hyperparameters?
3. Which is the most suitable method for accuracy assessment of classification results?

Specific-objective 3:

1. Which alternative state-of-the-art approaches exist?
2. Which approach performs better and in which aspects of the performance measures?

1.4. Innovation aimed at

CNNs were introduced by LeCun and colleagues who applied back-propagation in recognizing handwritten zip code digits provided by the U.S postal service (LeCun et al., 1998). Convolutional Neural Networks (CNNs) is a class of deep learning architectures inspired by the structure of mammals' visual cortexes. Deep CNNs became famous thanks to the work of Krizhevsky et al. (2012) in the LSVRC-2010 image classification contest. Although more popular in computer vision and pattern recognition, researchers involved in the analysis of remotely sensed images have applied CNNs with considerable successes (Danilla, 2017; Mboga et al., 2017; Alshehhi et al., 2017). There are no publications found in literature dealing with agricultural boundary detection using FCNs. This research is motivated by the recent successes of CNNs in solving problems where individual classes take different scales, orientations, texture and spectral properties.

2. LITERATURE REVIEW

This chapter explores the theoretical background for this research. Section 2.1 presents the definitions of a boundary in different domains. A brief review of existing edge detectors, state-of-the-art contour detectors and segmentation algorithms is presented. Existing CNN and FCN architectures are also discussed in this chapter. Finally, other relevant studies in remote sensing domain are highlighted.

2.1. Boundary concepts

In a legal perspective, a boundary can be defined as "the utmost limit of lands, whereby the same are known and ascertained; the imaginary line which divides two pieces of land from one another (Duhaime, 2017). "The line is generally, but not necessarily, marked or indicated on the surface of the land by a wall, fence, ditch or another object". The Oxford English Dictionary (2017) defines a boundary as a line which marks the limits of an area; a dividing line. A more precise definition is given by Crommelinck et al. (2016) as a "dividing entity with a spatial reference that separates adjacent land plots". In this research, we disregard imaginary boundaries and focus on visible boundaries which can be detected from VHR satellite imagery.

In the land administration domain, boundaries are divided into two broad categories, fixed boundaries and general boundaries (Dale & McLaughlin, 1988). Fixed boundaries have their spatial positions determined and recorded with a higher level of accuracy. Direct surveying techniques are used to determine fixed boundaries. Direct surveys involve ground measurements using theodolites, total stations and global navigation satellite system (GNSS). The second category, general boundaries, have their precise spatial position left undetermined. Indirect techniques are applied in the determination of general boundaries which are visible from remotely sensed data such as aerial or satellite imagery.

Common boundaries between agricultural fields include fences, roads, hedges, footpaths, open areas, certain crop types, rivers, canals and water drainages (Bennett et al., 2010). Information on these boundaries can play an important role in various agricultural applications such as precision farming and yield forecasting (Jain et al., 2013).

2.2. Methods of boundary detection

2.2.1. Edge and Contour detection

An edge can be defined as a discontinuity in grey-level, colour, texture, etc. in an image (Sonka et al., 1999). Detecting contours refers to finding boundaries between objects or segments in images. Edge detection approaches can be classified into two; search-based and zero-crossing based edge detectors. Search based methods are also known as first-order derivative-based edge detection methods. They detect edges by measuring the intensity gradient at a point in the image in two steps; firstly by the gradient magnitude, which gives the strength of the edge as the amount of the difference between pixels in the neighbourhood and secondly by gradient orientation which is the direction of the greatest change. The Canny (1986) is one of the most outstanding search-based edge detectors because of its noise reduction capabilities. Canny consists of a Gaussian blur which removes small texture artefacts, a non-maximum suppression which makes edges more precise by thinning them, and a double threshold hysteresis that categorizes edges as weak or strong. Weak edges are eliminated if they do not meet the set threshold. A more detailed elaboration of the Canny detector is done in Section 4.3.2. Other first-order edge detectors include the Roberts (1965), Sobel and Prewitt (Parker, 2010).

Zero-crossing based, also known as the second order derivative-based edge detectors, search for zero crossings in the second derivative of the image to find edges. Zero crossings have the advantage of providing closed contours. They are, however, extremely sensitive to noise. The Laplacian of Gaussian stands out in this category of edge detectors. It applies a Gaussian smoothing filter followed by a derivative operation (Juneja & Sandhu, 2009).

While previous contour detection approaches quantify the presence of an edge at a given image location through local measurements alone, more recent methods take into account colour, brightness and texture information. These multiple cues were taken into consideration by Maire et al. (2008) at a local and global image scales through spectral partitioning. Contours which were unrecognisable using image information at a local scale were able to be detected on a global scale.

The globalized probability of a boundary (gPb) is often considered a state-of-the-art contour detection method in computer vision domain (Jevnisek & Avidan, 2016; Zhang et al., 2013). gPb involves detecting contours and assigning probabilities (Arbeláez et al., 2011). Crommelink et al. (2017) studied the transferability of gPb contour detection to remote sensing applications. Their study involved detecting visible cadastral boundaries on RGB images captured using UAV platforms. The whole processing pipeline (gPb-owt-ucm) was considered in their study which is what they referred to as gPb contour detection. Object contours were detected at impressive completeness and correctness rates of up to 80%. Motivated by this work, we use this technique (gPb-owt-ucm) as one of our comparison algorithms.

2.2.2. Image segmentation

Segmentation refers to the process of subdividing images into spatially continuous, disjoint and homogenous regions with regard to spatial or spectral characteristics (Blaschke et al., 2004). Many image segmentation procedures have been proposed for different applications. However, only a few lead to quantitatively convincing results (Sourav et al., 2016; Kaur, 2015). The reason is that in most cases, the regions of interest are highly heterogeneous. Image segmentation methods can be divided into pixel- based, edge-based and region-based segmentation methods (Blaschke et al., 2004).

Pixel-based methods include image thresholding and segmentation in feature space. Edge-based segmentation methods find edges between regions and determine segments as regions within these edges. The first step in edge-based segmentation is edge detection, which has been described in detail in Section 2.2.1. The difference between edge-based segmentation and contour detection is that the later uses abrupt discontinuities in pixel values(edges) to define the extents of regions, which is the output. Contour detection, on the other hand, aims at finding the boundary between two regions and involves postprocessing edge detection results.

Region-based segmentation algorithms can be divided into region growing, merging and splitting techniques and their combinations. Region growing techniques aggregate pixels starting with a set of seed points. Neighbouring pixels are joined to the initial regions until a specific set threshold is achieved. The threshold is a homogeneity criterion or a combination of size and homogeneity. Region growing algorithms often suffer from lack of control over break off criterion for the growth of a region (Aguilar et al., 2016). In region merging and splitting, the image is divided into sub regions which are joined or split based on their properties. The idea behind region merging is joining segments starting with their initial regions. Region splitting algorithms divide segments into smaller units if they are not homogenous as required starting from the input which normally consists of large segments.

2.2.3. Convolutional Neural Networks (CNNs or ConvNets)

CNNs are made up of one or more convolutional layers that have learnable weights and biases. Standard architectures use a series of convolutional layers that extract feature maps, which are then flattened into a one-dimensional vector and fed to a fully-connected network (Goodfellow et al., 2015). The input to a convolutional layer is an image of dimensions $W \times W \times D$ where $W \times W$ is the height and width of the image and D is the number of channels. The convolutional layer is made up of K filters of size $F \times F$ where F is smaller than the dimensions of the image W. The filters are convolved with the image to produce feature maps which are equal to the number of filters (K). Each feature map is subsampled with average of max pooling with a stride s, where the value of $s > 1$. Subsampling with a stride = 1 leads to feature maps equal to the input image. After the subsampling layers (pooling layers), a bias is added and non linearity is applied to each feature. Non-linearities are also called activation functions and include Relu, Sigmoid, tanh etc. At the end of the networks, fully connected layers are inserted in standard CNN architectures. A more detailed description of the building blocks is found in Section 4.1.

CNN architectures

In the remote sensing domain, Bergado et al. (2016) proposed a CNN architecture for urban scene classification using high resolution aerial images. Alshehhi et al. (2017) suggested a single-patch CNN architecture for extraction of roads and buildings from high resolution remote sensing imagery. Mboga et al. (2017) proposed an architecture for detecting informal settlements from VHR satellite imagery. The authors compared their method with other state-of-the-art methods like SVM with RBF kernel and recorded better performance after presenting their method with many training samples and a large number of convolutional layers.

Other popular architectures in the literature include AlexNet developed by Krizhevsky et al. (2012), VGGNet by Simonyan & Zisserman (2014), LeNet, ZF Net, GoogLeNet and ResNet (Li et al., 2017). We begin by presenting the two most popular architectures.

AlexNet

Table 2.1: Representation of AlexNet architecture.

Layer	Dimensions	Parameters			
		No. of filters	Filter dimensions	Stride	Pad
Input	227×227×3	-	-	-	-
CONV -1	55×55×96	96	11 × 11	4	0
Max pool 1	27×27×96	-	3×3	2	-
Norm 1	27×27×96	-	-	-	-
CONV -2	27×27×256	256	5×5	1	2
Max pool 2	13×13×256	-	3×3	2	-
Norm 2	13×13×256	-	-	-	-
CONV -3	13×13×384	384	3×3	1	1
CONV -4	13×13×384	384	3×3	1	1
CONV -5	13×13×256	256	3×3	1	1
Max pool 3	6×6×256	-	3×3	2	-
FC6	4096 neurons	-	-	-	-
FC7	4096 neurons	-	-	-	-
FC8	1000 neurons	-	-	-	-

Table 2.1 summarises the AlexNet architecture developed Krizhevsky and colleagues (Krizhevsky et al. 2012). This architecture is perhaps the most popular CNN architecture to date. It was submitted in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) outperformed other models by big margins. Their top 5 error was recorded as 16% compared to the closest challenger with 26%.

Other hyperparameters used in the AlexNet included; use of the Relu activation function, a dropout of 0.5, batch size 128, Stochastic Gradient Decay (SGD) Momentum of 0.9, a Learning rate of 0.01 then reduced it to 0.1 manually upon plateauing of validation accuracy and L2 weight decay of 0.0005.

VGGNet

Simonyan & Zisserman (2014) shown that the depth of the network is a critical component for good performance of a CNN. Increasing the depth of a network is, however, more expensive because it increases the number of parameters, therefore, using more memory. Table 2.2 shows a brief illustration of VGG16 architecture submitted at ILSVRC in 2014.

Table 2.2: Representation of VGG16 architecture

Layer	Dimensions	Parameters	
		No. of filters	Filter dimensions
Input	224×224×3	-	-
Conv 1-1	224×224×64	64	3×3×3
Conv 1-2	224×224×64	64	3×3×64
Max pool 1	112×112×64	-	-
Conv 2-1	112×112×128	128	3×3×64
Conv 2-2	112×112×128	128	3×3×128
Max pool 2	56×56×128	256	3×3×128
Conv 3-1	56×56×256	-	-
Conv 3-2	56×56×256	256	3×3×256
Conv 3-3	56×56×256	256	3×3×256
Max pool 3	28×28×256	-	-
Conv 4-1	28×28×512	512	3×3×256
Conv 4-2	28×28×512	512	3×3×512
Conv 4-3	28×28×512	512	3×3×512
Max pool 4	14×14×512	-	-
Conv 5-1	14×14×512	512	3×3×512
Conv 5-2	14×14×512	512	3×3×512
Conv 5-3	14×14×512	512	3×3×512
Max pool 5	7×7×512	-	-
FC6	1×1×4096	-	-
FC7	1×1×4096	-	-
FC8	1×1×1000	-	-

2.2.4. Fully Convolutional Networks (FCN)

As described in section 2.2.3, traditional CNNs have fully connected layers with fixed dimensions. One shortcoming of the fully connected layers is that they discard the spatial information. The output of a CNN is a 1-dimensional distribution over classes (for Softmax regression) i.e they predict a single label. The idea

in a patch-based CNN is to extract small patches or subsets of the image then apply a CNN model on each patch to predict their central pixel. These labels are then arranged in a 2-D layout as the outputs (Fu et al., 2017).

Long et al. (2015) proposed an FCN model replacing the fully connected layers in a CNN with convolutional layers. The FCN models can be trained to predict all the pixels in an image or patches, therefore, the 2-D structure is preserved. A patchwise trained FCN predicts the labels of all the pixels of the entire patch, unlike the CNN which predicts the label of the central pixel. Experiments have shown that FCN performs better than CNN (Fu et al., 2017; Persello & Stein, 2017). Other advantages of FCN over CNN are : FCNs are easy to implement since its possible to train an entire image at a time. Lastly FCNs are less computationally expensive. CNNs predict the label of a pixel per patch therefore more computations are required to predict the labels of all pixels in a patch and the entire image.

FCN-DKs

Persello & Stein (2017) introduced a novel deep FCN for the detection of informal settlements in VHR images. Their proposed FCN consisted of dilated kernels of increasing spatial support. Using large filters increases the number of parameters therefore increasing the cost of training the networks. A high number of parameters also reduces the generalization capability of the networks. Downsampling technique is one of the strategies of correcting this problem. Persello & Stein proposed using dilated kernels (DKs) instead of downsampling. Using dilated kernels increases the spatial support without increasing the number of memory parameters. This increases the number of learnable parameters per layer without increasing the computational cost. Table 2.3 summarises the work on FCN-DKs.

Table 2.3: Architecture of FCN-DKs

Layer	Module type	Dimension	Dilation	Stride	Pad
DK1	Convolution	5 ×5 ×8 ×16	1	1	2
	lReLU				
	Max-pool	5 ×5		1	2
	Convolution	5 ×5 ×16 × 32	2	1	4
DK2	lReLU				
	Max-pool	9 ×9		1	4
	Convolution	5 ×5 ×32 ×32	3	1	6
DK3	lReLU				
	Max-pool	13 ×13		1	6
	Convolution	5 ×5 ×32 ×32	4	1	8
DK4	lReLU				
	Max-pool	17 ×17		1	8
	Convolution	5 ×5 ×32 ×32	5	1	10
DK5	lReLU				
	Max-pool	21 ×21		1	10
	Convolution	5 ×5 ×32 ×32	6	1	12
DK6	lReLU				
	Max-pool	25 ×25		1	12
	convolution	1×1 ×32 ×2	1	1	0
Class	Softmax				

2.3. Relevant studies in agricultural boundary detection

Alemu (2016) proposed a line segment detection (LSD) algorithm for delineating farm field boundaries and detecting crop rows from very high-resolution satellite images. He performed an accuracy assessment by defining two error ratios; a ratio for missing detections and a ratio for false detection for boundaries. He obtained 0.78 and 0.73 respectively. He made a conclusion that the errors were too high to be acceptable in practical applications.

Davidse (2015) experimented with different approaches to extract agricultural field boundaries from WorldView-2 satellite imagery. He applied techniques from PCA (Principal component analysis), Image segmentation and edge detection. His study area was in Sougoumba, Mali, West Africa. He noted that the study area was difficult because of a highly heterogeneous landscape. Image segmentation performed better than the other methods, but he also noted there was the risk of over-segmentation.

Machine learning approaches have also been proposed for agricultural parcel delineation. García-Pedrero et al. (2017) proposed an agglomerative segmentation methodology using the SLIC (Simple Linear Iterative Clustering) algorithm. Experiments were done on a WorldView-2 satellite imagery acquired on 3 December 2011 for agricultural fields in the Chilean central valley. They noted a huge potential for training machine learning algorithms to do what a human operator would do in a boundary delineation task. One of their main suggestions for future research involves determining optimal features to train a machine learning methodology.

Turker and Kok (2013) used a perceptual grouping approach to develop a methodology for extraction of boundaries within agricultural fields from SPOT imagery for a region in north-west Turkey. They used the Canny edge detector to detect edge pixels. Validation of their approach was done on SPOT4 and SPOT5 images. The overall matching accuracies between the reference data and the automatically extracted boundaries were 82.6% and 76.2% for SPOT5 and SPOT4 images respectively.

3. DATA AND SOFTWARE

This chapter describes the study area, raw datasets, ground reference data and software used.

3.1. Study area

This research was applied on a dataset involving agricultural farms in a village called Kofa in Kano state, Northern parts of Nigeria. Kofa area is characterised by a highly heterogeneous landscape. Farmers practice mixed farming; the fields are small in sizes with irregular shapes, there are variable planting dates and trees are present in nearly all the fields. Crops in this region include sorghum, rice, beans, millet, groundnut, pepper, maize, soybeans, moringa and cowpea (Agro News Nigeria, 2018). The rainy season is experienced in May, June, July and September. Dry periods are in January, February and August. The warmest month is April while the coolest month is August. September is the wettest month while January is the driest month. The fields are located approximately between 8° 14' 30" E to 8° 16' 0" E and 11° 34' 0" N to 11° 32' 30" N.

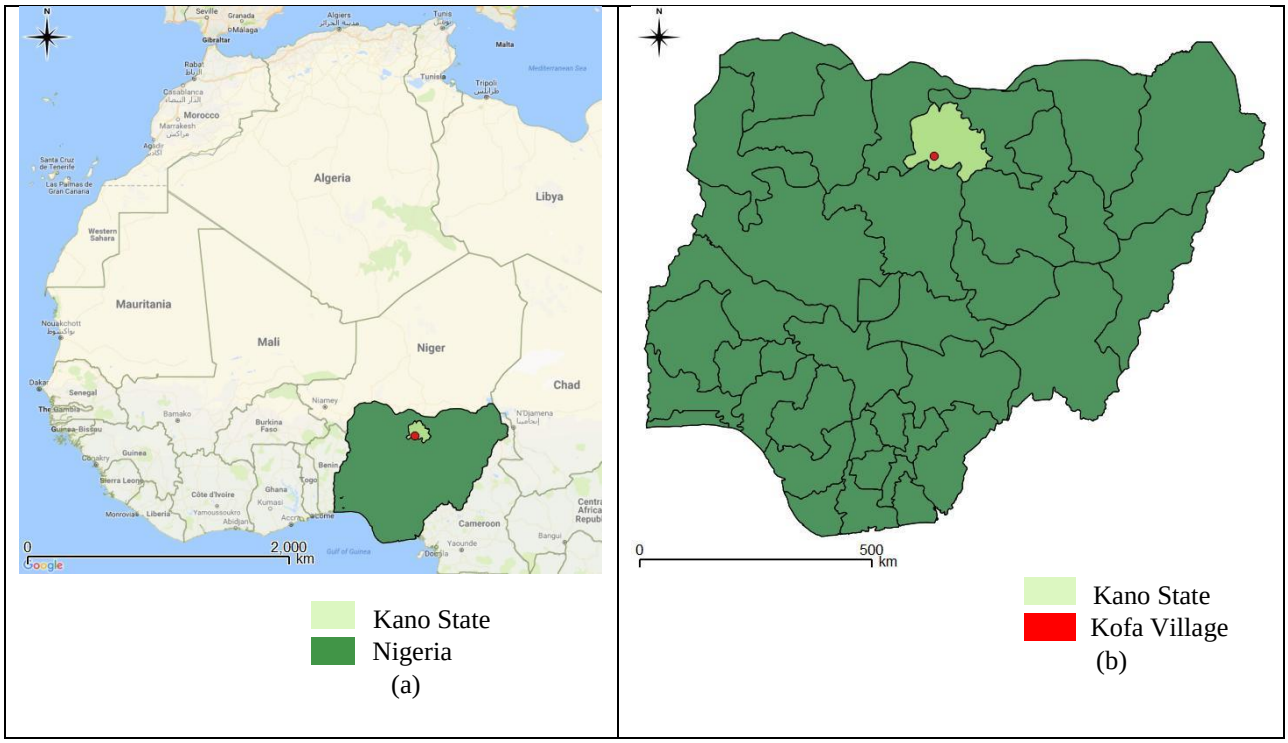


Figure 3.1: Study area: (a) Location of Nigeria in West Africa (b)Location of Kofa in Kano state, Nigeria.

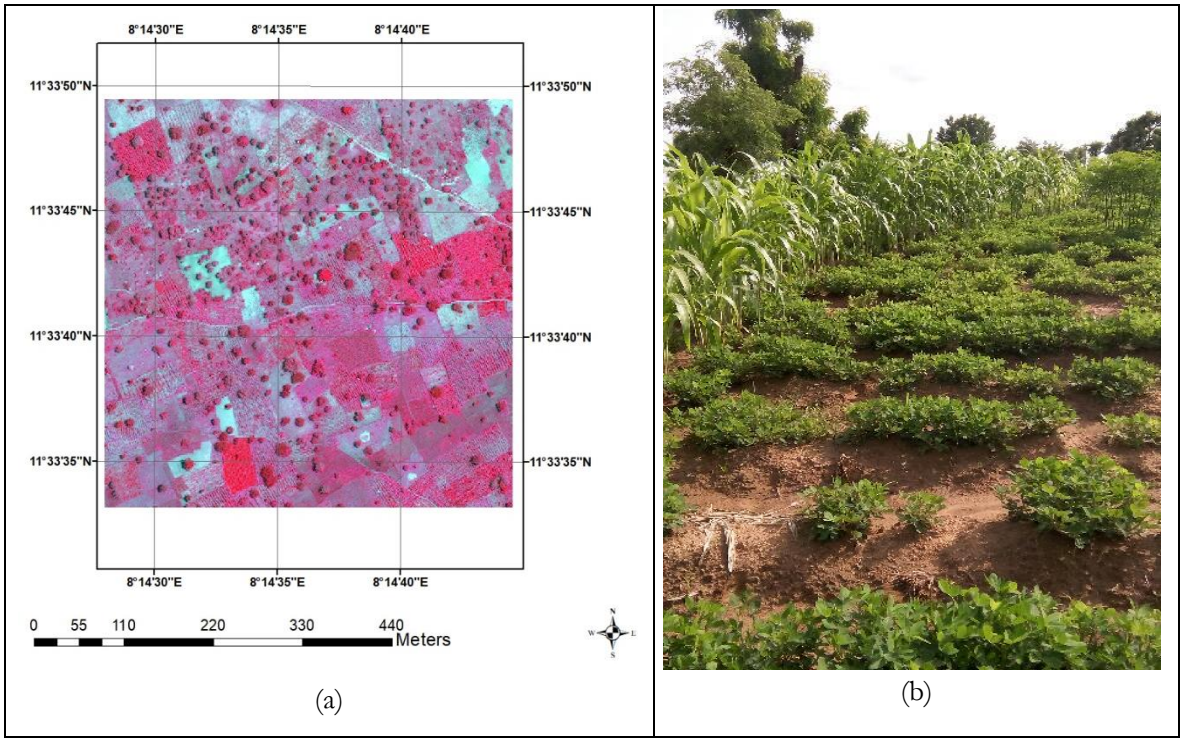


Figure 3.2: Study area: (a) A tile extracted from WorldView3(bands 7, 5, 3) satellite image captured over Kofa, Nigeria on 25-Sep-2015 (b)A photograph captured in one of the agricultural fields in Kofa showing mixed crops with trees between the fields

3.2. VHR Satellite imagery

A total of 9 WorldView-3 satellite images acquired during different stages of the cropping period (May – November) were available as shown in Table 3.1. This data was acquired in the STARS project for the years 2014 - 2015. The images were renamed using their acquisition dates for easier identification. WorldView-3 sensor specifications are listed in Table 3.2.

