Automated Land Use and Land Cover Map Production: A Deep Learning Framework

by

Victor Alhassan

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

in the Department of Applied Computer Science

Automated Land Use and Land Cover Map Production: A Deep Learning Framework

by

Victor Alhassan

Supervisory Committee

Dr. S. Ramanna, Supervisor
(Department of Applied Computer Science)

Dr. C. J. Henry, Surpervisor
(Department of Applied Computer Science)

Dr. C. Bidinosti, Member
(Department of Physics)

Dr. C. Storie, External Member
(Dept. of Geography, University of Winnipeg)

Supervisory Committee

Dr. S. Ramanna, Supervisor
(Department of Applied Computer Science)

Dr. C. J. Henry, Surpervisor
(Department of Applied Computer Science)

Dr. C. Bidinosti, Member
(Department of Physics)

Dr. C. Storie, External Member
(Dept. of Geography, University of Winnipeg)

ABSTRACT

In this thesis, we present an approach to automating the creation of land use and land cover (LULC) maps from satellite images using deep neural networks that were developed to perform semantic segmentation of natural images. This work is important since the production of accurate and timely LULC maps is becoming essential to government and private companies that rely on them for large-scale monitoring of land resource changes. In this work, deep neural networks are trained to classify each pixel of a satellite image into one of a number of LULC classes.

The presented deep neural networks are all pre-trained using the ImageNet Large-Scale Visual Recognition Competition (ILSVRC) datasets and then fine-tuned using ∼19,000 Landsat 5/7 satellite images of resolution 224×224 taken of the Province of Manitoba in Canada. The initial results achieved was 88% global accuracy. Furthermore, we consider the use of state-of-the-art generative adversarial architecture and context module to improve accuracy. The result is an automated deep learning framework that can produce LULC maps images significantly faster than current

semi-automated methods.

The contributions of this thesis are the observation that deep neural networks developed for semantic segmentation can be used to automate the task of producing LULC maps; extensive experimentation of different FCN architectures with extensions on a unique dataset; high classification accuracy of 90.46%; and a thorough analysis and accuracy assessment of our results.

Keywords: Deep Learning, Land Use, Land Cover, Maps, Classification, Deep Convolutional Neural Networks, Satellite Images.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

As I conclude this research journey, I would like to express my heartfelt gratitude to Prof. Chris Henry and Prof. Sheela Ramanna, for their immense support and guidance from day one to this very end, without them, none of this would be possible. I consider it a privilege and blessing to be under the tutelage of two distinguished professors. To Prof. Chris Henry and Prof. Sheela Ramanna, I say a very big thank you!

My studies at the University of Winnipeg has been most rewarding and fulfilling because of the following organizations, administrators, instructors, colleagues, and friends. I'd like to thank the Queen Elizabeth Diamond Jubilee Scholarship Program for the huge financial support and the leadership and community engagement opportunities it offered me. I pass my gratitude to my former instructors Prof. Yangjun Chen, Prof. Simon Liao, Prof. Christopher Bidinosti and to all the other valued members of the Applied Computer Science Department including Ms. Connie Arnhold and Mr. Daniel Shepherd who all contributed to my success during the duration of my studies. Special thanks to our Faculty Dean Prof. Mavis Reimer and the Graduate Studies Officers Sandy Peterson, Ms. Deanna England, Ms. Dagmawit Habtemariam, Mr. Eric Benson, Allison Norris, the Faculty of Graduate Studies and the University of Winnipeg family. I would like to thank my colleagues Muthu Palaniappan, Mallikarjun Swamy, Damilola Aleshinloye, Andrew Curtis and Daeyoun Kim who all contributed to this research work. Many thanks to GeoManitoba, which is a department in the Manitoba provincial government for their support and contributions to this work. Also, I'd like to thank the examining committee including Prof. Ramanna, Prof. Henry, Prof. Bidinosti and Prof. Christopher Storie for all the time and effort they will donate in examining this work.

Finally, I'm grateful to my parents and family for sticking with me through thick and thin. Thanks to everyone else whom I haven't named that has been with me throughout this journey. Thank you city of Winnipeg, Manitoba, and Canada!

Victor Alhassan

DEDICATION

To God and my awesome Parents...

Chapter 1

Introduction

Due to the decreasing cost and increasing resolution of satellite images, many government and private industries are turning to Land Use and Land Cover (LULC) maps as an important tool for large-scale monitoring of land resource changes. These maps are vital in areas such as flood forecasting, urban and rural land-use planning, resource management, and disaster management and planning [1]. Land cover refers to natural assets present on the earth's surface (e.g. water bodies, vegetation, bare soil) while land use refers to man-made areas that utilize land cover resources (e.g. roads, agriculture, urban infrastructures). Satellite imagery or remote sensed imagery in general contain tangible information in the form of features which can be extracted to provide a basis for analysis and processing. The various satellite sensors (*e.g. SPOT, IRS, Landsat, IKONOS*) available today provide a wide variety of satellite data based on spatial, spectral and temporal features. The quality of features of satellite data are characterized by their spatial, spectral and temporal resolutions. For instance, each pixel may correspond to an area on the ground (e.g. $30m \times 30m$ satellite image means each pixel represents a small square on ground of that size). Spectral resolution, on the other hand, refers to the magnitude of light energy emitted from an object on the ground recorded in different frequency and *bands*.

The process of creating LULC maps is initiated by classifying features into land use or land cover types and is one of the most common applications of remote sensing. Machine learning, and in particular, deep learning has proven to be a major breakthrough for remote sensing [2]. Deep learning uses *deep* neural network (DNN) which are characterized by a neural network with many more layers and many more parameters than a traditional neural network [3]. The use of deep learning algorithms to solve previously existing problems has been increasingly popular and successful in the

last decade [4, 5]. Successful application of DNNs in various fields include robotics [6], speech recognition [7], image processing [8], natural language processing [9]. DNN applications in remote sensing and geoanalytics can be in categorized in the following four core areas : image preprocessing, pixel-based classification, target recognition, scene understanding. Recent surveys on deep learning for remote sensing can be found in [2, 10].

Deep Convolutional Neural Network (DCNN) pioneered by [11] is a variant of traditional neural networks where the inputs to the network are images. The term *Convolutional* implies that this network consists of layers performing the mathematical operation of convolution generally used in digital signal and image processing. Specifically, Deep Convolutional Neural Networks (DCNNs) have excelled in computer vision problems [12, 13]. The problem of semantic segmentation translates directly to the problem of LULC mapping in which satellite image pixels are classified into a number of LULC classes. LULC classification typically aims to assign global labels to scenes, in this process spatial information is discarded from the output. In contrast, LULC map production involves producing maps in which spatial information is duly represented to present contextual meaning. Research related to LULC classification and deep learning include [14, 15, 16], and those related to LULC map production include [17, 18, 19]. Most studies considering LULC map production employ traditional unsupervised, semi-supervised, supervised (per-pixel or object based) approaches like *K-means Clustering, Decision Tree and Maximum Likelihood Classifier* [20] and require considerable user input to ensure high classification accuracies. These semi-automated approaches can be prone to error, take a significant amount of training and classification time, and suffer from a lack of consistency – especially when multiple analysts are involved in the process and/or when dealing with large multi-scene areas. Moreover, it is difficult to improve accuracy and efficiency of these methods as they typically rely on pixel pattern matching and manual user input.

One of the most exciting ideas in deep learning in recent times is the notion of dueling Neural Networks proposed by Ian Goodfellow. This is known as the generative adversarial network (GAN). This architecture takes two neural networks and pits them against each other in a digital cat-and-mouse game[1]. In this thesis, a case-study exploring two architectures (DCNN and GAN) for LULC map production of Landsat 5/7 multi-spectral satellite image dataset is presented. Both spatial and spectral information of medium resolution ($30m$) and multispectral (*6 bands*) satellite

[1] https://www.technologyreview.com/lists/technologies/2018/

imagery are used.

1.1 Problem Definition

In [21, 22], fully convolutional network (FCN) originally developed by [23] for semantic segmentation was modified and adapted for automating the production of LULC maps on the Landsat 5/7 Manitoba satellite image data. Specifically, FCN-8 Visual Geometry Group (VGG-16), Conditional Random Fields (CRFs)-RNN, and Dilation 8 (both frontend and context) networks were used. The best global accuracy achieved was 88%. In this thesis, we consider the use of state-of-the-art generative adversarial network architecture for producing LULC maps from Landsat 5/7 Manitoba satellite image data to improve the accuracy of classification of LULC maps.

1.2 Proposed Approach

We take advantage of the ability of DCNNs to learn low-level and high-level features in a hierarchical topology to produce desired output. In the proposed process, each pixel in a satellite image must be classified into a number of LULC classes (*e.g.* deciduous forest, marshland, fens, *etc.*. This process directly maps to Semantic Segmentation which is understanding an image at a pixel level by associating each pixel to a number of predetermined classes (e.g. *chair, dog, cat*). Our approach presented here is based on the **Fully Convolutional Network** introduced in [23].

In this thesis, we use three DCNNs (VGGNet [24], GoogLeNet [25], ResNet [26]) referred to as *base networks* for LULC mapping and analysis (see, Fig 1.1). The networks examined in this paper were originally designed for image classification and object detection tasks. We adapt the base networks into FCNs (labelled as FCN-[VGG, ResNet, GoogLeNet]) that take arbitrary sized input and produce spatial output maps [23]. We introduce two specific extensions to the base networks: a context module and an adversarial extension. The context-module developed by [27] is a standalone plug-in that aggregates context information for better performance. This module was used to extend our *base networks* for improved performance. The second extension involves positioning the *base networks* in an adversarial setting [28] by adding a discriminator network. This ensemble of *base networks* and their extensions form our proposed deep learning framework. The *motivation* for this work is three fold:

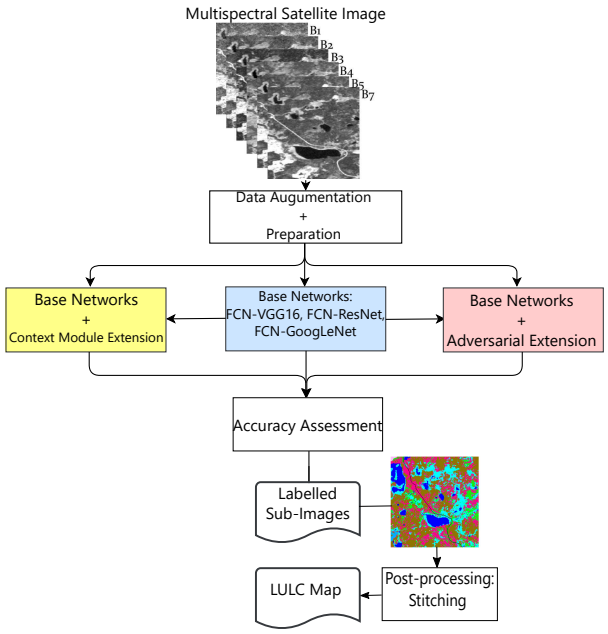


Figure 1.1: Overview of proposed deep learning framework for LULC mapping and analysis

- To propose a generalized deep learning framework to reduce the time spent by GeoManitoba on producing LULC maps.
- To investigate and compare deep learning architectures to obtain high classification accuracy.
- To observe the effectiveness of deep learning neural networks in remote sensing and geoanalytics application.

1.3 Contributions

Our contributions are the following:

- The observation that deep neural networks developed for semantic segmentation can be used to automate the task of producing LULC maps.
- Extensive experimentation of different FCN architectures with extensions on a unique dataset.
- High classification accuracy 90.46% achieved by our approach.

- A dramatic reduction in the time it take to produce these LULC maps (from 4800 hours to 8 minutes and 42 seconds).
- A thorough analysis and accuracy assessment on results of our deep learning framework both quantitatively and qualitatively.

1.4 Thesis Layout

The rest of this thesis is outlined as follows:

Chapter 2 describes the foundations of Neural Networks and Convolution Neural Networks in a limited scope, covering areas that relates to our proposed solution. It also includes the training process of deep learning networks.

Chapter 3 reviews relevant related works.

Chapter 4 covers the proposed *base networks* for LULC map production. It also includes our proposed novel extensions to the *base networks*.

Chapter 5 gives the implementation details discussing data acquisition, preparation, augmentation and the experimental setup.

Chapter 6 provides the results, analysis and comparisons based on quantitative and qualitative evaluations.

Chapter 7 concludes the thesis, summarizes the work done and provides possible future research directions.

Chapter 2

Foundations

In this chapter, we discuss the foundations of Neural Networks (NNs) describing briefly their loose relationship with biological processes and their abstraction and formulation. Subsequently we discuss the "Deep" Convolutional Neural Network, a variant of traditional NNs popularly used in computer vision tasks. We describe the layers present in a "Deep" Convolutional Neural Network and their functions. Lastly, we describe the learning and training process of deep learning models such as "Deep" Convolutional Neural Network.

2.1 Neural Network

A single layer neural network also called a *Perceptron* [29] shown in Fig. 2.1, considers a set of input and computes its weighted sum. The output of the weighted sum is passed through a linear/nonlinear function which acts as a threshold that triggers response in form of an output. Biological processes are often used as sources of inspiration for solving computational problems [30]. Neural networks (NNs) follow this narrative in that NNs are inspired by biological nervous systems. Although this connection is loose in that the NNs only share some key similarities with its biological counterparts, otherwise NNs are constricted to a mathematical abstraction. One such similarity is that the building blocks of both systems are computational devices or *neurons* that are highly interconnected, and the connections between neurons determine the function of the network [31].

In the perceptron diagram of Fig. 2.1, the inputs are denoted by $\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ with corresponding weights denoted by $\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n$. An applied bias is de-

noted by b and f connotes activation function, while y represents output. Subsequently, a perceptron is mathematically expressed as

$$y = f(\boldsymbol{w}_i \cdot \boldsymbol{x}_i + b). \tag{2.1}$$

From Eq. (2.1) f is a step function given as

$$f(\cdot) = \begin{cases} 1, & \text{if } \boldsymbol{w}_i \cdot \boldsymbol{x}_i + b > 0 \\ 0, & \text{otherwise,} \end{cases} \tag{2.2}$$

that defines behavior of the perceptron as a binary classifier.

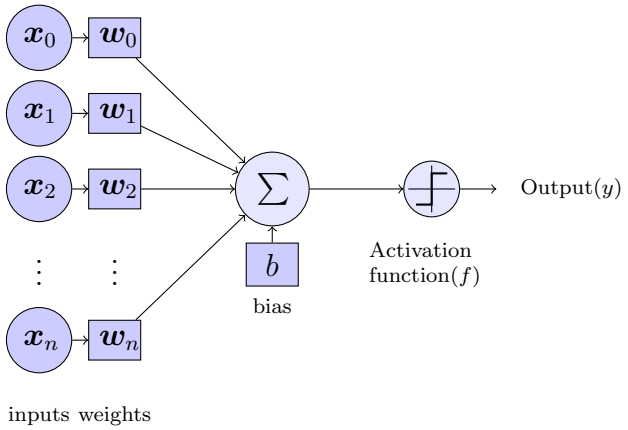


Figure 2.1: A single layer Neural Network also know as a Perceptron.

The perceptron can be trained iteratively to classify a set of inputs. A learning rule is applied to a perceptron by adjusting its weights and biases towards a particular goal. The perceptron learning rule was developed by Rosenblatt [29], and is presented in Algorithm 2.1.0 [32]. The single layer neural network provides intuition for solving very simple computations and can only classify linearly separable sets of vectors. The limitations of a single layer perceptron was pointed out by Minsky and Papert [33]. However, stacking multiple layers of perceptrons produces very powerful models able to tackle complex tasks and overcome the inadequacies of a single layer perceptron.

