Preprint for Neural Computing and Applications (published online 17 July 2019). DOI: 10.1007/s00521-019-04349-9.

Neural Computing and Applications manuscript No.
(will be inserted by the editor)

A Deep Learning Framework for Land-Use/Land-Cover Mapping and Analysis using Multispectral Satellite Imagery

Victor Alhassan[1] · Christopher Henry[1] · Sheela Ramanna[1] · Christopher Storie[2]

Received: date / Accepted: date

Abstract In this article, we present an approach to Land use and Land cover (LULC) mapping from multispectral satellite images using deep learning methods. The terms satellite image classification and map production, although used interchangeably have specific meanings in the field of remote sensing. Satellite image classification describes assignment of global labels to entire scenes, whereas LULC map production involves producing maps by assigning a class to each pixel. We show that by classifying each pixel in a satellite image into a number of LULC categories we are able to successfully produce LULC maps. This process of LULC mapping is achieved using deep neural networks pre-trained on the ImageNet Large-Scale Visual Recognition Competition (ILSVRC) datasets and fine-tuned on our target dataset, which consists of Landsat 5/7 multispectral satellite images taken of the Province of Manitoba in Canada. This approach resulted in 88% global accuracy. Performance was further improved by considering the state-of-the-art generative adversarial architecture and context module integrated with the original networks. The result is an automated deep learning framework that can produce highly accurate LULC maps images significantly faster than current semi-automated methods. The contribution of this article includes 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 Neural Networks, Satellite Images.

1 Introduction

This paper presents an approach to classify pixels obtained from satellite images using deep neural networks developed for semantic segmentation. This approach is directly applicable to the creation of Land-Use and Land-Cover (LULC) maps. The

[1]Department of Applied Computer Science · [2]Department of Geography,
University of Winnipeg, 515 Portage Ave, Winnipeg, Manitoba R3B 2E9 Canada
E-mail: alhassan-v@webmail.uwinnipeg.ca · {ch.henry, s.ramanna, c.storie}@uwinnipeg.ca

presented solution is increasingly important since the abundance and affordability of satellite imagery has led to many government and private industries using LULC maps as a fundamental tool for large-scale monitoring of land resources changes. For instance, these maps are vital in areas such as flood forecasting, urban and rural land-use planning, resource management, and disaster management and planning [1]. In this regard, land-use classes are considered man-made areas (*e.g*, roads, agriculture, cities) while land-cover classes are identified as natural earth resources (*e.g.* water, forests, marshlands, bogs).

The problem of semantic segmentation of digital images translates directly to the problem of LULC map production in which satellite image pixels are classified into a number of LULC classes. Traditionally, machine learning algorithms such as k-means clustering and maximum likelihood classifier [2, 3] have been used to classify satellite images or produce meaningful maps from satellite data. However, the use of deep learning methods to solve domain specific problems has been increasingly popular and successful in the last decade [4, 5]. Specifically, deep convolutional neural networks (DCNN) have excelled in computer vision tasks [6, 7]. The ability of DCNNs to learn low-level and high-level features in a hierarchical manner makes them suitable for computer vision problems such as semantic segmentation of images considered by [8].

In [9, 10], the fully convolutional network (FCN) originally developed by [8] for semantic segmentation was modified and adapted for automating the production of LULC maps. In this paper, DCNNs originally designed for image classification and object detection tasks are adapted into FCNs that take arbitrary sized input and produce image segmentations [8], *i.e.* each pixel is labeled with a LULC class (see Fig 1). Specifically, three pre-trained DCNNs (VGGNet [11], GoogLeNet [12], ResNet [13]) – referred to as *base networks* – are used for LULC map production and analysis. Moreover, this paper introduces two specific extensions to the base networks: a context module and an adversarial extension with the intent to further improve the quality of the produced LULC maps. The context-module developed by [14] is a standalone plug-in configured with convolution layers. By adding this module, features flowing from the base networks are processed to uncover more context information while preserving resolution. The second extension involves positioning the *base networks* in an adversarial setting [15] by adding a discriminator network. With this extension a new loss function is formulated and optimized to impose higher-order consistency across labeled pixels. Each extension is added to augment performance of the base networks. This ensemble of *base networks* and their extensions form our proposed deep learning framework.