Table 3.1: Available WorldView-3 satellite images

	Image ID	Bands	GSD pan(m)	GSD ms (m)
1	WV3_26-Nov-2014_ms	8	0.5	2.0
2	WV3_8-May-2015_ms	8	0.3	1.2
3	WV3_22-May-2015_ms	8	0.3	1.2
4	WV3_3-Jun-2015_ms	8	0.3	1.2
5	WV3_24-Jul-2015_ms	8	0.3	1.2
6	WV3_10-Sep-2015_ms	8	0.5	2.0
7	WV3_12-Sep-2015_ms	8	0.5	2.0
8	WV3_25-Sep-2015_ms	8	0.5	2.0
9	WV3_21-Nov-2015_ms	8	0.5	2.0

Table 3.2: WorldView-3 Sensor Specifications (DigitalGlobe, 2017)

Sensor Bands	Sensor Resolution (GSD)	Swath Width	Revisit Frequency (at 40°N Latitude)
• Panchromatic: 450-800 nm • 8 Multispectral: Coastal: 400 - 450 nm Blue: 450 - 510 nm Green: 510 - 580 nm Yellow: 585 - 625 nm Red: 630 -690 nm Red Edge: 705 - 745 nm Near-IR1: 770 - 895 nm Near-IR2: 860 - 1040 nm • 8 SWIR: (1195 nm - 2365 nm) • 12 CAVIS Bands: (405 nm - 2245 nm) desert clouds, aerosol-1, aerosol-2, aerosol-3, green, water-1, water- 2, water-3, NDVI-SWIR, cirrus, snow	**Panchromatic:** 0.31 m GSD at Nadir 0.34 m at 20° Off-Nadir **Multispectral:** 1.24 m at Nadir, 1.38 m at 20° Off-Nadir	At nadir: 13.1 km	Less than one day at 1 m GSD or 4.5 days at 20° off-nadir or less

3.3. Pre-processing

A selection had to be made for the most appropriate image. Images taken during the early stages of cropping period, when the fields are almost bare, had little information therefore could not be used for analysis. One WorldView-3 satellite image (WV3_25-Sep-2015_ms) was chosen as the most appropriate for experiments leading to extraction of visible boundaries. The reference shapefile was overlaid in this image and the visible boundaries in the image seen to be coinciding perfectly.

Multi spectral datasets were pan sharpened using the Gram-Schmidt algorithm. Resampling was based on nearest neighbour because it is less computationally expensive. Gram-Schmidt integrates the geometric information of the panchromatic band (the higher spatial resolution image) with spectral information of the multi-spectral bands (higher spectral information images). The resulting image (pan sharpened) is an image with the best properties of both images types, high spatial and high spectral resolution. Image WV3_25-Sep-2015_ms had a GSD of 0.5 m for its panchromatic band and 2.0 m for its multi spectral bands. The resulting image after pan sharpening had a spatial resolution of 0.5 m. Other pan-sharpening algorithms include NNDiffuse, SPEAR pan sharpening and PC spectral sharpening. Gram-Schmidt pan-sharpening outperforms the other pan sharpening methods in maximising image sharpness and minimising colour distortion, however, it is more computationally expensive (Maurer & Street, 2013).
We extracted 5 tiles from Image WV3_25-Sep-2015_ms and labelled them as TR1,TR3,TR4 TS2 and TS5, where TR means the tiles were used to train the networks and TS for testing. Each tile has dimensions 1000 ×1000 pixels and covering an area on the ground of 0.5 × 0.5 km.

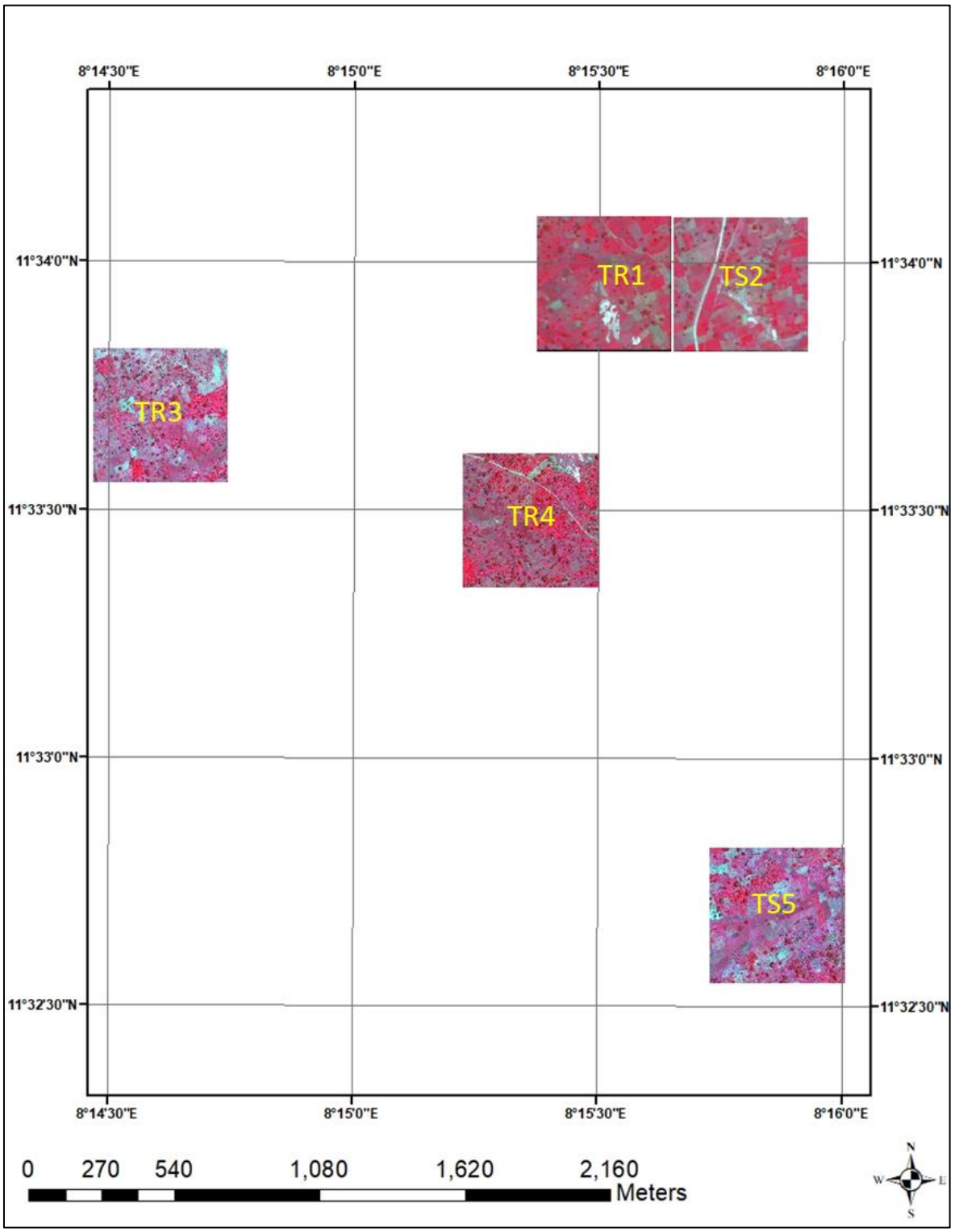


Figure 3.3: Location of tiles TR1, TR3, TR4 and TS2, TS5 in Kofa region

3.4. Reference datasets

Ground truth data for the 5 tiles was available from the STARS project. This data was availed as polygons with fields information in shapefile (.shp) format. The information includes crop types and field boundaries. Fields which had more than 1 crop planted had that information included. Table 3.3 gives an overview of a total number of agricultural fields in each tile.

Table 3.3: Number of farms in each tile

Tiles	No. of fields
TR1	214
TS2	188
TR3	151
TR4	244
TS5	232

Figure 3.4 shows the raw reference data for TR1, which is the same case for the other tiles.

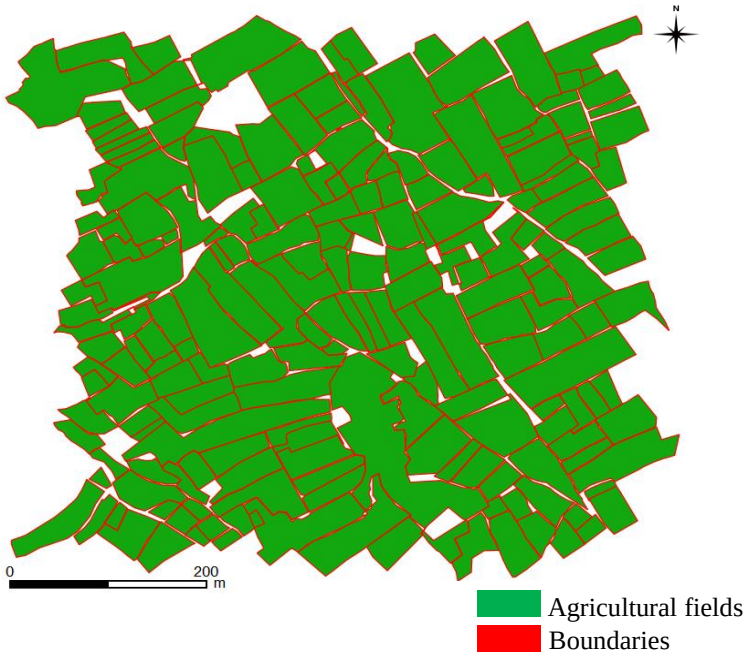


Figure 3.4: TR1 raw reference data

The reference dataset had to was converted to a raster format with equal thickness for the boundary class and a spatial resolution corresponding to that of the pre-processed VHR image. Visible boundaries in the VHR image are not of uniform width. Some boundaries are broad, e.g. boundaries coinciding with paths and others are thin e.g. boundaries at the transition between two adjacent agricultural fields. A choice for

the thickness of buffering had to be made to consider the variations in thickness of boundaries and ensure positional accuracy of detections. FCNs accept as input raster images inorder to make pixel-wise reference to the labels. Two techniques were used to obtain a raster image of boundary and non boundary labels from the raw shapefile. The first technique involved converting the raw shapefile from polygons to lines which generated dual lines for adjacent plots giving a false impression of the true thickness of the boundaries. "Collapse dual lines to centreline technique" in ArcMap was used correct this. The second technique involved converting the raw shapefile to raster format(tiff) directly then applying morphological thinning to obtain the central pixel in the boundary pixels. The raster was again converted to vector to enable buffering to a desired thickness. Morphological thinning was faster than the previous strategy especially when FCN experiments demanded more data. The resulting single lines in both strategies were buffered using 0.5 m, 1 m and 2 m radius. A decision was made to use 1 m buffer for further analysis because of accuracy requirements and spatial resolution of the available VHR satellite image. The buffer was labelled as the boundary class (Class 1) and the region not covered by the buffer (the agricultural fields) labelled as the non-boundary class (Class 2). New shapefiles covering the extents of each tile were created and a union performed with their corresponding 1 m buffer outputs from the previous step. Sections of fields outside the coverage were cropped out. The result was vector files with the same spatial coverage as the VHR image tiles. The next step involved converting the vector files to raster format(.tiff). Feature to raster conversion tool in ArcMap was used to perform this. The output cell size was set as 0.5 m, same as the pre-processed VHR image. Figure 3.5 shows the prepared reference images of the tiles considered in this research.

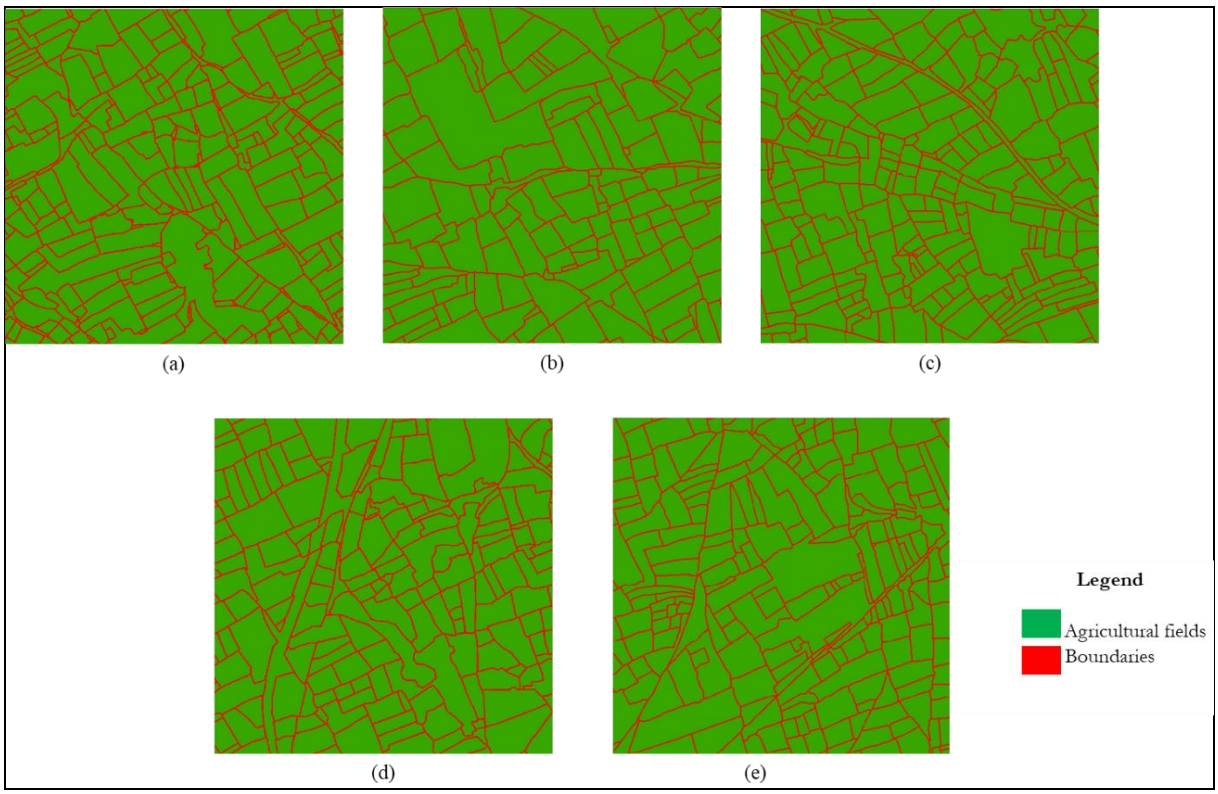


Figure 3.5: (a) GT1 (b) GT3 (c)GT4 (d)GT2 (e)GT5 are the ground truth images for tiles TR1, TR3, TR4, TS2 and TS5 respectively.

3.5. Software

The deep learning framework was based on MatConvNet-1.0-beta23. MatConvNet is a third party Matlab toolbox developed by Vedaldi & Lenc (2015) implementing CNNs for different applications.

The networks were trained using NVIDIA's CUDA GPUs (Quadro M1000M).

ArcGIS version 10.5.1 was used for the preparation of the reference data.

Morphological thinning was performed using C++ and called from R using package Rcpp.

ENVI version 5.3.1 was used for pan-sharpening the VHR satellite imagery. ENVI was also used to convert raw RGB bands extracted from the multispectral images to tiff image format for experiments on contour detectors.

ERDAS imagine 2016 was used for sub-setting data.

eCognition software version 9.3 was used to implement multi-resolution segmentation algorithm.

Linux operating system was used to run the Matlab precompiled open-source package for globalised Pb experiments developed by UC Berkeley Computer Vision Group (Arbeláez et al., 2013).

4. METHODS

This chapter gives a detailed description of the methods and experiments carried out towards achieving the main objective of detecting visible agricultural field boundaries from VHR satellite images.

4.1. FCN Design experiments

The architecture was developed from FCN-DKs, described in Table 2.3 by Persello & Stein (2017) as the foundation.

4.1.1. Data on hand

Figure 3.5 shows the available VHR satellite images and Figure 3.5 shows the prepared data for experiments. All the tiles (TR1, TS2, TR3, TR4, TS5) have dimensions 1000 x 1000 pixels for both the raw VHR satellite data and the reference data. The reference raster grid was derived from the input image. Tiles TR1, TR3 and TR4 were used for training and validation while tiles TS2 and TS5 were used for testing the performance of the implemented algorithms. In this research, tiles used for training the networks are abbreviated as TR while those used for testing are abbreviated as TS.

4.1.2. Hyper-parameter optimization

Preliminary experiments were carried out to help in fine-tuning parameters and design an optimal architecture. TR1 and TR2 were used for the initial experiments to minimise the training time. FCN-DK3 illustrated in Table 2.3 was used as the starting point. Training and validation sets were generated from TR1 while TS2 was used for testing. Patches were used as inputs to the networks. Figure 4.1 shows an overview of an FCN architecture in the context of agricultural field boundary detection.

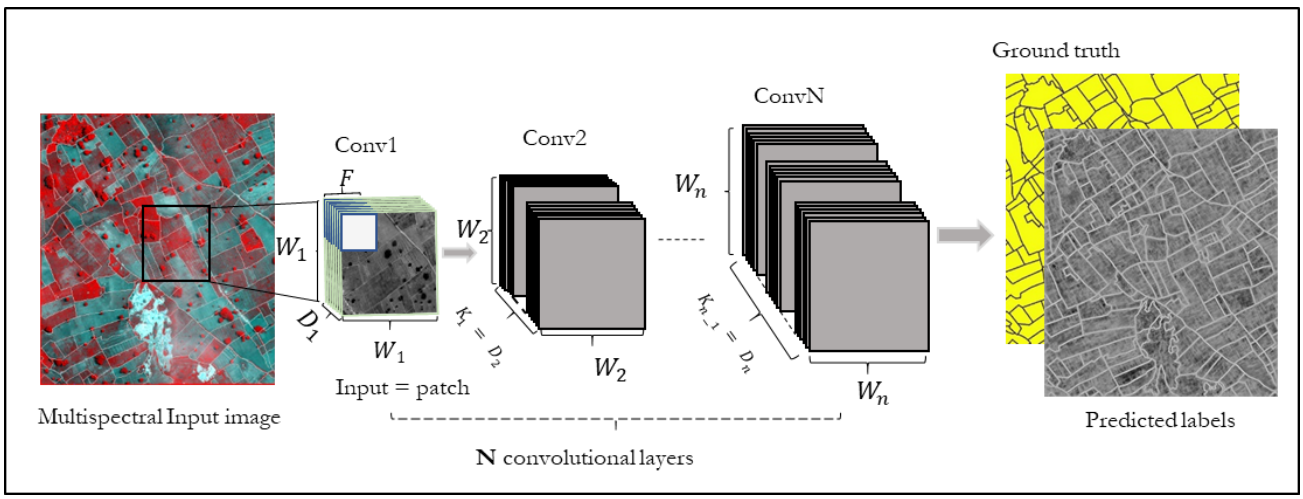


Figure 4.1: Overview of the adopted FCN architecture: patch samples were used as inputs in the networks. Each convolutional layer has a bank of filters which convolve with the inputs producing feature maps of dimensions equal to the inputs.

Input layer

The input layer holds the raw pixel values of the input patches. This layer has dimensions $W \times W \times D$ where $W \times W$ is the patch size and D is the depth of a patch, which is equal to the number of channels of the multi spectral VHR satellite image.

Convolutional layers

The convolutional layer (Conv layer) is the core building block in a convolutional neural network. A convolutional layer consists of K learnable filters (a set of weights) of dimension $F \times F$. A filter on the first layer would have smaller dimensions than the dimensions of the input patch.

Each filter is convolved across the width and height of the input volume during the forward pass. A 2D activation map of filter responses of every spatial position is output. The total number of the activation maps is equal to the total number of filters. The network learns filters which activate when they detect some visual feature such as edges of some orientation.

The hyper-parameters of a convolutional layer include; the number of filters which corresponds to the depth of the output volume, the filter dimensions, the stride, S , which is the step by which we slide the filter. A stride of 1 moves the filter 1 pixel at a time resulting in an output of the same dimensions as the input. A stride greater than 1 produces smaller output volumes. Another hyperparameter is zero-padding, P, which refers to filling border rows and columns of an input patch with zeros. It is useful because we can control the size of output feature map.

Setting zero padding as $P = \frac{(F-1)}{2}$ when the stride is $S = 1$ ensures that the input volume and output volume have the same size. The last hyperparameter of convolutional layers is dilation, d, which involves inserting spaces between cells of filters. Dilated convolutions increase the effective receptive field without increasing the number of parameters in the networks.

A convolutional layer N accepts a volume of size $W_{n-1} \times W_{n-1} \times D_{n-1}$ and outputs a volume of size $W_n \times W_n \times D_n$ where, $W_n = \frac{W_{n-1} - \mathrm{F} + 2\mathrm{P}}{S+1}$ and $D_n = K_{n-1}$.

We applied a stride of 1 to obtain output volumes with the same spatial dimensions as the input volumes throughout the networks. Zero padding the input volumes in the convolutional layers was done equal to 2 × dilation factor to ensure full coverage of input volumes with the dilated kernels. The filter dimensions were kept consistent as 5 × 5 for all convolutional layers apart from the output convolutional layer which was reduced to 1 × 1. The number of kernels in the first convolutional layer was set as 16, then increased to 32 from the second layer then retained constant for the next Conv layers. The output layer had 2 filters for the two classes.