A multi-layer perceptron (MLP) shown in Figure 2.2 combines two or more perceptrons into a single framework. This multi-layer neural network consists of at least one hidden layer of neurons that is not directly connected to both the input nodes and output nodes [32]. This network is capable of approximating non-linear functions and thus finds more applications than a perceptron. The success in training

Algorithm 2.1.0 Perceptron learning algorithm.

1: Initialize weights $\boldsymbol{w}_i(0)(0 \leq i \leq N-1)$ and bias b to small random values. Here $\boldsymbol{w}_i(n)$ is the weight from input i at iteration n and b is external bias.

2: Activate perceptron by applying input $\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, ...\boldsymbol{x}_{N-1}$ with desired output $d(n)$

3: Calculate output y of the perceptron

$$y = f(\sum_{i=0}^{N-1} \boldsymbol{w}_i \boldsymbol{x}_i + b)$$

4: Update weights

$$\boldsymbol{w}_i(n+1) = \boldsymbol{w}_i(n) + \eta[d(n) - y(n)]\boldsymbol{x}_i(n)$$

where

$$d(n) = \begin{cases} 1, & \text{if } y = \boldsymbol{w}_i(n)\boldsymbol{x}_i(n) + b > 0 \\ 0, & \text{otherwise,} \end{cases}$$

η is learning-rate parameter, a positive constant less than unity.

5: Repeat step 2 for each input

MLPs stems from the backpropagation algorithm re-introduced in [34]. MLPs learning procedure assumes a surpervised learning paradigm in which they are trained on a set of input-target pairs using iterative optimization algorithms (e.g., *Gradient Descent*) to minimize differences (error) between target and predicted value. The error described as a cost function can be modeled in variety of ways (e.g., *Mean Square Error*) depending on the task. Backpropagation in itself is a differentiation algorithm for computing derivative of multivariate function (*gradients*). The learning process of MLPs using backpropagation algorithm incorporates two distinct stages; the first stage where error between predicted and target is propagated backwards through the network in order to evaluate derivatives and the second stage of weight adjustment using the calculated derivatives [35].

2.2 Deep Convolutional Neural Network

Deep Convolutional Neural Network (DCNN) pioneered by [11] is a variant of traditional NNs that builds upon MLPs. The term "Deep" connotes multiple layers and "Convolutional" represents layers preforming the mathematical operation convolution generally used in digital signal and image processing. Like its counterparts in

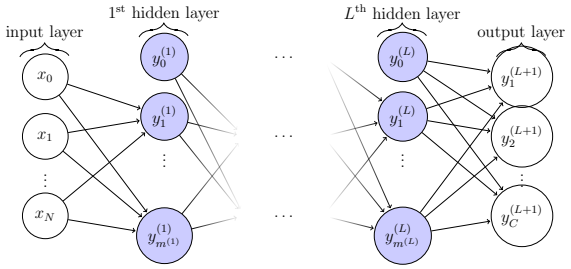
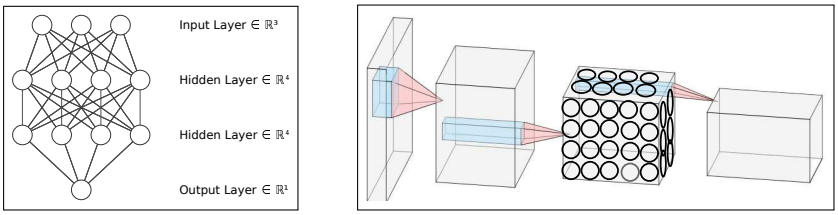


Figure 2.2: A multi-layer perceptron with N input units and C output units. The l^{th} hidden layer contains $m^{(l)}$ hidden units.



(a) (b)

Figure 2.3: (a). Traditional Neural Network (b). DCNN showing a 3D arrangement of neurons. A traditional neural network have all neurons fully connected from layer to layer while the neurons of DCNNs are connected to local regions from layer to layer [36].

Section 2.1, it is inspired by biological processes. DCNNs are structured differently from MLPs; they are designed to take input as images. Assuming a digital image with dimensions width (w), height (h), depth (d), the neurons present in the layers of a DCNN are arranged in 3 dimensions: $(h \times w \times d)$ (see *e.g.* Fig. 2.3). Considering high-dimensional input such as images, the fully connected neurons arrangement in MLPs becomes computationally expensive and impractical; DCNNs alleviates this problem with its arrangement of neurons connecting to local regions of input volume in which weights are shared [36]. These properties among others, makes DCNN invariant to geometrical transformations that may be present in image inputs.

The computational power of DCNN lies in their building components which consists of various layers performing different operations. Using ALexNet [12] DCNN model as a case study, we describe the building layers of DCNNs in detail in the following Sections.

2.2.1 Convolutional Layer

This layer performs convolution operations on input to yield output feature maps (see *e.g.* Fig 2.4). The local connectivity of the neurons present in this layer is characterized by the local receptive field of the neuron referred to as kernel/filter size [36]. The filter size is a hyper-parameter decision dependent on the architectural design of the DCNN. Given an image of any dimension size (e.g. $[227 \times 227 \times 3]$) we can compute the spatial dimension size of the output feature map following a convolution operation. Given the following parameters: Height of input image $H_i = 227$, Width of input image $W_i = 227$, Depth of input image $D_i = 3$, Number of Filters $K = 96$, Filter size $F = 11$, Stride S (i.e. the number of pixels the filter slide in each spatial dimension) $= 4$, Padding P (i.e the number of pixels (zeros) added to border of the input) $= 0$,

$$Width\ of\ output\ W_o = \frac{(W_i - F + 2P)}{S} + 1, \tag{2.3}$$

$$Height\ of\ output\ H_o = \frac{(H_i - F + 2P)}{S} + 1, \tag{2.4}$$

$$Depth\ of\ output\ D_o = K, \tag{2.5}$$

based on AlexNet architecture the resulting dimension size of the output feature map is $[55 \times 55 \times 96]$. The number of neurons in this layer is $55 * 55 * 96 = 290,400$. The weight sharing that occurs in this layer implies that the neurons in each depth slice uses the same weights and bias; this means $F * F * D_i$ weights per filter, for a total of $(F * F * D_i + 1) * K$ weights and biases (parameters) [36]. Thus, the amount of parameters present in this layer is $34,944$.

Briefly, we discuss other miscellaneous convolution operations *Dilated Convolution and Transpose Convolution* implemented in this thesis.

Dilated Convolutions

Dilated convolution also know as *atrous convolution* incorporates an additional hyperparameter called dilation "rate" d that inflates the filter described in Section 2.2.1 by inserting spaces between the filter elements, usually there are $d-1$ spaces inserted between kernel elements such that $d = 1$ corresponds to a regular convolution [37].

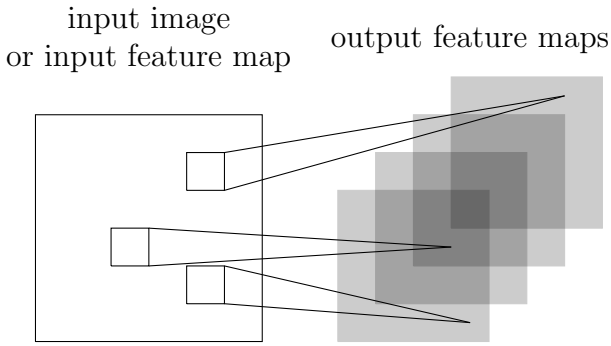


Figure 2.4: Illustration of a single convolutional layer. If layer l is a convolutional layer, the input image (if $l = 1$) or a feature map of the previous layer is convolved by different filters to yield the output feature maps of layer l.

We can reformulate the filter size F in this context as $\hat{F} = F + (F-1)(d-1)$. Substituting F in Eqs. (2.3) and (2.4) while following the convolution operation described in Section 2.2.1 effectively computes the dimension size of the output feature map in this context.

Transpose Convolution

Transpose convolution also known as deconvolution or fractionally strided convolution attempts to recover the spatial size of its initial input volume or feature map while performing a convolution operation. This is very important because it projects feature maps to a higher-dimensional space useful in Encoder-Decoder architectures described in Section 4.1. For instance, we attempt to recover the initial spatial resolution of the output feature map obtained in Section 2.2.1. By reworking Eqs. (2.3) and (2.4) into

$$Width\ of\ output\ \hat{W}_o = S(H_i - 1) + F, \tag{2.6}$$

$$Height\ of\ output\ \hat{H}_o = S(H_i - 1) + F, \tag{2.7}$$

where W_i & $H_i = 55$ equivalent to the output spatial $(H \times W)$ resolution size from Section 2.2.1, by plugging in the values we recover our initial input spatial resolution as $[227 \times 227]$.

2.2.2 Non-Linearity Function

The convolution layer is usually followed with a non-linear function which may or may not be described as a layer. Also known as an activation function, this function activates/fires neurons based on the satisfaction of its set conditions or properties. At the convolutional layer the output computed is a set of linear activations after which a non-linear function is applied to introduce non-linearity in the system. A popular non-linear function applied in DCNNs is *Rectified Linear Unit* (ReLU) introduced by [38]. The ReLU applies a function $f(x) = max(0, x)$ (see *e.g.* Fig 2.5) to all components of an input volume or feature map. Other non-linear activations functions that exists are logistic function *a.k.a* sigmoid, hyperbolic tangent (tanh) and Leaky ReLU [1]. The AlexNet model uses the ReLU activation function.

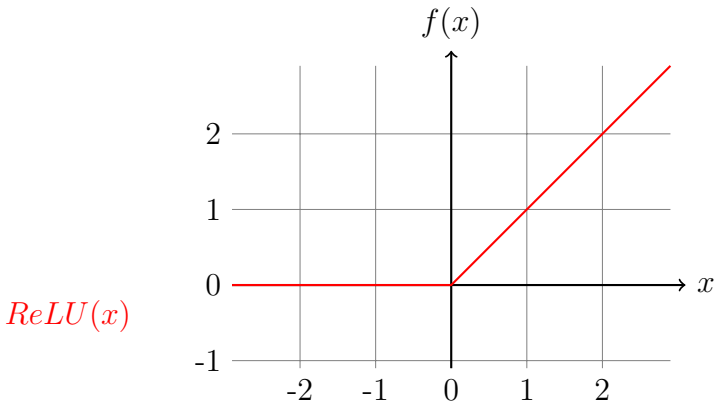


Figure 2.5: Visual representation of a ReLU activation function.

2.2.3 Pooling Layer

In this layer, sub-sampling is performed on input volume by using a pooling function which replaces components of an input volume or feature map at a certain location with a summary statistic (*Max, Sum or Average*) of the nearby elements. It could be used to perform spatial reduction on feature maps while preserving important details. The most common pooling operation is *Max Pooling* which computes the maximum output within a rectangular neighborhood [3] (see *e.g.* Fig 2.6), AlexNet model uses this operation. Pooling introduces the property of *translation invariance* to DCNNs.

[1] http://cs231n.github.io/neural-networks-1/

This means with pooling incorporated in a DCNN, it is able to detect geometrical shifts and translation of objects present in a feature map.

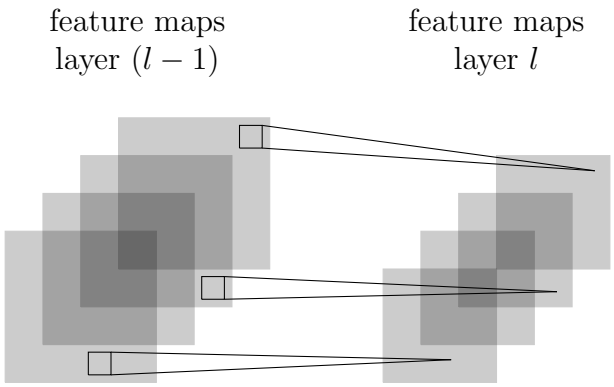


Figure 2.6: Illustration of a pooling and subsampling layer. If layer l is a pooling and subsampling layer and given $m_1^{(l-1)} = 4$ feature maps of the previous layer, all feature maps are pooled and subsampled individually. Each unit in one of the $m_1^{(l)} = 4$ output feature maps represents the maximum within a fixed window of the corresponding feature map in layer $(l-1)$.

Other layers typically implemented in deep convolutional "classification" networks such as AlexNet are *Dropout and Fully-Connected Layers*.

Dropout Layer

A DCNN model must be able to perform well on previously unseen inputs, this is called *generalization*; usually this is not always the case with *Overfitting and Underfitting* factors posing as problems to this property [3]. *Underfitting* occurs when the model fails to capture the underlying distribution of the training set, while *Overfitting* occurs when the model fits too well to the training set and fails to generalize on test data previously unseen. Both factors lead to poor results. In order to tackle overfitting a *regularization* technique is employed. Regularization in this context is any modification made to a learning algorithm or model in order to prevent the risk of overfitting. A Dropout layer is introduced as a regularization technique to tackle overfitting. The central idea to this approach involves "dropping out" neurons along with their connections to reduce co-adaptations of neuron units [39].

Fully-Connected Layers

This layer basically replicates a traditional neural network as neurons are fully interconnected to each other. They can better summarize information processed by lower-level layers in view of final output [14].

2.3 Training

Similarly, to MLPs described in Section 2.1 the learning procedure of DCNNs involves adjusting weights based on a loss or cost function to approximate a target function. This training and learning process proceeds in a number of steps; formulating a loss function to model differences between approximate (predicted) and target, choosing an iterative optimization algorithm to minimize error, random initialization of weights and finally backpropagation of errors to adjust weights accordingly.

2.3.1 Loss Function

The formulation of a loss function is implied based on the specific task of interest. DCNNs are often used for classification tasks (two class (binary) or multi-class classification problems). Common loss functions include mean squared error (L2 loss) and cross entropy loss functions. Given n number of classes, $\hat{\mathbf{y}}$ is a vector of n predictions generated from input vector $\mathbf{x}$ and $\mathbf{y}$ is the vector of n true labels, the mean squared error(MSE) loss function ℓ_{mse} is computed as

$$\ell_{mse}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2, \tag{2.8}$$

similarly, cross entropy loss function ℓ_H is computed as

$$\ell_H(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{n} y_i \ln(\hat{y}_i). \tag{2.9}$$

The majority of popular DCNNs such as AlexNet used in image classification employ the cross entropy loss function.

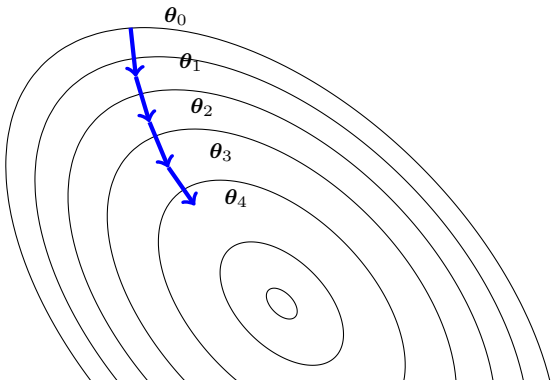


Figure 2.7: Illustration of gradient descent given a sequence of $\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \ldots, \boldsymbol{\theta}_{n+1}$ assuming $\boldsymbol{\theta}_*$ as stationary point which represents a local minimum in which $\ell = 0$.

