UNIVERSITY OF AGDER

Deep Convolutional Neural Networks for Semantic Segmentation of Multi-Band Satellite Images

Fredrik Bore
Andreas Taraldsen

SUPERVISORS
Assoc. Prof. Morten Goodwin
PhD Candidate Jahn Thomas Fidje

Master's Thesis
University of Agder, 2018
Faculty of Engineering and Science
Department of ICT

UiA
University of Agder
Master's thesis

Faculty of Engineering and Science
Department of ICT

Abstract

Semantic segmentation of images is of increasing interest in the field of computer vision and machine learning. Accurate and efficient segmentation methods is required for many of todays modern applications. This thesis provides a review of deep learning methods for semantic segmentation of satellite images. Firstly, we compare different state-of-the-art methods. Next, we explore the benefits of using multiple spectral bands of data as compared to the traditional RGB bands. Finally, a look at future possibilities with segmentation using capsule networks.

Acknowledgements

We would like to express our very great appreciation to Assoc. Prof. Morten Goodwin for his valuable and constructive suggestions during the writing of our thesis. His willingness to give his time so generously has been very much appreciated.

We are also particularly grateful for the assistance given by PhD. Candidate Jahn Thomas Fidje. His useful and constructive recommendations helped us take the right decisions.

Finally, we would like to thank Kristian Måkestad for his input.

Table of Contents

List of Figures

List of Tables

Part I

Research Overview

Chapter 1

Introduction

The amount of satellite launches increases each year containing communication devices, cameras and sensors. Used by governments and private companies to do multiple tasks. In context to the development of the way humans work it will be useful to utilize these satellites in a manner that would benefit the users even more than it is today.

The development in the field of artificial intelligence and deep learning has grown exponentially the last years. More computational power and data available than ever before creates opportunity to evolve our knowledge and usage of this field. Earlier both artificial intelligence and deep learning was just ideas that could not be implemented in a proper manner to be used in an industrial way. Now it has changed and is already being used in the industry. Deep learning is proven to solve multiple problems efficiently and will a be crucial technique in the future for problem solving and automation.

Representation and identification of objects from satellites with high speed and accuracy would make an impact on large and important industries. The current systems deliver a good accuracy based on the methods being used. However, the limitation of the algorithms is that they classify and detect regions of interests and mark these by the result of classification. This will only be a limitation when working with high resolution images that have small details. To achieve a even greater result and a pixel-perfect accuracy, a more advanced approach is needed.

This thesis explore the possibilities of deep learning in image segmenta-

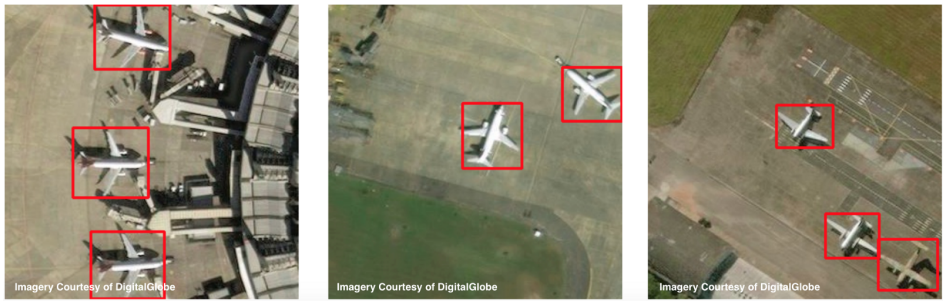


Figure 1.1: Result of classifying a region within an image[1]

tion and compare state-of-the-art algorithms in semantic segmentation using satellite images with multiple spectral bands of data. There will be multiple tests to evaluate the different approaches and modifications of algorithms are also going to be discussed.

1.1 Problem statement

Semantic segmentation is a relatively new area of research and is still being further developed. A result of this is a myriad of solutions to the same problems and that all try to outperform each other. There are as of today no blueprint of what is the best approach.

Usually when testing image classification algorithms, normal images picturing items in normal perspective are being used. Every experiment done in this thesis use aerial satellite imagery because we will like to see how these approaches perform in an environment that could be a reality. The consequence of this could be a lower accuracy and performance than in the original papers.

Solving problems like image classification and semantic segmentation requires data. The used dataset is delivered by Defence Science and Technology Laboratory owned by the Ministry of Defence of the United Kingdom. It includes normal 3-bands and extraordinary 8-bands imagery. This dataset is available on kaggle.com[2], after being a competition in 2017. Every image has however a dimension of 1km x 1km which makes the objects smaller compared to objects from other popular datasets such as Cityscapes[3], VOC2012[4] and CamVid[5].

The primary goal of this thesis is to examine the best state-of-the-art algorithm regarding semantic segmentation used on satellite imagery. The results will show the differences on using 3-band and 8-band and potential flaws in the current solutions.

Due to the fact that almost all approaches on semantic segmentation does not experiment with dataset of satellite imagery, we want to find the most appropriate approach for this specific usage. We will experiment with different approaches, multiple number of spectrum bands and trying to modify these to fit our data.

Research questions

1. Which state-of-the-art deep learning image segmentation algorithm has the best accuracy on satellite images?
2. Does increasing the number of spectrum bands from 3 to 8 increase classification accuracy?
3. Does a out-of-the-box implementation of CapsNet work for semantic segmentation?

Hypotheses

1. Newer approaches will give higher accuracy than older ones.
2. An increased amount of spectral bands will increase accuracy, but training time will be increased.
3. Semantic segmentation with CapsNet should work, but the accuracy will decrease.

1.1.1 Contributions

This thesis explores the current state-of-the-art for semantic segmentation of satellite images. In addition it explores the performance benefits of using several spectral bands of data.

1.1.2 Report Outline

Chapter 2

Theoretical Background

2.1 Digital image processing

Computer vision started in the late 1960's at universities and institutes that are trying to achieve artificial intelligence. At that time this field of study was trying to replicate the human visual system as a step toward to implement intelligence into robots. The plan was to finish in the end of summer in 1966, as a summer project with the idea of attaching a camera to a computer and get it to describe what it saw.[6] Computer Vision has since that time been forked into many new areas of artificial intelligence.

Digital image processing is the topic of this thesis and it allows algorithms to process digital images. When the work on digital image processing started there was developed applications to satellite imagery, medical imaging and character recognition amount other things. The cost of processing was high compared to the hardware available in the beginning but changed in the 1970's and cheaper computer was released and dedicated hardware became available. The algorithms at that time was simple and basic mathematics, and are not classified as artificial intelligence. In digital image processing there are several different tasks that can be made such as classification, localization, object detection and segmentation.