Pooling layer

Pooling (also called subsampling or downsampling) reduces the dimensions of each feature maps but retains the learned information from the convolutional layers. Pooling layers are embedded between successive convolutional layers in an FCN architecture. There are different types of pooling: max pooling, average pooling etc (Li et al., 2017). In Max pooling, a window is defined e.g 2×2 . The largest element from the

feature map is picked within that window. Average pooling averages the elements in that window instead of picking the maximum element. Pooling continually reduces the spatial size of the inputs, therefore reducing the number of parameters and computation in the network. Pooling layers control overfitting. Apart from max pooling, other functions include average pooling, which returns the arithmetic mean of received signals and L2-norm pooling. A pooling layer requires 2 hyperparameters: 1) Spatial extent, F, and 2) Stride, S. This layer accepts a volume of size $W_1 \times H_1 \times D_1$ and outputs a volume of size $W_2 \times H_2 \times D_2$ where $W_2 = \frac{W_1 - F}{S+1}$, $H_2 = \frac{H_1 - F}{S+1}$ and $D_2 = D_1$.

Springenberg et al. (2014) proposed CNNs without pooling layers in a bid to achieving a simple architecture which consists of convolutional as the only heavy computational layers.

Batch Normalization layer

During training, the distribution of each layer inputs changes since the parameters of the previous layers change. This leads to a slow training process making it a necessity to use lower learning rates and precise parameter initialization. Ioffe & Szegedy (2015) proposed incorporating normalization for each training mini-batch as a feature of the architecture. Batch normalization permits use of higher learning rates and less attention on initialization. Batch normalization also acts as a regularizer eliminating the need for Dropout. We leveraged on these benefits and introduced batch normalization layer after every convolutional layer throughout the networks.

Activation Functions

Activation functions are nodes added to the output of any layer in an FCN (Sharma, 2018). Activation functions define the output given a set of inputs. Non-linear activations are used because they are capable of approximating any function. They break linearities in the network, allowing it to learn more complex functions than a linear activation would do. The most common non-linear activation functions include the Sigmoid, ReLU and Leaky ReLU.
The sigmoid, also called Logistic activation function, takes an S-shape and is expressed as:

$$f(x) = \frac{1}{1+e^{-x}} \quad 4.1$$

Sigmoid activation functions take a real number and normalize it into the range between 0 and 1. Large negative numbers become 0 and large positive numbers become 1.

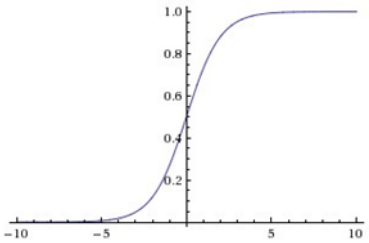


Figure 4.2: Sigmoid function

Sigmoid non-linearity, however, suffers some drawbacks: they saturate and "kill gradients". The gradient at either tail, 0 or 1 is zero therefore when the activation of the neuron saturates at those regions, no signal will flow through it to its weights and to data. Secondly, sigmoid outputs in the later stages of an FCN are not zero-centred. This introduces zig-zagging effect during gradient descent, such that its either all positive

or negative gradient on the weights. It is because of these short-comings that the sigmoid non-linearity is less popular in recent research outputs.

Most recent CNNs and FCNs use Rectified Linear Units(Relu) non-linearities (Krizhevsky et al., 2012; Li et al., 2017). ReLU is expressed as;

$$f(x) = \begin{cases} 0, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases} \quad 4.2$$

It has an output equal to 0 if the input is less than 0 and an output equal to input if its greater than 0. Krizhevsky et al. (2012) found ReLU to accelerate the convergence of stochastic gradient descent by a factor of 6 compared to sigmoid function. This was attributed to its linear, non-saturating form. Relu activation functions have however been known to be sensitive to large gradients. Setting a high learning rate may 'kill' many neurons such that they will never activate over the entire training dataset (Li et al., 2017).

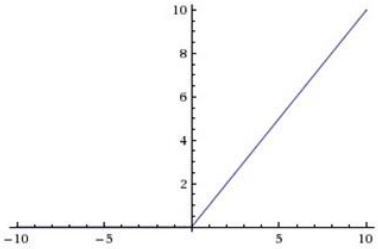


Figure 4.3: Relu activation function

Leaky Rectified Linear Units (Leaky ReLU) fixes the "dying ReLU" problem(Maas et al., 2013). When $x < 0$, the function computes a small negative slope (leak). The leak increases the range of the Relu function. Leaky Relu is expressed as:

$$f(x) = \begin{cases} c\,x, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases} \quad 4.3$$

Where, c is the leak factor

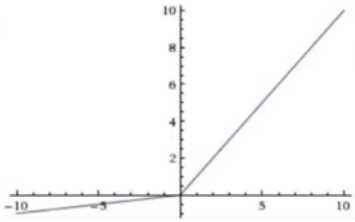


Figure 4.4: Leaky ReLU activation function

We used the Leaky ReLU non-linearities in our networks because of its agility over the other activations.

Softmax layer

The Softmax layer is an activation function added as the final (output) layer to perform classification (Bishop, 2013). This function takes in a number of score values equal to the number of classes $z_k, k = 1 \ldots K$, then squashes them into values in the range between 0 and 1. The sum of these values equal to 1, representing a probability distribution over K possible outcomes.

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad 4.4$$

Where z is a vector of the class inputs (in our case we had 2 classes, therefore, the elements in z were 2). c indexes the output classes, therefore j = 1, 2.

Dropout layer

Overfitting is a big challenge in deep neural networks. Dropout is a regularization technique used during training to address this problem by dropping units, along with their connections, with a probability $1 - p$ or keeping them with a probability p, so that a reduced network is left (Srivastava et al., 2014). Training on data is done on the reduced network.

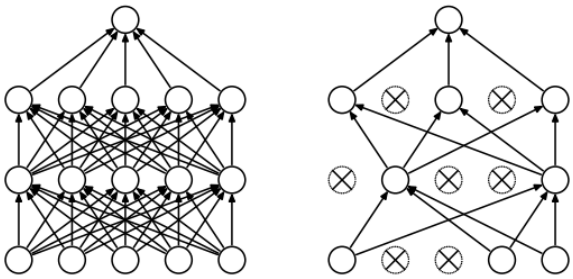


Figure 4.5: Applying dropout to a neural network

In practice, dropout improves the performance of FCNs. Training speed is improved significantly. Dropout can also enable a model to learn more robust features that can better generalize new data. We applied dropout with a rate of 0.5 in the last classification layer.

4.1.3. Training the networks on the full dataset

In section 4.1.2, we have shown how we investigated the various components of our FCN. We performed hyperparameter sensitivity experiments which included varying the patch size, sample size, depth of the network and discarding the pooling layers. These experiments were done using TR1 for the training set and TS2 for testing, to minimise computational cost. We rolled out our final implementation on the whole dataset to investigate the effect of adding more data on the networks. We train with 3 tiles (Tiles1,3 and 4 and test with 2 tiles (TS2 and 5).

4.1.4. Augmenting the full dataset

Data augmentation techniques aim at creating more training data artificially from real data by applying invariances. Augmentation techniques include cropping, rotating and flipping input images. We perform a rotation of 90^0 followed by a vertical flip for the 2000 training patches for each training tile. The result is a training dataset with 6000 training samples of 3 different orientations. The networks trained with the augmented dataset to increase rotational invariance.

4.2. Final Implementation

4.2.1. Network Architecture; FCN-DKConv6

Table 4.1shows our proposed architecture. The architecture is composed of convolutional layers followed by batch normalizations and Leaky Relu non-linearity. 6 convolutional layers were used in the final implementation. A 1×1 convolutional filter is used in the classification layer to predict labels.

Table 4.1: Final implementation; FCN-DKConv6

Networks	Layer	Weights(W×W×D×K)	Stride	Pad	Dilation	Receptive field
	Conv1	5 ×5 ×8 ×16	1	2	1	5
FCN-DKConv1	bnorm1	-	1		-	
	lReLU1	-	1		-	
	Conv2	5 ×5 ×16 × 32	1	4	2	13
FCN-DKConv2	bnorm2	-	1		-	
	lReLU2	-	1		-	
	Conv3	5 ×5 ×32 ×32	1	6	3	25
FCN-DKConv3	bnorm3	-	1		-	
	lReLU3	-	1		-	
	Conv4	5 ×5 ×32 ×32	1	8	4	41
FCN-DKConv4	bnorm4	-	1		-	
	lReLU4	-	1		-	
	Conv5	5 ×5 ×32 ×32	1	10	5	61
FCN-DKConv5	bnorm5	-	1		-	
	lReLU5	-	1		-	
	Conv6	5 ×5 ×32 ×32	1	12	6	85
FCN-DKConv6	bnorm6	-	1		-	
	lReLU6	-	1		-	
	conv	1×1 ×32 ×2	1		1	
Classification	dropout					
	Softmax					

4.3. Alternative approaches

In this section, we present the alternative approaches for boundary detection that were compared with FCN. Several factors were taken into account during the algorithm comparisons because in most real-world applications the quality of the results is not the only important measure of performance. Other important factors considered were speed and memory consumption. A trade-off exists between the quality of results and the time taken by an algorithm to produce them. In practice, to produce better results more complex models are required which means more computational costs and a longer runtime. Testing tiles TS2 and TS5 (used for testing FCN) were used in experiments of the alternative methods for a fair comparison of all the algorithms.

4.3.1. Global Probability of Boundary (gPb)

The globalized probability of a boundary(gPb) involves linearly combining multiscale probability of a boundary (mPb) with Spectral probability of a boundary (sPb). These 2 methods are developed from the original probability of a boundary (Pb) by Martin et al., (2004), which uses an oriented gradient signal to evaluate the strength of a contour through a set of pixels. Each pixel is examined locally. A region of pixels in a radius around the target pixel is located in the image, then divided into two with a straight line. The orientation of this line is set at angle θ. The two halves of the circular region are then examined independently. Pixel intensity histograms are generated from each half and the χ^2 distance between the two histograms calculated, whereby the result is called the gradient magnitude.
The probability of a boundary (Pb) is then extended to the multiscale probability of a boundary (mPb) which uses 4 different channels (intensity, colour a, colour b and textons) and 8 different orientations θ to perform boundary probabilities. The texture channel (texton) is produced by convolving the input image with 17 Gaussian derivative filters. K-means clustering is used to gather the pixels. The resulting cluster assignments replace pixel intensity information to form a new image, called a texton image, which shows the strongest edges. mPb is performed by linearly combining the 4 cues:

$$mPb(x, y, \theta) = \sum_s \sum_i \alpha_{i,s} G_{i,\sigma(i,s)}(x, y, \theta) \quad 4.5$$

where s shows the scales, i indexes the 4 feature channels, α is a constant and $G(x, y, \theta)$ is the gradient magnitude. This algorithm can be compressed to take the maximum θ at any particular pixel. The mPb is therefore defined as:

$$mPb(x, y) = max_\theta\{ mPb(x, y, \theta) \} \quad 4.6$$

The spectral probability of a boundary (sPb) is coined from Pb (Arbeláez et al., 2011). It incorporates global image information to determine the strength of each contour. A radius of local pixels around a target pixel is examined. Pixels which have a strong contour as determined by Pb are considered:

$$W_{ij} = \exp(-max_{p \epsilon \overline{ij}}\{ mPb(p) \}/\rho) \quad 4.7$$

Where W is a sparse symmetry affinity matrix, ρ a constant and $\overline{ij}$ is the line segment connecting pixels i and j.

To factor global image information, $D_{ii} = \sum_j W_{ij}$ is defined and eigenvectors $\{v_0, v_1, \dots, v_n\}$ of equation $(D - W)v = \lambda Dv$ solved.

The solution is factored in the final formulation of sPb detector;

$$sPb(x, y, \theta) = \sum_{k=1}^{n} \left(1 \Big/ \sqrt{\lambda_k} \right) \nabla_\theta \, \mathrm{v}_k(x, y) \quad 4.8$$

Where, v_k is an eigenvector and $1 \Big/ \sqrt{\lambda_k}$ is the weighting based on the entire image.

Combining mPb and sPb results to the globalized contour detector(gPb), taking advantage of the strengths of these two detectors. mPb is capable of extracting many contours while sPb picks the most salient ones. gPb is a linear combination of mPb and sPb and is defined as;

$$gPb(x,y,\theta) = \sum_s \sum_i \beta_{i,s} G_{i,\sigma(i,s)}(x,y,\theta) + \gamma . sPb(x,y,\theta) \quad 4.9$$

Where β and γ are constants.

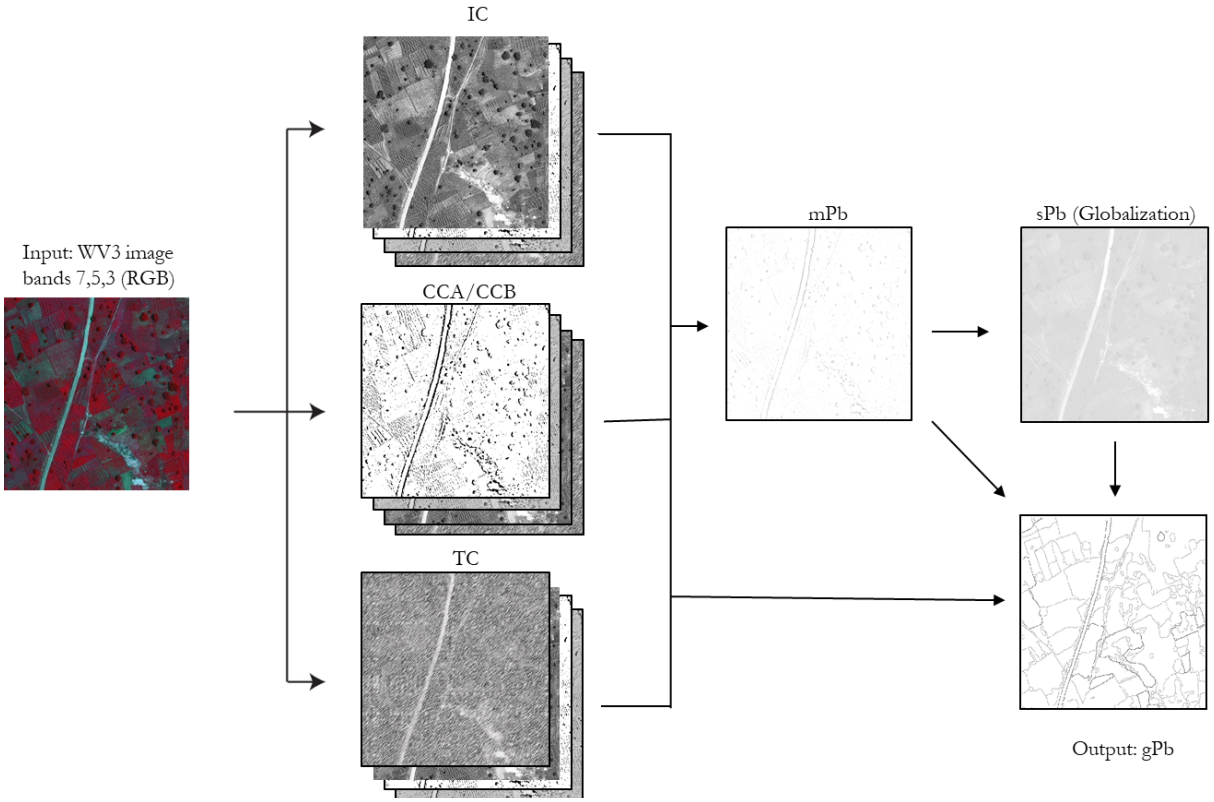


Figure 4.6: An illustration of the stages involved in gPb detector. The four channels used at different scales to calculate the probability of a boundary(mPb) are intensity channel (IC), colour a (CCA), colour b (CCB) and the texton channel (TC). The output of mPb is fed into sPb stage where they are combined into the gPb output.

After detection using the resulting gPb detector, the image is segmented using 1) an Oriented Watershed Transform (OWT), a variant of the watershed transform algorithm which generates regions from the oriented contours, and 2) Ultrametric Contour Map(UCM) which defines a hierarchical segmentation. OWT and UCM segmentation can be applied to the output of any contour detector. gPb-owt-ucm produces better results compared to other approaches like image segmentation (Mean Shift, Normalised Cuts) and edge detection (Prewitt, Sobel, Canny and Roberts) (Arbeláez et al., 2011). It is therefore referred to as a state-of-the-art contour detection method in computer vision (Zhang et al., 2013; Jevnisek & Avidan, 2016). In this thesis, we refer gPb-owt-ucm as gPb contour detection.

The publicly available open source algorithms from the Berkeley Segmentation Dataset and Benchmark by Arbeláez et al. (2013) were used for these experiments. Pansharpened images of WorldView3 prepared as explained in Section 3.3. We extract bands 7,5 and 3 to form an RGB composite. We reduce the dimensions of both tiles from 1000 x 1000 pixels to 800 x 800 pixels. These images are further sub-tiled into 4 tiles each to reduce computational cost.

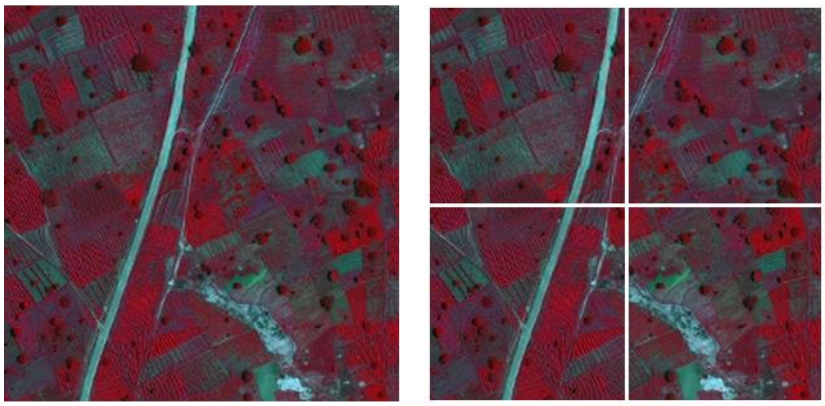


Figure 4.7: An illustration of TS2 divided into 4 sub-tiles

We apply gPb contour on each sub-tile, resulting in contour maps of each sub-tile. We then apply hierarchical image segmentation at a scale k, within the range [0, 1]. Each value of k generates different probabilities of detecting boundaries. The outputs in this stage are binary maps with 2 classes for boundary and non-boundary for each value of k tested.
The binary maps from each sub-tile are merged to form a single binary image of the same size as the input tile. This binary image is then vectorized and a 1m buffer applied to the detected boundary. This output is rasterised again. The result is a raster file with 2 classes, a boundary class with a uniform thickness of 4 pixels and the non-boundary class.

4.3.2. Canny detector

We implement the canny edge detector in 7 steps;

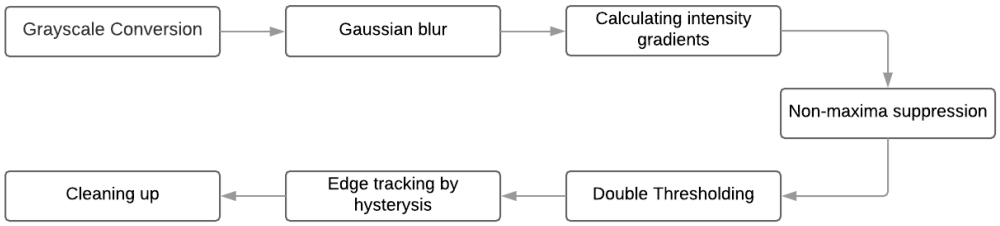


Figure 4.8: Procedure for canny edge detector

The input image is first converted to grayscale. Pixel intensities are 8 bits with a range between 0 to 255. We perform a gaussian blur to reduce undesired noise in the image. Presence of noise can lead to detection of false edges. We apply a 3x 3 filter and vary the standard deviation through trial and error to see where noise is reduced effectively.

$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Gradient magnitudes and directions are calculated at around every pixel in the image. A high magnitude of gradient means that pixel values are changing rapidly, therefore, implying the possibility that the pixel lies on a boundary. The direction of the gradient shows the edge orientation. We determine gradients using a Sobel filter.

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A \qquad \text{and} \qquad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

Where * denotes a convolution operation between the filter and the image A

The magnitude and angle of directional gradients are then calculated as;

$$|G| = \sqrt{G_x^2 + G_y^2} \qquad 4.10$$

$$\angle G = \arctan\left(\frac{G_y}{G_x}\right) \qquad 4.11$$

The image magnitude results in thick edges. Non-maximum suppression is performed to thin out the edges. It works by finding the pixel with maximum intensity value in an edge. Other pixels which do not meet this condition are set to zero.

Double thresholding removes noise further from the result of maximum suppression. Two thresholds are set; an upper threshold and a lower threshold. The upper threshold marks the most salient edges. Edge tracking hysteresis determines which weak edges are true edges. Weak edges connected to strong edges are classified as true edges. Weak edges which are not connected to strong edges are removed.

4.3.3. Multiresolution segmentation (MRS) in eCognition software

We used a technique called multiresolution segmentation in eCognition software version 9.3. Multiresolution segmentation is a bottom-up region-merging technique starting with one-pixel objects (Rejaur & Saha, 2008). This means starting with the seed pixels, followed by numerous subsequent iterations, small pixel objects are merged into larger ones.

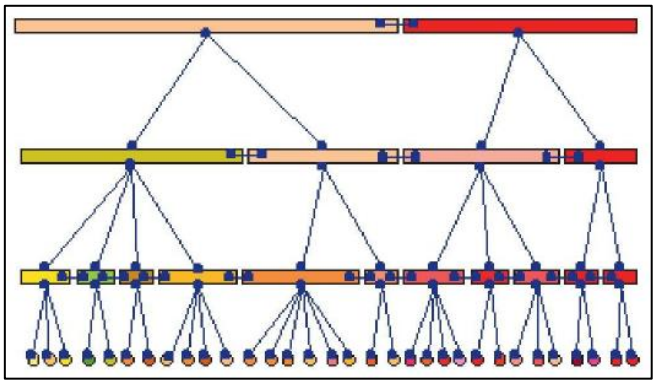


Figure 4.9: Hierarchical structure of image objects (Karakış et al., 2018)

Figure 4.9 shows a hierarchical frame and how the adjacent objects and the sub or super objects affect each other. The merging decision is based on local homogeneity criterion.