2.3.2 Optimization

After defining a loss function, the next step to training a model is optimization which involves minimizing the loss function. *Gradient Descent* is an iterative optimization algorithm popularly deployed in machine learning/deep learning whose goal is to find optimal parameters weights denoted as $\boldsymbol{\theta}$ for minimizing a loss function. Given a loss function $\ell(\boldsymbol{\theta})$ parameterized with $\boldsymbol{\theta}$ the partial derivative is computed as $\frac{\partial \ell}{\partial \boldsymbol{\theta}}$ showing the slope of the loss function with-respect-to (w.r.t) to $\boldsymbol{\theta}$ (see, *e.g.* Fig 2.7). In order to minimize ℓ, we move/step in the direction of the negative gradient. The gradient of ℓ w.r.t $\boldsymbol{\theta}$ is the vector containing all of the partial derivatives, denoted by $G = \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta})$. Subsequently, given a learning rate η which is a positive scalar determining the movement size [3] the gradient descent is computed as

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}).$$

There are variations of the gradient descent algorithm depicted in Algorithm 2.3.1 [40] that builds upon its basic implementation (*e.g.* Batch Gradient Descent, Stochastic Gradient Descent, Mini-batch Gradient Descent). Furthemore, in deep learning various gradient based optimizing algorithms tailored for deep learning implementations have been put forward, [41] provides an overview on such techniques. A popular optimization algorithm employed in deep learning is the *Adam Optimizer* [42].

Algorithm 2.3.1 The general gradient descent algorithm.

Input: Given initial parameters $\boldsymbol{\theta_0}$ and number of iterations T
Output: final parameters $\boldsymbol{\theta}_*^{(T)}$

1. **for** $t = 0$ **to** $T - 1$
2. estimate $\frac{\partial \ell}{\partial \boldsymbol{\theta}_{n+1}}$
3. compute $\boldsymbol{\theta}_{n+1}^{(t)} = \boldsymbol{\theta}_n - \nabla_{\boldsymbol{\theta}_n} \ell(\boldsymbol{\theta}_n)$
4. select learning rate η
5. $\boldsymbol{\theta}_{n+1}^{(t+1)} := \boldsymbol{\theta}_n^{(t)} + \eta \Delta \boldsymbol{\theta}_n^{(t)}$
6. **return** $\boldsymbol{\theta}_*^{(T)}$

2.3.3 Intialization

In deep learning the difference as to whether an optimization algorithm converges to a global minimum (a point of lowest absolute error) or diverges is very much dependent on the parameters initialization. Some techniques used for initializing parameters are described in [43]. Random initialization from a uniform distribution $U[-\sigma, \sigma]$ where σ is a chosen hyper parameter is widely used and it is the preferred initialization technique employed in this thesis work.

2.3.4 Backpropagation

Backpropagation briefly introduced in Section 2.1 is the quintessential means of computing the gradients of a deep learning model. Backpropagation algorithm incorporates an optimizing algorithm to form the learning procedure of a model. We give an intuition into backpropagation below folllowing [35, 40].

Take for example a layer l of a model that computes an approximate function $y_i^l = f(\boldsymbol{\theta}_i^l \boldsymbol{x}_i^l)$ given a single parameter $\boldsymbol{\theta}_i$ whose derivative is desired, and input $\boldsymbol{x}_i^l$ whose derivative is not required. Parameterizing ℓ loss function with respect to $\boldsymbol{\theta}_i^l$ gives $\ell(\boldsymbol{\theta}_i^l)$. We can compute the gradient of $\ell(\boldsymbol{\theta}_i^l)$ applying chain rule:

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_i^l} = \frac{\partial \ell}{\partial y_i^l} \frac{\partial y_i^l}{\partial \boldsymbol{\theta}_i^l}. \tag{2.10}$$

We define errors δ to be

$$\delta_l \equiv \frac{\partial \ell}{\partial y_i^l}, \tag{2.11}$$

Equation (2.10) can be applied recursively to evaluate the δ for hidden layers as follows:

$$\delta_l \equiv \frac{\partial \ell}{\partial y_i^l} = \sum_{l+1} \frac{\partial \ell}{\partial y_i^{l+1}} \frac{\partial y_i^{l+1}}{\partial y_i^l}, \tag{2.12}$$

the computation runs for all layers $l+1$. Given derivative f' of the non-linearity f backpropagation formula is formulated as

$$\delta_l = f'(y_i^l) \sum_{l+1} \theta_i^l \delta_{l+1}. \tag{2.13}$$

The error backpropagation algorithm depicted in [40] is reproduced here in Algorithm 2.3.1.

Algorithm 2.3.1 Error backpropagation algorithm for a layered neural network represented as computation graph $G = (V, E)$.

(1) For a sample $(\boldsymbol{x}_n, \boldsymbol{y}_n^*)$, propagate the input $\boldsymbol{x}_n$ through the network to compute the outputs $(\boldsymbol{v}_{i_1}, \ldots, \boldsymbol{v}_{i_{|V|}})$ (in topological order).

(2) Compute the loss $\mathcal{L}_n := \mathcal{L}(\boldsymbol{v}_{i_{|V|}}, \boldsymbol{y}_n^*)$ and its gradient

$$\frac{\partial \mathcal{L}_n}{\partial \boldsymbol{v}_{i_{|V|}}}. \tag{2.14}$$

(3) For each $j = |V|, \ldots, 1$ compute

$$\frac{\partial \mathcal{L}_n}{\partial \boldsymbol{w}_j} = \frac{\partial \mathcal{L}_n}{\partial \boldsymbol{v}_{i_{|V|}}} \prod_{k=j+1}^{|V|} \frac{\partial \boldsymbol{v}_{i_k}}{\partial \boldsymbol{v}_{i_{k-1}}} \frac{\partial \boldsymbol{v}_{i_j}}{\partial \boldsymbol{w}_j}. \tag{2.15}$$

where $\boldsymbol{w}_j$ refers to the weights in node i_j.

Chapter 3

Related Works

This chapter gives an overview of deep learning and popular modern architectures used in solving problems that cut across various domains. Subsequently, deep learning based works for semantic segmentation is discussed since it forms the basis of LULC map production explored in this thesis. Furthermore, works directly related to the problem considered in this thesis are presented.

3.1 Deep Learning

The term *deep* neural networks (DNN) characterizes a neural network with many more layers and many more parameters than a traditional neural network. DNNs exhibit a massively parallel structure which lend themselves naturally to efficient parallel implementations [44]. The recent success of DNNs is fuelled in part by graphics processing units (GPU) ability to implement parallel computations with speed and efficiency. The work presented here is inspired by the successful application of DNNs in various fields: robotics [6], speech recognition [7], image processing [8], natural language processing [9]. The remote sensing community can also benefit from DNNs to solve complex problems because of its ability to learn low-level and high-level features in a hierarchical manner. Zhang et al [10] summarizes DNN applications in remote sensing and geoanalytics in four core areas: 1) image preprocessing, 2) pixel-based classification, 3) target recognition, and 4) scene understanding. DNNs in this work are applied to the problem of pixel-based classification.

3.1.1 Modern Architectures

In 2012, a deep learning model named *AlexNet* [12] was created and used for image processing tasks, this model won the 2012 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [13] and largely popularized deep convolutional neural networks (DCNNs). AlexNet builds on the insight of LeNet [11] which is widely recognized as the first convolutional neural network (CNN). DCNNs are a variant of DNNs modelled after the visual cortex of a cat [45]. DCNNs make an explicit assumption the network input is an image. This assumption is reflected in the network architecture of DCNNs. For instance, the neurons present in a convolution layer of a DCNN are arranged in three dimensions: width, height, depth; where each neuron is connected to a local region in the input volume and all neurons share the same weights. This arrangement is far more efficient in terms of memory and computation concerns in contrast to traditional NNs where each neuron is connected to each neuron in the previous layer and has its own set of weights [36].

A DCNN consists of a sequence of layers, very similar to DNNs, that uses convolution in place of general matrix multiplication in at least one of their layers [3]. For example, AlexNet consists of five convolutional and max-pooling layers, and three fully-connected layers with a final 1000-way softmax layer that completes the model [36]. The success of AlexNet inspired the creation of many other DCNNs in an attempt to win subsequent ImageNet challenges and improve accuracy. Often these networks achieved better results by building deeper networks. As a result, ensuing architectures all have many more layers.

Simonyan and Zisserman [24] proposed a uniform DCNN architecture called VGGNet containing 16-19 weight layers as compared to AlexNet with 8 layers. Its layers are arranged in a top to bottom topology that consists of stacked (3×3) convolution layers and (2×2) pooling layers followed by three fully-connected layers. Accordingly, this simple and uniform design proves to be effective as a feature extractor as evidenced in our results (see *e.g.*, Chapter 6). Similarly, [25] proposed a novel architecture called GoogLeNet that goes even deeper. This network introduces a network topology design for its layers described as an *inception module*. Inception modules add a different complexity to the network design that generally strays from the tradition of stacking layers in a uniform structure. The design of the inception module is described in Chapter 4. GoogLeNet uses 9 inception modules stacked on each other in addition to occasional max-pooling layers with a stride of 2. One of the goals of

this architecture was to reduce the amount of computation, which is evidenced by the fact that it has less parameters than AlexNet and VGGNet. GoogLeNet won the 2014 ILSVRC for classification in which VGGNet was runner-up. One downside of GoogLeNet is that the large amount of layers causes the network to fit too well to the training data, thereby loosing its ability to generalize to new samples not contained in the training set. This problem is referred to as *overfiting*. In particular, GoogLeNet network did not scale well to our task (see *e.g.*, Section 6).

Intuitively, one may assume that increasing the depth of DCNN should improve performance. However, in practice this is not the case. During back-propagation for these large networks, the computed gradients of the loss function (which are used to update layer weights) begin to vanish in the early layers of the network, which results in poor performance. This problem was observed in [26] and subsequently the authors proposed a residual learning framework that takes into consideration identity mappings from previous layers. Typically, other DCNNs do not take into account residual input from previous layers but simply learn a direct mapping of input x to output y with function $H(x)$. The authors propose Residual Network (ResNet) conditioned to learn the residual function $F(x) = H(x) - x$. The residual function is recast into $H(x) = F(x) + x$ and ResNet tries to learn this instead of a direct mapping $H(x) = x$. To this end, the authors hypothesize that it is easier to optimize the residual mapping than to optimize the direct unreferenced mapping. ResNet won the 2015 ILSVRC on classification. VGGNet, GoogLeNet and ResNet forms an integral part in this thesis.

3.2 Deep Learning for Semantic Segmentation

Pixel-wise classification tasks, such as semantic segmentation, aim to understand images at the pixel level. Semantic segmentation of digital mages involves assigning classes (*e.g.* road, grass, cat, dog) to individual pixels in an image. The goal is to cluster image pixels that belong to the same perceptual objects within the image thereby giving contextual meaning to the pixels. Segmentation algorithms that generally do not use DNNs are referred to as traditional approaches. Thoma [46] gives an overview of traditional *unsupervised* methods for segmentation, including k-means, decision forests and support vector machine algorithms. Modern DCNN architectures such as AlexNet [12], VGGNet [24] and GoogLeNet [25] have attained remarkable success in image classification tasks, and based on this achievements Long

et al [23] adapts modern DCNN architectures (AlexNet, VGGNet, GoogLeNet) into fully convolutional networks (FCN) for use in semantic segmentation.

Typically, classifier networks (such as AlexNet, VGGNet, GoogLeNet) take fixed-sized inputs and produce non-spatial outputs. This means the fully connected layers present in this networks have fixed dimensions that do not relate to the original spatial coordinates of the input image. For the purpose of semantic segmentation, Long et al [23] cast the fully connected layers into fully convolutional layers, *i.e.* the network can take input of any size and produce spatial output maps. However, the output maps produced are coarse due to the sub-sampling layers present in the network. In order to solve this problem, the authors define skip connections that combines deep feature-rich coarse maps with appearance information from shallow layers of the network. The result is a network able to produce more accurate and detailed semantic samples. Three skip architectures (FCN-32s, FCN-16s, FCN-8s) combine information from different shallow layers of the network producing finer output maps that contain high-level information. These output maps are then up-sampled by transpose convolutions to the original resolution of the input image [37]. Long *et al.* [23] train the FCNs through the use of transfer learning [47], which is the process of using a network trained on a larger dataset (in this case on ImageNet), and then fine tuning it with a smaller dataset.

Yu and Koltun [27] adapts the VGG-based network proposed in [23] by removing pooling and striding layers and making heavy use of dilated convolutions in subsequent layers. Furthermore, a context module that uses dilated convolutions to systematically aggregate multiscale-contextual information while retaining resolution was introduced. In addition results are reported in [22] based on the architectures described in [27].

CRFasRNN, described in [48], tackles dense pixel prediction using Conditional Random Fields (CRFs). When used in the context of pixel-wise label prediction, CRFs models pixel labels as random variables that form a Markov Random Field (MRF) when conditioned upon an image. More specifically, they formulate each step in an iteration of the mean-field algorithm [49] as stack of CNN layers in a dense CRF. The result is that the iterative mean-field inference is considered as a Recurrent Neural Networks(RNN). Furthermore, this formulation is combined with FCN-8s of [23] and trained end-to-end. Results are also reported in [22] based on CRFasRNN [48].

DeepLab v2 [50] explores the use of convolution with up-sampled filters called *atrous convolution* to enlarge the field of view without increasing the number of param-

eters, which is the same behaviour as the dilated convolutions in [27]. Furthermore, atrous spatial pooling (ASPP) is introduced for multi-scale processing, and CRFs are used as a post-processing step to improve localization performance. DeepLab v3 [51] further improves on the later. Another popular architecture for segmentation is Unet [52] which builds on FCNs [23] for bio-medical segmentation.

3.3 Deep Learning for LULC Map Production

As in many domains, the success of DNNs has prompted researchers to use them for problems in the field of remote sensing. A particular task of interest in the remote sensing community is LULC map production. The authors of [17] proposed an approach of using FCNs for classification of high resolution remote sensing imagery into a number land use/cover classes. The FCN model proposed in this approach uses dilated convolutions and is modified for multi-scale classification. Furthermore, this approach incorporates CRFs in a post-processing step which takes into account more spatial cues with the end goal of improving accuracy. The work of [18] decouples the task of land use and land cover production individually and tackles each task separately. For land cover classification of multi-spectral remote sensing imagery, the authors adopts SegNet model [53] and compares different variants, in contrast LiteNet model [54] and other variants is used for land use classification. Remote sensing imagery can be categorized based on spatial and spectral resolution, [19] classified hyperspectral remote sensing imagery to produce LULC maps. Typically, this type of satellite images contains 10s to 1000s of bands, the authors, in this case, proposed a deep learning framework that includes a Deep Belief Network(DBN) which learns deep representations and CRFs that considers spatial information trained end-to-end similar to the approach of [48] for LULC classification. Similarly, Alam et al [55] incorporate CRF with CNN into a common framework for hyperspectral image segmentation.

The pixel-wise classification method of satellite imagery presented in this paper produces LULC maps. In contrast there exist methods that assign global labels to satellite imagery for LULC classification. The difference between both processes lies in the output, LULC maps production retains spatial resolution while generic LULC classification does not, usually the latter is a step before LULC map production. In this light, it is worth mentioning such related works different from a map production approach that also use deep learning.

Castelluccio et al [14] explored the use of DCNNs for classification of remote sensing scenes using two contemporary architectures CaffeNet and GoogleNet, Two datasets UC-Merced and Brazilian Coffee Scenes were considered, each dataset with unique features. UC-Merced spatial and spectral characteristics (high resolution, low level features, and RGB color space) were closely matched to general optical images while Brazilian Coffee Scenes includes a special band- Near Infrared band (NIR) typically found in remote-sensing data. CaffeNet and GoogleNet were implemented to classify both datasets independent of each other UC-Merced (21 classes), Brazilian Coffee Scenes (4 classes). Results from both models outperformed other state of the art classical techniques paired with the same data and classification problem.

Basu et al [15] proposed a novel classification framework for classifying satellite images called DeepSat. There were 150 features extracted from two datasets (SAT-4 and SAT-6) containing four bands - Red, Green, Blue and NIR. Some features extracted include energy, entropy, homogeneity, contrast, maximum probability, saturation, intensity, and image channels. All features were normalized to lie in the range [0, 1] before being fed to a Deep Belief Network(DBN) classifier trained using Contrastive Divergence algorithm [56]. This network outperforms the classical DBN on the target dataset.