This paper presents a case study in exploring recent work on DCNNs for LULC map production of Landsat 5/7 multi-spectral satellite images using LULC classes defined by GeoManitoba[1]. The presented work is an extension of [10], where FCN-8 VGG-16, CRF-RNN, and Dilation 8 (both frontend and context) networks were used on the Landsat 5/7 Manitoba satellite image data for LULC map production. The best solution reported in [10] produced an average accuracy of 88%. The motivation of this work is to improve the results reported [10] by: i) investigating other deep learning architectures to further improve the classification accuracy, ii) propose a generalized deep learning framework to reduce the time spent pro-

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

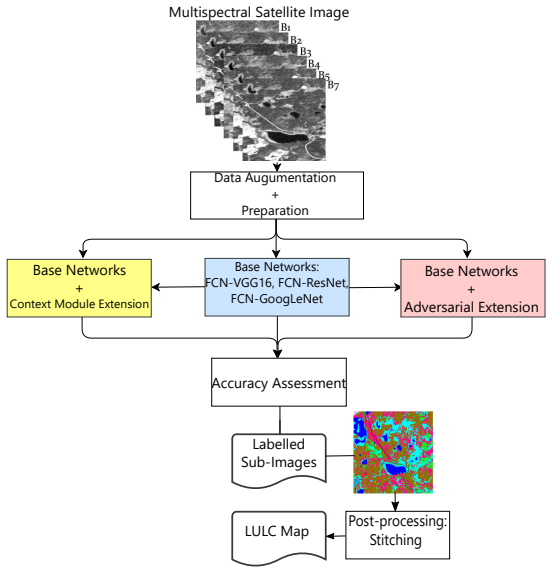


Fig. 1 Overview of proposed deep learning framework for LULC mapping and analysis

ducing LULC maps, and iii) to provide more effective DCNN architectures for LULC map production of 16-bit Landsat 8 GeoManitoba data. The contribution of our work is as follows: i) an improvement in the classification of global accuracy from 88% to 90.46%, ii) extensive experimentation of different FCN architectures with extensions on a unique dataset, and iii) a dramatic reduction in the time it takes to produce these LULC maps (from 4800 hours to 8 minutes and 42 seconds). The main significance of these contributions is the increase in accuracy. As is well known, improvements in accuracy become exceedingly difficult as a system approaches 100% accuracy. For example, consider the ImageNet classification challenge. From 2012 to 2015, the accuracy increased from 84.7% to 96.4% with an average increase of 3.9% per year. In 2016 and 2017, the accuracy went from 97.0% to 97.5% with an average increase of 0.5% increase per year after 2015.

This paper is organized as follows: In Section 2, works related to deep learning methods for semantic segmentation in general and for LULC mapping and analysis in particular, are presented. In Section 3, a detailed description of base networks used in the research is given. In Section 4, the proposed extensions to the base networks are discussed, followed by implementation details in Section 5. Results are discussed in Section 6. We conclude the paper in Section 7.

2 Related Works

The works related to this paper is presented in two separate sub sections. The first section is devoted to deep learning-based works for semantic segmentation since it forms the basis for LULC mapping and analysis in this paper. The second section discusses deep learning techniques specifically for LULC classification and analysis.

2.1 Deep Learning for Semantic Segmentation

Pixel-wise classification tasks, such as semantic segmentation, aim to understand images at a pixel level. Semantic segmentation of digital images 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 [16] 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 [6], VGGNet [11] and GoogLeNet [12] have attained remarkable success in image classification tasks and based on this achievements Long et al [8] 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 these 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.* 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 [17]. Long *et al.* [8] train the FCNs through the use of transfer learning [18], 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. Next, [14] adapts the VGG-based network proposed in [8] 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 [10] based on the architectures described in [14]. Conditional Random Fields as Recurrent Neural Networks (CRFasRNN) described in [19], 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 [20] as a 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 [8] and trained end-to-end. Results are also reported in [10] based on CRFasRNN [19]. Next, DeepLab v2 [21] explores the use of convolution with up-sampled filters called *atrous convolution* to enlarge the field of view without increasing the number of parameters, which is the same behaviour as the dilated convolutions in [14]. 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 [22] further improves on the latter. Another popular architecture for segmentation is Unet [23] which builds on FCNs [8] for biomedical segmentation.

2.2 Deep Learning for LULC Mapping and Analysis

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 mapping and analysis. The authors of [24] 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 [25] 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 author adopt SegNet model [26] and compares different variants, in contrast LiteNet model [27] and other variants are used for land-use classification. Remote sensing imagery can be categorized based on spatial and spectral resolution, [28] 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 [19] for LULC classification. Similarly, Alam et al [29] incorporate CRF with CNN into a common framework for hyperspectral image segmentation.

The terms *satellite image classification* and *map production* have specific meaning in the field of remote sensing. Satellite image classification describes assignment of global labels to entire scenes. In this process spatial information is discarded from the output since only a single class label is associated with the scene. In contrast, LULC map production involves producing maps by assigning a class to each pixel. Thus, spatial information is maintained to provide contextual meaning. The pixel-wise classification method of satellite imagery presented in this paper differs from generic satellite image classification. In this light, it is worth mentioning such related works differ from the map production approach that also uses deep learning. Castelluccio *et al.* [30] explored the use of DCNNs for classification of remote sensing scenes using two contemporary architectures CaffeNet and GoogleNet. Here, two datasets, UC-Merced and Brazilian Coffee Scenes, were considered since each dataset has 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 include the near-infrared (NIR) band 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.* [31] proposed a novel classification framework for classifying satellite images called DeepSat. Here, 150 features

were extracted from two datasets (SAT-4 and SAT-6) containing four bands (red, green, blue and NIR). Some of the features extracted included 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 [32]. This network outperforms the classical DBN on the target dataset. Marmanis *et al.* [33] explores 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.