The results of multiresolution segmentation are determined by 3 main factors: the heterogeneity criteria or scale parameter, that determines the maximum allowed heterogeneity for the resulting segments, secondly, the weight of the colour and shape criteria in the segmentation process, and finally the weight of the

compactness and smoothness criteria. The optimal determination of these three often considered abstract parameters is not easy to carry out (Aguilar et al., 2016).

The clustering process in multiresolution segmentation is pairwise, whereby the underlying optimization procedure minimises the weighted heterogeneity Nh of resulting objects, where N is the size of the segment and h an arbitrary definition of heterogeneity (Karakis et al., 2004). The smallest growth of the defined heterogeneity can be described as the merging of adjacent image objects in each iteration. If this growth exceeds the defined scale parameter then the process stops. Multiresolution segmentation is a local optimization procedure.

Spectral or colour heterogeneity can be expressed as;

$$h = \sum_c w_c \sigma_c \quad 4.12$$

where heterogeneity is given by the sum of standard deviations of spectral values in each layer weighted with the weights.

Spectral and spatial heterogeneity is combined in order to reduce deviation from a crisp shape. Describing heterogeneity in terms of deviation from a crisp shape, the ratio of the border length l and the square root of the total number of pixels $\sqrt{n}$ forming this image object can be expressed as:

$$h = \frac{l}{\sqrt{N}} \quad 4.13$$

Furthermore, heterogeneity can be described as the ratio of the de facto border length l and the shortest possible border length b given by the bounding box of an image object parallel to the raster.

$$h = \frac{l}{b} \quad 4.14$$

More information about the mathematical formulation of multiresolution segmentation can be found in (Baatz & Schäpe, 2000; Tian & Chen, 2007)

4.4. Accuracy evaluation

We used the precision-recall framework to assess the performance of our proposed methodology on the selected dataset in comparison to the ground truth. Precision-recall is best suited for binary classification tasks with highly imbalanced classes (López et al., 2013; Hossin & Sulaiman, 2015). The non-boundary class had far much more pixels compared to boundary class. The boundary class in the reference dataset was prepared with a uniform width of 4 pixels or 2m GSD for both training and testing tiles. Section 3.4 describes how the reference data was prepared. p, r, F, α, β denotes precision, recall, F-score, false positive rate (type I error) and false negative rate (type II error) respectively. TP, FP, FN denotes the total number of true positives, false positives and false negatives, respectively.

$$\mathrm{p} = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \quad 4.15$$

$$\mathrm{r} = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \quad 4.16$$

$$\mathrm{F} = 2 \times \frac{\mathrm{p} \times \mathrm{r}}{\mathrm{p+r}} \quad 4.17$$

$$\alpha = \frac{\mathrm{FP}}{\mathrm{FP + TN}} \quad 4.18$$

$$\beta = \frac{\mathrm{FN}}{\mathrm{TP + FN}} \quad 4.19$$

True Positives are the boundary pixels predicted correctly by the algorithms i.e. the hits. True Negatives are the correct rejections, i.e. non-boundary pixels predicted correctly. False Positives are the false alarms or type I error i.e., non-boundary pixels predicted as boundary pixels. False Negatives are the miss or type II error, i.e. The boundary pixels predicted as non-boundary pixels. Precision gives an indication of the correctly detected boundaries in relation to the total boundaries detected. Precision can be thought of as a measure of exactness. A low precision indicates that there is a large number of false positives. Recall gives the ratio of correctly detected boundaries to the total number of boundaries in the reference. Recall can be thought of as a measure of completeness. A low recall indicates many false negatives. The F-score also referred to as the F-measure gives the harmonic mean of precision and recall. F- score is the more preferred measure to avoid treating precision and recall separately since it is possible to increase the value of one at the expense of the other.

5. RESULTS AND ANALYSIS

Findings from the experiments done in Chapter 4 are reported in this chapter. We begin by presenting the results of hyperparameter sensitivity analysis in section 5.1 which involve varying the patch size and sample size, varying the depth of the networks, experiments with pooling and without pooling layers, increasing the amount of training data and augmenting the training data. Results from boundary detection alternatives are shown in section 5.3 and finally a comparison of all the methods used is presented in section 5.4.

5.1. Hyper-parameter sensitivity analysis

We begin this section by presenting the results of the initial experiments. These experiments were done on a small dataset to minimise computational cost as explained in section 4.1.2. Hyperparameter sensitivity analysis experiments involved varying the patch size, number of training samples and number of convolutional layers. Training and validation sets were extracted from tile TR1 while evaluation of the algorithms was done on testing tile TS2 for the initial experiments.

The strategy employed in fine-tuning was varying a single parameter while the rest were held constant. The first parameter to be investigated was the patch size. FCN-DK3 was trained with stochastic gradient descent with a momentum of 0.9 and 1000 patch samples. The random sampling strategy was used to build training and validation sets. A sample represents a patch. The channels were normalized in the range [0,1]. The objective function was observed to plateau sufficiently at iteration 200. The first 170 epochs were trained with a learning rate of 1.0×10^{-3} and the remaining 30 epochs with a learning rate of 1.0×10^{-4}. **Error! Reference source not found.** shows the values of the learning parameters used in experiments on FCN-DK3.

Table 5.1: Learning hyperparameters for experiments on network FCN-DK3

Hyper-parameter	Values
Number of epochs	Step1 170
	Step2 200
Learning rate	Step1 - 1.0×10^{-3}
	Step2 - 1.0×10^{-4}
Momentum	0.9
Weight decay	0.0005

5.1.1. Patch size experiments

The patch size defines the extent where contextual information is considered when assigning labels an entire patch. Table 5.2 presents the results showing the effect of variation of the patch size fed into network FCN-DK3. From this table, we observe that patch size 95×95 produces the best accuracies and the lowest errors among the different experiments on the patch sizes. Increasing the size of the patch led to an improved F-score up to a maximum of 0.171 and 0.130 for training tile TR1 and testing tile TS2 respectively. Increasing the patch size from 95×95 to 115×115 led to a drop in F-score by 0.074 for TR1 and 0.049 for TS2. This is despite the expectation that the F-score would increase since a bigger patch size provides for a bigger window for more contextual information.

Table 5.2: Patch size experiments for FCN-DK3

Tiles	Patch size	Type I error	Type II error	Precision	Recall	F-score
TR1	**25 x 25**	0.007	0.942	0.552	0.058	0.105
	75 x 75	0.009	0.922	0.561	0.079	0.137
	95 x 95	0.01	0.9	0.58	0.101	0.171
	115 x 115	0.006	0.947	0.548	0.053	0.097
TS2	**25 x 25**	0.005	0.975	0.375	0.025	0.047
	75 x 75	0.014	0.928	0.387	0.072	0.121
	95 x 95	0.014	0.923	0.414	0.077	0.130
	115x 115	0.007	0.956	0.442	0.044	0.081

A patch size of 95 x 95 was therefore fixed to tune the other parameters and was used in the final implementation. Figure 5.1**Error! Reference source not found.** graphically shows the trend generated by varying the patch sizes, and Figure 5.2 shows the output maps from these experiments.

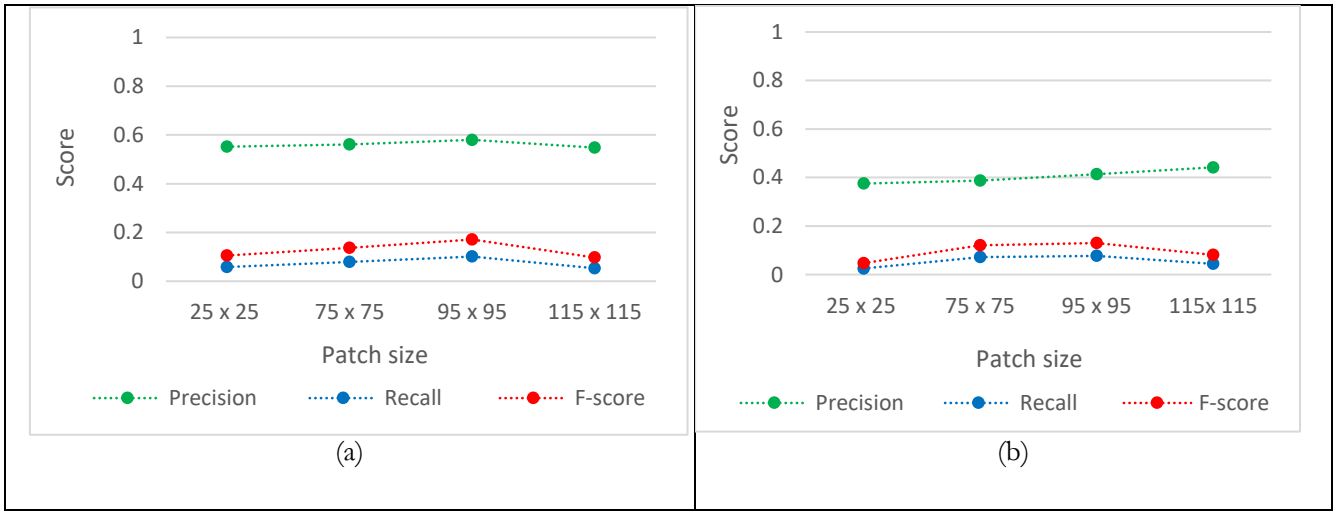


Figure 5.1: **(a)** and **(b)** are the results showing the effects on the accuracies for training tile TR1 and testing tile TS2 respectively upon variation of the size of patch for network FCN-DK3.

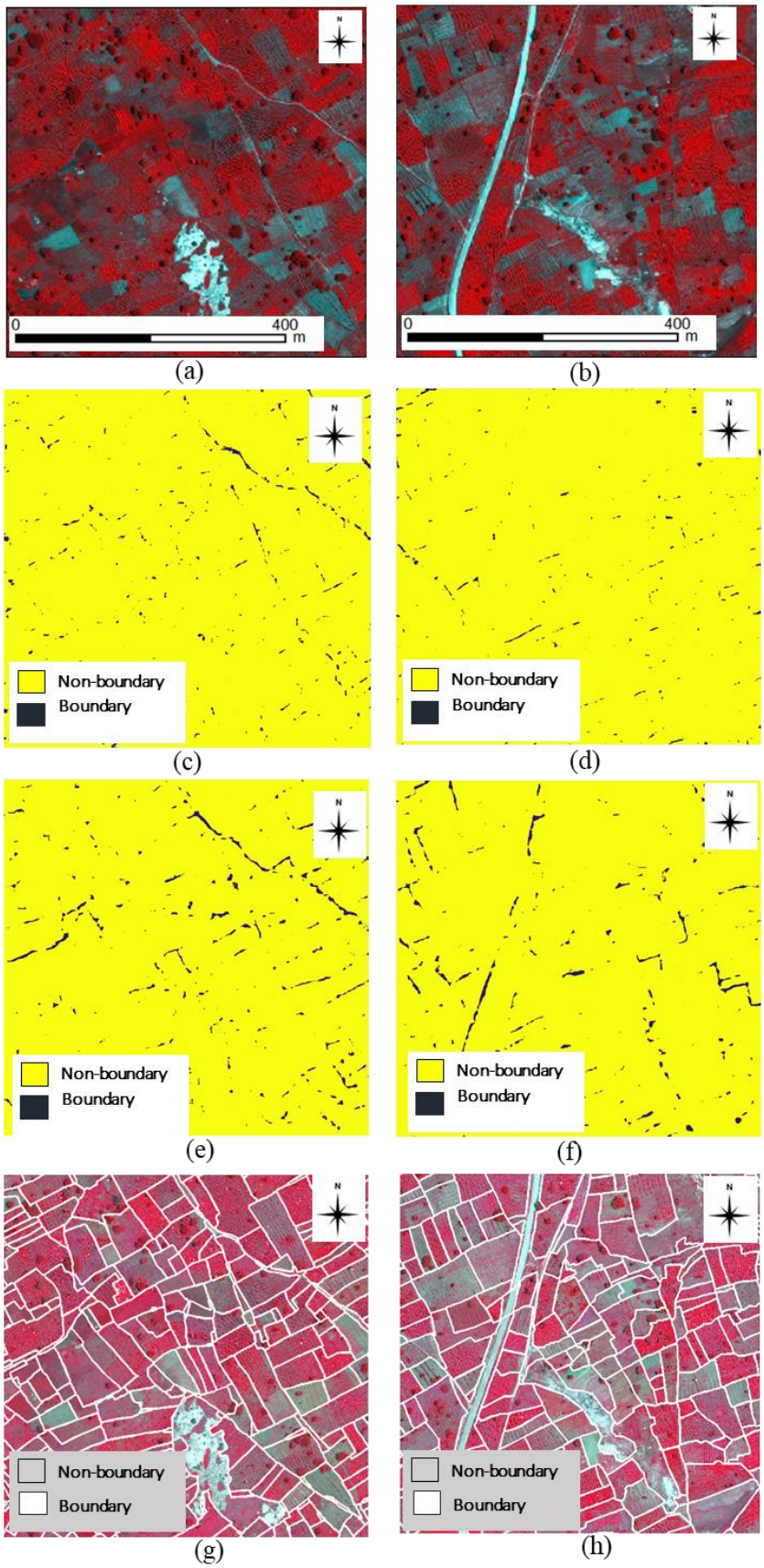


Figure 5.2: A comparison between detected boundaries using a patch size of **(c, d)** 25×25 and **(e, f)** 95×95 for training tile **(a, c, e, g)** TR1 and **(b, d, f, h)** testing tile TS2. **(a, b)** are the raw WV3 imagery (bands 7,5,3) and **(g, h)** are the ground truths.

5.1.2. Number of patch samples

We experimented with different sizes of training sets (500, 1000, 2000, 4000 and 6000) extracted from the training tile TR1 through random sampling. A small number of patch samples presents insufficient training data in the network. Table 5.3 shows that increasing the sample size improves precision, recall and F-score. The 2 types of errors are reduced significantly from a sample size of 500 to 6000. There is a slight improvement, however, between 4000 and 6000 samples.

Table 5.3: Results for experiments on sample size variation; Network FCN-DK3.

Tiles	No. of patches	Type I error	Type II error	Precision	Recall	F-score
TR1	500	0.002	0.990	0.425	0.010	0.020
	1000	0.010	0.900	0.580	0.100	0.171
	2000	0.030	0.685	0.596	0.3152	0.412
	4000	0.039	0.484	0.651	0.516	0.576
	6000	0.039	0.428	0.672	0.572	0.618
TS2	500	0.001	0.994	0.367	0.006	0.013
	1000	0.014	0.923	0.414	0.077	0.130
	2000	0.028	0.863	0.378	0.137	0.201
	4000	0.062	0.755	0.332	0.245	0.282
	6000	0.062	0.766	0.321	0.234	0.291

Training time increases from about 40 minutes using 500 samples to about $12\frac{1}{2}$ hours using 6000 samples. FCNs are computationally very expensive. It is for this reason that the next hyperparameter optimisation experiments we use 2000 samples which is a compromise between accuracy and cost. Other strategies of increasing accuracies are investigated while considering cost. In the final implementation, we use 6000 patch samples and compare with the other strategies employed. Figure 5.3. is a graphical visualization of the accuracies produced by sample size experiments and the output maps are presented in Figure 5.4.

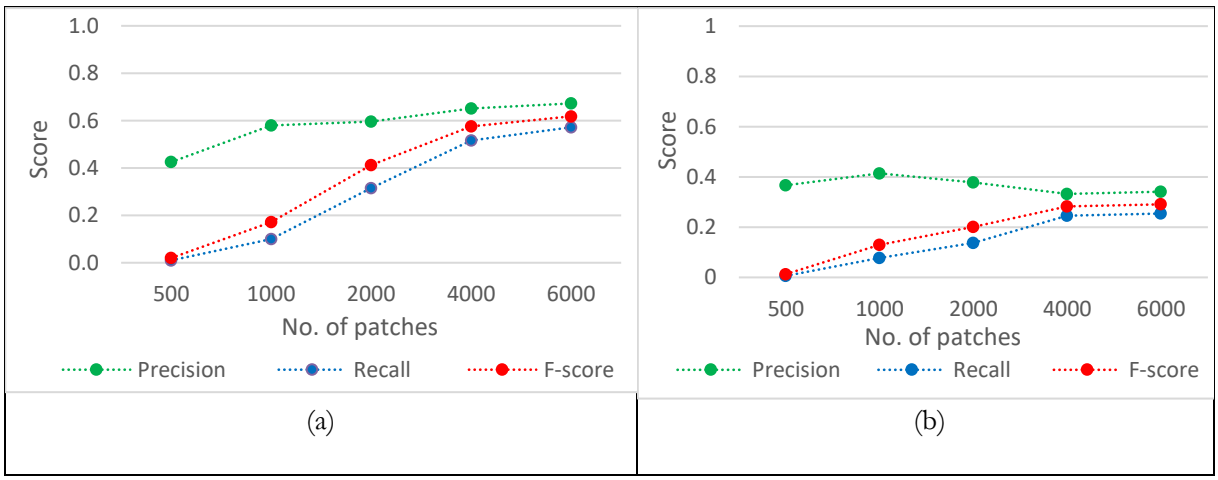


Figure 5.3: Accuracies for **(a)** training tile TR1 and **(b)** testing tile TS2 produced by the variation of the number of patch samples on network FCN-DK3.

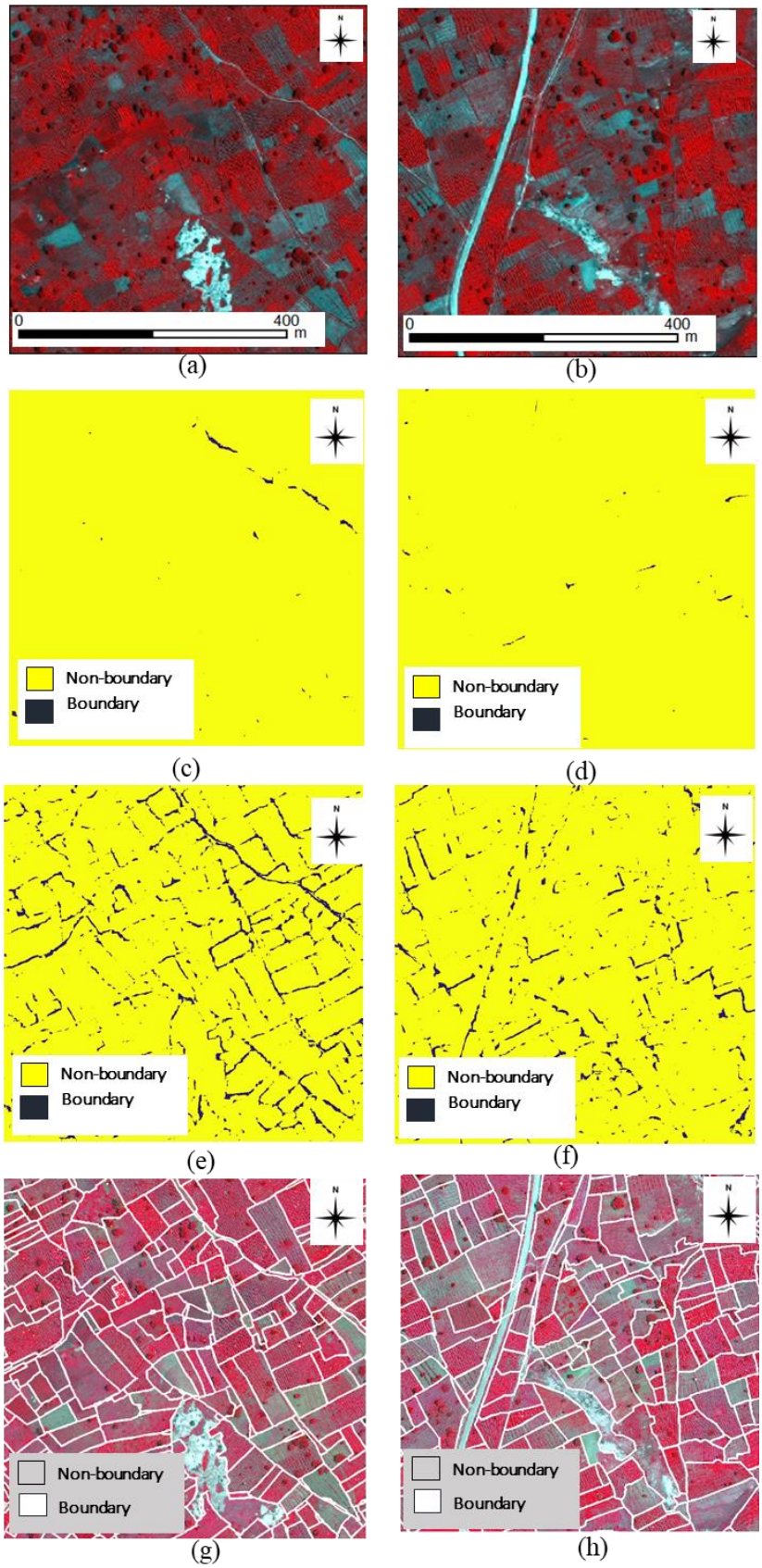


Figure 5.4: Output maps for sample size experiments for FCN-DK3: **(a, c, e, g)** training tile TR1 and **(b, d, f, h)** testing tile TS2. **(c, d)** are maps for patch size 500, **(e, f)** maps for patch size 2000, **(a, b)** are the raw input WV3 satellite images and **(e, f)** the ground truths

5.1.3. Varying the depth of FCN-DK6

Network depth refers to the number of successive computational layers from the input to the output layer. We varied the depth of the networks, keeping the patch size fixed as 95 × 95 and a constant number of patch samples as 2000. Table 5.4 shows the results of obtained by these experiments. FCN-DK6 produces the highest precision, recall, F-score and the lowest type I and II errors compared to the other networks. These experiments show that the depth of a network is crucial for its success. The challenge with deep networks is however the cost of training them. Training FCN-DK6 took around 8 hours and 20 minutes while FCN-DK2 took around 2 hours. From the depth of 4 convolutional layers, results appear to change by a small margin till the 6th convolutional layer, which was the final depth considered in our experiments. Experiments on depth of the network are repeated with the other experiments preceded by this set using a different variant of the DKs proposed in this research to investigate the generalization capability of all the networks at different levels.