Marmanis et al [16] explore a system of extracting representations from DCNNs pretrained on ImageNet dataset for classification of remote sensing images. The system follows a two stage classification scheme. In the first stage, original training data is fed into a pretrained DCNN model. Information obtained from a set of deep activations in the last layers of the pretrained DCNN is then fused to a single vector reshaped into a 2-D array. In the second stage, this information is received by a CNN supervised classifier with labels to classify images. This system has three positive implications; richer information obtained in the deeper layers of pretrained networks contributes to higher classification accuracy, information fusion from different layers influences accuracy since multiple scales of relevant information exists at these layers, by reshaping the single vector into 2-D array a reduction of parameters occurs and features are better processed by the CNN classifier.

Chapter 4

Base Networks for LULC Map Production

This chapter introduces three popular DNN architectures which we refer to as *base networks*. Specifically, VGGNet [24], GoogLeNet [25] and ResNet [26] classification networks play an important role in the solution proposed here as they are used as the base networks for the semantic segmentation approach defined in thesis. Furthermore, modifications and extensions to the networks based on the problem considered in this thesis are discussed.

4.1 Modifications to Base Networks

The base networks (*VGGNet, GoogLeNet, ResNet*) were originally designed for the purpose of image classification, object detection and are identified as classifications networks that produce non-spatial outputs. They can be re-purposed for semantic segmentation task by re-interpreting fully connected layers as fully convolutional layers [23]. Here, these base networks are adapted into FCNs (labelled as FCN-[VGG, ResNet, GoogLeNet]) that take arbitrary sized input and produces semantic maps [23]. This adaptation is rather trivial and the output maps produced are coarse with spatially reduced dimension size. To produce good representative semantic maps, the coarse output is passed through a stack of transpose convolution layers which increases dimension size and connects coarse output to dense pixels [23]. In this design, the first part of the network (*i.e.* the base network) is referred to as an encoder that acts a feature extractor encoding input information into a compressed vector,

and the second part is considered a decoder that performs upsampling of the compressed vector to match input spatial dimension. This network structure is depicted in Fig. 4.1.

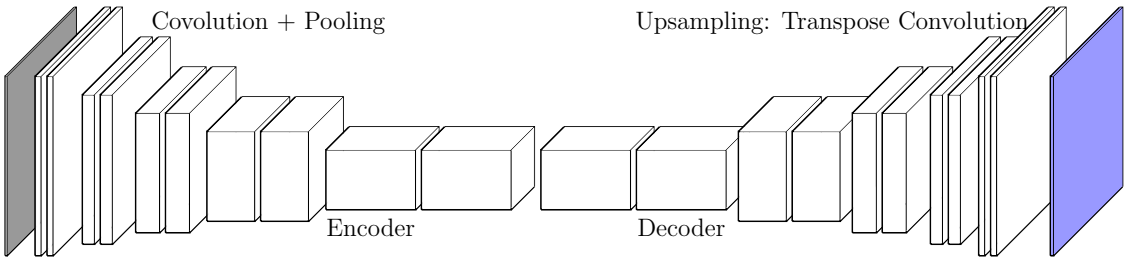
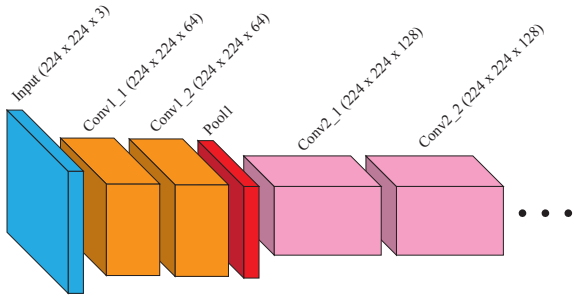


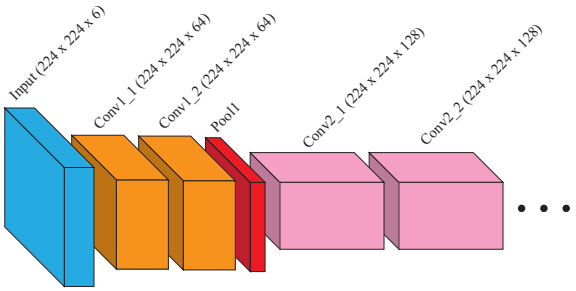
Figure 4.1: FCNs: Encoder-Decoder Architecture

The original base networks are structured to take arbitrary sized input ($H \times W \times 3$) suited for general-purpose optical images that have three channels (depth) red, green and blue (RGB). The Landsat 5/7 satellite images used in this work are intrinsically different in that they contain six channels: red, green, blue, and three infrared. Due to this fact, the input layer of the original base networks were modified to take input as ($H \times W \times 6$). Originally, the base networks were trained on the ImageNet dataset containing ∼1.2 million RGB-based images. We take weights of the base networks that were pre-trained on the ILSVRC and double the number of parameters on the first layer and initialize these new weights randomly from a uniform distribution. In other words, only weights for each depth slice in this layer doubles. For instance, consider a network where the first convolution layer has dimensions ($3 \times 3 \times 3 \times 64$). By doubling weights of the depth slice, the layer dimensions become ($3 \times 3 \times 6 \times 64$). Note, only the depth slice weights were modified to accommodate three additional channels.

All other parameters remain the same, for instance assuming the first layer (i.e the input image) is of dimension size $224 \times 224 \times 3$ doubled to $224 \times 224 \times 6$, the number of neurons does not increase because the subsequent layer dimensions remain $224 \times 224 \times 64$ only the number of weights for this layer doubles since $3 \times 3 \times 6 \times 64 \sum 64 = 3520$ (see e.g. Fig 4.2) . In addition, all fully-connected layers were removed from the original base networks, which follows the approach described in [23]. In other words, these classifiers were adapted for dense prediction and up-sampling using transpose convolutions.



(a)



(b)

Figure 4.2: Example of network modifications performed in this work. (a) First six layers of the FCN-8 using RGB image input (*i.e.* 3 bands), and (b) first six layers of the FCN-8 with Landsat 5/7 (*i.e.* 6 bands).

4.2 FCN-VGG

The VGG network architecture was named after the Visual Geometry Group at the University of Oxford. It is a popular network that secured second position in the ILSVRC14 classification task [24]. In this work, the VGG-16 layer net is re-purposed for semantic segmentation (in the same manner as Long et al [23]). VGGnet consists of convolution layers, max-pooling layers, and fully convolutional layers. In regards to VGG-16, the encoder part of the network has 16 layers while the decoder part has 3 layers. The convolution layers are stacked top to bottom to receive corresponding input from each layer, and the convolution kernel size is fixed with varying depth $(3 \times 3 \times X)$. Max-pooling layers are positioned in between convolution layers to down-sample input coming from the previous layer. The max pooling kernel size is fixed at $(2 \times 2 \times X)$, where the depth size is the same as the previous layer. Fully convolutional layers appear at the end that produce feature maps, called sore maps, containing contextual meaning. Additionally, score maps are passed through a stack of up-sampling layers which increase dimensionality of the output to match

original input. This final process is called transpose convolution with strides [37]. To summarize, the same FCN-8 architecture described by [23] is used in this work with only a modification of the first layer as described above. The network is fully illustrated in Table 4.1, which shows operations across all layers of the FCN-VGG-16 network.

Table 4.1: Input/output characteristics of the FCN-VGG-16 network architecture used in this work.

layer name	kernel size / stride	output size
input(224 x 224 x 6 satellite image)		
conv1.x	{3 x 3, 64 / 2} x 2	(224 x 224 x 64)
pool1	{2 x 2 / 2}	(112 x 112 x 64)
conv2.x	{3 x 3, 128 / 1} x 2	(112 x 112 x 128)
pool2	{2 x 2 / 2}	(56 x 56 x 128)
conv3.x	{3 x 3, 256 / 1} x 3	(56 x 56 x 256)
pool3	{2 x 2 / 2}	(28 x 28 x 256)
conv4.x	{3 x 3, 512 / 1} x 3	(28 x 28 x 512)
pool4	{2 x 2 / 2}	(14 x 14 x 512)
conv5.x	{3 x 3, 512 / 1} x 3	(14 x 14 x 512)
pool5	{2 x 2 / 2}	(7 x 7 x 512)
conv_fc.x	{1 x 1, 4096 / 1} x 2	(7 x 7 x 4096)
conv_final	{1 x 1, 19 / 1}	(7 x 7 x 19)
upsampling layers (transpose convolutions with stride)		
tconv_fuse_pool4	{4 x 4, 512 / 2}	(14 x 14 x 512)
tconv_fuse_pool3	{4 x 4, 256 / 2}	(28 x 28 x 256)
tconv_final	{16 x 16, 19 / 8}	(224 x 224 x 19)

4.3 FCN-ResNet

The ResNet architecture introduced residual connections between layers of deep convolutional networks allowing network depths to be increased. This network won 1st place in the ILSVRC14 classification task [26]. In this work, a 101 layer deep net-

work variant of ResNet (ResNet-101) is adapted for dense pixel classification. The convolutional layers in this network are defined in a bottleneck architecture that has a kernel size of $(1\times1\times X), (3\times3\times X), (1\times1\times X)$ stacked together. For each residual connection a stack of convolutional layers is defined, where the $(1 \times 1 \times X)$ kernels are responsible for dimensionality reduction and restoration. With this bottleneck design, max pooling layers are eliminated in between convolutional layers. Furthermore, in this work, the last average pooling layer and fully connected layer are removed. The output score maps are passed through a stack of up-sampling layers with skip connections from previous layers (again following the approach of [23]). Otherwise, the ResNet-101 base architecture (without fully connected layer and softmax classifier) described in [26] is retained, and only the first layer modified. The network architecture is fully defined in Table 4.2

Table 4.2: Input/output characteristics of the FCN-ResNet-101 network architecture used in this work.

layer name	kernel size / stride	output size
input(224 x 224 x 6 satellite image)		
conv1	{7 x 7, 64 / 2} x 2	(112 x 112 x 64)
maxpool	{3 x 3 / 2}	(56 x 56 x 64)
conv2.x	{1 x 1, 64 — 3 x 3, 64 — 1 x 1, 256} x 3	(56 x 56 x 256)
conv3.x	{1 x 1, 128 — 3 x 3, 128 — 1 x 1, 512} x 4	(28 x 28 x 512)
conv4.x	{1 x 1, 256 — 3 x 3, 256 — 1 x 1, 1024} x 23	(14 x 14 x 1024)
conv5.x	{1 x 1, 512 — 3 x 3, 512 — 1 x 1, 2048} x 3	(7 x 7 x 2048)
conv_fc	{1 x 1, 19}	(7 x 7 x 19)
upsampling layers (transpose convolutions with stride)		
tconv_fuse_conv4.x	{4 x 4, 1024 / 2}	(14 x 14 x 1024)
tconv_fuse_conv3.x	{4 x 4, 512 / 2}	(28 x 28 x 512)
tconv_final	{16 x 16, 19 / 8}	(224 x 224 x 19)

4.4 FCN-GoogLeNet

GoogLeNet, also called *Inception*, arranges the operational layers in a network topology in which multiple convolution layers (with pooling) are structured into modules.

This novel network structure, proposed by [25], won 1st place in the ILSVRC15 classification task. Although new improvements [57, 58] have been made on the original Inception network [25], the very first version with 22 layers was used in this work for dense pixel classification in order to simplify implementation. Each module contains multiple convolution layers with kernel sizes $(1 \times 1 \times X)$, $(3 \times 3 \times X)$, $(5 \times 5 \times X)$ and $(3 \times 3 \times X)$ and max pooling layer connected in parallel. This network is shown in Fig. 4.3. The $(1 \times 1 \times X)$ convolution kernels performs dimension reduction along the depth vector of the input, reducing the number of parameters, to make the network computationally efficient.

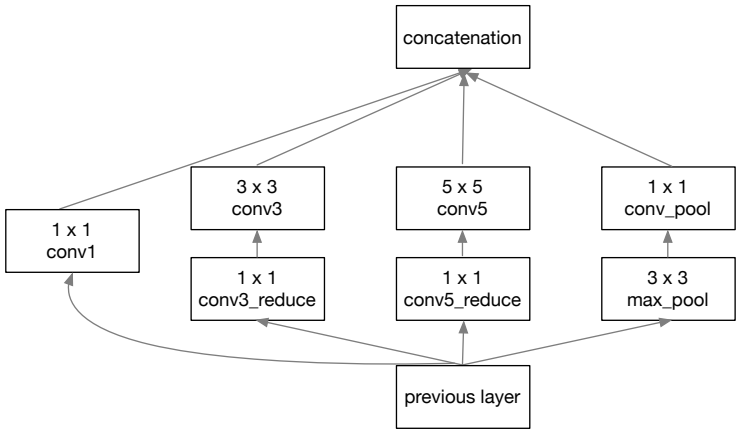


Figure 4.3: Architecture Design of Inception Module

Multiple operational layers in the module learn discriminative patterns of the feature maps, and, at the end, all resulting features maps from the parallel connections are concatenated. Just as with the previous networks, the original architecture was used for this work. Again, the first layer of the network was modified for 6-channel input, and the last layers of the network, including the average pooling layer, linear layer and softmax classifier were removed before passing the output from the last inception module into a stack of up-samling layers with skip connections following the approach of [23]. Table 4.4 shows details of the network structure.

4.5 Extensions to Base Networks

Extensions were made to the base networks to further improve accuracy. Specifically, two methods are presented with the sole aim of improving performance. Each method is tested individually for each base network, subsequently an ensemble of both meth-

ods is tested for each base networks. The results achieved from the extensions is a major improvement to our previous results reported in [22] and adds to the performance of base networks presented (see *e.g.*, Section 6). In this light, we present both extensions in the following subsections.

4.5.1 Context Module Extension

The first extension was to add a vestige plugin called a context module [27]. This module uses dilated convolutions to systematically aggregate multi-scale contextual information without losing resolution. This module is appended to the end of the base networks and takes up-sampled feature maps as input and passes them through a series of dilated convolution layers. This module proved to be effective in improving accuracy (as reported in Section 6). The architecture of this module is summarized in Table 4.3.