Depending on the task there are different approaches. Often, the more complex the tasks are the longer time it takes to run. Therefore it is important

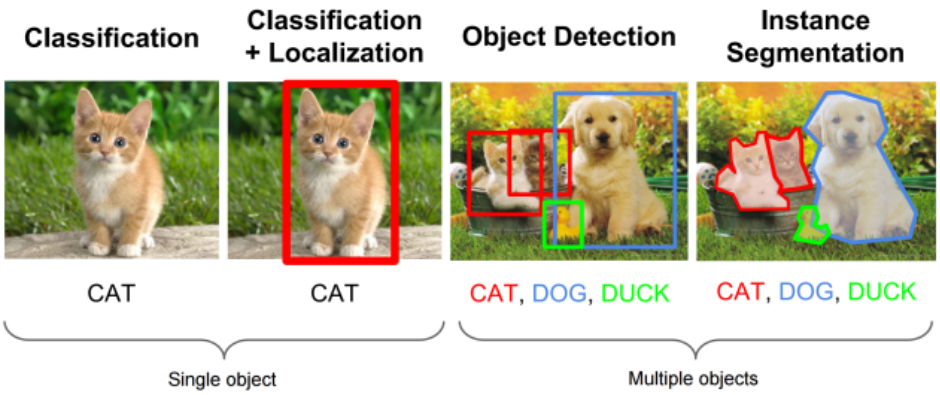


Figure 2.1: Overview over different technologies[7]

to choose the most fitting method for each specific task.

Classification The goal is to classify a single object in the input image. "Is there a cat in this image?"

Classification + Localization This combination result in not only can the algorithm say if it is an object in the image, but can also localize the single object.

Object Detection Makes it possible to find multiple objects in an image and classify them to correct classes.

Segmentation Doing the same job as object detection, but gives a much more accurate pixel-perfect result. These algorithms classify each pixel

The approaches of digital image processing has changed and deep neural networks are being used in all of the best performing algorithms available.

2.2 Convolutional Neural Networks

Convolutional neural networks is a class of deep feed-forward neural networks that has been applied to analyzing visual imagery. Being able to

extract features throughout the network makes them an essential part of image and video recognition algorithms and as of now is the most viable option for this operation. Being integrated in almost every solution for solving the classification problem

The networks were designed by the core principles equal to the biological processes. The functionality allows the algorithms to connect patterns between neurons that resembles the organization of the animal visual cortex. A individual neuron responds to stimuli only in a restricted region of the visual field. The visual fields of various neurons overlap in a way that it covers all of the visual field.

Various different neural network consist of three different types of layers, input layer, output layer and a number of hidden layers. Typically the hidden layers consists of a series of convolutional layers, pooling layers, fully connected layers and normalization layers. It will vary from algorithm to algorithm because the architecture that the algorithms use and its purpose.

Convolutional Make a convolution operation on the input and sends it to the next layer. The operation matches the response of an individual neuron to visual stimuli[8]. This layer is also the core building block of a CNN with its parameters consist of a set of kernels/filters.

Pooling Both local and global pooling exists. Combine outputs from multiple neurons in a layer and sends it to one neuron in the next layer[9]. The pooling is a kind of a non-linear down-sampling with max pooling as a popular example.

Fully Connected Connects every neuron in one layer with every neuron in the next.

Because CNN perform with good accuracy, it makes CNN obvious choice for most algorithms within the field of image and video recognition. Popular choices such as ResNet[10], VGG[11] and Inception[12] are being well used and deliver acceptable performance. The development has increased by the years but some problems and limitations the the solutions are still there, which hinders the system to carry out the core ideas of being a replication of the human visual system.

CNNs operates on complex tasks. The computational cost are therefore high and to make a real-time object detection require sacrifices on the algorithms.

Due to the complexity of the neural network and the way the neurons learn gradually training data is essential to have large amounts of. The algorithms are being used in applications as they are today, but reducing the necessary amount of data could potentially make room for new applications and field of use.

Chapter 3

State-of-the-Art

Working with such a popular field as deep neural network, results in a massive development ratio compared to other areas of research. Semantic segmentation is a relatively new method of deep learning and computer vision. Therefore not only one approach leads the way. Multiple methods have state-of-the-art performance.

3.1 Architectures

This sub-chapter contains a listed structured overview over the state-of-the-art deep learning architectures. They are sorted by the age of the architecture with the oldest coming first. ResNet being the first, with DenseNet right after and the sub-chapter ending with Inception. These are all popular choices among deep learning approaches. Ending with an overview of the architectures.

3.1.1 ResNet

In earlier stages of image classification using convolutional networks, AlexNet and VGG were two popular networks. The publication of AlexNet was seen as a massive breakthrough in the field at that time. From the release of AlexNet in 2012 and until 2015 those two were the ones who delivered the

best accuracy and performance.[11][13]

Developed by a team from Microsoft Research, they published an approach *Deep Residual Learning for Image Recognition* with the intention to overcome the underlying problems of VGG and AlexNet. The obstacles that AlexNet and VGG have difficulties with, are scalability within the network. When deeper networks started converging, degradation problems were exposed. With the network depth increasing, accuracy gets saturated and then degrades rapidly.[11][13][10]

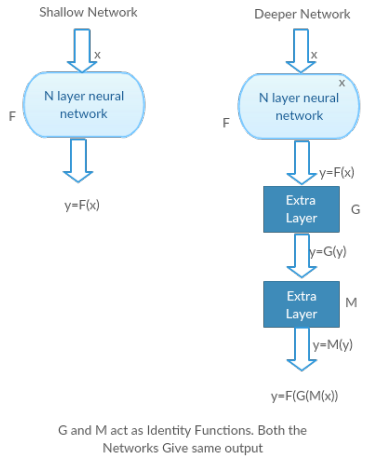


Figure 3.1: Identity function with both shallow and deep network[14]

ResNet's core idea was to implement residual blocks "identity shortcut connection" that skips one or more layers. Figure 3.1 shows the complexity with multiple layers without this approach. Instead of having a network learning a direct mapping of $x \rightarrow y$ using a function $H(x)$, which is a few stacked non-linear layers[10]. ResNet defines the residual function using:

$$x_\iota = H_\iota(x_{\iota-1}) + x_{\iota-1} \tag{3.1}$$

When the identity mapping reaches optimum it pushes the residuals to zero and fits an identity mapping.