Table 5.4: Results for experiments on the depth of the networks FCN-DK2 to FCN-DK6

Tiles	Layers	Type I error	Type II error	Precision	Recall	F-score
TR1	DK2	0.006	0.940	0.570	0.060	0.108
	DK3	0.030	0.685	0.596	0.315	0.412
	DK4	0.029	0.301	0.769	0.699	0.732
	DK5	0.024	0.273	0.806	0.727	0.765
	DK6	0.024	0.242	0.811	0.758	0.784
TS2	DK2	0.012	0.954	0.327	0.046	0.080
	DK3	0.028	0.863	0.378	0.137	0.201
	DK4	0.057	0.768	0.339	0.232	0.275
	DK5	0.049	0.786	0.356	0.214	0.268
	DK6	0.050	0.785	0.357	0.215	0.269

Figure 5.5 shows the trend generated by the accuracies as the network depth was increased from 2 to 6 convolutional layers.

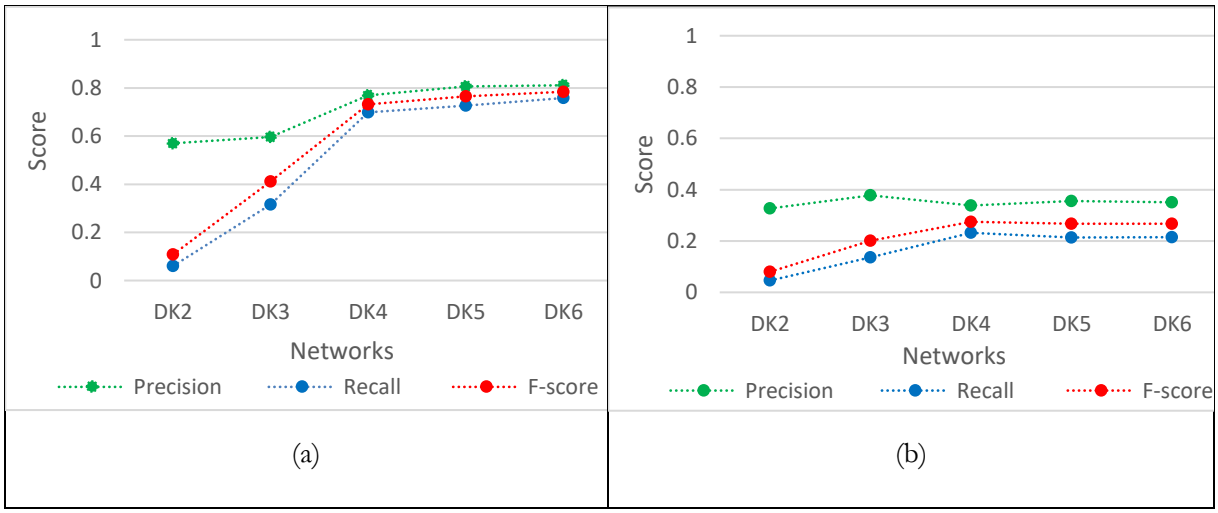


Figure 5.5: Accuracies for **(a)** training tile TR1 and **(b)** testing tile TS2 produced by the variation of the depth of the networks FCN-DK2 to FCN-DK6

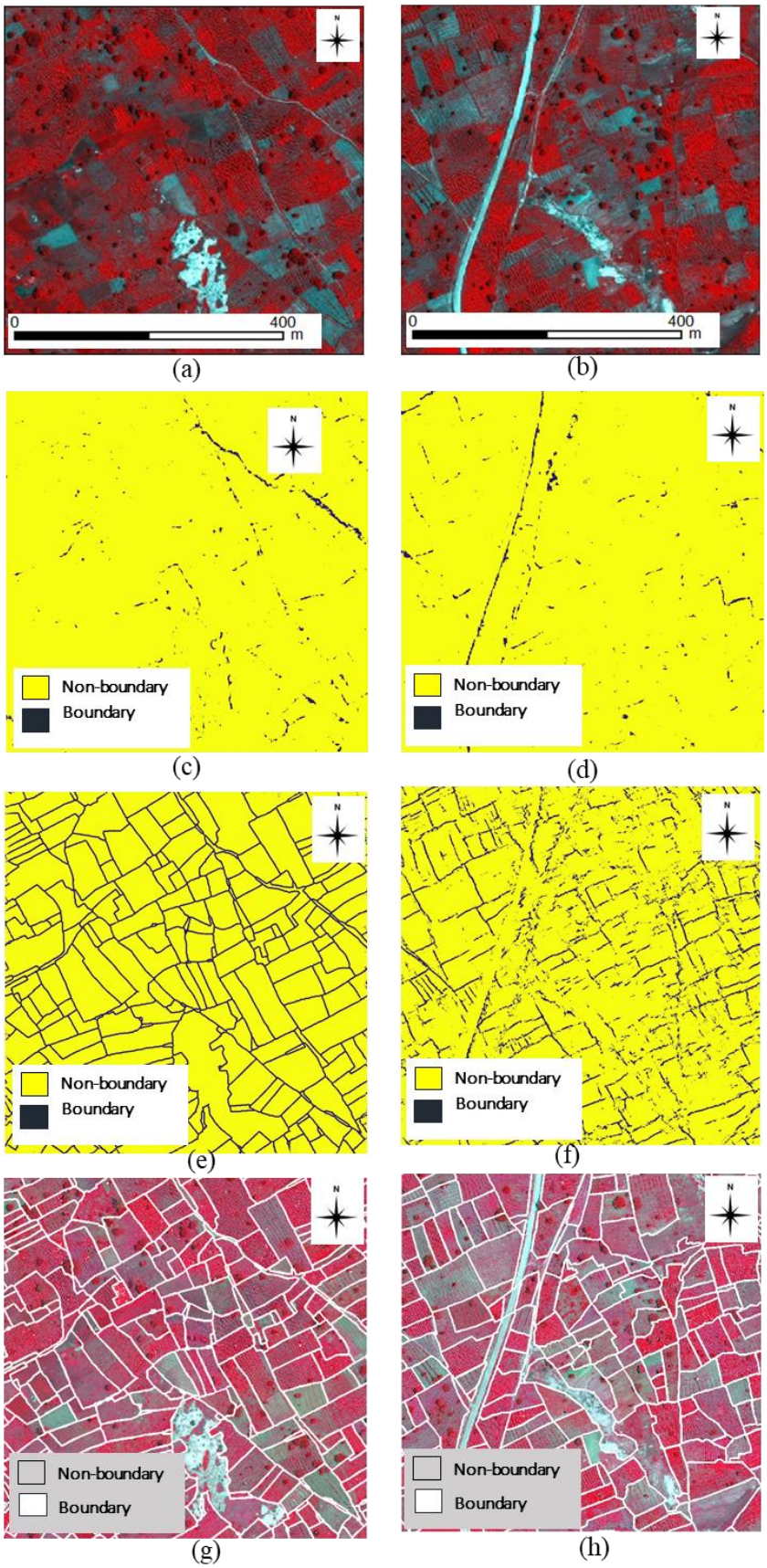


Figure 5.6: Output maps for network depth experiments for **(c, d)** FCN-DK2 and **(e, f)** FCN-DK6: **(a, c, e, g)** are results from training tile TR1 and **(b, d, f, h)** from testing tile TS2. **(a, b)** are the raw input WV3 images and **(g, h)** are the ground truths.

5.1.4. Discarding the pooling layers; FCN-DKs vs FCN-DKConvs

We aimed for a more efficient network which is made up of convolutional layers as the only heavy computational blocks in the networks. As explained in Section 4.1.2, computation in convolutional layers involves convolutions between a bank of filters and the input image. Batch normalization was applied immediately after the convolutional layers followed by Leaky Relu non-linearities. The original FCN-DKs had max pooling layers inserted after the non-linearities. Max pooling serves to reduce the number of parameters by downsampling the inputs which enables detection of features to be scale invariant. Another benefit of max pooling is to control overfitting. Although pooling layers have that benefit, it comes at a computational cost. The adopted FCN-DKs were designed to output feature maps equal to the input image. Upsampling modules were not required. This made investigations on the pooling layers necessary to establish their significance in the networks. Experiments involved dropping a few pooling layers and eventually dropping them all. This experiment formed the basis of our new variant of FCN-DKs which we called FCN-DKConvs. Table 4.1 illustrates the parameters of this new variant of the DKs. Figure 5.7 explores outputs from raw predicted labels from both networks. The raw output scores were normalized in the range [0 ,1] to produce boundary probability maps.

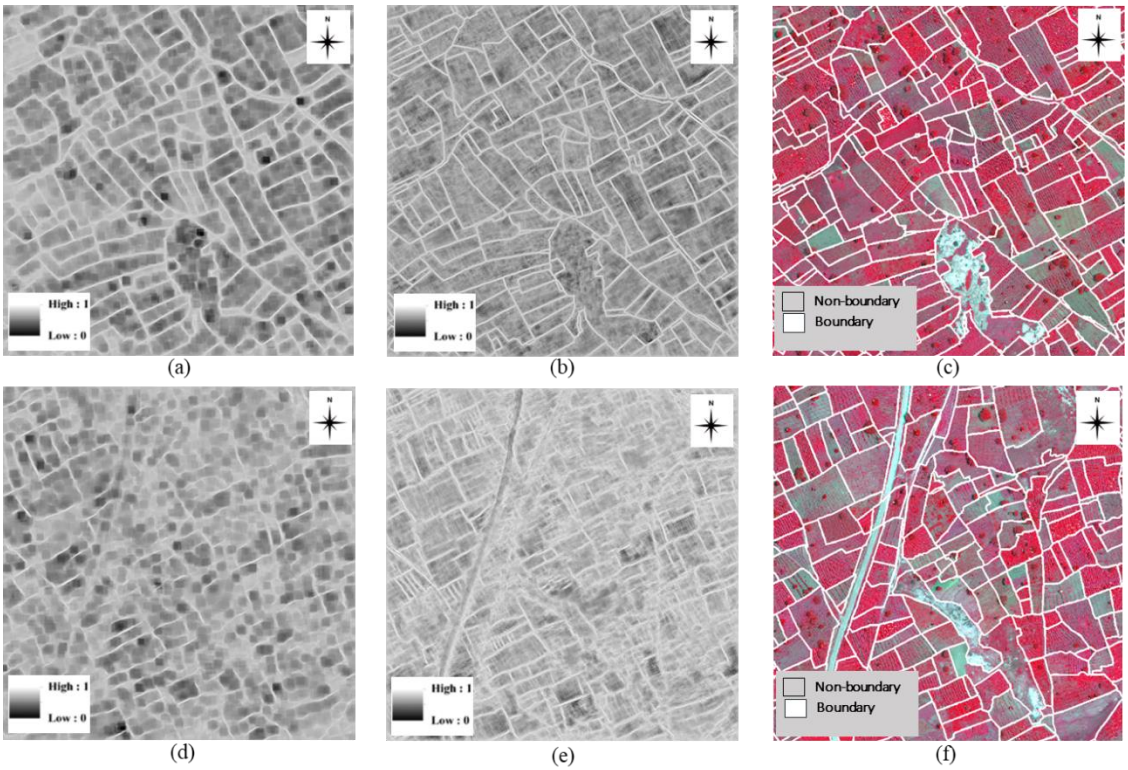


Figure 5.7: Boundary probability maps of **(a, d)** FCN-DK6 vs **(b, e)** FCN-DKConv6. **(a, b, c)** are training tiles TR1 and **(d, e, f)** are testing tiles TS2 and **(c, f)** are the ground truths

Table 5.4 shows the results obtained from experiments on networks FCN-DK-Conv1 – 6. Figure 5.7 shows a comparison of the two series of networks. FCN-DKConvs outperformed FCN-DKs on both training TR1 and testing TS2 results. Unlike FCN-DKs which show optimal performance on testing tile TS2 at the 4th convolutional layer network, FCN-DKConvs perform optimally at the 6th convolutional layers.

Table 5.5: TS1 and TR2 results for FCN-DKConv1 to FCN-DKConv6

Tiles	Networks	Type I error	Type II error	Precision	Recall	F-score
TR1	DKConv2	0.015	0.811	0.645	0.189	0.292
	DKConv3	0.017	0.32	0.85	0.68	0.756
	DKConv4	0.012	0.126	0.909	0.874	0.891
	DKConv5	0.004	0.028	0.976	0.972	0.974
	DKConv6	0.004	0.027	0.975	0.975	0.975
TS2	DKConv2	0.026	0.882	0.358	0.118	0.177
	DKConv3	0.053	0.774	0.35	0.226	0.275
	DKConv4	0.056	0.744	0.365	0.256	0.301
	DKConv5	0.062	0.712	0.365	0.288	0.322
	DKConv6	0.068	0.699	0.358	0.301	0.327

Figure 5.8 shows a comparison of the two series of networks. FCN-DKConvs outperformed FCN-DKs on both training TR1 and testing TS2 results.

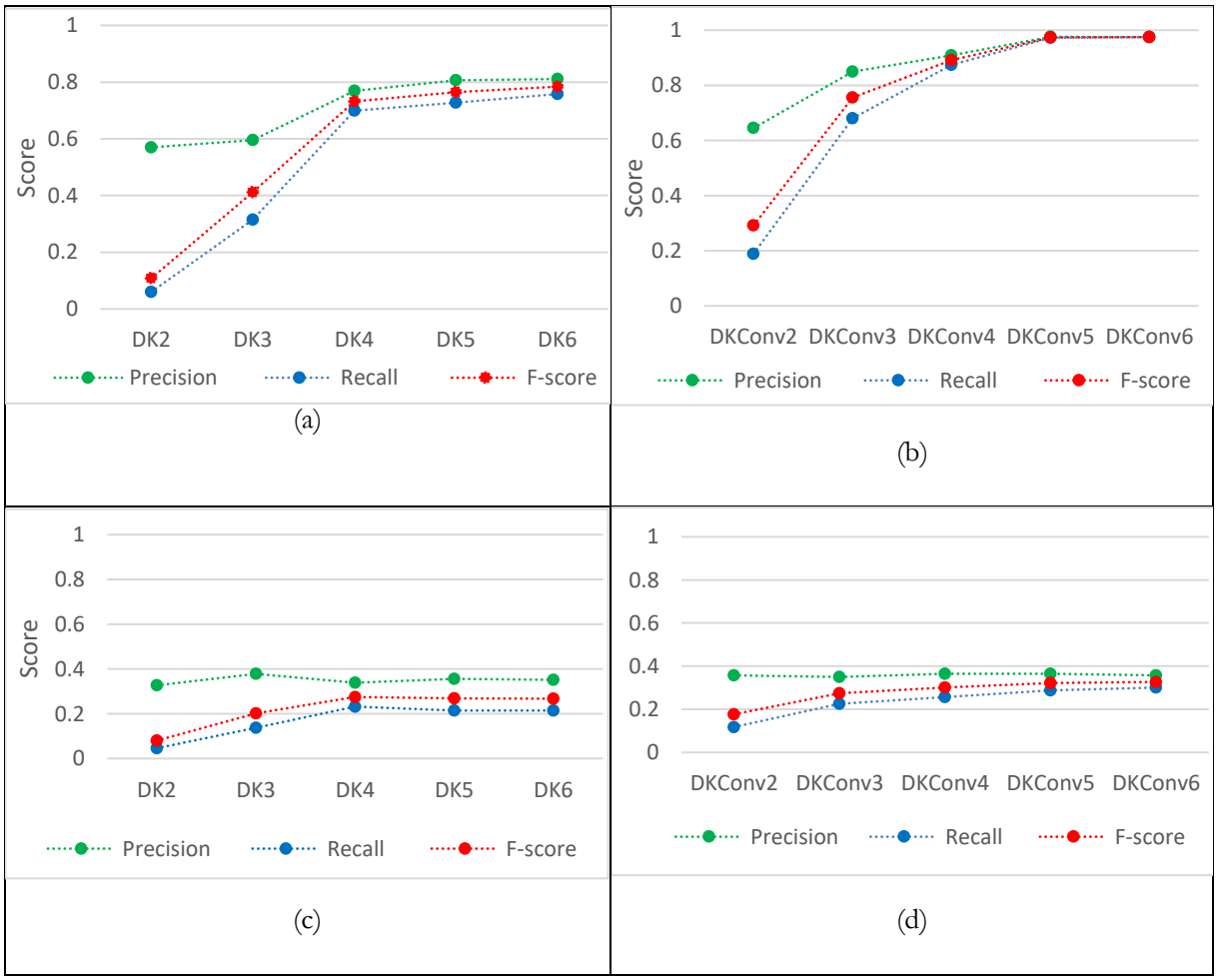


Figure 5.8: Side by side comparison of the accuracies of FCN-DKs vs FCN-DKConvs: **(a, b)** are results for training tile TR1 and **(c, d)** testing tile TS2. **(a, c)** are results for networks DK2 -DK6 and **(b, d)** for DKConv1 -DKConv6.

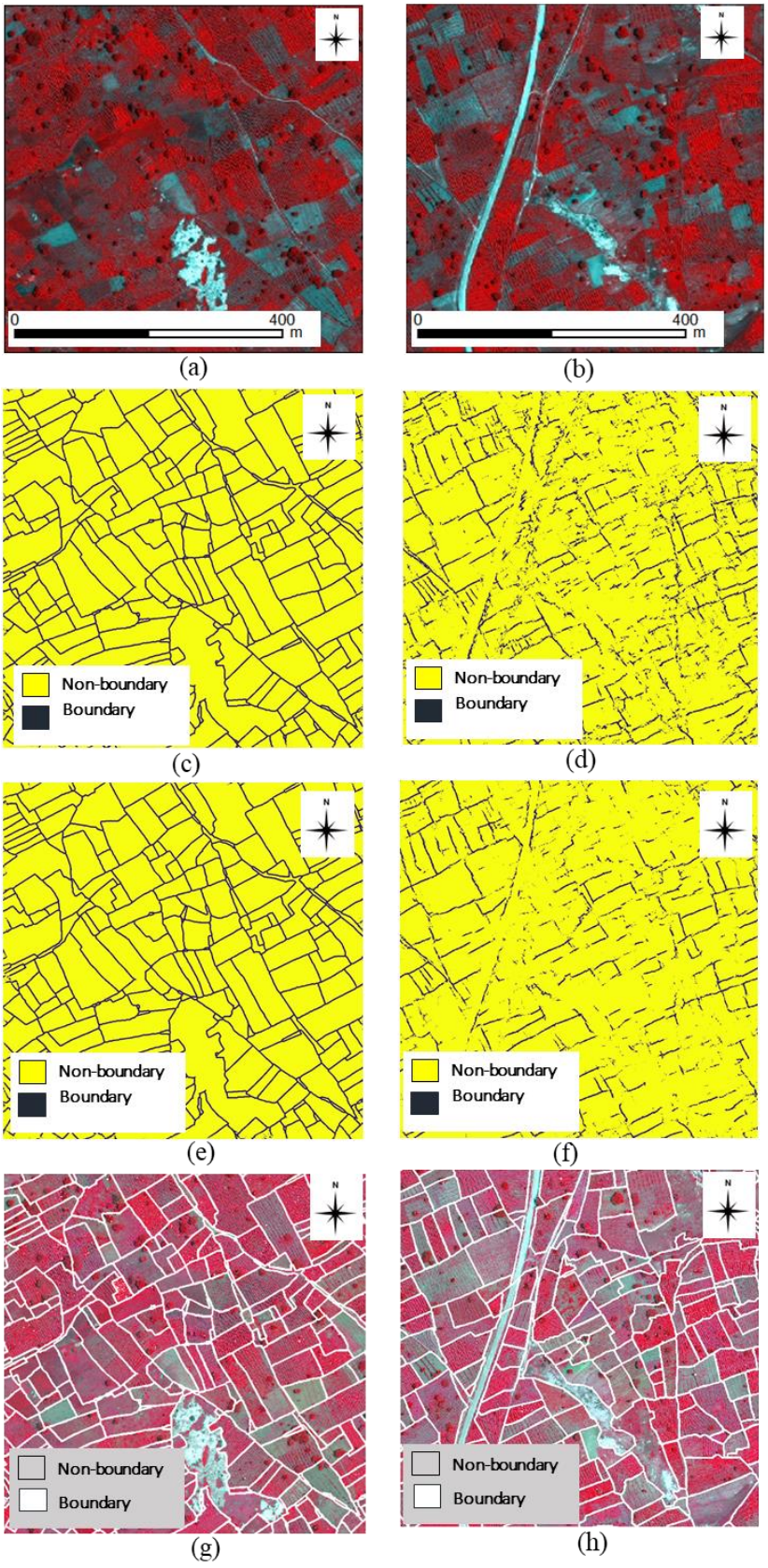


Figure 5.9: Outputs maps for **(c, d)** FCN-DK6 vs **(c, d)** FCN-DKConv6. **(a, c, e, g)** are from training tile TR1 and **(b, d, f, h)** testing tile TS2. **(a, b)** are the raw WV3 input images and **(e, f)** are the ground truths.

5.1.5. Training FCN-DKConvs on 1 tile Vs training on the full dataset -3 tiles

Figure 5.8 (b, d) illustrates results obtained from training FCN-DKConv6 using 1 tile, TR1. We rolled out the whole series of the networks, adding more training tiles (TR1, TR3 and TR4) and added an additional testing tile, TS5 in addition to TS2. This set of experiments was carried out to investigate the effect of adding even more of training data. 2000 randomly sampled patches were picked from each of the training tiles. The results obtained are as shown in Table 5.6.