Table 4.3: Input/output characteristics of the context module used in this work

layer name	kernel size / dilation	output size
input(224 x 224 x 19 feature maps)		
ctx_1	{3 x 3, 38 / 1}	(224 x 224 x 38)
ctx_2	{3 x 3, 38 / 1}	(224 x 224 x 38)
ctx_3	{3 x 3, 76 / 2}	(224 x 224 x 76)
ctx_4	{3 x 3, 152 / 4}	(224 x 224 x 152)
ctx_5	{3 x 3, 304 / 8}	(224 x 224 x 304)
ctx_6	{3 x 3, 608 / 16}	(224 x 224 x 608)
ctx_7	{3 x 3, 608 / 1}	(224 x 224 x 608)
ctx_8	{3 x 3, 19 / 1}	(224 x 224 x 19)

4.5.2 Adversarial Extension

The semantic segmentation networks were further extended by adding an adversarial network. Specifically, each of the FCN-[VGG, ResNet, GoogLeNet] networks were trained alongside a discriminator network that discriminates between ground-truth and predicted output (classes). This process mimics a GAN [28] architecture. To state this another way, our FCN (*i.e.* either FCN-[VGG, ResNet, GoogLeNet])

is redefined as a generator and is combined with a convolutional neural network-based discriminator to form a GAN. The architecture of the discriminator is adapted from [59], wherein configuration for the layers follow the pattern: Convolution-Batch Normalization-Leaky Rectified Linear Unit (ReLU). In addition, the discriminator uses receptive fields of size 256 x 256 pixels, which proved effective on input with 224×224 pixels. The discriminator network function is to discriminate between the ground-truth maps as *real* and predicted maps as *fake* while the generator network function is to the fool the discriminator by producing maps as close to ground-truth as possible. Each network is trained independently of each other with the main goal of propagating signals that encourages the generator network to produce better and more accurate results. This process is depicted in Fig. 4.4

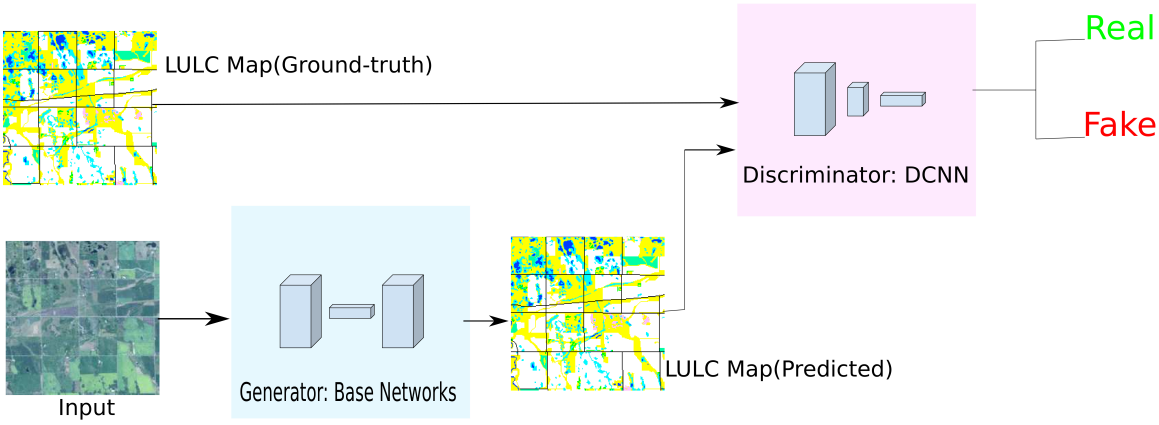


Figure 4.4: Adversarial Extension

As a result of the structural changes made to our networks, the loss functions for the generator and discriminator are based on the approach by Luc et al [60]. The generator loss function combines multi-class entropy loss, a standard loss function used in our base networks, with a binary class entropy loss. Formally, for ground-truth (denoted by $\mathbf{y}$) and predicted output (denoted by $\hat{\mathbf{y}}$) C denotes the number of classes (see *e.g.*, Fig. 5.2), y_{ic} is the correct probability i for class c and $\hat{y}_{ic}$ is the predicted probability i for class c the multi-class entropy loss is measured as:

$$\ell_{mce}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{c=1}^{C} y_{ic} \ln(\hat{y}_{ic}). \tag{4.1}$$

Similarly, z denotes the binary probability for predicted output (0) and ground-truth (1) and $\hat{z}$ represents predicted probability between (0 and 1) the binary class entropy loss is defined as

$$\ell_{bce}(z, \hat{z}) = -(z \ln(\hat{z}) + (1 - z) \ln(1 - \hat{z})). \tag{4.2}$$

Given a dataset N containing $\mathbf{x}_n$ number of training images and corresponding ground-truth maps $\mathbf{y}_n$ the generator model $g(\cdot)$ is a trainable function which can be interpreted as a conditional probability model $g(\mathbf{x}_n) = \boldsymbol{P}(\mathbf{y}_n \mid \mathbf{x}_n)$. The generator $g(\cdot)$ is trained to produce target maps by minimizing multi-class entropy loss. In contrast, the discriminator model assuming binary classification with a trainable function $d(\cdot)$ can be interpreted as a joint probability model $d(\mathbf{x}_n, \mathbf{y}_n) = \boldsymbol{P}(0, 1)$. The discriminator model predicts that y_n is the ground-truth label map of x_n by assigning labels as $(real = 1)$ to ground-truth and discriminates label maps $g(x_n)$ produced by the generator by assigning labels as $(fake = 0)$. Training the discriminator translates to minimizing the following loss function:

$$\sum_{n=1}^{N} \ell_{bce}(d(\mathbf{x}_n, \mathbf{y}_n), 1) + \ell_{bce}(d(\mathbf{x}_n, g(\mathbf{x}_n)), 0).$$

The generator in an adversarial role does not only minimize the multi-class entropy loss, but also aims to degrade performance of the discriminator by producing very similar outputs to corresponding ground-truth. Training the generator translates to minimizing the following loss function:

$$\sum_{n=1}^{N} \ell_{mce}(g(\mathbf{x}_n), \mathbf{y}_n) + \lambda \ell_{bce}(d(\mathbf{x}_n, g(\mathbf{x}_n)), 1),$$

note, λ is applied as a constant regularization function.

Lastly, the parameters $(\boldsymbol{\theta}_g, \boldsymbol{\theta}_d)$ of the generator and discriminator respectively are

adjusted by minimizing a hybrid loss function defined as

$$\ell(\boldsymbol{\theta}_g, \boldsymbol{\theta}_d) = \sum_{n=1}^{N} \ell_{mce}(g(\mathbf{x}_n), \mathbf{y}_n) - \lambda(\ell_{bce}(d(\mathbf{x}_n, \mathbf{y}_n), 1) + \ell_{bce}(d(\mathbf{x}_n, g(\mathbf{x}_n)), 0)).$$

Table 4.4: FCN-GoogLeNet network architecture used in this work.

layer name	kernel size / stride	output size
input(224 x 224 x 6 satellite image)		
conv1	{7 x 7, 64 / 2} x 2	(112 x 112 x 64)
maxpool	{3 x 3 / 2}	(56 x 56 x 64)
conv2.x	{3 x 3, 192 / 1}	(56 x 56 x 192)
maxpool	{3 x 3 / 2}	(28 x 28 x 192)
inception(3a)	{conv1, 64 / 1} {conv3_reduce, 96 / 1} {conv3, 128 / 1} {conv5_reduce, 16 / 1} {conv5, 32 / 1} {conv_pool, 32 / 1}	(28 x 28 x 256)
inception(3b)	{conv1, 128 / 1} {conv3_reduce, 128 / 1} {conv3, 192 / 1} {conv5_reduce, 32 / 1} {conv5, 96 / 1} {conv_pool, 64 / 1}	(28 x 28 x 480
maxpool	{3 x 3 / 2}	(14x 14 x 480)
inception(4a)	{conv1, 192 / 1} {conv3_reduce, 96 / 1} {conv3, 208 / 1} {conv5_reduce, 16 / 1} {conv5, 48 / 1} {conv_pool, 64 / 1}	(14 x 14 x 512)
inception(4b)	{conv1, 160 / 1} {conv3_reduce, 112 / 1} {conv3, 224 / 1} {conv5_reduce, 24 / 1} {conv5, 64 / 1} {conv_pool, 64 / 1}	(14 x 14 x 512)
inception(4c)	{conv1, 128 / 1} {conv3_reduce, 128/ 1} {conv3, 256 / 1} {conv5_reduce, 24 / 1} {conv5, 64 / 1} {conv_pool, 64 / 1}	(14 x 14 x 512)
inception(4d)	{conv1, 112 / 1} {conv3_reduce, 144/ 1} {conv3, 288 / 1} {conv5_reduce, 32 / 1} {conv5, 64 / 1} {conv_pool, 64 / 1}	(14 x 14 x 528)
inception(4e)	{conv1, 256 / 1} {conv3_reduce, 160 / 1} {conv3, 320 / 1} {conv5_reduce, 32 / 1} {conv5, 128 / 1} {conv_pool, 128 / 1}	(14 x 14 x 832)
maxpool	{3 x 3 / 2}	(7 x 7 x 832)
inception(5a)	{conv1, 256 / 1} {conv3_reduc, 160 / 1} {conv3, 320 / 1} {conv5_reduc, 32 / 1} {conv5, 128 / 1} {conv_pool, 128 / 1}	(7 x 7 x 832)
inception(5b)	{conv1, 384 / 1} {conv3_reduc, 192 / 1} {conv3, 384 / 1} {conv5_reduc, 48 / 1} {conv5, 128 / 1} {conv_pool, 128 / 1}	(7 x 7 x 1024)
conv_final	{1 x 1 x 19 / 1}	(7 x 7 x 19)
upsampling layers (transpose convolutions with stride)		
tconv_fuse_inception(4e)	{4 x 4, 832 / 2}	(14 x 14 x 832)
tconv_fuse_inception(3b)	{4 x 4, 480 / 2}	(28 x 28 x 480)
tconv_final	{16 x 16, 19 / 8}	(224 x 224 x 19)

Chapter 5

Implementation Details

5.1 Dataset Acquisition

Landsat 5/7 satellite images of the southern agricultural growing region of Manitoba (see the red outline in Fig. 5.1) are used to evaluate and compare the accuracy of the networks described here. This area is referred to as *southern extent of Manitoba*, and the size of this region is approximately 148,800 km^2. These images differs from typical digital RGB images and as such necessitated the base network modifications described in Chapter 4, this satellite imagery contains three additional infrared channels known as spectral bands. The additional bands captures the spectral response of the various objects (e.g. *water, grass, conifer and deciduous trees*) that make up the earths land surface. The magnitude of energy that the objects reflects or emits across a range of wavelengths is called its spectral response pattern and its recorded in the spectral bands [1]. The satellite imagery (multispectral) used in this work contains 6 bands. The characteristics of the bands are contained in Table 5.1. Each pixel in the satellite image represents 30 m x 30 m square area of land.

The dataset acquired contains raw Landsat 5/7 satellite images (*i.e.* unclassified data) and LULC (*i.e.* labelled, ground-truth data). The labelled data was created using semi-automated methods (see [61]) and ground-truthed by GeoManitoba[2]. A total of eighteen Landsat 5/7 scenes (see the green outlines in Fig. 5.1) were used to produce the maps. LULC maps are produced from satellite images by classifying each pixel in the satellite image to one of several land-use labels (see *e.g.*, Fig. 5.3). Example labels include water, grassland, marsh, deciduous, coniferous, road, and

[1] https://www.e-education.psu.edu/natureofgeoinfo/node/1906

[2] A government agency mandated to create land-use/land-class maps of the province of Manitoba

Table 5.1: Bands: Spatial and Spectral Resolution of Landsat5/7 Data

Bands	Spectral Resolution (μm)	Spatial Resolution (meter)
Blue (B)	0.450 to 0.515	30
Green (G)	0.525 to 0.605	30
Red (R)	0.630 to 0.690	30
Near Infrared (NIR)	0.750 to 0.900	30
Shortwave Infrared (SWIR)	1.550 to 1.750	30
Shortwave Infrared 2 (SWIR2)	2.090 to 2.350	30

agriculture. The full list is given in Fig. 5.2, and an example of a GeoManitoba LULC map created from Landsat 5/7 data from 2004 is given in Fig. 5.4. The GeoManitoba dataset was augmented with satellite images containing clouds and a new class (*clouds*) was added to the list provided by GeoManitoba. This was done to prevent the networks from misclassifying clouds into one of the other land-use classes.

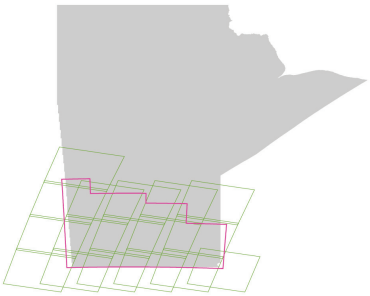
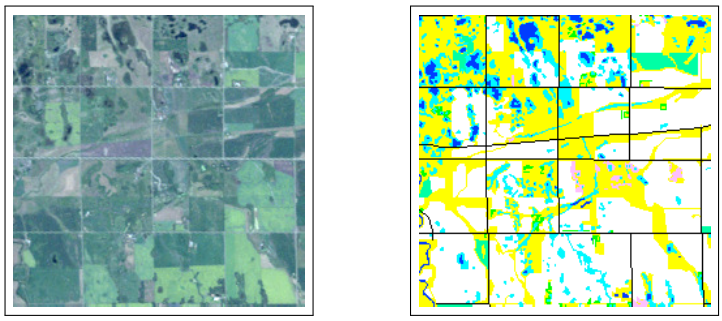


Figure 5.1: Province of Manitoba with the southern agricultural growing region (red) and the associated Landsat 5/7 scenes that cover this area (green).

Figure 5.2: GeoManitoba land-use classes

5.2 Data Preparation and Augmentation

A key component to successfully training deep neural networks is the availability of sufficient training data. For example, all base networks presented in this work were originally trained on the ImageNet dataset [13], consisting of ∼1,2 million images



(a) (b)

Figure 5.3: Example LULC map. (a) RGB components of a Landsat 7 satellite image of Manitoba, and (b) the GeoManitoba LULC map produced from (a).

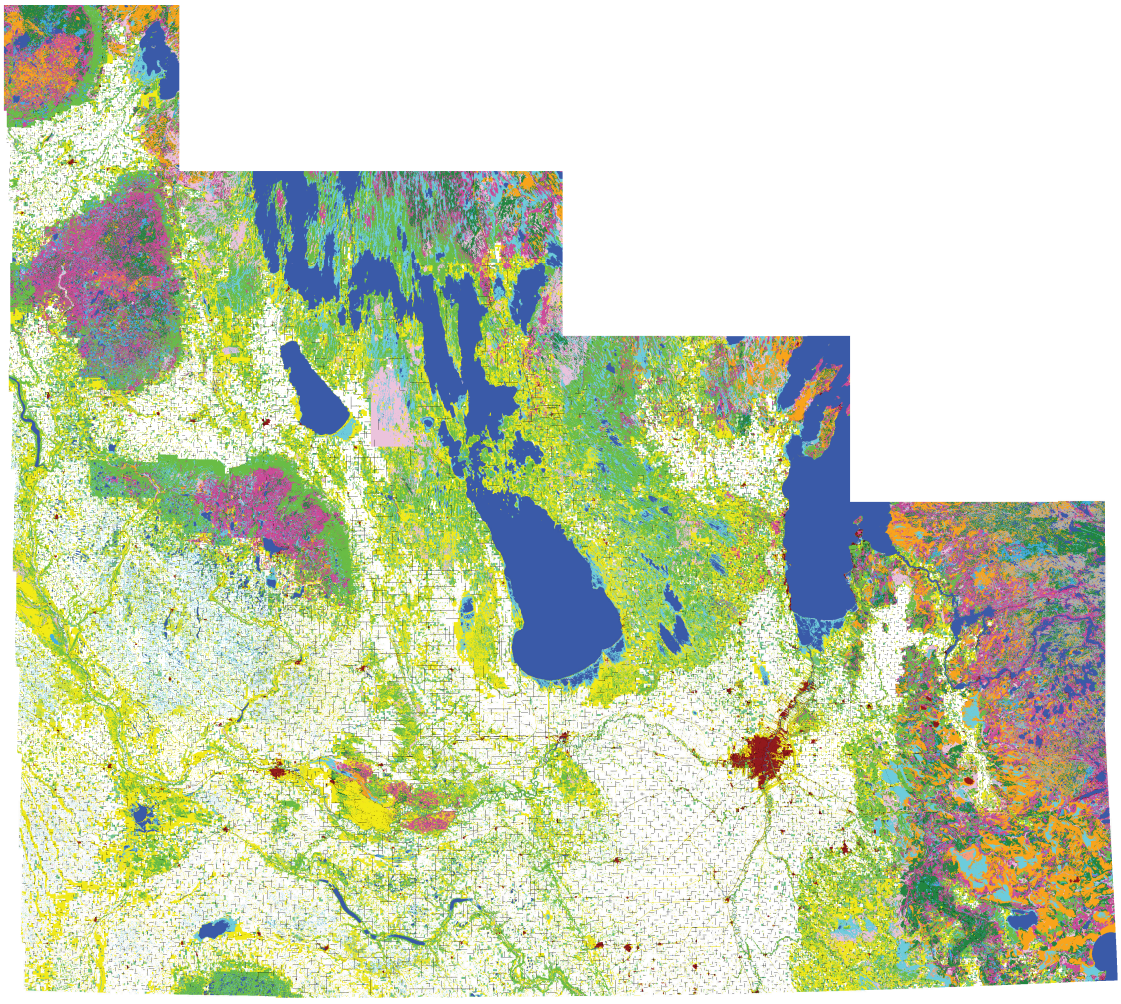
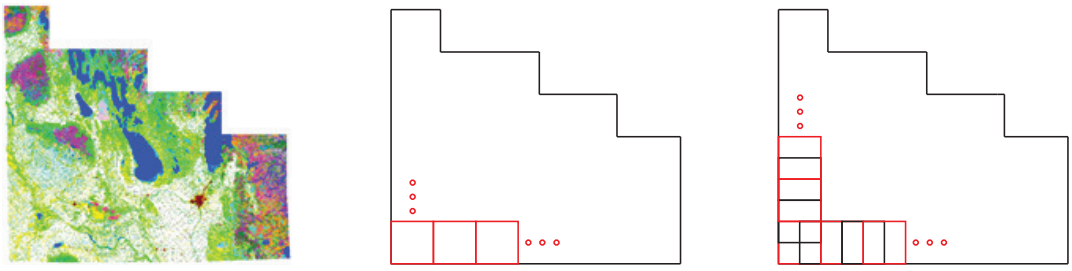