With these actions and modifications the ResNet outperforms the current state-of-the-art convolutional networks. Since the publication the approach

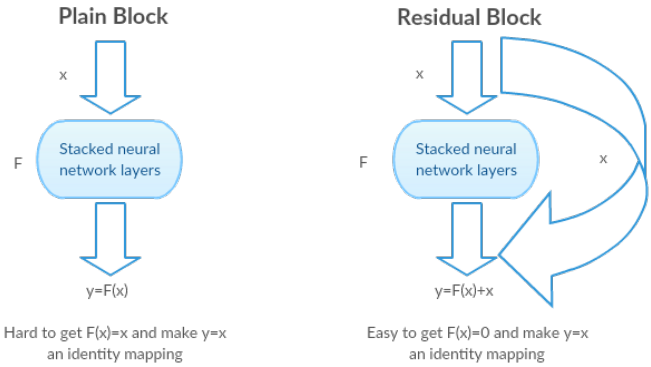


Figure 3.2: Residual block compared to a plain block[15]

has been well used and even replaced VGG in the object detection algorithm Faster R-CNN[16].

3.1.2 DenseNet

Facebook AI Research developed a convolutional network architecture named DenseNet which in 2017 won the best paper award at Computer Vision and Pattern Recognition(CVPR)[17]. Their architecture did as ResNet did before them and introduced a new block, Dense Block. [10][18]

DenseNet connects each layer to every other layer in a feed-forward fashion. Traditional convolutional networks with N layers also has N connections going from one layer to another. In this network the number of direct connections is $N(N+1)/2$. In every layer the feature maps of all the previous layers are being used as inputs and their own feature maps are used as inputs to all subsequent layers.[18]

As a direct result of the architecture and the dense patterns this approach requires fewer parameters than traditional convolutional networks hence it is no need to relearn redundant feature maps. While ResNet has a large amount of parameters because every layer has its own weights. DenseNet explicitly differentiates between added information and preserved information. The layers are very narrow compared to other networks which adds a small set of feature maps to the collective knowledge of the network and

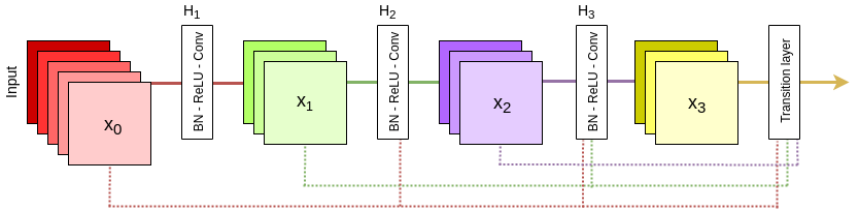


Figure 3.3: A dense block. Each layer takes all preceding feature map as input

lets the remaining maps unchanged. The final classifier makes a conclusion based on all feature maps in the network.[18]

A big advantage of DenseNet is the improved information and gradient flow throughout the network. Each layer has access to the gradients using the loss function and input signal which makes it implicit deep supervision.[19]

A major difference to ResNet, this network introduces concatenation of feature maps where ResNet has a skip-connection with identity function. This is why this architecture is named Densely Connected Convolutional Networks(DenseNet).[10][18]

$$x_\iota = H_\iota([x_0, x_1, ..., x_{\iota-1}]) \tag{3.2}$$

3.1.3 Inception

The first version of Inception, at that time named GoogLeNet, was released in 2014. The hallmark of this architecture is the improved streamlining of the computing resources within the network. In the way they designed the architecture allowed a more deeper and wider network without effecting the computing resources and keeps it constant.[12][20]

Google managed this by introducing Inception modules, a new type of block into the network. These modules have multiple and small convolutions to reduce the number of parameters compared to other state-of-the-art approaches while still having a deeper architecture. [12]

Multiple versions of Inception have been released, with the latest called

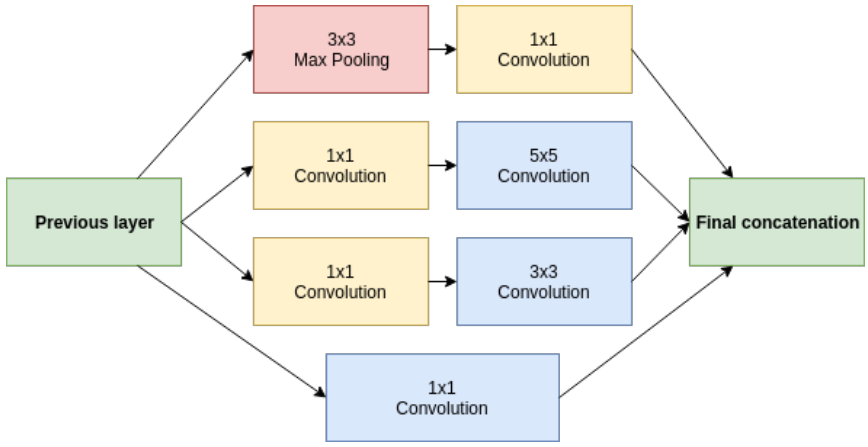


Figure 3.4: Inception module with dimension reduction

Inception-v4. The Inception architecture is highly tunable and that which was utilized by tuning the layer sizes carefully to balance the computation between sub-network models. The introduction of TensorFlow in Inception-v4 did that the most recent models was trained without partitioning the replicas. This is possible due to optimization of memory used by back-propagation and carefully considering what tensors are needed for computation and structuring to reduce the number such tensors[12].

In the same paper as Inception-v4, Google introduced Inception-ResNet. An architecture based on the combination of previous Inception and residual blocks. By doing this the training time was heavily reduced[10].

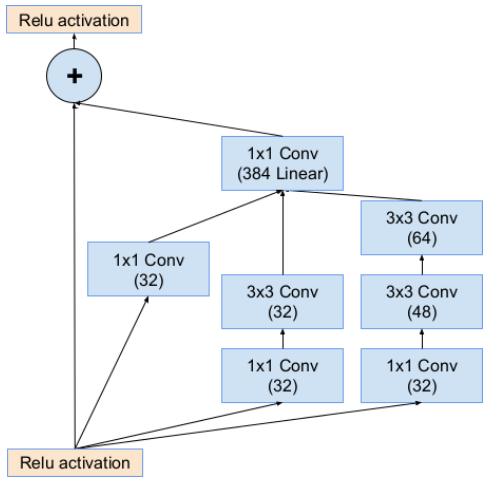


Figure 3.5: The schema for 35x35 grid module of Inception-ResNet network[12]

Architecture	Developed by	Released
AlexNet	SuperVision Group w/ Geoffrey Hinton	2012
ResNet	Microsoft Research	2015
Inception v4	Google Research	2016
DenseNet	Facebook AI Research	2017

Table 3.1: Overview of popular models

3.2 Semantic segmentation

There are multiple approaches of semantic segmentation and the most recent and popular methods are written about in this chapter. There will be explanations of the algorithms with models. At the end of the sub-chapter there are a overview of who developed them and when it was released.