Table 5.6: Results of training networks FCN-DKConv2 to FCN-DKConv6 on the full dataset

Tiles	Networks	Type I error	Type II error	Precision	Recall	F-score
TR1	DKConv2	0.015	0.862	0.568	0.138	0.222
	DKConv3	0.024	0.577	0.715	0.423	0.531
	DKConv4	0.024	0.223	0.820	0.777	0.798
	DKConv5	0.015	0.122	0.889	0.878	0.883
	DKConv6	0.010	0.069	0.926	0.931	0.929
TS2	DKConv2	0.020	0.886	0.407	0.114	0.178
	DKConv3	0.038	0.785	0.416	0.215	0.283
	DKConv4	0.060	0.706	0.378	0.294	0.331
	DKConv5	0.061	0.689	0.389	0.310	0.345
	DKConv6	0.060	0.686	0.394	0.314	0.349
TR3	DKConv2	0.005	0.940	0.514	0.060	0.107
	DKConv3	0.013	0.716	0.675	0.284	0.400
	DKConv4	0.017	0.256	0.812	0.744	0.777
	DKConv5	0.012	0.128	0.880	0.872	0.876
	DKConv6	0.007	0.072	0.925	0.927	0.926
TR4	DKConv2	0.010	0.908	0.566	0.092	0.159
	DKConv3	0.020	0.689	0.682	0.311	0.427
	DKConv4	0.024	0.267	0.804	0.733	0.766
	DKConv5	0.016	0.138	0.880	0.861	0.870
	DKConv6	0.010	0.071	0.928	0.928	0.928
TS5	DKConv2	0.011	0.95	0.363	0.05	0.088
	DKConv3	0.036	0.794	0.413	0.206	0.275
	DKConv4	0.062	0.702	0.371	0.298	0.330
	DKConv5	0.058	0.716	0.378	0.284	0.323
	DKConv6	0.057	0.709	0.385	0.291	0.331

Testing tile TS2 is picked for comparison of the results because the tile has been used in both experiments. Accuracies improve significantly using more training tiles and errors are reduced as illustrated in Figure 5.8 (b, d) illustrates results obtained from training FCN-DKConv6 using 1 tile, TR1. We rolled out the whole series of the networks, adding more training tiles (TR1, TR3 and TR4) and added an additional testing tile,

TS5 in addition to TS2. This set of experiments was carried out to investigate the effect of adding even more of training data. 2000 randomly sampled patches were picked from each of the training tiles. The results obtained are as shown in Table 5.6.The F-score for DKCOnv6 increases by 0.022, precision increases by 0.036, recall core increases by 0.013, type II error is reduced by 0.014 and type I error is reduced by 0.01.

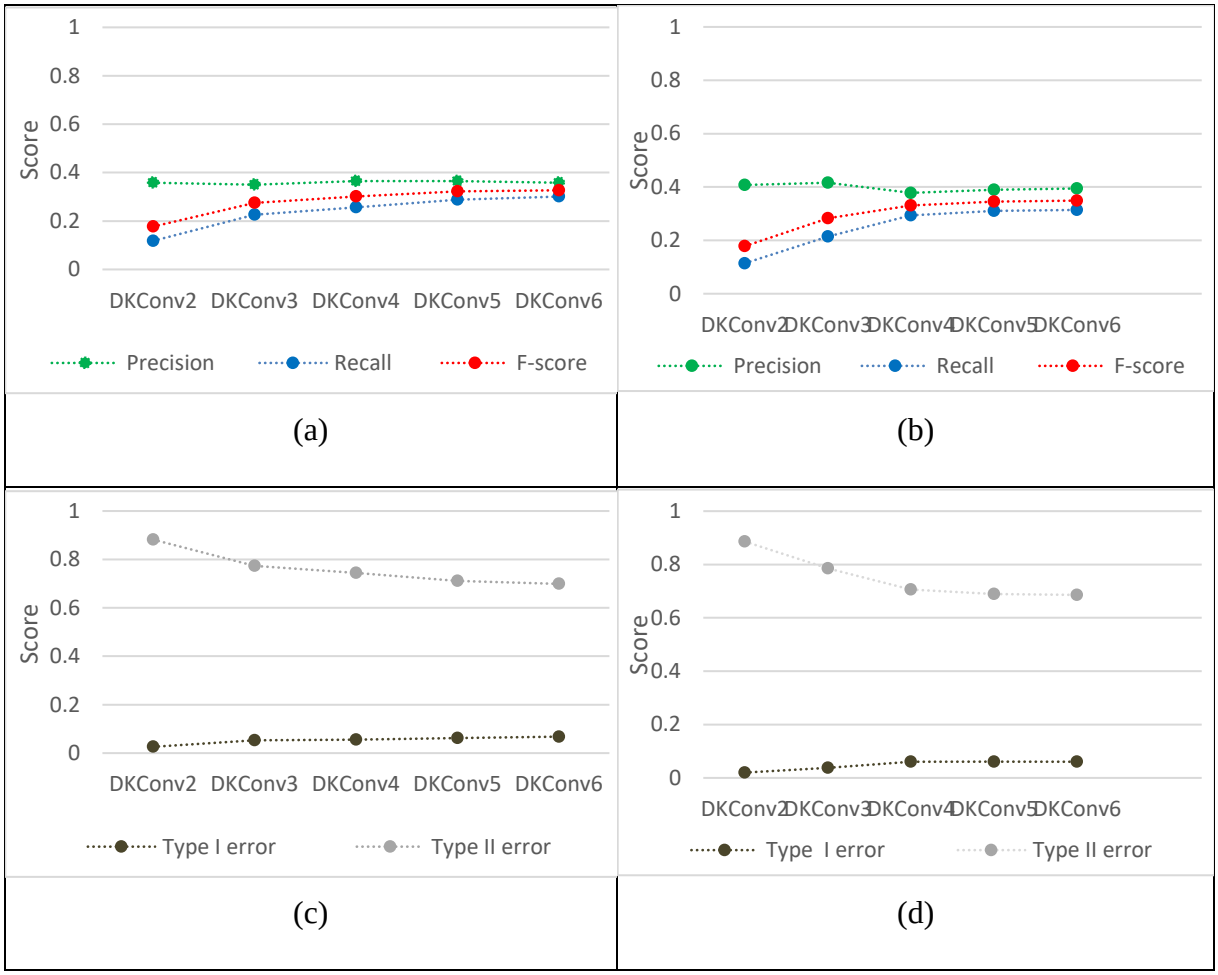


Figure 5.10: Results from testing tile TS2: **(a, c)** DKConvs series trained on tile TR1 only and **(b, d)** DKConvs series trained on tiles TR1, TR3 and TR4. **(a, b)** shows the obtained accuracies while **(c, d)** shows their corresponding errors.

5.1.6. Augmented training set Vs non-augmented

We trained the networks with augmented datasets to increase rotational invariance. 2000 random patches were picked from each tile then rotated by 90^0 then 180^0. Each tile, therefore, generated 6000 samples at 3 different orientations. Results from training set TR1 and testing sets TS2 and TS5 were picked for comparisons with the previous experiments on non-augmented data. Table 5.7 shows that there is an improved accuracy in TR1 from an F-score of 0.929 to 0.934. TS2 records an improved F-score as well from 0.349 to 0.358. Unexpectedly, TS5 drops its F-score from 0.331 to 0.322. Figure 5.11 show the trend generated by the accuracies of the results of augmented dataset experiments.

Table 5.7: Results of FCN-DKConvs trained using 3 tiles with patches rotated at 3 different orientations.

Tiles	Layers	Type I error	Type II error	Precision	Recall	F-score
TR1	DKConv2	0.008	0.905	0.611	0.095	0.165
	DKConv3	0.022	0.663	0.684	0.337	0.451
	DKConv4	0.024	0.259	0.814	0.741	0.776
	DKConv5	0.014	0.096	0.903	0.904	0.903
	DKConv6	0.010	0.063	0.930	0.937	0.934
TS2	DKConv2	0.016	0.922	0.379	0.078	0.129
	DKConv3	0.037	0.806	0.396	0.194	0.261
	DKConv4	0.071	0.684	0.361	0.316	0.337
	DKConv5	0.071	0.677	0.365	0.323	0.343
	DKConv6	0.073	0.655	0.372	0.345	0.358
TS5	DKConv2	0.009	0.956	0.370	0.044	0.078
	DKConv3	0.034	0.817	0.401	0.183	0.251
	DKConv4	0.068	0.722	0.336	0.278	0.304
	DKConv5	0.064	0.722	0.350	0.278	0.310
	DKConv6	0.068	0.703	0.351	0.297	0.322

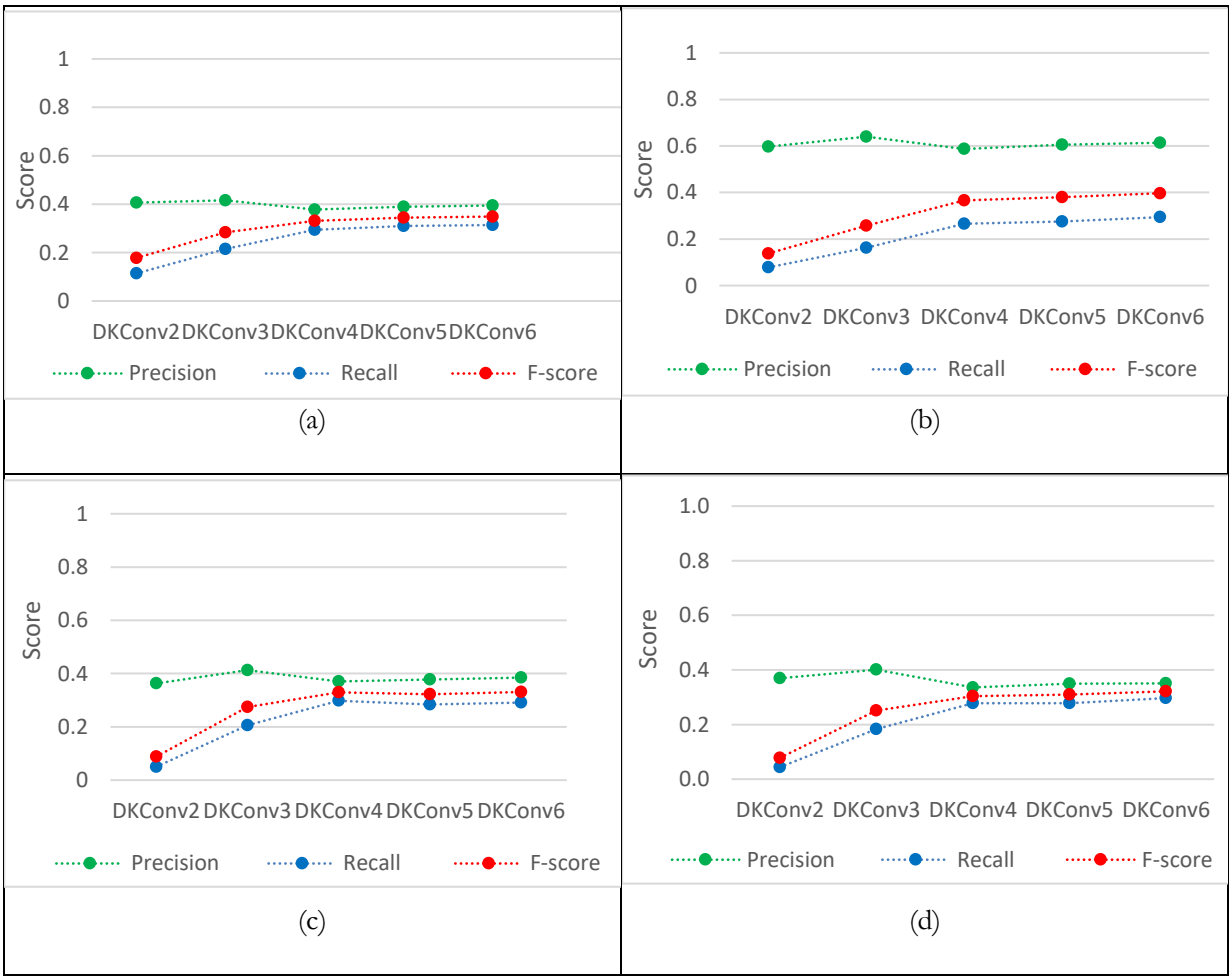


Figure 5.11: Side by side comparison of the accuracies of networks FCN-DKConv2 to FCN-DKConv6. **(a, b)** are results from testing tile TS2 and **(c, d)** are from testing tile TS5. **(a, c)** are results for networks trained using non-augmented data while **(b, d)** are for networks trained using augmented data.

5.2. Final implementation

In section 5.1 we have shown how we investigated the various components of our proposed approach. Experiments involved varying the patch size and sample size, varying the depth of the networks, experiments with pooling and without pooling layers, increasing the amount of training data by increasing the number of training tiles and finally augmenting the training data. We chose the best performing experiment for our final implementation. Experiment 5.1.5 emerged as the best implementation. As explained in Section 5.1.2, the costs involved in training the networks made it impossible to do all experiments using the best performing number of patch samples. In the final implementation we consider the 6000 patch samples. This experiment investigated the effect of increasing the number of training tiles from 1 to 3 tiles. Increasing the depth of the networks was also investigated. In terms of network depth, network FCN-DKConv6 produced the best results, therefore, we picked this network for our final implementation and for comparison with other alternative methods. To begin with, we show the results of this network in Table 5.8.

Table 5.8: Final results for network FCN-DKConv6

Tiles	Type I error	Type II error	Precision	Recall	F-score
TR1	0.008	0.052	0.944	0.948	0.946
TS2	0.057	0.688	0.407	0.312	0.353
TR3	0.006	0.051	0.941	0.949	0.945
TR4	0.007	0.053	0.945	0.947	0.946
TS5	0.056	0.712	0.389	0.288	0.331

Training tiles produce a near perfect performance with an average F-score of 0.946, a very low type I error averaging 0.007 and type II error of 0.052. Testing tile TS2 produces a better F-score of 0.353 than TS5 which produces 0.331. this is because TS5 is a more difficult test tile. A comparison with other competing approaches is based on the performance of this network on the two testing tiles. This provides for an unbiased evaluation of the network with regard to its generalization capability. Output maps from the network are shown in Figure 5.12 and Figure 5.13.

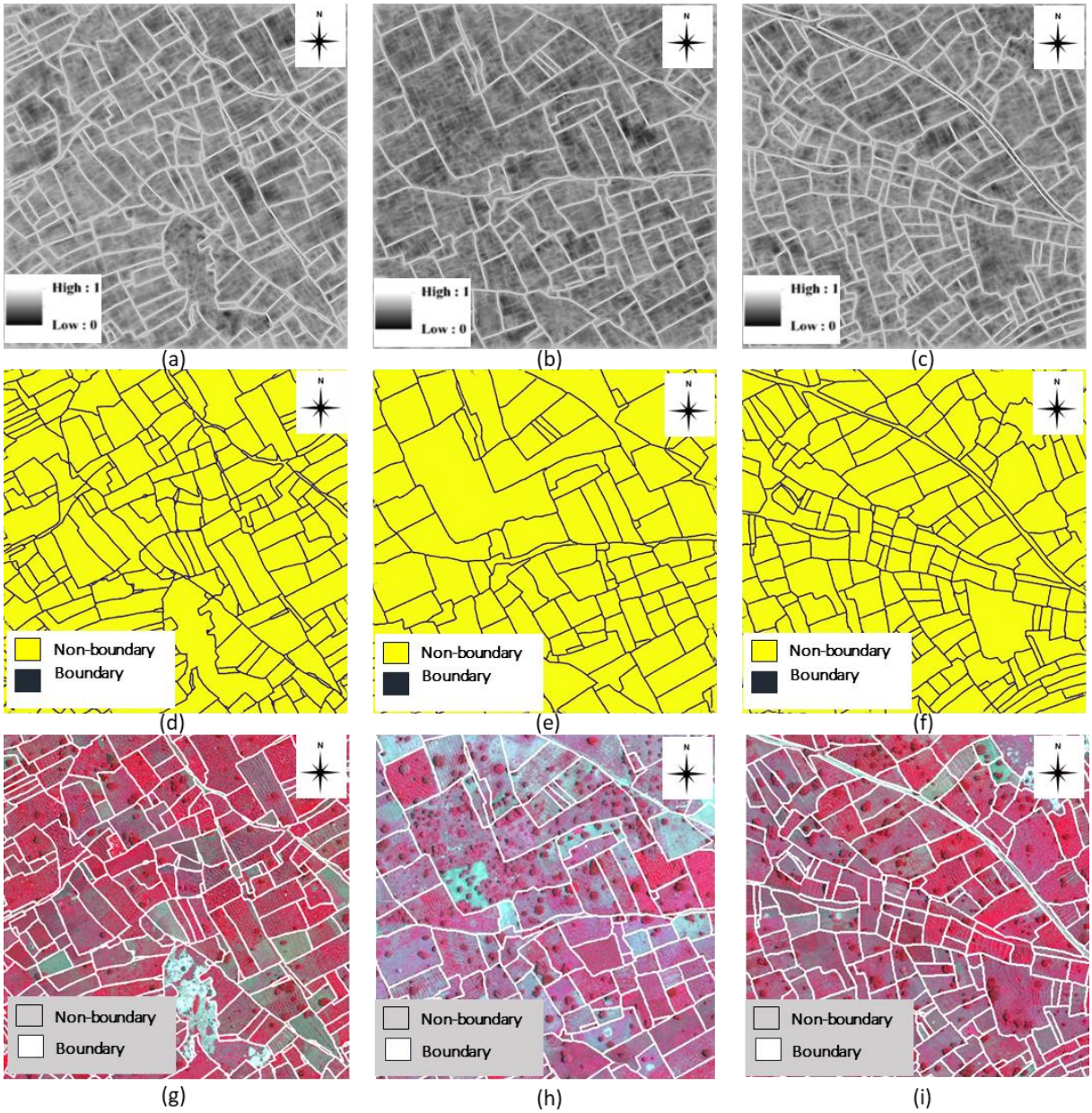


Figure 5.12: Results of training tiles **(a, d, g)** TR1, **(b, e, h)** TR3, **(c, f, i)** TR4. **(a, b, c)** are boundary probability output maps from predicted class scores **(d, e, f)** are the final classified maps from the network and **(g, h, i)** are ground truths.

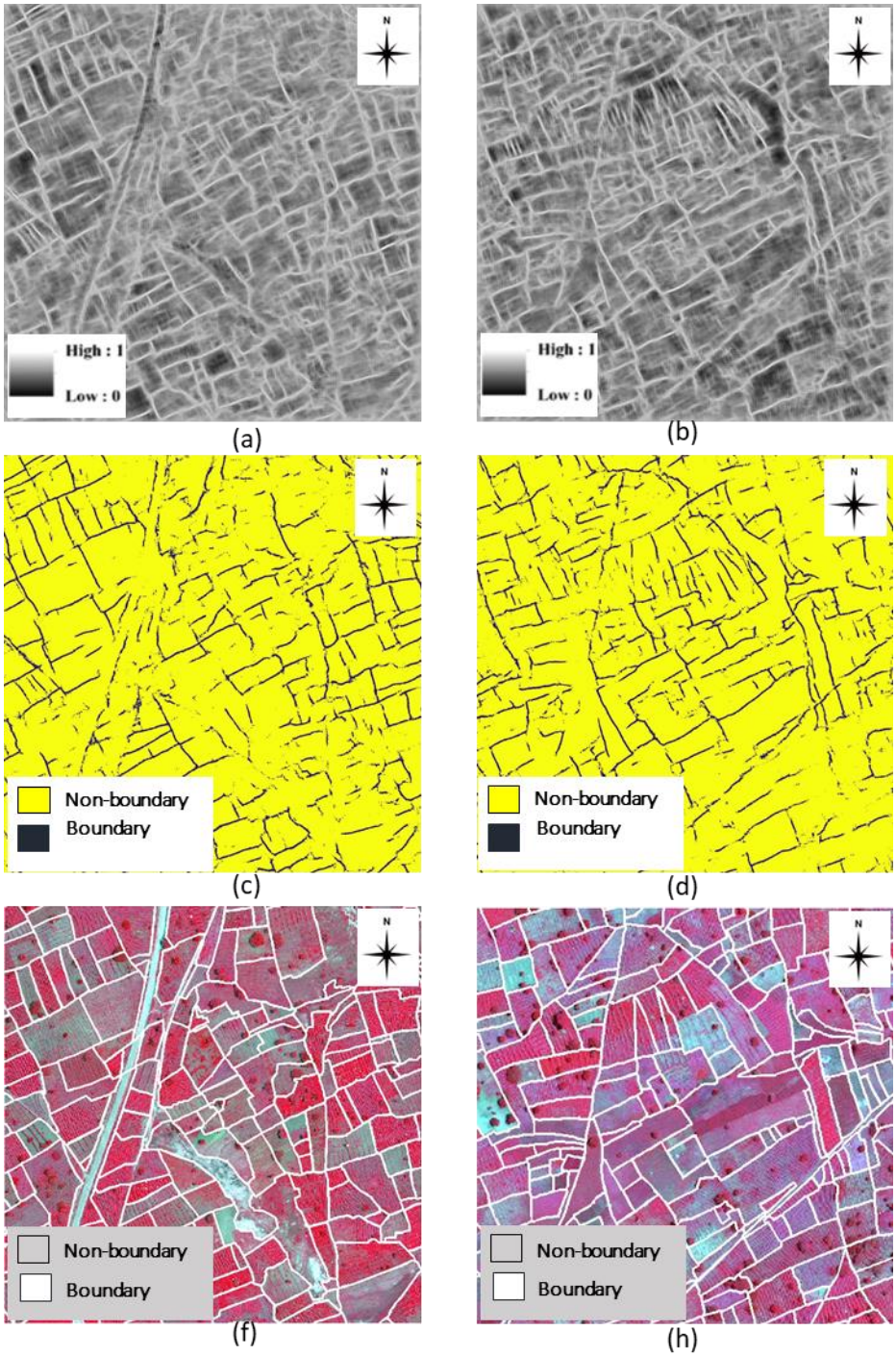


Figure 5.13: Results of testing tiles **(a, c, f)** TS2 and **(b, d, h)** TS5. **(a, b)** are the boundary probability maps **(c, d)** are the classified maps from network and **(f, h)** the ground truths.

5.3. Alternative approaches

5.3.1. gPb contour detector

The outputs from gPb contour detection are probabilities which are a result of combining mPb and sPb as described in section 4.3.1. Testing tiles TS2 and TS5 were used for gPb experiments. The algorithm consumes a lot of memory therefore the tiles were reduced to manageable dimensions of 800 × 800 pixels. A threshold k, determines the contours transferred from the raw Ultrametric contour maps to output binary boundary maps. The parameter k was varied between 0.05 and 0.6. This is because visual inspection of outputs of gPb with a value of k below 0.05 transfers so many unwanted contours including tree crowns and intra-field crop rows as boundaries as illustrated in Figure 5.14. A high value of k transfers few contours which are the most pronounced in the image as illustrated in Figure 5.15.

Figure 5.14: gPb contour detection results for threshold k = 0.05. So many contours are transferred from the ucm to boundary map.