Figure 5.4: 2004 LULC map provided by GeoManitoba.

and 1000 categories. The LULC map provided by GeoManitoba had a resolution of $(13777 \times 16004 \times 6)$, which posed some problems. Firstly, the original base networks were structured to take input of (224×224). Secondly, maintaining the original resolution of our dataset meant only 3 training examples were available to work with. Subsequently, these problems were solved by dividing both the raw satellite image and corresponding LULC maps into tiles of size $(224 \times 224 \times 6)$. This approach proved to be effective in solving both issues. The process used to produce the individual tiles from the full southern extent is depicted in Fig. 5.5. The first set of tiles were produced by the method shown in Fig. 5.5(b), namely the tiles were non-overlapping. To further increase the size of the dataset, the tiling process in Fig. 5.5(c) was used to produce more tiles. In this case, tiles were overlapped by half the size of the network input resolution, *i.e.* $224/2 = 112$. Moreover, by starting this process in each of the four corners of the full map depicted in Fig. 5.5(a), the total number of tiles generated in Fig. 5.5(c)was able to be increased by more than $4\times$ due to the fact that the resolution of the map in Fig. 5.5(a) is not a multiple of 224.



(a) (b) (c)

Figure 5.5: Illustration depicting the tiling process. (a) GeoManitoba's LULC map of southern Manitoba, (b) non-overlapping tiles, and (c) 1/2 tile overlap.

5.3 Experimental Setup

All networks were trained and evaluated using the Tensorflow deep learning framework [62] on a NVIDIA Digits DevBox [3] containing four Titan X GPUs with 12GB of memory per GPU, 64 GB DDR4 RAM, and a Core i7-5930K 3.5 GHz processor. Training time for each network took an average period of 6-10 days. A mini-batchsize of 2 was maintained across all networks due to GPU memory constraints. Subse-

[3] `https://developer.nvidia.com/devbox`

quently, learning rates of $10^{-4}, 10^{-5}, 10^{-9}$ were used for experimentation, and the best results were achieved by starting with a learning rate of 10^{-4}, maintaining it for 100 epochs, and then reducing the learning rate to 10^{-5} for another 100 epoch, thereby completing training after 200 epochs. The Adam optimization algorithm[42] was used to update network weights since it allowed the networks to converge quickly when compared to RMSprop and (SGD + Momentum) optimizer [41].

Chapter 6

Results, Analysis and Comparisons

In this chapter, we complete analysis and evaluations of our base networks and extensions identified in Chapter 4. Accuracy assessment is performed on our best network identified from quantitative evaluations. In addition, sample results from each network are compared. At the end we discuss our results and make a case for our deep learning framework introduced over the course of this work based on quantitative and qualitative evaluations.

6.1 Evaluation Metrics

The dataset consists of 18,054 training images and 958 validation images following a 80/20 split. The networks evaluated are as follows: Base networks (consisting of FCN-[VGG-16, ResNet-101, GoogLeNet]), base networks + context module, base networks + adversarial network, and base networks + context module + adversarial network. All networks are described in Chapter 4. The following common segmentation metrics were aggregated over validation images (predicted and corresponding ground-truth). Let n_{ij} be the number of pixels having ground-truth label i whose prediction is predicted label j. Also, $t_i = \sum_{j=1}^{C} n_{ij}$ denotes the total number of pixels labeled with i where C is the number of classes, n_{ii} is the number of pixels labeled correctly and n_{ji} is the number of pixels wrongly labeled.

Global pixel accuracy is the ratio of correctly classified pixels to total pixels summed over all classes and it is defined as

$$\frac{\sum_{i=1}^{C} n_{ii}}{\sum_{i=1}^{C} t_i}.$$

We compute the per-class accuracy metric as

$$\frac{1}{C}\sum_{i=1}^{C}\frac{n_{ii}}{t_i},$$

which measures the ratio of correctly classified pixels in each class to total pixels, averaged over all classes. Lastly, the Mean IOU defined as

$$\frac{1}{C}\sum_{i=1}^{C}\frac{n_{ii}}{t_i+\sum_{j=1}^{C}n_{ji}-n_{ii}},$$

which measures the average intersection over union (IOU) over all classes. Here, IOU is the ratio of correctly classified pixels to the total number of pixels that are assigned a class by the ground-truth and predicted.

6.2 Quantitative Evaluations and Comparisons

Beginning with the base networks, Table 6.1 shows the results based on the validation set. FCN-ResNet-101 performed best with a global accuracy of 88.25% slightly outperforming FCN-VGG16. FCN-GoogLeNet performed worst with a global accuracy of 62.13%, which is attributed to the network overfitting the training set. Both dropout [39] and batch normalization [63] were attempted to prevent overfitting, however neither approach was able to improve the results. Next, the base networks were extended to include an adversarial component (as described in Section 4.5). These results are reported in Table 6.2. Observe that the improvement due to the adversarial network was minimal, but that all networks did improve. In this category FCN-VGG16 performed best with a 0.93% increase over just the base network. Continuing on, Table 6.3 presents the results on extending the base networks with a context module. In this case the improvements were more significant, and, again, were all better than the previous two results. FCN-ResNet-101 performed best on the global accuracy metric while FCN-VGG16 performed best on both mean accuracy and mean IOU metrics. Finally, Table 6.4 gives the results from extending the base network with both the context module and adversarial network. The results obtained in this category provided the best overall result, with only VGG-16 performing worse than just using the context module. More specifically, FCN-ResNet-101 performed best among networks from all the categories.

Table 6.1: Results from the base networks

	Global Accuracy	Per-Class Accuracy(Mean)	Mean IOU
VGG-16	87.99%	81.50%	72.19%
ResNet-101	**88.25%**	**81.92%**	**73.53%**
GoogLeNet	62.13%	37.13%	29.06%

Table 6.2: Results from the base networks + adversarial network

	Global Accuracy	Per-Class Accuracy(Mean)	Mean IOU
VGG-16	**88.92%**	**83.26%**	**74.17%**
ResNet-101	88.40%	82.56%	74.10%
GoogLeNet	62.19%	37.15%	29.08%

Table 6.3: Results from the base networks + context

	Global Accuracy	Per-Class Accuracy(Mean)	Mean IOU
VGG-16	90.32%	**83.93%**	**75.77%**
ResNet-101	**90.38%**	83.81%	75.37%
GoogLeNet	77.64%	59.64%	49.04%

Table 6.4: Results of comparison between base networks + context + adversarial network

	Global Accuracy	Per-Class Accuracy(Mean)	Mean IOU
VGG-16	90.34%	83.63%	75.36%
ResNet-101	**90.46%**	**84.14%**	**75.66%**
GoogLeNet	77.71%	59.81%	49.20%

6.3 Accuracy Assessment and Analysis

The results generated from the best network in form of an *error matrix* or *confusion matrix* presented in Tables 6.5 and 6.7 provides a base for LULC map analysis. The error matrix takes different forms and shows the percent accuracy and total number of pixels classified for each LULC class. Note, for Tables 6.5 and 6.7, the rows correspond to ground truth and the columns to the best network prediction. The *No Data* class (see Tables 6.5 to 6.7) is introduced for the network to be able to classify the pixels around the edge of a Landsat scene correctly when working with real data. Note, this class is easily classified by the network as observed from the high accuracy since its label is very distinguishable from other labels. Some other classes easily classified by the network are *Agriculture, Water, Treed Bog, Forage and Fens.* The deep learning framework performs well for both land-use and land-cover classes which makes it very robust. *Burns* is identified as the class with the lowest accuracy. Moreover, *Burns* accounts for only $2.2 \times 10^{-4}\%$ of the total labels in the validation set. Likewise, the *Burns* category represent only 1.8×10^{-06} of the total pixels in the ground-truth labels provided by GeoManitoba. Thus, an explanation to its low accuracy is the fact that *Burns* is significantly underrepresented in the original dataset making it quite difficult to classify by the network. Going by the data in Table 6.6 the randomly sampled validation set used for testing offers a fair representation of the total labels in the ground-truth for each class. Notice, from the error matrix that misclassification between similar classes is at a minimum in this network. In addition, our best network reliably detects features like roads that exist at single pixel level.

Table 6.5: Percent accuracy for each class from FCN-ResNet-101 + Context + Adversarial Network.

	No Data	Agriculture	Deciduous	Water	Grass	Mixedwood	Marsh	Tbog	Trock	Conifer	Burns	Open Deci.	Forage	Cultural	Cutovers	Gravel	Road	Fens	Cloud
No Data	**99.94**	0.01	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Agriculture	0.00	**96.72**	0.49	0.02	1.62	0.01	0.47	0.00	0.00	0.00	0.00	0.02	0.12	0.01	0.00	0.00	0.50	0.00	0.00
Deciduous	0.00	1.30	**87.41**	0.33	5.76	2.07	1.30	0.06	0.01	0.13	0.00	0.95	0.23	0.02	0.10	0.02	0.30	0.00	0.00
Water	0.00	0.11	0.52	**97.06**	0.50	0.17	1.36	0.03	0.05	0.07	0.00	0.07	0.01	0.01	0.00	0.01	0.02	0.00	0.00
Grass	0.00	3.96	5.10	0.26	**86.72**	0.33	1.43	0.01	0.00	0.06	0.00	0.47	0.58	0.08	0.01	0.02	0.95	0.00	0.01
Mixedwood	0.00	0.03	4.93	0.28	0.83	**84.33**	1.60	0.99	0.93	4.22	0.00	1.11	0.02	0.02	0.38	0.01	0.31	0.00	0.00
Marsh	0.00	3.78	3.27	2.05	4.30	1.88	**81.65**	1.01	0.09	0.44	0.00	1.13	0.19	0.00	0.03	0.00	0.17	0.01	0.01
Tbog	0.00	0.00	0.44	0.09	0.04	2.30	1.53	**90.37**	0.81	3.50	0.00	0.78	0.00	0.00	0.09	0.00	0.05	0.00	0.00
Trock	0.00	0.00	0.28	0.61	0.04	8.97	0.62	3.37	**74.36**	11.17	0.00	0.08	0.00	0.01	0.35	0.00	0.14	0.00	0.00
Conifer	0.00	0.01	0.57	0.23	0.27	9.10	0.70	2.95	2.59	**82.13**	0.00	0.78	0.00	0.01	0.48	0.01	0.17	0.00	0.00
Burns	0.00	7.55	1.89	0.00	14.15	1.89	4.72	1.89	0.00	0.00	**51.89**	0.00	0.94	0.00	0.94	0.00	14.15	0.00	0.00
Open Deci.	0.00	0.49	6.94	0.23	3.67	3.13	2.06	0.83	0.03	1.12	0.00	**80.96**	0.11	0.00	0.14	0.03	0.24	0.00	0.01
Foreage	0.00	2.28	1.20	0.03	4.00	0.03	0.34	0.00	0.00	0.00	0.00	0.09	**91.46**	0.01	0.01	0.00	0.53	0.00	0.01
Cultural	0.03	2.02	1.39	0.36	3.89	0.38	0.13	0.08	0.01	0.04	0.00	0.04	0.14	**88.26**	0.02	0.04	3.12	0.00	0.05
Cutovers	0.00	0.01	2.77	0.05	0.40	5.52	0.50	0.73	0.74	2.81	0.00	0.98	0.03	0.00	**85.14**	0.02	0.29	0.00	0.00
Gravel	0.00	3.54	5.63	3.56	8.80	1.35	0.48	0.18	0.03	0.63	0.00	1.70	0.09	0.52	0.43	**71.91**	1.11	0.00	0.03
Road	0.01	10.54	3.14	0.11	10.35	1.40	0.65	0.11	0.07	0.36	0.00	0.40	0.98	0.86	0.11	0.03	**70.88**	0.00	0.01
Fens	0.07	0.00	0.03	0.01	0.04	0.17	3.54	0.79	0.09	0.01	0.00	0.00	0.00	0.00	0.07	0.00	0.09	**95.08**	0.00
Cloud	0.00	1.87	3.64	0.12	8.00	0.13	1.84	0.02	0.00	0.03	0.00	0.57	0.51	0.13	0.01	0.23	0.58	0.00	**82.30**

Table 6.6: Total ground-truth (*gt*) labels in *2004 LULC map* and total ground-truth labels in the validation (*val*) set for each class.

	Labels (*gt*)	**% Labels** (*gt*)	**Labels** (*val*)	**% Labels** (*val*)
No Data	67328000	30.54	1759952	3.66
Agriculture	46492000	21.09	14637515	30.45
Deciduous	21248000	9.64	6651303	13.84
Water	15454000	7.01	4714705	9.81
Grass	23441000	10.63	6931812	14.42
Mixedwood	9610600	4.36	2842707	5.91
Marsh	9515700	4.32	2904884	6.04
Tbog	5594000	2.54	1451776	3.02
Trock	1554400	0.70	383216	0.80
Conifer	5910500	2.68	1575915	3.28
Burns	390	0.00	106	0.00
Open Deci.	4610900	2.09	1226973	2.55
Foreage	4819400	2.19	1452454	3.02
Cultural	694080	0.31	224161	0.47
Cutovers	1020800	0.46	340413	0.71
Gravel	122860	0.06	29651	0.06
Road	2982000	1.35	913966	1.90
Fens	36163	0.02	6955	0.01
Cloud	51590	0.02	20144	0.04

Table 6.7: Total number of pixels classified for each class from FCN-ResNet-101 + Context + Adversarial Network.