3.2.1 Fully Connected Networks (FCN)

CNNs is not only superior for image classification, but also making way for other local tasks with a more structured output. Object detection is

an example of a network who use a convolutional neural network and has structured output with their bounding boxes[1][21].

As being seen as the first useful semantic segmentation algorithm, Fully Connected Networks for Semantic Segmentation by J. Long, E. Shelhamer and T. Darrell[22], used findings and research on CNNs as their foundation of the work. They developed an approach that predicted each pixel in the input image. This was the first approach that worked with trained end-to-end for pixel-wise prediction and from supervised pre-trained.

Before this algorithm was released the other approaches used fixed-sized inputs and the fully connected layers of these networks had fixed dimensions and throw away spatial coordinates. In FCN these layers are viewed as convolutions with kernels that cover the whole input region and casts them into fully convolutional networks. This lets the input be any size and output a classification map.

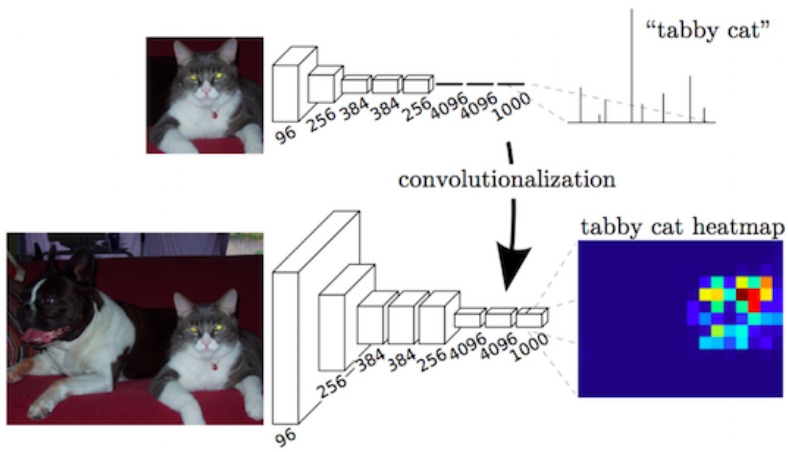


Figure 3.6: Fully Convolutional Network Architecture[22]

The segmentation architecture casts classifiers into fully connected network then increase them for dense prediction with pixel-wise loss and train for segmentation by fine-tuning[22].

When this paper was released in November 2014 the algorithm outperformed the current state of the art performance and in the aftermath considered as the first real segmentation algorithm and more recent algorithms has used their ideas and principles.

In our thesis FCN is being used and experimented with to give a baseline for the results.

3.2.2 U-Net

Based on the previous approaches like FCN, a more elegant architecture was built. U-Net made modifications and extended the FCNs architecture so that the network would need fewer training images and yield more precise segmentation[22][23].

The motivation and idea behind this approach was to supplement a usual contracting network by successive layers, such that pooling operators was replaced by upsampling operators because these layers increases the resolution of the outputs. One of the largest modifications in U-Net's architecture is in the upsampling. Here there are a large number of feature maps that gives the network the opportunity to propagate context information to a higher resolution layer. This is why the architecture model is shaped like an U[23].

U-Net was developed for bio-medical image segmentation tasks and won ISBI cell tracking challenge 2015[24], by training the network on transmitted light microscopy imagery[23].

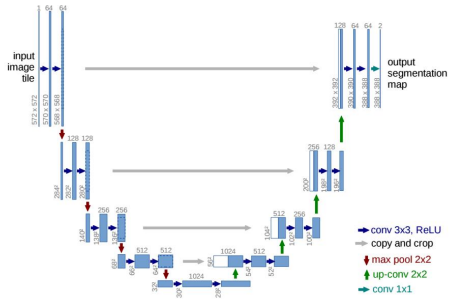


Figure 3.7: U-Net Structure[23]

3.2.3 Tiramisu

A fairly new approach used DenseNet(section 3.1.2) and FCN together for better performance. DenseNet has proven it self with image recognition

and the developers wanted to see how the architecture would perform when dealing with the problem of semantic segmentation[18][25].

In the architecture of DenseNet the down-sampling is already there. To recover from the input spatial resolution, the FCNs give an upsampling path including of convolutions, upsampling operations and skip connections. In this new solution, FC-DenseNet, the convolution operations are replaced with the known dense blocks and the upsampling is referred to as transition up[22][25].

A transition up module is a set of transposed convolutions that upsamples the previous feature maps. These features maps are to be concatenated to the ones coming from the skip connection to create a new dense block. Hence the upsampling path has a higher spatial resolution. The growth in number of features is too memory consuming and is a limitation[25].

This limitation is overcome by not concatenating its output with the dense block input. Thus, the transposed convolution is applied only to the feature maps obtained by *the last dense block and not to all feature maps concatenated so far*[25]. All information contained by previous block is summarized by the last dense block.

The main semantic segmentation architecture, FC-DenseNet103, consists mainly by three blocks of operations, layer, transition down and transition up[25].

Layer
Batch Normalization
ReLU
3 x 3 Convolution
Dropout p = 0.2

Transition Down
Batch Normalization
ReLU
1 x 1 Convolution
Dropout p = 0.2
2 x 2 Max Pooling

Transition Up
3x3 Transposed Convolution *stride* = 2

Table 3.2: Building blocks of fully convolutional DenseNets. From left to right: layer used in the model, Transition Down and Transition Up[25]

This architecture are seen as very deep(103 layers) and with other approaches very slow. But hence the 10 fold reduction of parameters compared to other state-of-the-art architecture the network depth is not an issue. It is proven better performance on urban scene understanding datasets as CamVid and Gatech without further post-processing, temporal information

and pre-training[5][25].

3.2.4 PSPNet

Scene parsing is based on semantic segmentation and is a fundamental topic in computer vision. The goal of scene parsing is to provide a complete understanding of the scene by predicting the label, location and shape of the objects. The previous state-of-the-art scene parsing frameworks are in most times based on the FCN. The usage of CNNs comes with some challenges, it having a hard time to consider diverse scenes and unrestricted vocabulary. To overcome the challenges, a paper called Pyramid Scene Parsing Network (PSPNet) was released[22][26].

PSPNet are based on FCN for pixel prediction. On top of that they have extended the pixel-level feature to a designed global pyramid pooling, where the local and global values combined makes the final prediction more reliable. In addition to that they have included an optimization strategy with deeply supervised loss[26].