3 Base Networks for LULC mapping and analysis

This section introduces three popular DNN architectures which we refer to as *base networks*. Specifically, VGGNet [11], GoogLeNet [12] and ResNet [13] are popular networks that have been submitted to past ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [7]. Each architecture was originally designed for the purpose of image classification and object detection. These 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 [8] (recall the base network in [8] was a VGGNet). These base networks were originally designed as classification networks that produce non-spatial outputs.

These classification networks can be re-purposed for semantic segmentation tasks by re-interpreting fully connected layers as fully convolutional layers [8]. Here, these base networks are adapted into FCNs (labeled as FCN-[VGG, ResNet, GoogLeNet]) that take arbitrary sized input and produce semantic segmentations [8]. This adaptation is rather trivial and the output segmentation images produced are coarse with spatially reduced dimension size. To produce good representative spatial output maps, the coarse output is passed through a stack of transpose convolution layers which increases dimension size and connects coarse output to dense pixels [8]. In this design, the first part of the network (*i.e.* the base network) is referred to as an encoder that acts as 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. An example of this network structure is the FCN-VGG depicted in Fig. 2.

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

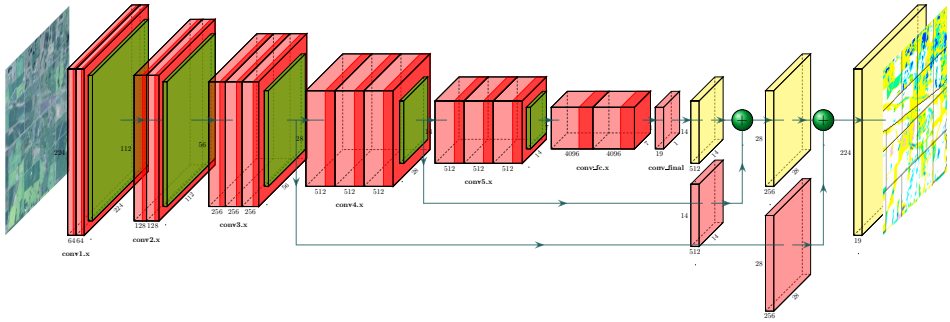


Fig. 2 FCN-VGG: Encoder-Decoder Architecture

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. In addition, all fully-connected layers were removed from the original base networks, which follows the approach described in [8]. In other words, these classifiers were adapted for dense prediction and up-sampling using transpose convolutions. The details of each network architecture and modifications are described below.

3.1 FCN-VGG

The VGG network architecture VGG 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 [11]. In this work, the VGG-16 layer net is re-purposed for semantic segmentation (in the same manner as Long et al [8]). 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 score 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 [17]. To summarize, the same architecture as described by [8]

is used in this work with only a modification of the first layer as described above. The network is fully illustrated in Table 1, which shows operations across all layers of the FCN-VGG-16 network.

Table 1 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)

3.2 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 [13]. In this work, a 101-layer deep network 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 $(1 \times 1 \times X), (3 \times 3 \times X), (1 \times 1 \times X)$ 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 [8]). Otherwise, the ResNet-101 base architecture (without fully connected layer and softmax classifier) described in [13] is retained, and only the first layer modified. The network architecture is fully defined in Table 2

Table 2 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)

3.3 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 [12], won 1st place in the ILSVRC15 classification task. Although new improvements [34, 35] has been made on the original Inception network [12], 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. 3. The $(1 \times 1 \times X)$ convolution kernels perform dimension reduction along the depth vector of the input, reducing the number of parameters, to make the network computationally efficient.

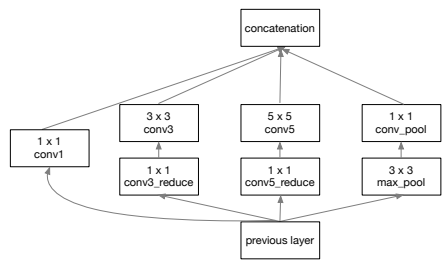


Fig. 3 Architecture Design of Inception Module

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-sampling layers with skip connections following the approach of [8]. Table 3 shows details of the network structure.

Table 3 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)

4 Proposed Extensions to Base Networks

A major contribution of this paper is the network extensions introduced 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 methods is tested on each of the base networks. The results achieved from the extensions are a major improvement to our previous results reported in [10] 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.

4.1 Context Module Extension

The first extension was to add a vestige plug-in called a context module. This module introduced in [14] is explicitly designed as a plug-in system consisting of dilated convolution layers stacked from top to bottom. By incorporating dilated convolution layers, the convolution operation is modified to include a dilation factor which expands the field of view exponentially while its parameters grows linearly [17]. With little increase to the complexity of the base networks, we append this module at the end to take up-sampled feature maps as input and pass them through a series of these layers to expose more contextual information thereby increasing accuracy. The architecture of this module is summarized in Table 4. Note that dilation factors and convolution filters size considered in designing this module is the same as in [14].