	No Data	**Agriculture**	**Deciduous**	**Water**	**Grass**	**Mixedwood**	**Marsh**	**Tbog**	**Trock**
No Data	**1758971**	190	78	197	69	122	25	11	59
Agriculture	84	**14157681**	71724	3212	237394	900	69454	6	1
Deciduous	279	86654	**5813921**	21761	382855	137432	86353	4118	841
Water	80	5337	24731	**4575969**	23623	8200	63912	1567	2159
Grass	200	274527	353524	17722	**6011492**	23065	99092	492	157
Mixedwood	78	994	140067	7890	23666	**2397357**	45366	28181	26397
Marsh	76	109753	94864	59408	125030	54739	**2371839**	29236	2627
Tbog	18	16	6447	1370	527	33438	22180	**1311902**	11688
Trock	3	10	1085	2330	136	34365	2366	12915	**284969**
Conifer	19	110	9040	3617	4211	143348	11030	46514	40878
Burns	0	8	2	0	15	2	5	2	0
Open Deci.	10	5989	85121	2866	45070	38367	25330	10222	391
Foreage	5	33103	17473	487	58096	427	4955	11	1
Cultural	73	4530	3118	796	8716	852	281	177	25
Cutovers	0	23	9436	154	1364	18798	1710	2495	2530
Gravel	0	1049	1669	1057	2608	400	143	54	9
Road	58	96309	28664	960	94551	12828	5980	1016	669
Fens	5	0	2	1	3	12	246	55	6
Cloud	0	377	733	24	1612	27	371	5	0

	Conifer	**Burns**	**Open Deci.**	**Forage**	**Cultural**	**Cutovers**	**Gravel**	**Road**	**Fens**	**Cloud**
No Data	107	0	26	7	0	5	0	85	0	0
Agriculture	63	3	3429	18105	1403	8	346	73523	0	179
Deciduous	8544	3	63326	15602	1587	6679	1264	19832	0	252
Water	3177	3	3422	463	677	127	509	746	0	3
Grass	4295	11	32849	40047	5742	551	1332	65957	0	757
Mixedwood	119871	2	31612	497	615	10790	389	8923	0	12
Marsh	12811	1	32710	5602	109	818	93	4818	191	159
Tbog	50773	2	11259	33	14	1358	27	683	41	0
Trock	42819	3	315	9	32	1335	4	520	0	0
Conifer	**1294279**	2	12350	43	113	7640	89	2631	0	1
Burns	0	**55**	0	1	0	1	0	15	0	0
Open Deci.	13752	0	**993402**	1324	21	1746	360	2899	0	103
Foreage	30	0	1319	**1328423**	173	73	21	7729	0	128
Cultural	93	0	82	315	**197838**	47	98	6999	0	121
Cutovers	9565	2	3344	92	15	**289821**	60	1004	0	0
Gravel	188	0	505	27	153	128	**21323**	328	0	10
Road	3324	3	3623	8982	7837	1047	246	**647786**	6	77
Fens	1	0	0	0	0	5	0	6	**6613**	0
Cloud	6	0	114	103	27	2	47	117	0	**16579**

6.4 Qualitative Evaluation and Comparisons

Output LULC map samples from each network are presented for visual comparison with corresponding ground-truth maps. This is to show how the predicted output differs from its corresponding ground-truths at a human-level perceptual understanding. The best network (*i.e.* FCN-ResNet101 + Context + Adversarial) produces very similar LULC maps to the ground-truths and as such validates the accuracy results obtained from quantitative evaluations in Section 6.2 (see *e.g.* Fig. 6.4).

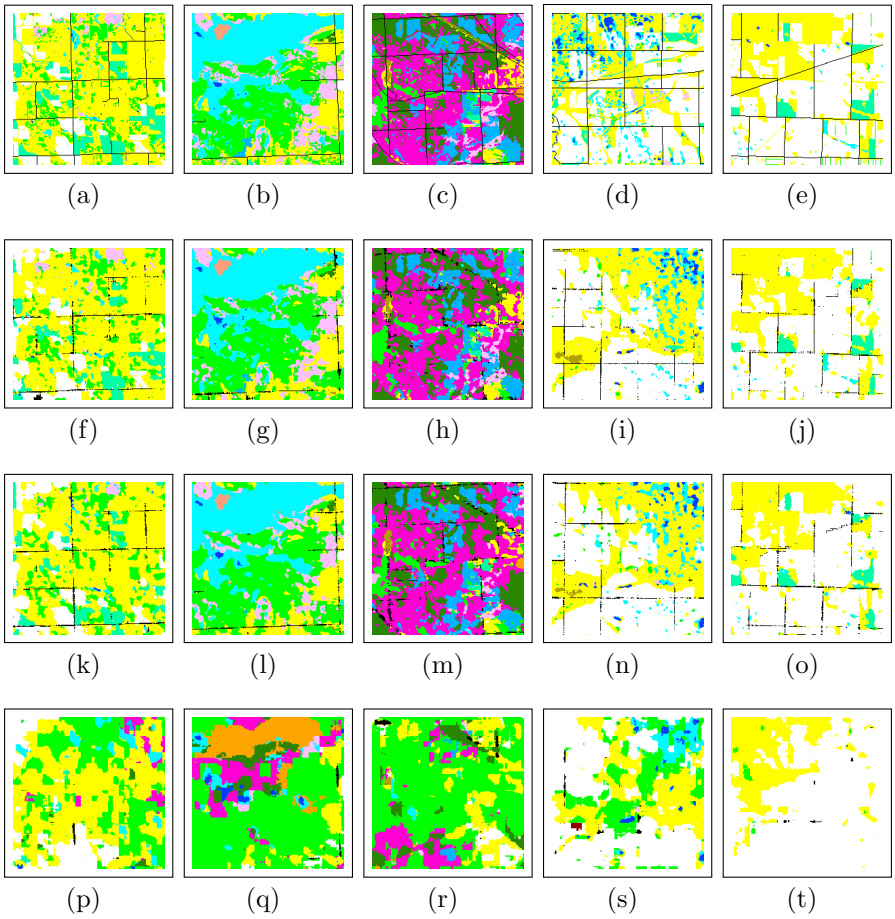


Figure 6.1: Sample validation set results for base networks. (First row) ground-truth labellings, (Second row) result from FCN-ResNet101, (Third row) result from FCN-VGG16 and, (Fourth row) result from FCN-GoogLeNet.

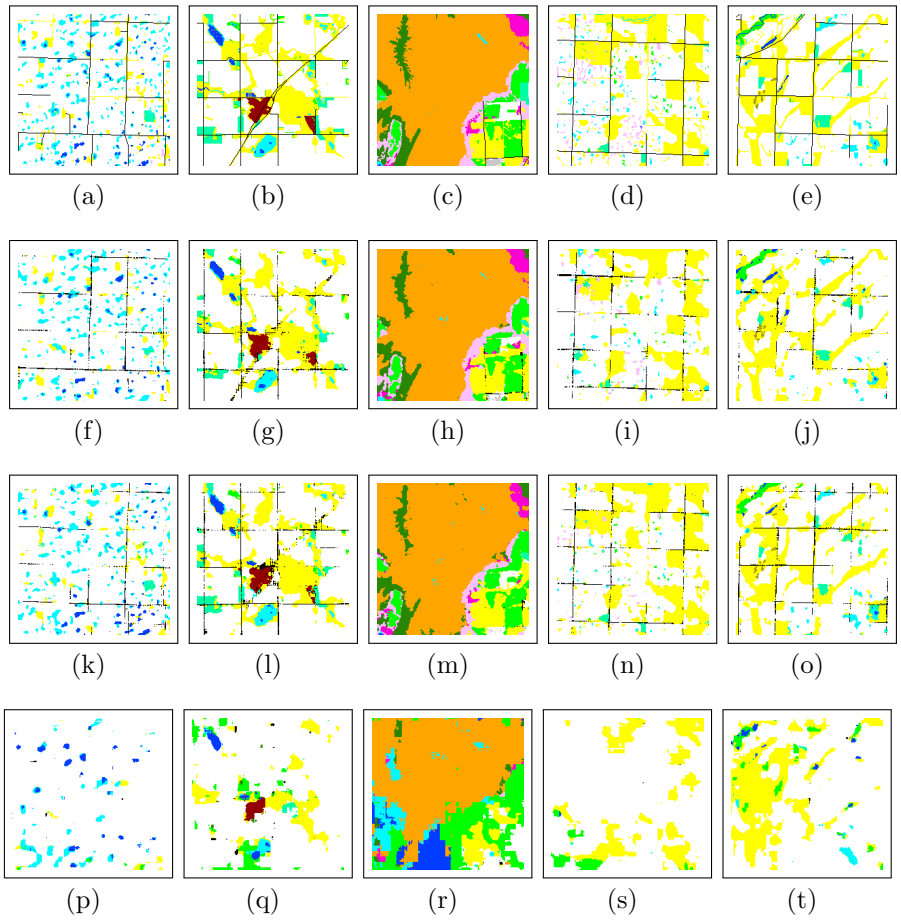


Figure 6.2: Sample validation set results for base networks + adversarial. (First row) ground-truth labellings, (Second row) result from FCN-VGG16 + adversarial, (Third row) result from FCN-ResNet101 + adversarial and, (Fourth row) result from FCN-GoogLeNet + adversarial.

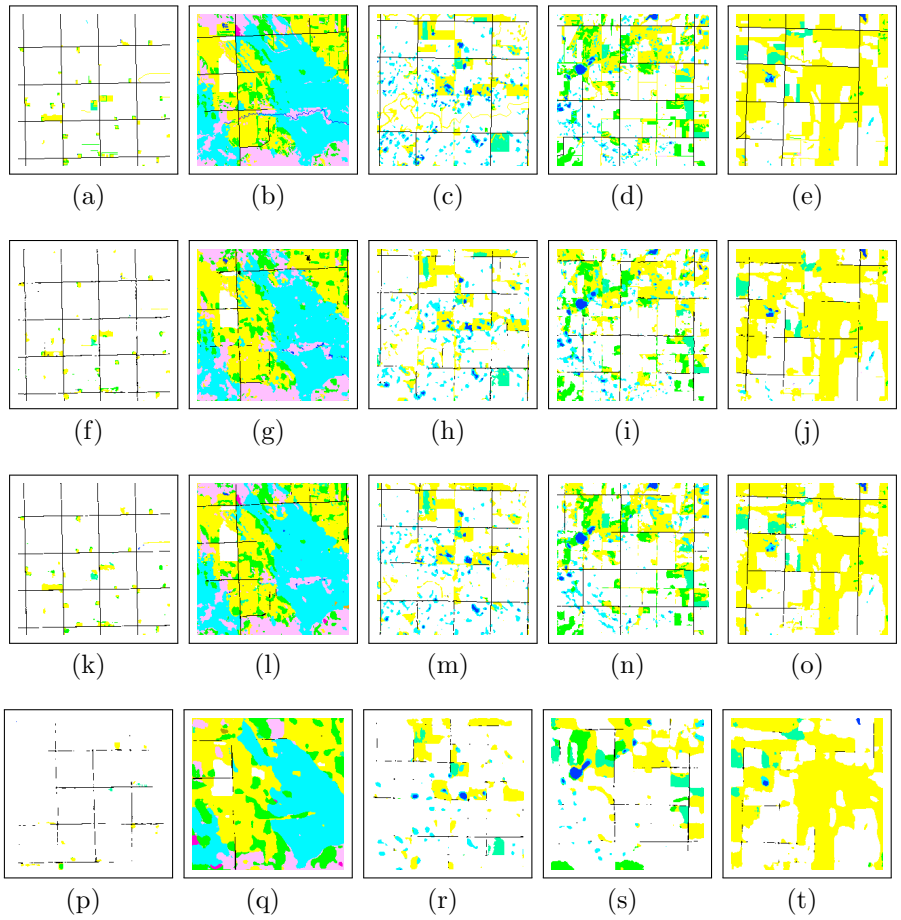


Figure 6.3: Sample validation set results for base networks + context. (First row) ground-truth labellings, (Second row) result from FCN-ResNet101 + Context, (Third row) result from FCN-VGG16 + Context and, (Fourth row) result from FCN-GoogLeNet + Context.

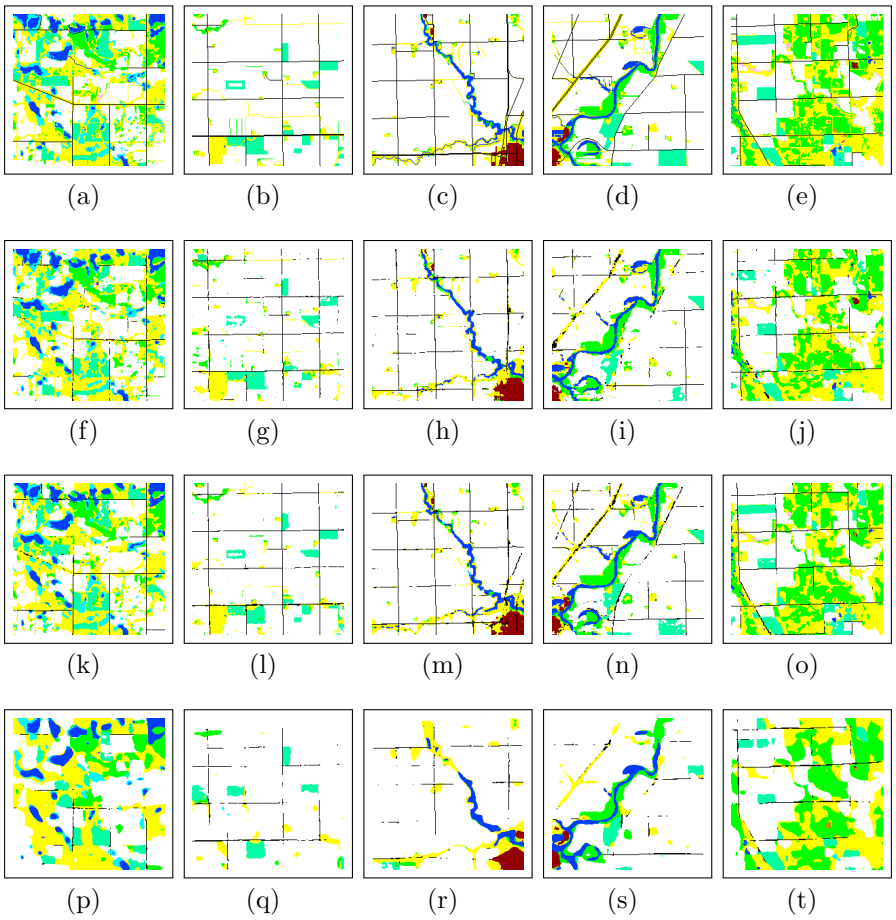


Figure 6.4: Sample validation set results for base networks + context + adversarial. (First row) ground-truth labellings, (Second row) result from FCN-ResNet101 + Context + Adversarial, (Third row) result from FCN-VGG16 + Context + Adversarial and, (Fourth row) result from FCN-GoogLeNet + Context + Adversarial.

6.5 Summary

The robustness of our deep learning framework is demonstrated in the fact the network is able to successfully classify the 2010 Landsat 7 dataset, which was not used in the training process since there are no corresponding labels for this year. Figure 6.5 and Fig. 6.6 shows a fully classified 2004 Landsat 7 from which the network was trained and a fully classified 2010 Landsat 7 dataset with which the network was not presented training examples. Overall, our proposed deep learning framework successfully discriminates and classifies very similar classes based on spectral cues and produces highly accurate maps.