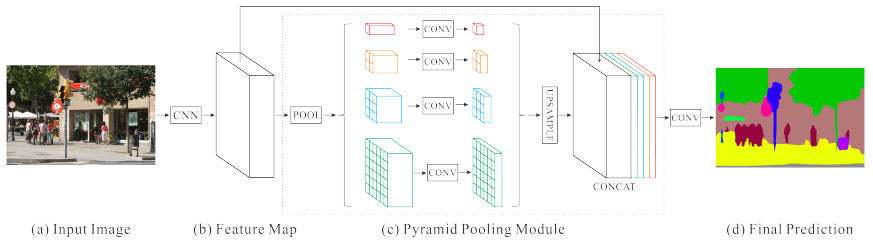


Figure 3.8: PSPNet Structure[26]

To address the challenge of reducing context information loss between different sub-regions they introduce Pyramid Pooling Module for global scene prior construction upon the final-layer-feature-map of the neural network. This module has operations under four different pyramid stages. Figure 3.8 shows the structure of the pyramid pooling module. The pooling highlighted with red is global pooling to generate a single bin output. The next pyramid level, labeled with orange, separates the feature map to different regions and forms pooled representation for different locations. The output of these four levels contains a feature map with varied sizes. The four levels in the pyramid pooling module has bin size of 1x1, 2x2, 3x3 and 6x6, from to to bottom illustrated in the figure above[26].

PSPNet is a proven and an effective pyramid scene parsing network for complex scene understanding[26].

3.2.5 DeepLab v3+

The most recent version of Google's DeepLab series was released the 7th of February 2018 and is called DeepLab v3+. The success of the previous versions and of what they call depthwise separable convolution, was their motivation to extend this feature even further. DeepLab has a novel encoder-decoder structure using Xception as feature map. Modification was done on the Xception to adapt it for segmentation tasks[27][28][29].

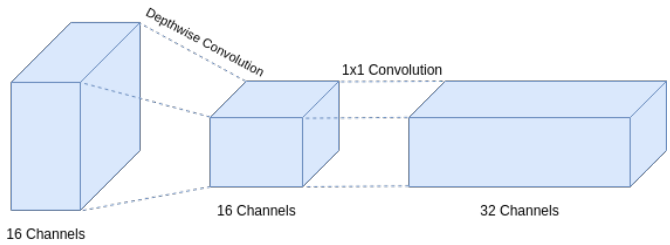


Figure 3.9: Depthwise Separable Convolution[29]

Depthwise Separable Convolutions consists of a convolution over each channel separately (as opposed to a standard convolution where every filter is used on every channel). This is followed by a 1x1 convolution that combines the output channels into a single output layer. This method reduces the number of parameters while still keeping the performance good. They are primarily used in real-time and mobile applications[28][29].

Example: Number of parameters with 16 input channels and 32 output channels
Normal convolution: $16 * 32 * 3 * 3 = 4608$
Depthwise separable: $16 * 3 * 3 + 16 * 32 * 1 * 1 = 656$

This seems to work very well if you look at the performance of Xception and DeepLab with Xception architecture[27][29].

In the proposed encoder-decoder structure, one can control the resolution of

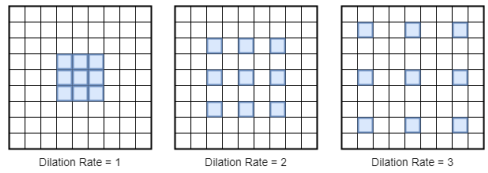


Figure 3.10: Dilation Rate Comparison

extracted encoder features by dilated convolutions. Also known as Atrous Convolutions [30]. Dilated Convolutions introduces another parameter in addition to the standard convolutional parameters, dilation rate. Dilation rate specifies a spacing between the values in a kernel (see figure 3.10). Dilated Convolutions allows the convolutional layer to get a larger field of view while keeping the computational cost the same. This is useful in real-time applications which segmentation often is a part of[29].

DeepLab is the most recent approach published that achieves state-of-the-art performance on segmentation tasks[29]. Using convolutional neural network with atrous convolution they can extract the encoder features at an arbitrary resolution depending on the available computation resources. DeepLab shows through experiments that they sat new state-of-the-art performance on PASCAL VOC 2012 benchmark dataset[4].

Algorithm	**Developed by**	**Released**
FCN	UC Berkeley	8 Mar 2015
U-Net	University of Freiburg	18 May 2015
SegNet	University of Cambridge	10 Oct 2016
Tiramisu	Multiple contributors	31 Oct 2017
PSPNet	The Chinese University of Hong Kong, SenseTime Group Limited	27 Apr 2017
DeepLab v3+	Google Research	8 Mar 2018

Table 3.3: Overview of semantic segmentation algorithms

3.3 Capsule network (CapsNet)

The idea of a capsule network has been worked on for decades by Geoffrey Hinton and in November 2017 a paper was released of the subject, *Dynamic routing between capsules*. Capsule Net(CapsNet) is a neural network based

on a system of capsules where a capsule is a group of neurons that recognize visual aspects that traditional convolutional neural networks do not. Regular CNN has its challenges, i.e using pooling layers makes you lose information and will not be able to see combinations. CNN also requires large datasets which is not efficient, due to the use of backpropagation. Hinton who first introduced CNN has tried to solve the challenges CNN contains.[31][32][33] [34].

In 2011 CapsNet was first introduced as a concept. In that paper back-propagation it was not used although it being used successfully by most of the best performing algorithms. Back-propagation is a method that adjusts weights so the algorithms can be trains with minimum errors[32].

A capsule encapsulate what is seen as important information about the state of the feature they are detecting in vector form as opposed to a scalar that a neuron outputs in traditional CNNs. It encodes the probability of detection of a feature equal to the length of the output vector, and the direction the vector points to is encoded by the state of the detected feature. This makes it possible to keep the probability unchanged while its orientation does, while the detected feature state varies[31].

Algorithm 1 Dynamic routing algorithm

procedure ROUTING($\widehat{u}_{j|i}, r, l$)
  for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
  for r iterations **do**
    for all capsule i in layer l: $c_i \leftarrow \text{softmax}(b_i)$
    for all capsule j in layer $(l+1)$: $s_j \leftarrow \sum_i c_{ij}\widehat{u}_{j|i}$
    for all capsule j in layer $(l+1)$: $v_j \leftarrow \text{squash}(s_j)$
    for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \widehat{u}_{j|i} \cdot v_j$
  end for
  return v_j
end procedure

In order to properly train a capsule network this paper proposes a dynamic routing algorithm. A capsule needs to decide how to send its output to a high-level capsule. It makes its decisions is by changing the scalar weight that will multiply its output vector, to be treated as input to the next higher-level capsule. Each weight is a non-negative scalar and for each lower-level capsule the sum of all weights equals to 1. These weights are determined

by the iterative dynamic routing algorithm[31].

Based on the pseudocode of Dynamic routing algorithm, this is the explanation line by line:

1. Takes all lower-level capsules l together with output $\widehat{u}$ and amount of routing iterations r.

2. Coefficient b_{ij} represent a temporary value that later will be updated iteratively. After procedure is complete this value will be located in c_{ij}. Initiated as zero.