Table 4 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)

4.2 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 [15] 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 [36], wherein configuration for the layers follows 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

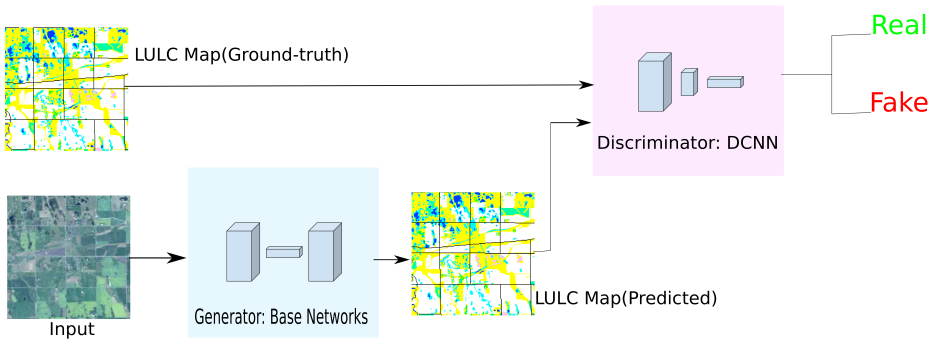


Fig. 4 Adversarial Extension

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.* [37]. 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, the loss function is based on the following: ground-truth (denoted by $\mathbf{y}$), predicted output (denoted by $\hat{\mathbf{y}}$), C denotes the number of classes (see, *e.g.*, Fig. 5), y_{ic} is the correct probability i for class c, and $\hat{y}_{ic}$ is the predicted probability i for class c. Thus, the multi-class entropy loss (denoted by ℓ_{mce}) is defined as

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

Next, z denotes the binary probability for predicted output (0) and ground-truth (1), and $\hat{z}$ represents predicted probability between (0 and 1). Then, the binary class entropy loss (denoted by ℓ_{bce}) is defined as

$$\ell_{bce}(z, \hat{z}) = -(z \ln(\hat{z}) + (1 - z) \ln(1 - \hat{z})). \tag{2}$$

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 loss function

$$\sum_{n=1}^{N} \ell_{bce}(d(\mathbf{x}_n, \mathbf{y}_n), 1) + \ell_{bce}(d(\mathbf{x}_n, g(\mathbf{x}_n)), 0). \tag{3}$$

The generator in an adversarial role not only minimizes 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 loss function

$$\sum_{n=1}^{N} \ell_{mce}(g(\mathbf{x}_n), \mathbf{y}_n) + \lambda \ell_{bce}(d(\mathbf{x}_n, g(\mathbf{x}_n)), 1), \tag{4}$$

note, λ is applied as a constant regularization function.

Furthermore, the parameters $(\boldsymbol{\theta}_g, \boldsymbol{\theta}_d)$ of the generator and discriminator respectively are adjusted by minimizing a hybrid loss function defined as

$$\ell(\boldsymbol{\theta}_g, \boldsymbol{\theta}_d) = \sum_{n=1}^{N} \ell_{mce}(g(\mathbf{x}_n), \mathbf{y}_n) - \lambda(\ell_{bce}(d(\mathbf{x}_n, \mathbf{y}_n), 1) + \ell_{bce}(d(\mathbf{x}_n, g(\mathbf{x}_n)), 0)). \tag{5}$$

Lastly, it is important to mention the convention followed for training the base networks in this adversarial setting. The base networks are pre-trained. In the adversarial setting the pre-trained weights are restored while the discriminator starts training from scratch. This can be re-described as fine-tuning the base networks with a discriminator. However, in this setting such fine-tuning is based on the redefined loss functions. By restoring the previous state of the base network which has reached a point of diminishing accuracy, any improvement in accuracy is easily attributed to the addition of a discriminator network.