To perform accuracy analysis, this output was further post processed. For a fair comparison with the previous method, FCN, the output was vectorised then buffered into an equal thickness of 2 m. The new file is then converted back to raster format for precision recall calculations. The output raster has its boundary class made of 4 pixels thick. Table 5.9 shows the results obtained by comparing the final output against the available reference data.

At a value of k = 0.1, the best performance is recorded in terms of F-score and precision. At this value of k however, recall and type II error are higher than values of k = 0.2 and 0.3. This is because at value k = 0.1 there are many unwanted contours which contribute to additional false positives.

Table 5.9: gPb results obtained by varying the value of parameter k

Tile	Value of k	Type I error	Type II error	Precision	Recall	F-score
TS2	0.05	0.093	0.807	0.342	0.193	0.247
	0.1	0.194	0.513	0.190	0.487	0.273
	0.2	0.204	0.441	0.106	0.559	0.178
	0.3	0.21	0.482	0.065	0.518	0.116
	0.4	0.214	0.513	0.041	0.487	0.076
	0.5	0.213	0.588	0.025	0.412	0.046
	0.6	0.215	0.667	0.014	0.333	0.027
TS5	0.05	0.094	0.842	0.385	0.158	0.224
	0.1	0.192	0.546	0.192	0.453	0.269
	0.2	0.2	0.467	0.115	0.533	0.189
	0.3	0.207	0.406	0.063	0.594	0.113
	0.4	0.215	0.783	0.013	0.217	0.024
	0.5	0.22	0.795	0.015	0.205	0.027
	0.6	0.22	0.793	0.009	0.208	0.017

Figure 5.15 shows the results of gPb at levels 0.1 and 0.5. the boundaries shown are after the buffering process. They have been overlaid on the raw image to improve visibility and make it easy to identify the spatial locations where detections have taken place.

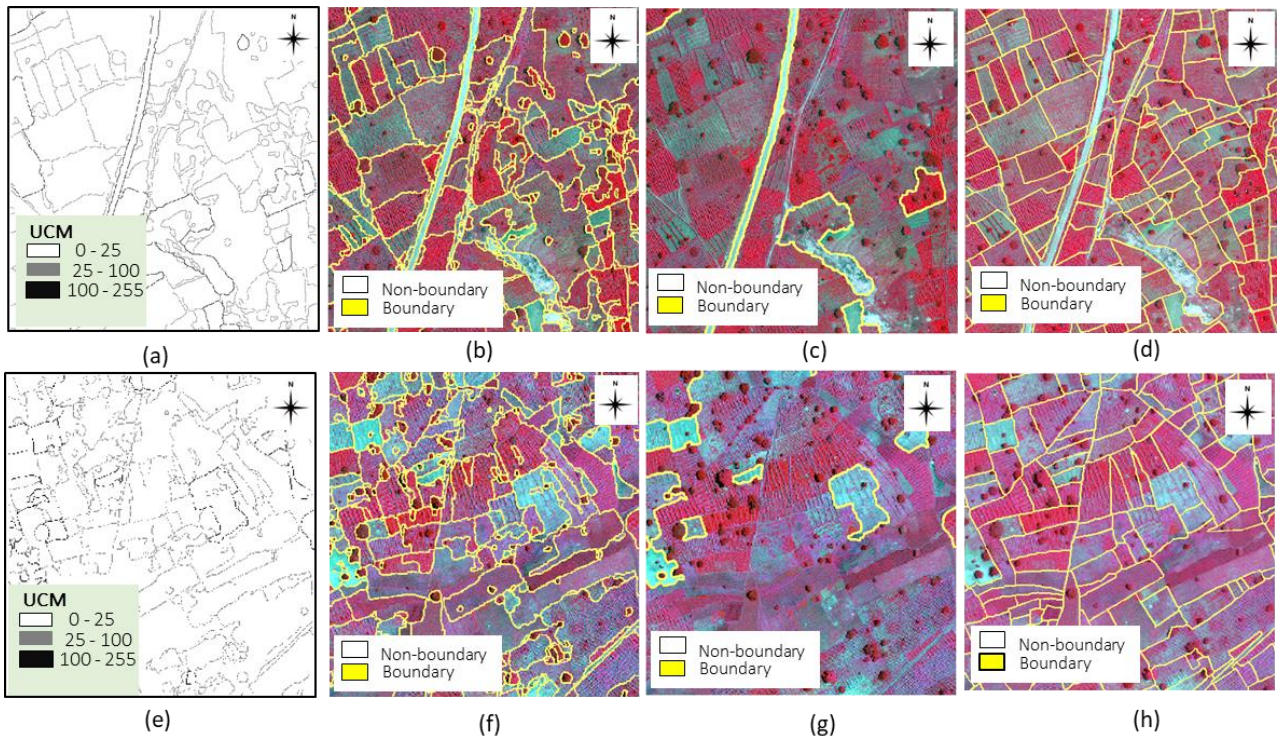


Figure 5.15: gPb detection results for; **(a, b, c, d)** TS2 and **(e, f, g, h)** TS5. **(a, e)** are ultrametric contour maps, **(b, f)** threshold k = 0.1 **(c, g)** k = 0.5 and **(d, h)** Ground truths.

5.3.2. Canny detector

Canny detector takes as input a grayscale image and produces an output map showing detected intensity discontinuities. The canny detector has three hyper-parameters namely, 1) the width of the Gaussian kernel used in the smoothing phase, 2) the upper and 3) lower thresholds used by the tracker. Increasing the width of the Gaussian kernel diminishes the sensitivity of the detector to noise, but reduces the amount of detail which can be detected in the image.

We started experiments with testing tile TS2 to tune canny detector hyperparameters. Just like we did with gPb detector, all the raw canny detection outputs were post processed by converting them to vector, thickening the boundary to 2m and then converting them back again to raster for accuracy analysis. Therefore, all the results posted are from the final post processed output. Using a Gaussian filter with a standard deviation of 1.0 and an upper threshold $T_1 = 255$, and a lower threshold $T_o = 1$, we obtain;

Table 5.10: Canny detector results; $\sigma = 1.0$, $T_1 = 255$ and $T_o = 1$

Tile	Type I error	Type II error	Precision	Recall	F-score
TS2	0.2163	0.7431	0.0216	0.2576	0.0409

The next experiments involved optimizing these 3 hyperparameters. Setting T_0 too high extracts the most salient edges and drops most weak edges. Setting T_1 too low increases quantity weak boundaries, therefore, increasing false positives and a very high type 1 error. Maintaining the standard deviation as 1.0 and $T_o = 1$, then reducing T_1 to 180, we obtain

Table 5.11: Canny detector results; $\sigma = 1.0$, $T_1 = 180$ and $T_o = 1$

Tile	Type I error	Type II error	Precision	Recall	F-score
TS2	0.2160	0.7440	0.0201	0.2571	0.0406

The three accuracy measures reduce as shown in Table 5.11. This is because many faint edges are detected leading to increased false positives, therefore, reducing the accuracies and increasing type II error. The difference is however too small because at this level canny detector fails in the highly complicated dataset.

In the next step, we tune standard deviation which is a component of the Gaussian smoothing. Gaussian smoothing in canny edge detector is useful for reducing noise. It reduces noise by controlling the amount of detail that appears in the edge image. Since our images are very noisy, we increase the standard deviation to 2.0 to check the performance of canny. The other parameters are maintained as $T_o = 1$, and $T_1 = 180$.

Table 5.12: Canny detector results; $\sigma = 2.0$, $T_1 = 180$ and $T_o = 1$

Tile	Type I error	Type II error	Precision	Recall	F-score
TS2	0.2160	0.7429	0.0206	0.2571	0.04074

There is a slight improvement from the previous experiment. We set out to investigate the effect of increasing the lower threshold, maintaining the standard deviation as 2.0 and T_1 as 180.

Table 5.13: Canny detector results; $\sigma = 2.0$, T_1 = 180 and T_o = 60, 100

Tile	**T_o**	Type I error	Type II error	Precision	Recall	F-score
TS2	60	0.2165	0.7436	0.0225	0.2582	0.0431
TS2	100	0.2162	0.7492	0.0206	0.2397	0.0386

In testing tile TS2, the best parameter combination is observed as $\sigma = 2.0$, T_1 = 180 and T_o = 60. This is because increasing T_o =60 discards unnecessary details. Increasing further to T_o =100 drops so many edges which are true positives hence the decreasing accuracy. We implemented this parameter set to the other testing tile TS5 and the results of both tiles are as shown in Table 5.14.

Table 5.14: Canny detector results for TS2 and TS5 for parameters, $\sigma = 2.0$, T_1 = 180 and T_o = 60

Tile	Type I error	Type II error	Precision	Recall	F-score
TS2	0.2165	0.7436	0.0225	0.2582	0.0431
TS5	0.2143	0.7923	0.0488	0.2113	0.0754

Despite the fact that canny edge detector finds the intensity discontinuities in an image, it is not guaranteed that these discontinuities correspond to actual edges of the object. Figure 5.16 shows the results obtained for both testing tiles hence the dismal performance. Most boundaries in both tiles have been missed by the detector. Only the very pronounced boundaries have been detected.

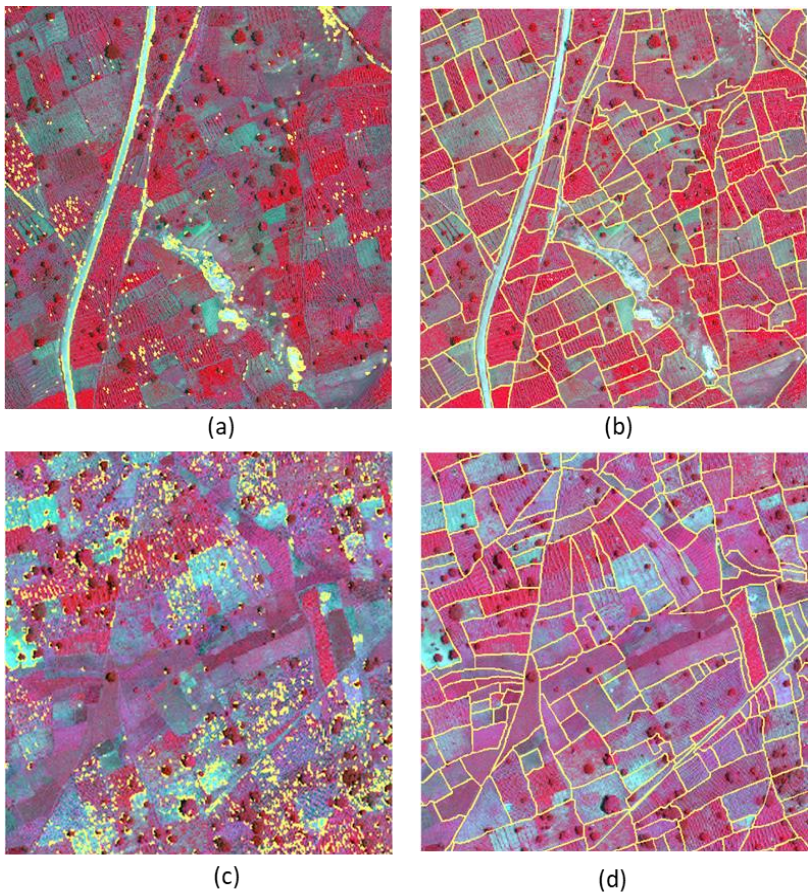


Figure 5.16: Canny detector results for **(a)** TS2 and **(c)** TS5 and their respective ground truths **(b** and **d)**

5.3.3. Image segmentation in eCognition software

The multi-resolution segmentation algorithm is able to generate image objects at any chosen resolution. MRS aims at minimising the average heterogeneity and maximising its respective heterogeneity, for a given number of image objects, to produce meaningful objects.

As described in Section 4.3.3, the outcome of multi-resolution segmentation is determined by the weights of colour and shape, scale parameter, and compactness. The weight of the colour or image layer weights indicates the level to which the bands influence the segmentation process. In these experiments, we were using Worldview3 VHR satellite imagery, described in detail in Section 3.2. Agricultural fields and boundaries are mostly covered by vegetation which has a high reflectance in the infrared region. It is for this reason bands 7(Near-IR1) and 8(Near-IR2) were assigned the highest recommended weight(4) in the literature (Rejaur & Saha, 2008; Aguilar et al., 2016). Bands 5 (Red) and 6(Red edge) were assigned an equal weight of 2 and bands 1 – 4 set to 1. The second hyperparameter is shape which factors how the spectral values of each multispectral band affect the heterogeneity of segmented objects. A high value of shape parameter produces segments which have more spatial homogeneity and less spectral homogeneity. Since our study area has a very heterogeneous landscape, the emphasis was less spectral uniformity. The shape parameter was therefore set as 0.8 as recommended in the literature for very heterogeneous landscapes. The third parameter is Compactness which determines how segments objects are going to be compact. A high value creates compact segments. This parameter was set as 5 to create a balance between compact and non-compact object segments. Lastly, the most sensitive parameter is the scale parameter (SP). The scale parameter determines the maximum allowable heterogeneity in segmented objects. A small value of scale parameter produces more homogeneous segments as compared to a larger value of the scale parameter. Values of SP were varied as 100, 150, 200, 250 and 300. A step size of 50 was chosen to enable a visible change in the segmented output. Differences in outputs were not easy to visually tell using a smaller step size. Values of SP below 100 led to over-segmentation while values above 300 led to under segmentation. Figure 5.17 shows the results obtained by varying the scale parameter.

Table 5.15: Multi resolution segmentation results

Tile	Scale Parameter (SP)	Type I error	Type II error	Precision	Recall	F-score
TS2	100	0.081	0.819	0.493	0.181	0.265
	150	0.084	0.784	0.400	0.216	0.280
	200	0.089	0.775	0.336	0.225	0.270
	250	0.091	0.764	0.299	0.236	0.263
	300	0.092	0.755	0.280	0.245	0.261
TS5	100	0.083	0.836	0.493	0.164	0.247
	150	0.088	0.809	0.374	0.191	0.253
	200	0.090	0.792	0.317	0.208	0.251
	250	0.094	0.770	0.243	0.230	0.236
	300	0.096	0.768	0.216	0.232	0.224

SP = 150 produces the best result for testing tile TS2 and TS5 as indicated in Table 5.15. Segmented results match best with reference data at this level of parameter combination. Type II errors are however quite high in all the parameter values tested. This is because most detections

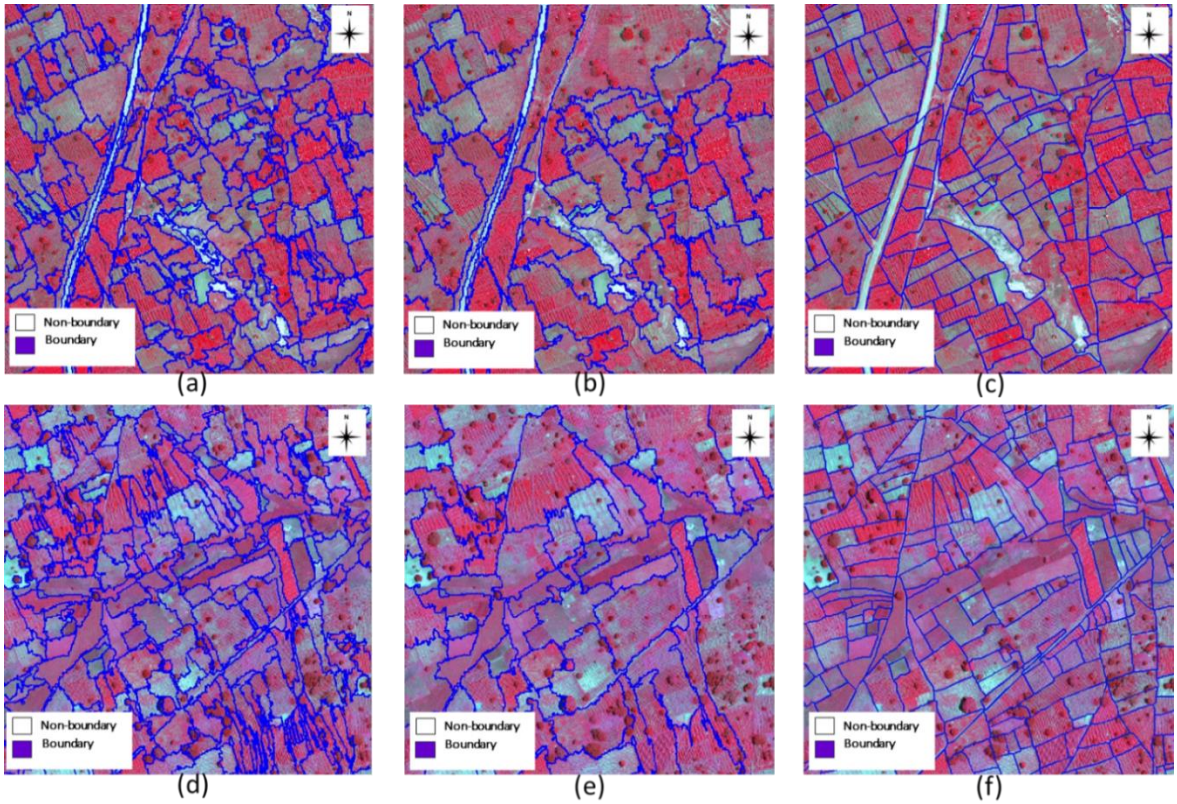


Figure 5.17: Multiresolution segmentation results; **(a, b, c)** TS2 and **(d, e, f)** TS5, **(a, d)** SP = 100 **(b, e)** SP = 300 **(c, f)** are the Ground truths for TS2 and 5 respectively.

5.4. Performance comparison

5.4.1. Precision, recall and F-score

As described in section 4.4, precision is a measure of exactness. An algorithm producing a high value of precision means it has detected more correct boundaries in relation to the total detected boundaries. Table 5.16 and Table 5.17 shows that FCN-DKConv6 outperformed the other algorithms in terms of precision, followed by multi-resolution segmentation then gPb and finally Canny. Recall is a measure of completeness. A high value of recall indicates that an algorithm has identified more boundaries in the reference dataset compared with other algorithms. A low recall value indicates presence of many false negatives. gPb outperformed the other algorithms in terms of recall. F-score is a combination of these two measures. F-score is more reliable as an overall measure of quality performance. FCN-DKConv6 produced the highest F-score among the considered algorithms therefore proving better when both in both exactness and completeness.

Table 5.16: Accuracies obtained by the considered algorithms on testing tile TS2

Algorithm	Precision	Recall	F-score
FCN-DKConv6	0.407	0.312	0.353
MRS	0.400	0.216	0.280
gPb	0.190	0.487	0.273
Canny	0.023	0.258	0.043

Table 5.17: Accuracies obtained by the considered algorithms on testing tile TS5

Algorithm	Precision	Recall	F-score
FCN-DKConv6	0.389	0.288	0.331
MRS	0.374	0.191	0.253
gPb	0.192	0.453	0.269
Canny	0.049	0.211	0.075

5.4.2. Visual inspection and comparison of output maps

We compared output maps of all the algorithms tested. FCN-DK6 was able to label tree crowns as non-boundaries as shown in Figure 5.13. Multi-resolution segmentation and gPb on the other hand had many false detections in terms of tree crowns and within field detections. Canny detector missed so many detections therefore produced dismal results especially in testing tile TS5.

5.4.3. Computational time and complexity

Comparing the performance of the methods in terms in terms of computational time and complexity, eCognition's multi-resolution segmentation was the fastest followed by canny detector. Both executed in under a minute per testing tile. gPb is a computational heavyweight although not as complex as FCN. Results from each testing tile, of dimensions 800×800 pixels, were obtained in about 15 minutes by gPb. FCN had the highest computational cost, which is a characteristic of most complex models. Training time ranged from 2 – 3 hours for hyper-parameter tuning experiments and 14 – 16 hours for experiments on full datasets using NVIDIA's CUDA GPUs (Quadro M1000M).

6. DISCUSSION

In this chapter, a discussion based on the analysis of the results is presented. We investigated the use of FCN for detection of visible agricultural field boundaries from VHR satellite images. Hyperparameter optimization experiments demonstrated that the patch size, the number of patch samples and the depth of the network have the highest influence on the accuracy of the results. Experiments on a patch size of 115 × 115 produced a lower accuracy compared to 95 × 95. The reduction in accuracy can be attributed to a growing number of parameters with a fixed sample size of 1000 for that set of experiments. Having growing number parameters and a fixed training data means the network is likely to underperform when the patch is increased further. For this reason, a patch size of 95 × 95 was maintained in all the later experiments. We noted the significance of a big training size by experimenting with different sample sizes and varying the number of training tiles. FCNs are data hungry. Increasing the number of patch samples improved the accuracy of the networks in Section 5.1.2. A large training set enables an FCN to learn more complex features and increase its generalization capabilities. For instance, Experiments in Section 5.1.4 involved training our proposed networks using training tile TR1 only. DKConv6 in this experiment achieved the highest F-score of 0.975 compared to the other experiments. On generalization capability at this level, the accuracy of testing tile TS2 was low compared to the other experiments involving the full dataset.

We studied the effect of the depth of the networks. FCN depth defines the number of layers doing convolutional computations on the dataset from the input layer to the output layer. A deep network allows for a hierarchy of useful spatial contextual features to be learnt (Simonyan & Zisserman 2014). We increased the dilation factor of the filters as the network grew deeper. For instance, the filters in the 6th convolutional layer were dilated with a factor of 6 while the 2nd convolutional layer the filters had a dilation of 2. The effective receptive field of filters in the 6th convolutional layer was 85 compared to 13 for filters in the 2nd layer. This means the deeper network is able to scan a larger area of an input image for more spatial-contextual information as opposed to the smaller network.

Experiments on discarding the pooling layers were very important for our final implementation. An improvement in accuracies and a reduced value of errors were noted when the pool was dropped. Max pooling with a stride greater than 1 reduces the spatial size of an input data which reduces the number of parameters. This helps in reducing the amount of computation in the network and in the process controls overfitting. However, pooling with a stride of 1 and maintaining an equal number of filters across the network means the input and outputs in each layer are the same, so parameters remain the same. Reducing the complexity of the model for the same number of parameters proved efficient in-terms of reducing training time from 8 hours 20 minutes for FCN-DK6 of experiment 5.1.3 to about 4 hours in network FCN-DKConv6 of experiment 5.1.4.