3. The for-loop indicates that the following steps will be repeated r times.

4. Calculates the vector value of c_i that represent all weights for lower-level capsules i. Softmax ensures that each weight c_{ij} has a non-negative value. After all weights c_{ij} is calculated for every lower-level capsule, it moves to line five.

5. Calculate a linear combination of vector inputs, weighted and determined in the previous step. This is done for all capsules in the higher-level.

6. Vector from previous steps are passed to the squash non-linearity makes sure the vector direction is preserved and produces the output vector v_j for all capsules of higher-level.

7. Looks at each higher-level capsule j and examines the input for so to update the corresponding weight b_{ij}.

In the original paper they tested their new approach by using the MNIST dataset. All architecture are therefore tailored to that specific task[31][35].

The CapsNet architecture can be split into two parts, encoder and decoder. The encoder part of the network processes the input and learns to encode it into a vector. This is where the capsules are located and consists of three layers. One convolutional layer for detection of features, a PrimaryCaps layer for producing combinations of the features from layer one and DigitCap layer(Digit because of MNIST dataset) transform the tensor input to a matrix[31].

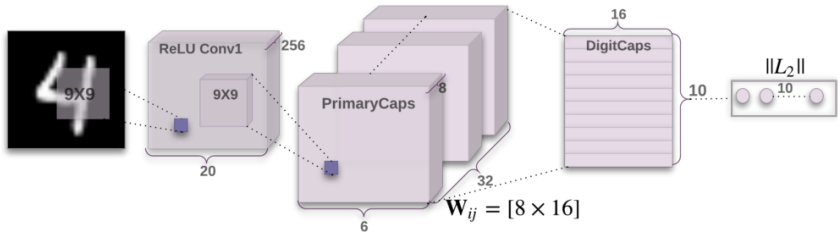


Figure 3.11: CapsNet's encoder architecture[31]

Part two of the architecture is the decoder. These last three layers are fully connected layers where all three uses the previous output as input to increases its output and parameters. In each fully connected layer the calculation is the same: number of parameters = (number of inputs + bias) x number of neuron in layer.[31]

This new approach shows great potential to replace the traditional convolutional neural nets by performing well on the MNIST dataset[35][31].

3.3.1 Capsules for Object Segmentation

There is not much work done on object segmentation using capsule networks and when we started working on this thesis no work had been published or released. On the 11th of April a paper called Capsules for Object Segmentation were released. They propose an approach, SegCaps, who is the first to use capsules instead of CNNs for object segmentation[36].

The authors extended the ideas of capsule networks. The original CapsNet architectures and dynamic routing algorithm is very computational expensive for memory and run-time. The number of parameters quickly escalates even for small inputs as MNIST and also CIFAR10 datasets[31][35][37].

SegCaps solves this by rewriting the dynamic routing algorithm in two ways.

1. Children are only routed to parents within a defined spatially-local kernel.
2. Transformation matrices are shared for each member of the grid within a capsule type but are not shared across capsule types.

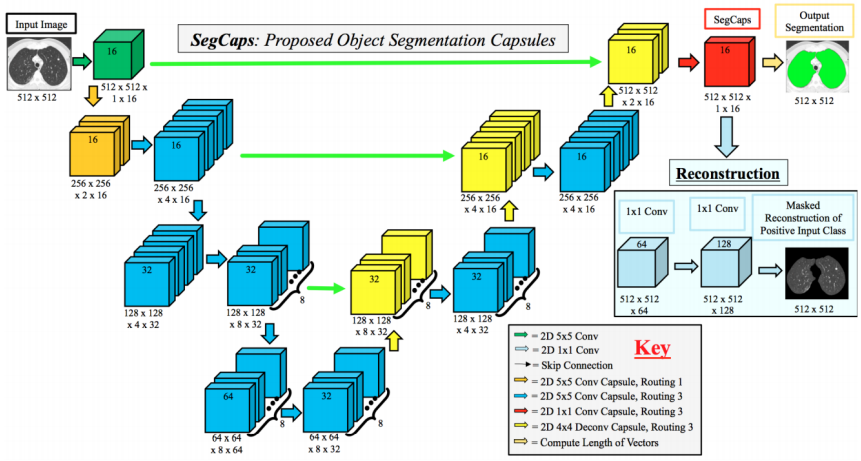


Figure 3.12: SegCaps architecture for object segmentation[36]

To make up for the lack of global connectivity due to the introduction of locally-constrained routing, they propose "deconvolutional" capsules that using transposed convolutions routed by the locally-constrained routing algorithm. The SegCaps architecture contains 95,4% fewer parameters than U-Net and 38.4% fewer than Tiramisu[36].

Algorithm 2 Locally-Constrained Dynamic Routing

procedure ROUTING($\widehat{u}_{xy|t_i^l}, d, l, k_h, k_w$)
  for all capsule types t_i^l within a $k_h \times k_w$ kernal centered at position (x, y) in layer l and capsule xy centered at position (x, y) in layer $(l+1)$: $b_{t_i^l|xy} \leftarrow 0$.
  for d iterations **do**
    for all capsule types t_i^l in layer l: $c_{t_i^l} \leftarrow \text{softmax}(b_{t_i^l})$
    for all capsule xy in layer $(l+1)$: $p_{xy} \leftarrow \sum_n r_{t_i^l|xy} \widehat{u}_{xy|t_i^l}$
    for all capsule xy in layer $(l+1)$: $v_{xy} \leftarrow \text{squash}(p_{xy})$
    for all capsule types t_i^l in layer l and capsules xy in layer $(l+1)$: $b_{t_i^l|xy} \leftarrow b_{t_i^l|xy} + \widehat{u}_{xy|t_i^l} \cdot v_{xy}$
  end for
  return v_{xy}
end procedure

Experimentally, SegCaps shows it results using LUNA16 dataset containing lung segmentation. They outperform the state-of-the-art algorithms slightly

and provides evidence that capsules can be better than CNNs on object segmentation[36].

3.4 Aerial imagery approaches

Approaches have been created for specific tasks using satellite imagery. Some of these tasks have a very specific target which would not be usable for others. These methods are developed using various technologies and all works well on their tasks.

An approach had as a goal to automatically extract road networks from aerial images. Using high-resolution images to compare their model to other pure segmentation algorithms. The observations shows that the pure segmentation methods have high error rates due to their noisy CNN output are hard to correct. They propose a solution named RoadTracer which uses an iterative search process by the help of CNN-based decision function to graph the road network directly from the output. According to the findings RoadTracer captures 45% more junctions in cities than other methods[38].