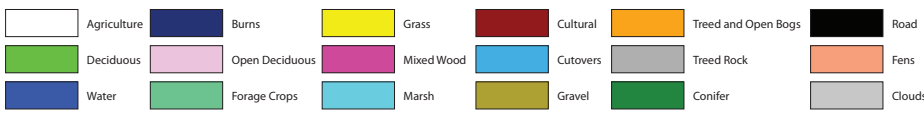


Fig. 5 GeoManitoba LULC Classes

5 Implementation Details

5.1 Dataset Acquisition

The dataset used in this work was originally described in [10] and is briefly described here. Specifically, Landsat 5/7 satellite images of the southern agricultural growing region of Manitoba (see the red outline in Fig. 6) 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 differ from typical RGB images since satellite data contains three additional infrared channels, namely *near infrared (NIR), shortwave infrared 1 (SWIR1), and shortwave infrared 2 (SWIR2)*. As was mentioned, this difference necessitated the base network modifications described in Section 3. Each pixel in the satellite image represents a 30 m x 30 m square area of land. The dataset contains raw Landsat 5/7 satellite images (*i.e.* unclassified data) and LULC (*i.e.,* labeled, ground-truth data). The labeled data was created using semi-automated methods used in [38] and ground-truthed by GeoManitoba. A total of eighteen Landsat 5/7 scenes (see the green outlines in Fig. 6) were used to produce the maps. Example labels include water, grassland, marsh, deciduous, coniferous, road, and agriculture. The full list is given in Fig. 5, and an example of a GeoManitoba LULC map created from Landsat 5/7 data from 2004 is given in Fig. 8. Additionally, the GeoManitoba dataset was augmented with satellite images containing clouds and a new class was added to the list provided by GeoManitoba. This was done to prevent the networks from misclassifying clouds into one of the other classes.

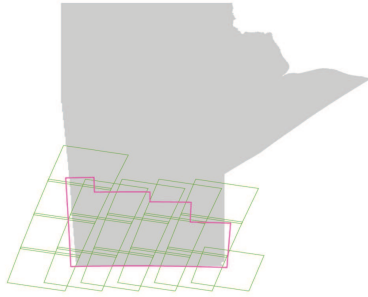
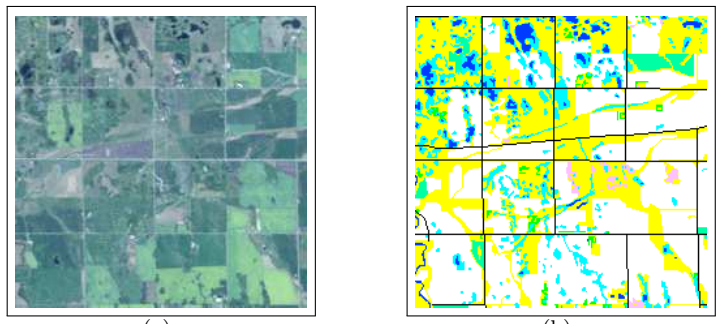


Fig. 6 Province of Manitoba with the southern agricultural growing region (red) and the associated Landsat 5/7 scenes that cover this area (green).

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 [7], consisting of ∼1,2 million images and 1000 categories. The LULC map provided by GeoManitoba had a resolution of (13777 × 16004 × 6), which posed some problems. Firstly, the original base



(a) (b)

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

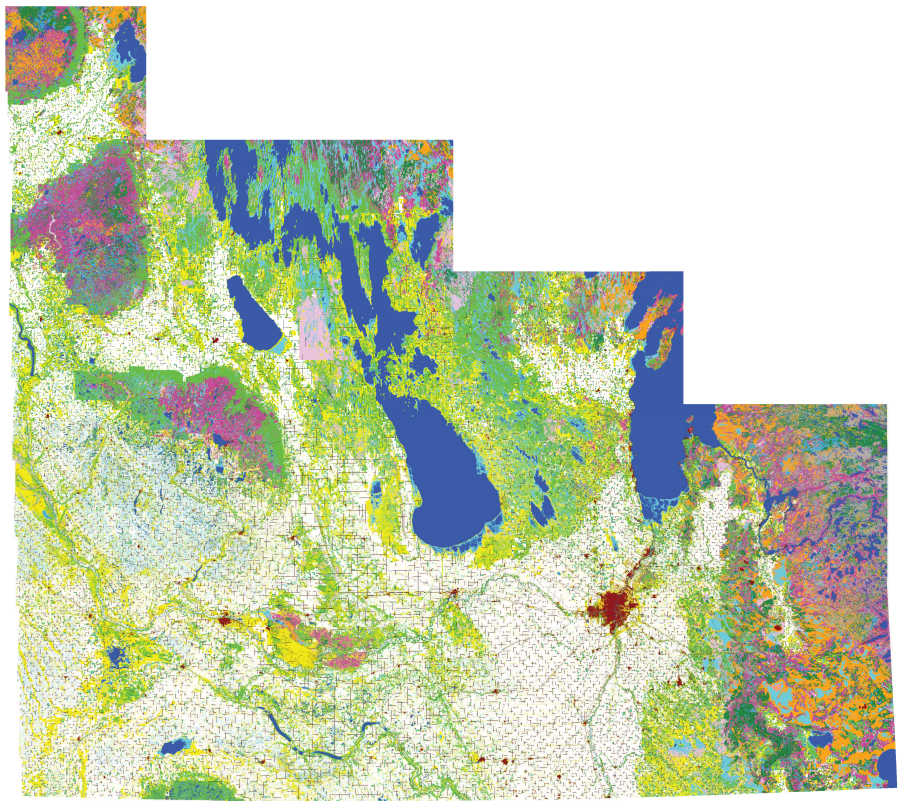


Fig. 8 2004 LULC map provided by GeoManitoba.