Data augmentation experiments unexpectedly did not improve the performance of the networks. We made the networks rotational invariant by rotating training data at 3 different angles. We created artificial data which was 3 times the original data. Long et al. (2015) tried augmentation by randomly mirroring and downscaling images. Their strategy didn't improve the results as well. Depending on a classification problem and the nature of the data, augmentation doesn't always work.

Our final implementation was compared with other alternative approaches to contour detection and segmentation approaches for boundary extraction. A comparison was drawn based on various factors like accuracy of the results, visual quality of the output maps and computational time and complexity. In terms of accuracy of the results, our proposed method outperformed the three contenders. Multi-resolution segmentation performed second best closely followed by gPb contour detection. Canny detector

performed poorest with the lowest accuracy. In terms of visual quality of the output maps, FCN outputs appear very regular and in the same thickness as the training data without any post processing technique. A few spots of wrong detections are visible within the fields which could be removed by a simple post-processing procedure. We did not integrate our proposed methodology with a post-processing procedure because the focus was more on an end-to-end FCN.

7. CONCLUSIONS AND RECOMMENDATIONS

7.1. Conclusions

The main objective of this study was to formulate a methodology for detection of visible agricultural field boundaries from VHR satellite images using a deep feature learning approach based on FCNs. We reviewed several convolutional networks and chose FCN-DKs. We used a WorldView-3 satellite image captured over Kofa area, in Kano state in the northern parts of Nigeria on the 25th September 2015. The effect of varying FCN's hyperparameters on the performance and ability to generalize, by presenting the networks with unseen data during training, were investigated. A detailed comparative analysis of the proposed approach against gPb, multi-resolution segmentation algorithm in eCognition and canny detector was conducted. Experiments shown that our method performed better than the contending algorithms.

The answers to research questions posed in section 1.3.1 are as follows:

1. Which CNN/FCN architectures have been proposed in the literature and how do they work?

   In section 2.2.3, we discussed how CNNs work and further discussed in detail two outstanding architectures in literature, namely AlexNet and VGGNet. FCNs were described in sub-section 2.2.4 as variants of CNN. The architecture of FCN-DKs was also elaborated in detail.

2. Which CNN/FCN architecture is appropriate for detection and extraction of agricultural field boundaries?

   We adopted FCN-DKs proposed by Persello & Stein (2017), modified the architecture and optimised the hyperparameters to enable the networks solve the problem of distinguishing between boundaries and non-boundaries in our dataset. As described in section 2.2.4, FCN-DKs produced impressive performance compared to conventional patch based CNNs.

3. How should the classes be defined?

   In section 3.4, we described how the reference datasets were prepared. Classes were divided into 2, the boundary class which was assigned a label =1 and the non-boundary class assigned the label =2. After preparations, the final reference dataset had its boundary made up of a raster with 4 pixels equal thickness. This was achieved by applying a buffer of 1m to a shapefile and exporting this shapefile with a cell resolution of 0.5 m, which was equal to the resolution of pansharpened VHR images.

4. What are the optimal dimensions of the network and learning hyperparameters?

   Experiments shown that a patch size of 95 × 95, a patch sample size of 6000 and a network depth of 6 convolutional layers produced the best results. The amount of training data was key. Using three training tiles TR1, TR3 and TR4 produced better results than using just one training tile TR1. Training the networks with a stochastic gradient descent with a momentum of 0.9 accelerated the training process. Training was divided into two steps for the 200 epochs. The first 170 iterations

were done using a learning rate of 1.0×10^{-3} and then reduced to a learning rate of 1.0×10^{-4} for the remaining 30 iterations. Convergence of training and validation losses was already achieved by the 170th iteration.

5. Which is the most suitable method for accuracy assessment of classification results?

   Precision, recall and F-score were used to quantify accuracies because they are better suited for binary classification tasks with highly imbalanced classes as explained in section 4.4. Values of the three measures were given between the range of 0 to 1, 0 being a very low accuracy and 1 being the highest attainable accuracy.

6. Which alternative state-of-the-art approaches exist?

   Three alternative approaches were selected to compare the performance of our proposed method with. They included gPb detector, multi-resolution segmentation algorithm in eCognition software and Canny detector.

7. Which approach performs better and in which aspects of the performance measures?

   FCN produces a better F-score and Precision than gPb, multi-resolution segmentation and canny detector. gPb outperforms the rest with regards to of recall. The F-score was used to determine the best performing algorithm since it's a function of both precision and recall therefore FCN was taken the best performer. Canny detector produced the poorest results.

7.2. Recommendations for future research work

For future works, we recommend the following:

Investigate integrating the proposed FCN with oriented watershed transform and ultrametric contour maps to produce closed contours.

Investigate the performance of the proposed FCN using other coarser resolution, freely available satellite images like sentinel-1 and sentinel-2

Investigate techniques of improving accuracy like integrating the FCN with classifiers capable of noise reduction like CRF

LIST OF REFERENCES

Agro News Nigeria. (2018). Nigerian States and Their Agricultural Produce. Retrieved 8 January 2018, from http://agronewsng.com/nigerian-states-and-their-agricultural-products/

Aguilar, M. A., Aguilar, F. J., García Lorca, A., Guirado, E., Betlej, M., Cichon, P., … Parente, C. (2016). Assessment of multiresolution segmentation for extracting greenhouses from WorldView-2 imagery. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *41*(June), 145–152. https://doi.org/10.5194/isprsarchives-XLI-B7-145-2016

Alemu, M. M. (2016). *Automated farm field delineation and crop row detection from satellite images (Unpublished master's thesis)*. Faculty of Geo-Information Science and Earth Observation, University of Twente, Enschede, Netherlands.

Alshehhi, R., Marpu, P. R., Woon, W. L., & Mura, M. D. (2017). Simultaneous extraction of roads and buildings in remote sensing imagery with convolutional neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, *130*, 139–149. https://doi.org/10.1016/j.isprsjprs.2017.05.002

Arbeláez, P., Maire, M., Fowlkes, C., & Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(5), 898–916. https://doi.org/10.1109/TPAMI.2010.161

Arbeláez, P., Maire, M., Fowlkes, C., & Malik, J. (2013). UC Berkeley Computer Vision Group - Contour Detection and Image Segmentation - Resources. Retrieved 13 December 2017, from https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

Baatz, M., & Schäpe, A. (2000). Multiresolution Segmentation : an optimization approach for high quality multi-scale image segmentation. *Angewandte Geographische Informationsverarbeitung XII, Beiträge Zum AGIT-Symposium Salzburg 2000*. https://doi.org/retrieved from : http://www.ecognition.com/sites/default/files/405_baatz_fp_12.pdf

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828. https://doi.org/10.1109/TPAMI.2013.50

Bennett, R., Kitchingman, A., & Leach, J. (2010). On the nature and utility of natural boundaries for land and marine administration. *Land Use Policy*, *27*(3), 772–779. https://doi.org/10.1016/j.landusepol.2009.10.008

Bergado, J. R. A., Persello, C., & Gevaert, C. (2016). A Deep Learning Approach to the Classification of Sub-Decimetre Resolution Aerial Images. *2016 IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, 1516–1519. https://doi.org/10.1109/IGARSS.2016.7729387

Bishop, C. M. (2013). *Pattern Recognition and Machine Learning. Journal of Chemical Information and Modeling* (Vol. 53). https://doi.org/10.1117/1.2819119

Blaschke, T., Burnett, C., & Pekkarinen, A. (2004). Image segmentation methods for object-based analysis and classification. *Blaschke, Thomas, Charles Burnett, and Anssi Pekkarinen. 'Image Segmentation Methods for Object-Based Analysis and classification.' Remote Sensing Image Analysis: Including the Spatial Domain. Springer Netherlands, 2004. 211-236.*, 211–236. https://doi.org/10.1007/978-1-4020-2560-0

Bowyer, K., Kranenburg, C., & Dougherty, S. (1999). Edge detector evaluation using empirical ROC curves. *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference On.*, *1*, 359 Vol. 1-359 Vol. 1. https://doi.org/10.1109/CVPR.1999.786963

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *8*(6), 679–698. https://doi.org/10.1109/TPAMI.1986.4767851

Crommelinck, S., Bennett, R., Gerke, M., Nex, F., Yang, M. Y., & Vosselman, G. (2016). Review of automatic feature extraction from high-resolution optical sensor data for UAV-based cadastral mapping. *Remote Sensing*, *8*(8). https://doi.org/10.3390/rs8080689

Dale, P. F., & McLaughlin, J. D. (1988). *Land information management: an introduction with special reference to cadastral problems in Third World countries*. Clarendon Press.

Danilla, C. (2017). *Convolutional Neural Networks for contextual denoising and classification of SAR images (unpublished master's thesis)*. Faculty of Geo-Information Science and Earth Observation, University of Twente, Enschede, Netherlands.

Davidse, J. (2015). *Semi-automatic detection of field boundaries from high resolution satellite imagery (master's thesis)*. Wageningen University and Research Centre, Netherlands. https://doi.org/80 09 09 173 090

de By, & R.A. (2015). STARS : Project overview : spurring an transformation for agriculture through

remote sensing.
DigitalGlobe. (2017). Satellite Sensors and Specifications | Satellite Imaging Corp. Retrieved 20 August 2017, from http://www.satimagingcorp.com/satellite-sensors/
Duhaime, L. (2017). Boundary definition. Retrieved 4 January 2018, from http://www.duhaime.org/LegalDictionary/B/Boundary.aspx
FAO. (2016). *Key to achieving the 2030 Agenda for Sustainable Development. Food And Agriculture*. https://doi.org/I5499E/2/04.16
Fu, G., Liu, C., Zhou, R., Sun, T., & Zhang, Q. (2017). Classification for high resolution remote sensing imagery using a fully convolutional network. *Remote Sensing*, *9*(5), 1–21. https://doi.org/10.3390/rs9050498
García-Pedrero, A., Gonzalo-Martín, C., & Lillo-Saavedra, M. (2017). A machine learning approach for agricultural parcel delineation through agglomerative segmentation. *International Journal of Remote Sensing*, *38*(7), 1809–1819. https://doi.org/10.1080/01431161.2016.1278312
Goodfellow, I., Bengio, Y., & Courville, A. (2015). Deep Learning. *Nature Methods*, *13*(1), 35–35. https://doi.org/10.1038/nmeth.3707
Grompone von Gioi, R., Jakubowicz, J., Morel, J.-M., & Randall, G. (2012). LSD: a Line Segment Detector. *Image Processing On Line*, *2*, 35–55. https://doi.org/10.5201/ipol.2012.gjmr-lsd
Hossin, M., & Sulaiman, M. N. (2015). a Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, *5*(2), 1–11. https://doi.org/10.5121/ijdkp.2015.5201
Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. https://doi.org/10.1007/s13398-014-0173-7.2
Jain, M., Mondal, P., DeFries, R. S., Small, C., & Galford, G. L. (2013). Mapping cropping intensity of smallholder farms: A comparison of methods using multiple sensors. *Remote Sensing of Environment*, *134*, 210–223. https://doi.org/10.1016/j.rse.2013.02.029
Jevnisek, R. J., & Avidan, S. (2016). Semi global boundary detection. *Computer Vision and Image Understanding*, *152*, 21–28. https://doi.org/10.1016/j.cviu.2016.07.004
Juneja, M., & Sandhu, P. S. (2009). Performance evaluation of edge detection techniques for images in spatial domain. *International Journal of Computer Theory and Engineering*, *1*(5), 614–622. https://doi.org/10.7763/IJCTE
Karakış, S., Marangoz, A. A., Buyuksalih, G., Karakis, S., Marangoz, A. A., & Buyuksalih, G. (2018). ANALYSIS OF SEGMENTATION PARAMETERS IN ECOGNITION SOFTWARE USING HIGH RESOLUTION QUICKBIRD MS IMAGERY. *Paper, Department of Geodesy and …*, (October), 1–5. https://doi.org/Retrieved from: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Analysis+of+Segmentation+Parameters+in+Ecognition+Using+High+Resolution+Quickbird+MS+Imagery#6
Kaur, H. (2015). Review of Remote Sensing Image Segmentation Techniques. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, *4*(4), 1667–1674.
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances In Neural Information Processing Systems* (pp. 1–9). Nevada, USA: Neural Information Processing Systems (NIPS).
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539
LeCun, Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2323. https://doi.org/10.1109/5.726791
Li, F.-F., Johnson, J., & Yeung, S. (2017). CS231n Convolutional Neural Networks for Visual Recognition. Retrieved 4 November 2017, from http://cs231n.github.io/convolutional-networks/
Long, J., Shelhamer, E., & Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, *8828*(c), 3431–3440. https://doi.org/10.1109/CVPR.2015.7298965
López, V., Fernández, A., García, S., Palade, V., & Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, *250*, 113–141. https://doi.org/10.1016/j.ins.2013.07.007
Lowder, S. K., Skoet, J., & Raney, T. (2016). The Number, Size, and Distribution of Farms, Smallholder Farms, and Family Farms Worldwide. *World Development*, *87*, 16–29. https://doi.org/10.1016/j.worlddev.2015.10.041
Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network

Acoustic Models. *Proceedings of the 30 Th International Conference on Machine Learning*, *28*, 6.
Maire, M., Arbel, P., Fowlkes, C., & Malik, J. (2008). Using Contours to Detect and Localize Junctions in Natural Images. *IEEE Conference on Computer Vision and Pattern Recognition*, 1–8. https://doi.org/10.1109/CVPR.2008.4587420
Martin, D. R., Fowlkes, C. C., & Malik, J. (2004). Learning to Detect Natural Image Boundaries Using Local Brightness and Texture Cues. *Pami*, *26*(1), 1–20. https://doi.org/10.1109/TPAMI.2004.1273918
Mathias, J., & Lemmens, P. M. (1996). A Survey on Boundary Delineation Methods. In P. W. Karl Kraus (Ed.), *ISPRS TC VII Symposium* (pp. 435–441). Vienna, Austria.
Maurer, T., & Street, N. Y. (2013). How To Pan-Sharpen Images Using the Gram-Schmidt Pan-Sharpen Method – a Recipe. *ISPRS Hannover Workshop 2013*, *XL*(May), 21–24.
Mboga, Persello, C., Bergado, J. R., & Stein, A. (2017). Detection of informal settlements from VHR images using convolutional neural networks. *Remote Sensing*, *9*(11). https://doi.org/10.3390/rs9111106
Mueller, M., Segl, K., & Kaufmann, H. (2004). Edge- and region-based segmentation technique for the extraction of large, man-made objects in high-resolution satellite imagery. *Pattern Recognition*, *37*(8), 1619–1628. https://doi.org/10.1016/j.patcog.2004.03.001
Nogueira, K., Miranda, W. O., & Santos, J. A. Dos. (2015). Improving Spatial Feature Representation from Aerial Scenes by Using Convolutional Networks. *Brazilian Symposium of Computer Graphic and Image Processing*, *2015–Octob*, 289–296. https://doi.org/10.1109/SIBGRAPI.2015.39
Oxford English Dictionary. (2017). Definition of boundary. Retrieved 12 October 2017, from https://en.oxforddictionaries.com/definition/boundary
Parker, J. R. (2010). *Algorithms for image processing and computer vision* (2nd ed.). New Jersey: Wiley Publishing ©2010. https://doi.org/10.1017/CBO9781107415324.004
Persello, C., & Stein, A. (2017). Deep Fully Convolutional Networks for the Detection of Informal Settlements in VHR Images. *IEEE Geoscience and Remote Sensing Letters*, *PP*(X), 1–5. https://doi.org/10.1109/LGRS.2017.2763738
Rejaur, R., & Saha, S. K. (2008). Multi-resolution segmentation for object-based classification and accuracy assessment of land use/land cover classification using remotely sensed data. *Journal of the Indian Society of Remote Sensing*, *36*(2), 189–201. https://doi.org/10.1007/s12524-008-0020-4
Roberts, L. Gi. (1965). *Machine perception of three-dimensional solids*. *PhD Thesis*. Retrieved from http://www.packet.cc/files/mach-per-3D-solids.html#*.
Rockson, G., Bennett, R., & Groenendijk, L. (2013). Land administration for food security: A research synthesis. *Land Use Policy*, *32*, 337–342. https://doi.org/10.1016/j.landusepol.2012.11.005
Salahat, E., & Qasaimeh, M. (2017). Recent advances in features extraction and description algorithms: A comprehensive survey. *Proceedings of the IEEE International Conference on Industrial Technology*, 1059–1063. https://doi.org/10.1109/ICIT.2017.7915508
Scott, G. J., England, M. R., Starms, W. A., 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. https://doi.org/10.1109/LGRS.2017.2657778
Sharma, S. (2018). Activation Functions: Neural Networks – Towards Data Science. Retrieved 5 January 2018, from https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
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
Sonka, M., Hlavac, V., & Boyle, R. (1999). Image Processing, Analysis, and Machine Vision Second Edition. *International Thomson*, (May 2015).
Sourav, D., Bhattacharyya, S., Chakraborty, S., & Dutta, P. (2016). Hybrid Soft Computing for Multilevel Image and Data Segmentation, 29–41. https://doi.org/10.1007/978-3-319-47524-0
Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for Simplicity: The All Convolutional Net, 1–14. https://doi.org/10.1163/_q3_SIM_00374
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958. https://doi.org/10.1214/12-AOS1000
Swetz, F. J. (2008). Surveying. In H. Selin (Ed.), *Encyclopaedia of the History of Science, Technology, and Medicine in Non-Western Cultures* (2nd ed., pp. 2063–2066). Springer Netherlands.
Tian, J., & Chen, D. -M. (2007). Optimization in multi-scale segmentation of high-resolution satellite images for artificial feature recognition. *International Journal of Remote Sensing*, *28*(20), 4625–4644.

https://doi.org/10.1080/01431160701241746

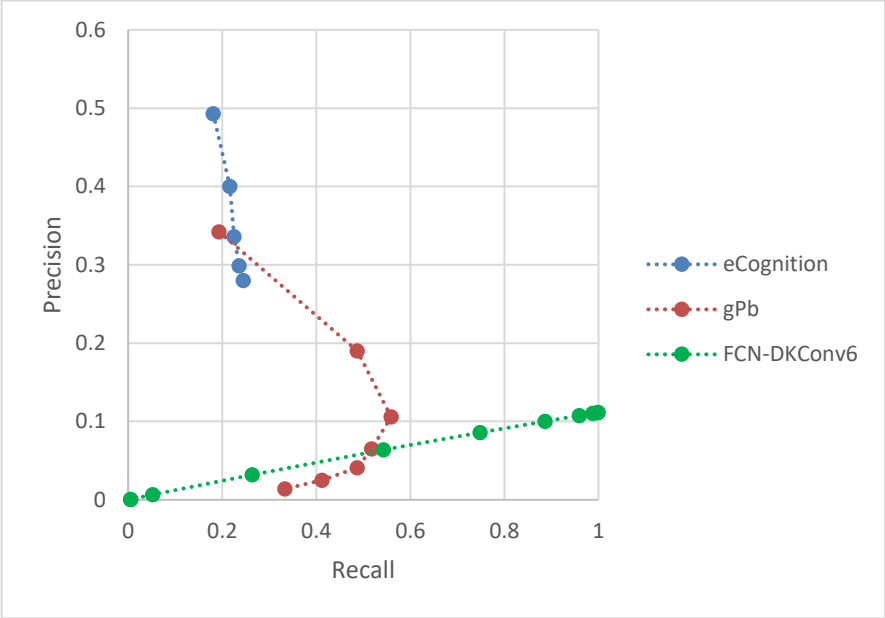
Turker, M., & Kok, E. H. (2013). Field-based sub-boundary extraction from remote sensing imagery using perceptual grouping. *ISPRS Journal of Photogrammetry and Remote Sensing*, *79*, 106–121. https://doi.org/10.1016/j.isprsjprs.2013.02.009

Vedaldi, A., & Lenc, K. (2015). MatConvNet: Convolutional Neural Networks for MATLAB. In *Proceedings of the 23rd ACM International Conference on Multimedia* (pp. 689–692). New York, NY, USA: ACM. https://doi.org/10.1145/2733373.2807412

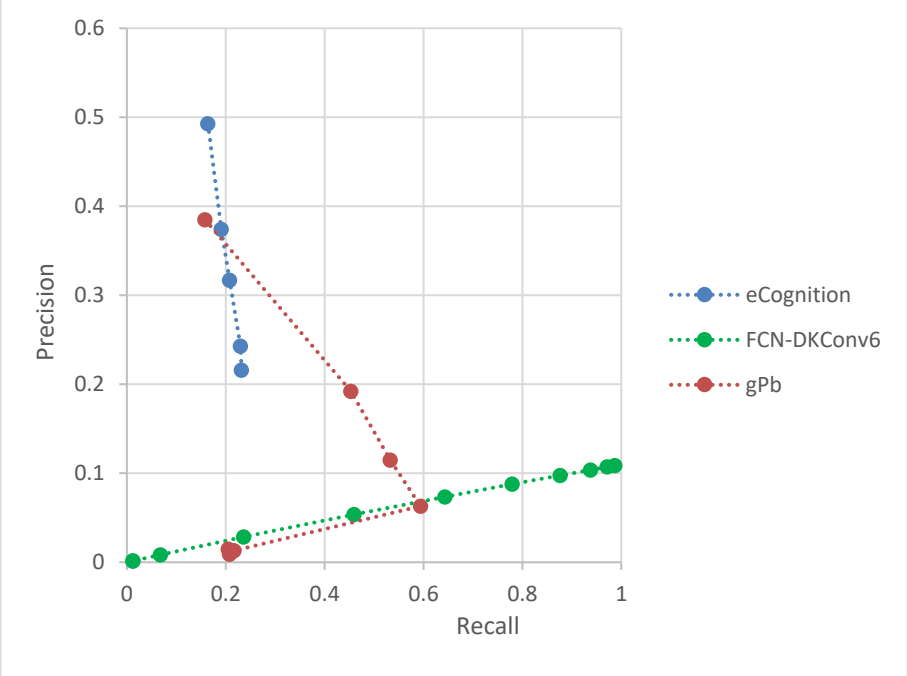
Zhang, X., Xiao, P., Song, X., & She, J. (2013). Boundary-constrained multi-scale segmentation method for remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing*, *78*, 15–25. https://doi.org/10.1016/j.isprsjprs.2013.01.002

8. APPENDIX

Precision recall curves



Tile2 precision recall curves



Tile5 precision recall curves