More traditional approaches have been done with datasets from unmanned aerial vehicles (UAVs) in order to detect airplanes in real-time from UAV applications. The well-used algorithm YOLO[39] was first tested on this dataset with positive results, performing a detection and classification accuracy on 84%. The authors proposed another CNN architecture with 24 convolution layers followed by two fully-connected layers at the end. This self composed network is more computationally efficient and deliver a accuracy of 97.8% at the chosen dataset[40].

Part II

Contributions

Chapter 4

Approach

The following chapter is about our apporoach and how we received the results and with what methods. The is information about the dataset such as which classes it contains, channels and distribution. Implementation details contains explaination of loss functions, which algorithms are being tested and how accuracy being calculated.

4.1 Data

Data used in this thes is gathered from the DSTL kaggle competition[2]. It features 450 images gathered from the WorldView-3 satellite[41]. Every image has 16 channels of data. 8 channels from the multispectral band (400-1040nm) with a resolution of 1.24m and 8 channels from the short-wave infrared band (1195-2365nm) with a resolution of 7.5m[1](table 4.1).

25 images from the full set of 450 are labeled into 10 classes (see Table 4.2). The remaining 425 are without annotations and are only used for testing without ground truth.

The dataset is heavily unbalanced as can be seen in figure 4.3. In addition there is almost no data on certain classes. We have therefore decided to remove both vehicle classes and combine **Flowing Water** with **Still Water**.

[1]WorldView-3 has a resolution of 3.7 m, but the data provided have reduced it to 7.5m.

Name	Wavelength	Resolution
Panchromatic	450-800nm	0.31m
Coastal	400-450nm	1.24m
Blue	450-510nm	1.24m
Green	510-580nm	1.24m
Yellow	585-625nm	1.24m
Red	630-690nm	1.24m
Red Edge	705-745nm	1.24m
Near-IR1	770-895nm	1.24m
Near-IR2	860-1040nm	1.24m
SWIR-1	1195-1225nm	7.5m
SWIR-2	1550-1590nm	7.5m
SWIR-3	1640-1680nm	7.5m
SWIR-4	1710-1750nm	7.5m
SWIR-5	2145-2185nm	7.5m
SWIR-6	2185-2225nm	7.5m
SWIR-7	2235-2285nm	7.5m
SWIR-8	2295-2365nm	7.5m

Table 4.1: Wavelengths and resolution for different bands

Buildings	Misc. Manmade structures
Road	Dirt Road / Trail
Trees	Crops
Flowing Water	Still Water
Large Vehicle	Small Vehicle

Table 4.2: Segmentation Classes

This results in a split of 22 training images, 2 validation images and 1 test image. The classes also get reduced to a total of 7.

Figure 4.1: Multispectral bands (400-1040nm)

Figure 4.2: Short-wave infrared bands (1195-2365nm)

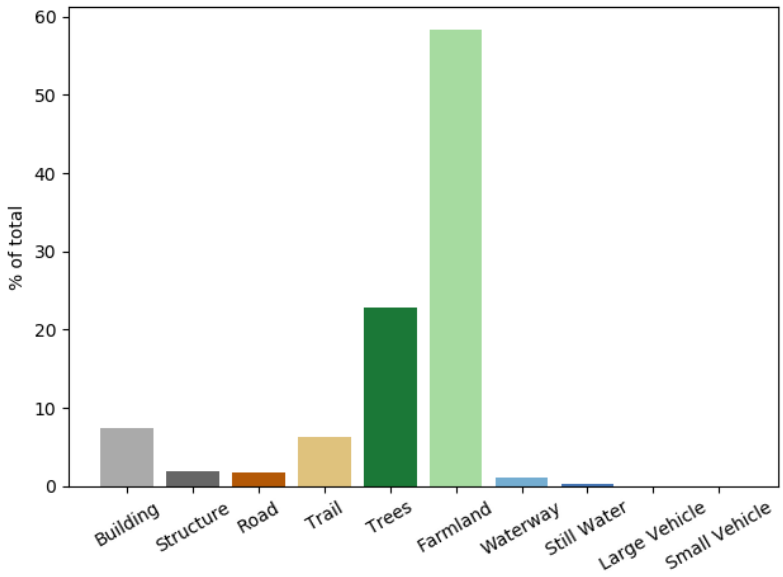


Figure 4.3: Dataset Class Distribution

4.2 Implementation Details

Adam[42] has been used as optimization function for all of the different networks and algorithms. It is an extension to the classic Stochastic Gradient Descent[43]. Adaptive learning rate reduces the need for tuning another hyperparameter while also generalizing better to sparse data like ours.

Adam is initialized with the default hyperparameters provided in the original paper (see Table 4.3) except for ϵ which uses the TensorFlow default.

α (Learning Rate)	0.001
β_1	0.9
β_2	0.999
ϵ	10^{-7}

Table 4.3: Adam Hyperparameters

Sigmoid (equation 4.1) has been used as the last activation function. Using the sigmoid activation function we generate a separate prediction map for each class. These are then combined in order to generate a full prediction.

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \tag{4.1}$$

Input size is set to 320x320x*channels* for all models unless otherwise is noted.

4.2.1 Algorithms

In this paper we are using all of the current state-of-the-art semantic segmentation algorithms to test and evaluate their performance on satellite imagery. Looking at the leaderboard from the DSTL Kaggle competition, we will find that most of the solution are U-Net based. Therefore it is important to include the approach in the experiments.

U-Net is mostly built up like the original paper with 4 downsampling and upsampling layers. We have however added zero padding in order to get

output predictions which are the same size as the input data. The Up Convolution (Deconvolution) has been replaced by a Upsamling and Convolution in order to avoid potential artifacts[44].

Regarding the Tiramisu, PSPNet, FCN and DeepLab, we used premade repositories and tweaked it to our advantage. A dynamic data loader was created that works with all the algorithms. All of the repositories delivers a code based on Keras.

4.3 Evaluation Metrics

4.3.1 Cross-entropy Loss

Cross-entropy loss is used in cases where a model outputs probabilities between 0 and 1. The loss increases when the prediction diverges from the ground truth.

$$-(y\log(p) + (1-y)\log(1-p)) \tag{4.2}$$

$$-\sum_{c=1}^{M} y_{o,c}\log(p_{o,c}) \tag{4.3}$$

We train our model to predict a separate prediction map for each class and then combining it into one prediction. With this method we can use binary cross entropy loss.