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. 9. The first set of tiles were produced by the method shown in Fig. 9(b), namely the tiles were non-overlapping. To further increase the size of the dataset, the tiling process in Fig. 9(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. 9(a), the total number of tiles generated in Fig. 9(b) was increased by more than $4\times$ due to the fact that the resolution of the map in Fig. 9(a) is not a multiple of 224. For example, the non-overlapping tiling produces ∼4000 plus tiles while the 1/2 tile overlap produces ∼17000 tiles.

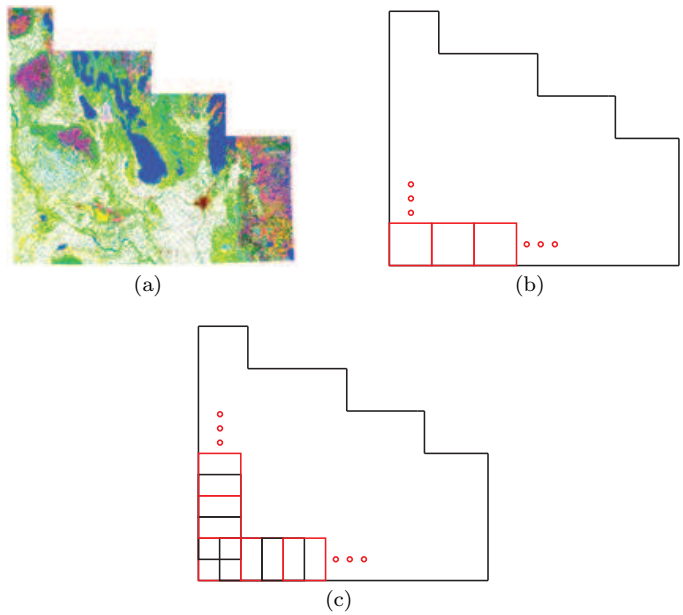


Fig. 9 Illustration depicting the tiling process. (a) GeoManitoba's LULC map of southern Manitoba, (b) non-overlapping tiles, and (c) 1/2 tile overlap.

5.3 Experimental Setup

All networks were trained and evaluated using the TensorFlow deep learning framework [39] on a NVIDIA Digits DevBox [2] 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-batch size of 2 was maintained across all networks due to GPU memory constraints. Subsequently, learning rates of $10^{-4}, 10^{-5}, 10^{-9}$ were used for experimentation,

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

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 epochs, thereby completing training after 200 epochs. The Adam optimization algorithm [40] was used to update network weights since it allowed the networks to converge quickly when compared to RMSprop and (SGD + Momentum) optimizer [41].

6 Experimental Results and Analysis

The dataset consists of training and validation sets. A larger set created from tiling is separated following an 80/20 split. Each set is pruned to remove empty tiles *i.e.,* tiles in which all pixel values equal zero. In total, 18054 images are used for training and 958 images for validation. As discussed, the networks evaluated 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. The following common segmentation metrics were aggregated over validation images (predicted and corresponding ground-truth).

Let n_{ij} be the number of pixels with ground-truth label i whose prediction is label j. Also, $t_i = \sum_{j=1}^{C} n_{ij}$ denotes the total number of pixels labeled with label 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. Firstly, global pixel accuracy is the ratio of correctly classified pixels to total pixels summed over all classes and it is defined as

$$\frac{\sum_{i=1}^{C} n_{ii}}{\sum_{i=1}^{C} t_i}. \tag{6}$$

Secondly, the per-class accuracy metric is computed as

$$\frac{1}{C} \sum_{i=1}^{C} \frac{n_{ii}}{t_i}, \tag{7}$$

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

$$\frac{1}{C} \sum_{i=1}^{C} \frac{n_{ii}}{t_i + \sum_{j=1}^{C} n_{ji} - n_{ii}}, \tag{8}$$

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 that class by the ground-truth and predicted.

Beginning with the base networks, Table 5 shows the results based on the validation set. FCN-ResNet-101 performed the best with a global accuracy of 88.25%, which slightly outperformed FCN-VGG16, while FCN-GoogLeNet performed worst with a global accuracy of 62.13%. Prior to experimentation, we hypothesized the consecutive classifiers networks VGG → GoogLenet → ResNet (that incorporated new architectures and improved on the weakness of their predecessors) to perform consecutively better, similarly to the ILSVRC challenge (92.7% → 93.3% → 96.4%, respectively). However, this was not the case with our