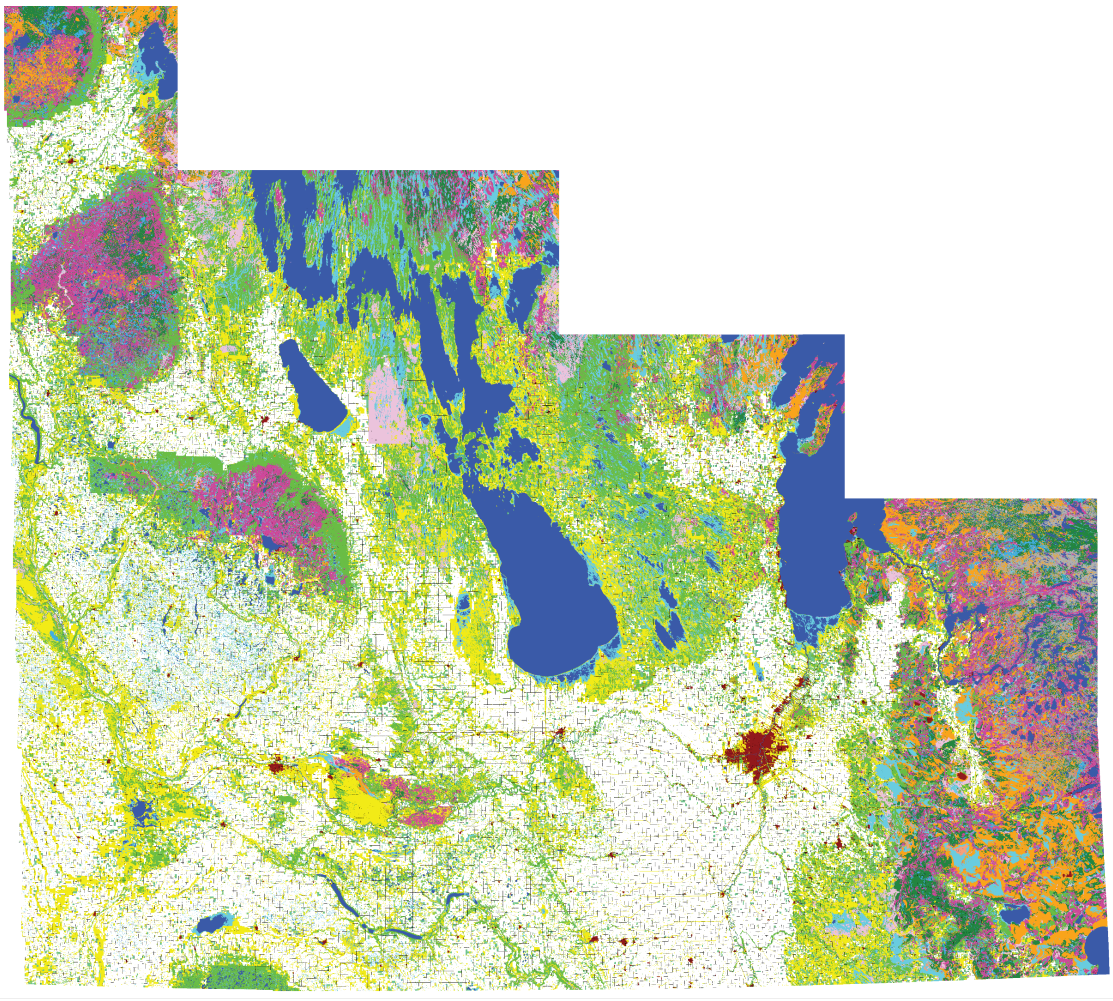


Figure 6.5: Final 2004 Landsat 7 LULC map produced by the best network using ToA input using both validation and training sets.

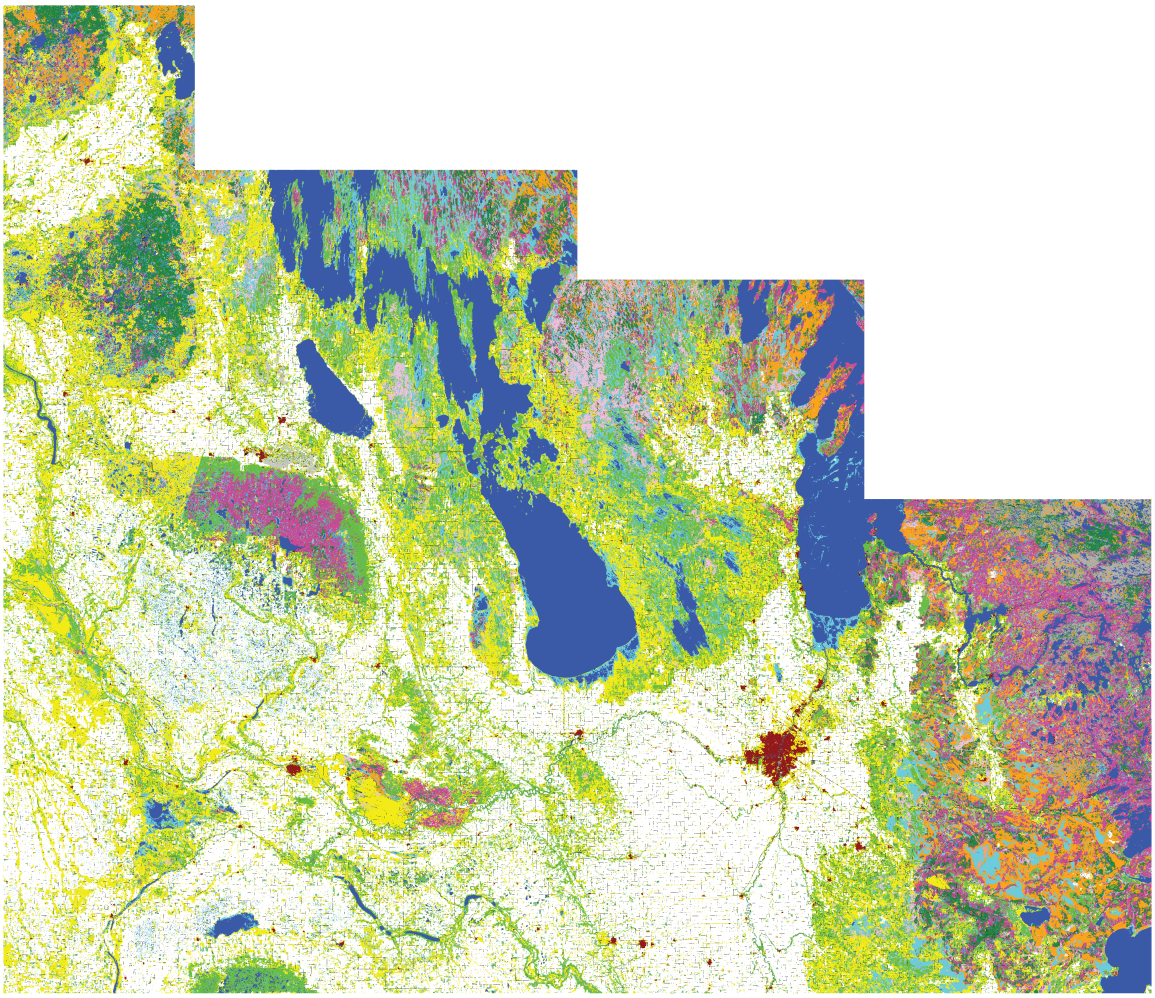


Figure 6.6: Final 2010 Landsat 7 LULC map produced by the best network trained on 2004 data using ToA input.

Chapter 7

Conclusions

In this thesis, we have presented a deep learning framework for LULC map production. Three deep neural networks denoted as *base networks* in this paper were adapted into FCNs and modified to take input Landsat 5/7 satellite images (6 *bands*) and produce fully classified LULC maps. Furthermore, the networks were extended to improve accuracy. Two extensions were made: 1.) A context module was added to the base networks, and 2.) An adversarial network was added to the base networks. Both extensions served to further improve accuracy and a combination of both extensions added to the base networks provided us with the best result of 90.46% for global accuracy. To demonstrate the robustness of the proposed deep learning framework we have presented thorough evaluations and analysis of results obtained. Furthermore, using our best network we classified and produced LULC map from previously unseen 2010 Landsat 5/7 raw satellite image of the southern extent of Manitoba. Additionally, this deep learning framework can take only 8 minutes and 42 seconds to produce a map of the southern extent of Manitoba with a trained model, effectively automating production. This represents a phenomenal reduction in the 4,800 hours required by the current semi-automated approach according to information made available to us from GeoManitoba. This goes to show that our work can be deployed in real world applications. Furthermore, an important observation is that the solution work presented here should be seen as a solution to freeing up people from the tedious task of producing LULC maps, rather than eliminating a job. This solution will allow technicians to focus on analysis of problems and results rather than performing repetitive pattern classification, tasks which people find tedious and are prone to error.

This work presented in this thesis is a continuous research work and may assume a number of possible future directions. In the immediate future we intend on adapting, modifying and extending other modern deep learning network architectures [64, 65] to further improve results. Additionally, future work will consist of performing real-world ground-truthing on the results to produce a statistical proof of the actual accuracy of the results. Finally, producing a valid solution for 16-bit Landsat 8 is of utmost importance because of the improved data quality of its imagery. This means the data quality (signal to noise ratio) and radiometric quantization (12-bits scaled to 16-bits) of the (Operational Land Imager (OLI) and the Thermal Infrared Sensor (TIRS))[1] is higher than previous Landsat 5/7 instruments, providing significant improvement in the ability to detect small differences in reflected or emitted energy recorded in the spectral bands. This entails creating a new dataset based on Landsat 8 imagery and producing LULC maps based on this data.

[1] https://lta.cr.usgs.gov/L8

Bibliography

[1] Treitz P, Rogan J (2004) Remote sensing for mapping and monitoring land-cover and land-use changean introduction. Progress in Planning 61(4):269–279

[2] Zhu XX, Tuia D, Mou L, Xia GS, Zhang L, Xu F, Fraundorfer F (2017) Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources. IEEE Geoscience and Remote Sensing Magazine 5(4):8–36

[3] Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. MIT Press

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

[5] LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436

[6] Levine S, Pastor P, Krizhevsky A, Ibarz J, Quillen D (2018) Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. The International Journal of Robotics Research 37(4-5):421–436

[7] Bahdanau D, Chorowski J, Serdyuk D, Brakel P, Bengio Y (2016) End-to-end attention-based large vocabulary speech recognition. In: Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on, IEEE, pp 4945–4949

[8] Dong C, Loy CC, He K, Tang X (2016) Image super-resolution using deep convolutional networks. IEEE transactions on pattern analysis and machine intelligence 38(2):295–307

[9] Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:14061078

[10] Zhang L, Zhang L, Kumar V (2016) Deep learning for Remote Sensing Data. IEEE Geoscience and Remote Sensing Magazine 4(2):22–40

[11] LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11):2278–2323

[12] Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25, pp 1097–1105

[13] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115(3):211–252

[14] Castelluccio M, Poggi G, Sansone C, Verdoliva L (2015) Land Use Classification in Remote Sensing Images by Convolutional Neural Networks. arXiv preprint arXiv:150800092 pp 1–11

[15] Basu S, Ganguly S, Mukhopadhyay S, DiBiano R, Karki M, Nemani R (2015) DeepSat. In: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '15, pp 1–10

[16] Marmanis D, Datcu M, Esch T, Stilla U (2016) Deep learning earth observation classification using ImageNet pretrained networks. IEEE Geoscience and Remote Sensing Letters 13(1):105–109

[17] 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):498

[18] Yang C, Rottensteiner F, Heipke C (2018) Classification of land cover and land use based on convolutional neural networks. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 4 (2018), Nr 3 4(3):251–258

[19] Zhong P, Gong Z, Schönlieb C (2016) A dbn-crf for spectral-spatial classification of hyperspectral data. In: 23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016, pp 1219–1224

[20] Lu D, Weng Q (2007) A survey of image classification methods and techniques for improving classification performance. International journal of Remote sensing 28(5):823–870

[21] Storie CD, Henry CJ (2018) Deep learning neural networks for land use land cover mapping. In: Proceedings of the 38th IEEE International Geoscience and Remote Sensing Symposium, p 4 pp., *accepted*

[22] Henry CJ, Storie C, Palaniappan M, Alhassan V, Swamy M, Aleshinloye D, Curtis A, Kim D (2018) Automated LULC Map Production using Deep Neural Networks, submitted

[23] Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 07-12-June-2015:3431–3440

[24] Simonyan K, Zisserman A (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:14091556

[25] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol 07-12-June-2015, pp 1–9

[26] He K, Zhang X, Ren S, Sun J (2016) Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 770–778

[27] Yu F, Koltun V (2015) Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:151107122

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

[29] Rosenblatt F (1958) The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review 65:386–408

[30] Benyus J (1997) Biomimicry: Innovation Inspired By Nature. HarperCollins

[31] Demuth HB, Beale MH, De Jess O, Hagan MT (2014) Neural Network Design, 2nd edn. Martin Hagan

[32] Lippmann RP (1988) An introduction to computing with neural nets. SIGARCH Comput Archit News 16(1):7–25

[33] Minsky M, Papert S (1969) Perceptrons: An Introduction to Computational Geometry. MIT Press

[34] Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533

[35] Bishop CM (2006) Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag

[36] Li F, Karpathy A (2015) CS231n: Convolutional neural networks for visual recognition. Course Notes, Stanford University

[37] Dumoulin V, Visin F (2016) A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:160307285

[38] Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 807–814

[39] 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

[40] Stutz D (2017) Learning shape completion from bounding boxes with cad shape priors. Master's thesis, RWTH Aachen University

[41] Ruder S (2016) An overview of gradient descent optimization algorithms. CoRR abs/1609.04747

[42] Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. CoRR abs/1412.6980

[43] Fernández-Redondo M, Hernandez-Espinosa C (2001) Weight initialization methods for multilayer feedforward. In: ESANN, pp 119–124

[44] Chetlur S, Woolley C, Vandermersch P, Cohen J, Tran J, Catanzaro B, Shelhamer E (2014) cuDNN: Efficient Primitives for Deep Learning. arXiv preprint arXiv: ... pp 1–9

[45] Hubel DH, Wiesel TN (1959) Receptive fields of single neurons in the cat's striate cortex. Journal of Physiology 148:574–591

[46] Thoma M (2016) A survey of semantic segmentation. arXiv preprint arXiv:160206541

[47] Yosinski J, Clune J, Bengio Y, Lipson H (2014) How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems 27, Curran Associates, Inc., pp 3320–3328

[48] Zheng S, Jayasumana S, Romera-Paredes B, Vineet V, Su Z, Du D, Huang C, Torr PH (2015) Conditional random fields as recurrent neural networks. In: Proceedings of the IEEE International Conference on Computer Vision, IEEE Computer Society, pp 1529–1537

[49] Krähenbühl P, Koltun V (2011) Efficient inference in fully connected crfs with gaussian edge potentials. In: Advances in Neural Information Processing Systems 24, Curran Associates, Inc., pp 109–117

[50] Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2018) Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence 40(4):834–848

[51] Chen LC, Papandreou G, Schroff F, Adam H (2017) Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:170605587

[52] Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, Springer, pp 234–241

[53] Badrinarayanan V, Kendall A, Cipolla R (2017) Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 39(12):2481–2495

[54] Paisitkriangkrai S, Sherrah J, Janney P, van den Hengel A (2016) Semantic labeling of aerial and satellite imagery. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 9(7):2868–2881

[55] Alam FI, Zhou J, Liew AW, Jia X, Chanussot J, Gao Y (2017) Conditional random field and deep feature learning for hyperspectral image segmentation. CoRR abs/1711.04483

[56] Carreira-Perpiñán Ma, Hinton GE (2005) On Contrastive Divergence Learning. Artificial Intelligence and Statistics 0:17

[57] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2015) Rethinking the Inception Architecture for Computer Vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 2818–2826

[58] Szegedy C, Ioffe S, Vanhoucke V, Alemi A (2016) Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In: AAAI, vol 4, p 12

[59] Isola P, Zhu JY, Zhou T, Efros AA (2017) Image-to-image translation with conditional adversarial networks. arXiv preprint

[60] Luc P, Couprie C, Chintala S, Verbeek J (2016) Semantic segmentation using adversarial networks. arXiv preprint arXiv:161108408

[61] Yifang B, Gong P, Gini C (2015) Global land cover mapping using earth observation satellite data: Recent progresses and challenges. ISPRS journal of photogrammetry and remote sensing (Print) 103(1):1–6

[62] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org

[63] Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR abs/1502.03167

[64] Huang G, Liu Z, v d Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 2261–2269

[65] Chen L, Zhu Y, Papandreou G, Schroff F, Adam H (2018) Encoder-decoder with atrous separable convolution for semantic image segmentation. CoRR abs/1802.02611