results, specifically with the dense pixel classification tasks defined in this work. Here, we observe max pooling operations as a drawback affecting performance, which is also identified in [14]. Although max pooling achieves translational invariance and yields computational savings, it is responsible for the loss of a lot of valuable information. More precisely, max pooling disregards relationship between pixels which is paramount for dense pixel classification. From the architectural definitions in Tables 1, 2, 3 each base network uses max pooling operations in varying degrees. Moreover, FCN-GoogLeNet liberally employs max pooling operations within its inception modules and across its structure. While this gives a computational advantage, it negatively impacts classification performance in the case of semantic segmentation of images. The speed in training FCN-GoogLenet is very noticeable compared to the other base networks, but performance gains quickly plateaus after a number of epochs. Initially, this problem was attributed to the network overfitting, and dropout [42] and batch normalization [43] were employed to combat overfitting and help with training. 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). These results are reported in Table 6. 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 7 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 8 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. Examples of the LULC maps produced by the best network **FCN-ResNet-101 + Context + Adversarial Network** is given in Figs. 10 & 11. Each extension achieves performance gains in varying degrees with the context module serving to aggregate contextual information and adversarial extension providing a means of learning the structure for high-order potentials to enforce label consistency. Consequently, we establish that combining the strength of both extensions leads to significant performance gains.

Table 5 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%

The results generated from the best network in form of an *error matrix* or *confusion matrix* presented in Table 9 provides a base for LULC map analysis. The error matrix shows the percent accuracy for each LULC class. The No Data class is easily classified by the network since its label is very distinguishable from other labels, *i.e.,* this class is characterized by a vector of all zeros. Some other classes

Table 6 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 7 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 8 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%

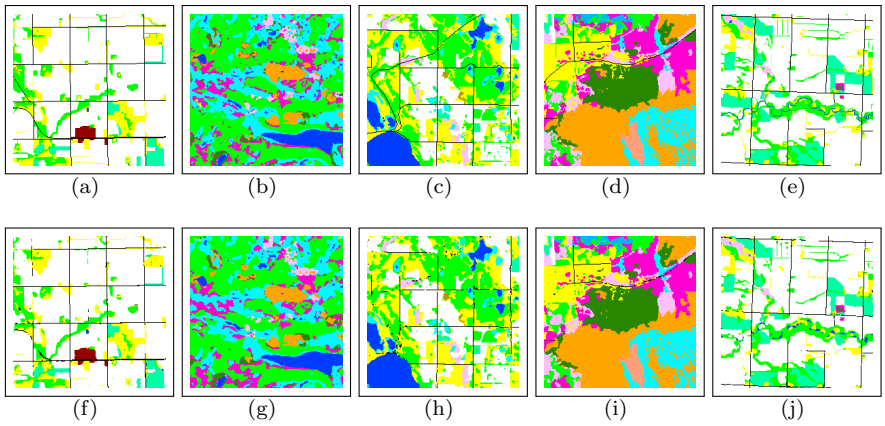


Fig. 10 Sample validation set results. (Top row) ground-truth labellings, and (bottom row) result from FCN-ResNet-101 + Context + Adversarial

easily classified by the network are Agriculture, Water, Treed Bog, Forage and Fens. In our previous work [10] Roads and Burns were identified as two classes with lowest accuracies, and, similarly, both classes achieve the lowest accuracies in this work. The class Road exists at a single pixel level while Burns is underrepresented in terms of the total number of pixels presented to the networks; we hypothesize these as the probable cause of low accuracies for each class. However, in comparison with the best network from our previous work, this new proposed network ensemble reliably detects features like roads that exist at a single pixel level and is able to consolidate underrepresented classes. The accuracy of Burns is improved from 44.34% to 51.89% and more noticeably the accuracy of Roads is increased from

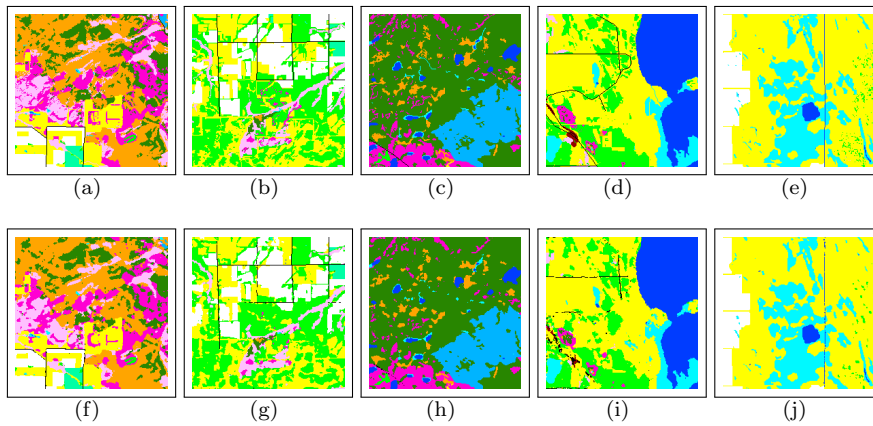
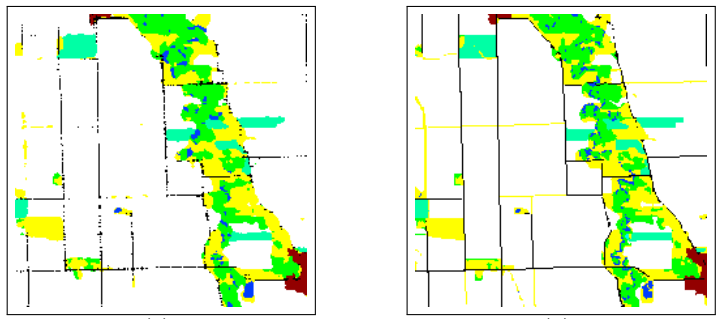


Fig. 11 Sample validation set results. (Top row) ground-truth labellings, and (bottom row) result from FCN-ResNet-101 + Context + Adversarial

Table 9 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**

57.20% to 70.88%. This is shown visually in Fig 12. Such gains can be attributed to the ensemble of extensions that were added to the base networks. The network employed in our previous work [10] losses fine details at the single pixel level, which is the same for the base networks examined without any extensions; this is due to a series of downsampling and upsampling operations causing fine detail loss. In the final analysis, we observe that by combining the extensions (Context + Adversarial), we are able to solve this problem. Overall, our proposed deep learning framework successfully discriminates and classifies very similar classes based on spectral cues and produces highly accurate maps. In addition, as a post-processing last step in our deep learning framework the produced individual maps are stitched back in place to recreate a larger LULC map same as in Fig. 8.



(a) (b)

Fig. 12 Comparison of LULC maps. (a) LULC map output from previous work [10] and (b) LULC map produced from FCN-ResNet-101 + Context + Adversarial

7 Conclusion

We have introduced a deep learning framework for LULC mapping and analysis. Three modern networks denoted as *base networks* in this paper were adapted into FCNs and modified to take input Landsat 5/7 satellite images with six bands. Furthermore, the networks were extended to improve accuracy by the extensions: 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 reported.

The results reported in this paper show that deep convolutional neural networks perform well in the production of LULC maps. In addition, it takes only 8 minutes and 42 seconds to produce a map of the southern extent of Manitoba with a trained model, effectively automating production. This is a phenomenal reduction considering that the current semi-automated approach takes 4,800 hours. Furthermore, by extending the networks, there was a marked improvement in the results compared to the networks without extensions. The best network with no extensions produced a global accuracy of 88.25% while the overall best network with a combination of the two extensions produced a global accuracy of 90.46%.

Future works will include adapting, modifying and extending other modern network architectures [44, 45] to further improve results. Additionally, training on 16-bit Landsat 8 dataset will be a part of future research.

8 Acknowledgements

This research has been supported by through a Queen Elizabeth II Diamond Jubilee scholarship and by the Natural Sciences and Engineering Research Council of Canada (NSERC) discovery grants 194376 and 418413. Moreover, this work would not have been possible without the support and cooperation of GeoManitoba.

References

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. 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
3. Mera D, Fernández-Delgado M, Cotos JM, Viqueira JR, Barro S (2017) Comparison of a massive and diverse collection of ensembles and other classifiers for oil spill detection in sar satellite images. Neural Computing and Applications 28(1):1101–1117
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. 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
7. 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
8. 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
9. 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., *in press*
10. Henry CJ, Storie CD, Palaniappan M, Alhassan V, Swamy M, Aleshinloye D, Curtis A, Kim D (2019) Automated lulc map production using deep neural networks. International Journal of Remote Sensing 40(11):4416–4440
11. Simonyan K, Zisserman A (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:14091556
12. 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
13. 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
14. Yu F, Koltun V (2015) Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:151107122
15. 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
16. Thoma M (2016) A survey of semantic segmentation. arXiv preprint arXiv:160206541
17. Dumoulin V, Visin F (2016) A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:160307285

18. 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
19. 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
20. 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
21. 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
22. Chen LC, Papandreou G, Schroff F, Adam H (2017) Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:170605587
23. 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
24. 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
25. 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
26. 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
27. 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
28. 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
29. Alam FI, Zhou J, Liew AW, Jia X, Chanussot J, Gao Y (2017) Conditional random field and deep feature learning for hyperspectral image segmentation. arXiv preprint arXiv:171104483
30. 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
31. 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
32. Carreira-Perpiñán Ma, Hinton GE (2005) On Contrastive Divergence Learning. Artificial Intelligence and Statistics 0:17
33. 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

34. 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
35. 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
36. Isola P, Zhu JY, Zhou T, Efros AA (2017) Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:161107004
37. Luc P, Couprie C, Chintala S, Verbeek J (2016) Semantic segmentation using adversarial networks. arXiv preprint arXiv:161108408
38. 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
39. 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
40. Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980
41. Ruder S (2016) An overview of gradient descent optimization algorithms. arXiv preprint arXiv:160904747
42. 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
43. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:150203167
44. 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
45. Chen L, Zhu Y, Papandreou G, Schroff F, Adam H (2018) Encoder-decoder with atrous separable convolution for semantic image segmentation. arXiv preprint arXiv:180202611