4.3.2 Accuracy

Accuracy is simply the number of correctly classified pixels out of every single pixel. This is not a good measurement for segmentation tasks. If it predicts everything as farmland it will get a excellent accuracy, but the result is not usable.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}^{2} \tag{4.4}$$

4.3.3 Jaccard Index

Jaccard Index, also known as Intersection over Union (IoU), was developed by Paul Jaccard in 1901[45]. It gives more weighting to correct classification and calculates an individual score for each class. Jaccard Index is useful both as a metric and a loss function because it gives equal weighting to every class. This mitigates some of the problems with unbalanced data.

$$\text{Jaccard Index} = \frac{TP}{TP + FP + FN} \tag{4.5}$$

The mean IoU is simply the mean of all individual class IoUs:
Mean IoU = $\frac{1}{M} \sum_{c=1}^{M} IoU(c)$

[2]TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative

Part III

Experiments and Results

Chapter 5

Experiments

This chapter contains results and thoughts of the experiments done in the thesis. Under each experiment there is an explanation of why the specific experiment is conducted, the results is represented with a table or graph and some reflection of the result. Some of these experiments are directly linked to the research questions.

5.1 Networks

To find the most accurate semantic segmentation algorithm for satellite imagery we had to do extensive research of the state-of-the-art algorithms in this field. Each of these was implemented and tested to get an overview over which one has best performance.

It has to be mentioned that due to the amount tests and with different algorithms we did not optimize the hyper parameters on each approach.

The results in Table 5.1 shows a clear sign of which approach has the best performance. The U-Net algorithm delivers a performance of 26.77 mIoU which is much better than the others and has also balanced performance on the different classes. The newer algorithms such as Tiramisu, PSPNet and DeepLab shows that the old algorithm U-Net perform a higher accuracy even if its older.

Network	Buildings	Structures	Road	Trail	Trees	Crops	Water	mIoU
U-Net	59.35	0.32	44.77	2.45	40.53	39.93	0.02	26.77
Tiramisu	49.42	0.00	0.02	0.00	25.85	0.00	0.00	10.74
FCN	51.84	0.29	0.00	0.01	28.23	4.18	0.00	12.45
PSPNet	15.00	0.00	27.20	0.00	0.33	70.40	0.00	16.55
Deeplab v3+	38.12	0.00	0.00	0.00	11.94	0.07	0.00	7.16

Table 5.1: Performance (Insersection over Union) with different network architectures

Most of the algorithms manage to classify buildings except PSPNet, which on the other hand has very good performance on crops.

Based on these results the following experiments are only tested on the best performing algorithm, U-Net.

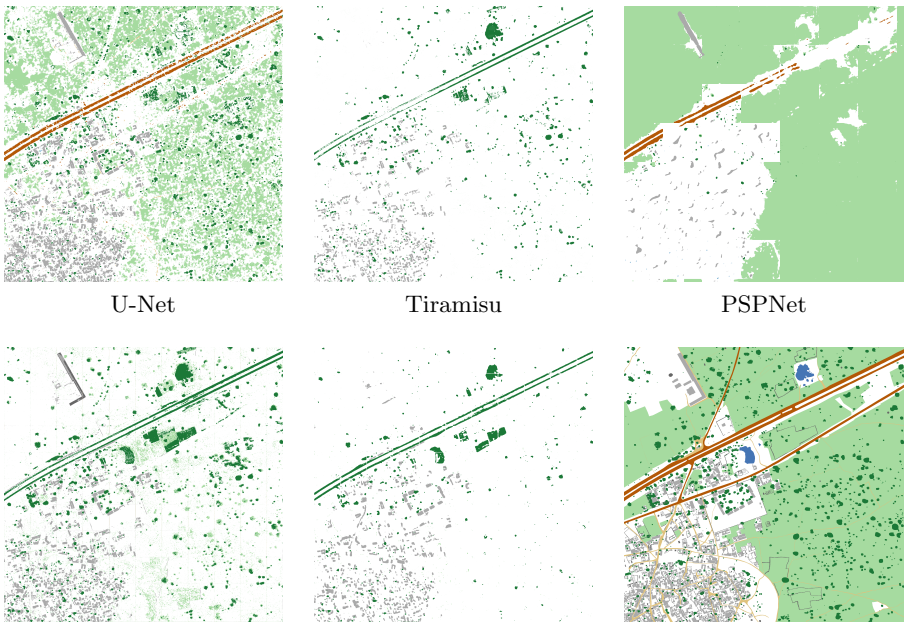


Figure 5.1: Predictions on different network architectures

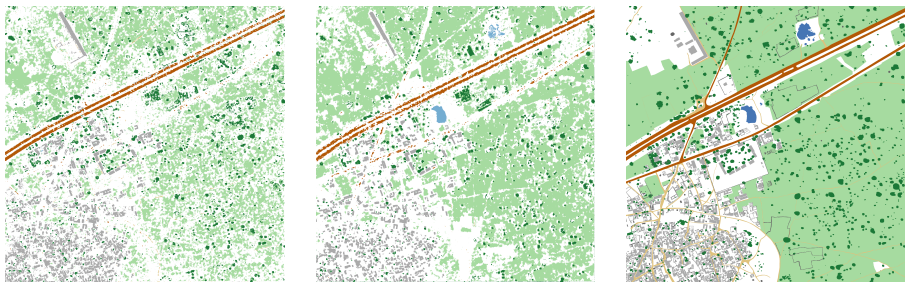
5.2 Spectral Bands

Satellites are capable of capturing light in other spectrums than visible light. Different elements like water and metal have different reflective properties and will probably be easier to classify using more channels. Experiments have therefore been run on 3-channel data (RGB) and 8-channel data.

Network	Buildings	Structures	Road	Trail	Trees	Crops	Water	mIoU
RGB	59.35	0.32	44.77	2.45	40.53	39.93	0.02	26.77
8-Band	59.08	0.00	49.15	0.51	37.60	59.30	63.27	38.42

Table 5.2: Performance with different bands of data

The results in Table 5.2 shows that using 8 spectral bands instead of 3 improves performance in certain aspects. RGB-bands have trouble finding water because in these images it has almost the same color and shape as trees. 8-band on the other hand use information from the bands we cannot see to find water with a greatly improved performance as can be seen in Figure 5.2. Training time using only 3 bands was 9 hours on a single K80 core compared to 13 hours for 8 bands of data.



U-Net U-Net 8 Channels Ground Truth

Figure 5.2: Predictions on different spectral bands

5.3 Data Amount

Machine learning algorithms are dependent on having lots of data to generalize from. Some algorithms are better at coping with this than others.

This experiment will test the performance degradation of U-Net when it has to work without data augmentation.

Network	Buildings	Structures	Road	Trail	Trees	Crops	Water	mIoU
With Augmentation	59.35	0.32	44.77	2.45	40.53	39.93	0.02	26.77
No Augmentation	52.79	1.00	50.93	0.00	33.23	15.95	33.04	26.71

Table 5.3: Performance with different amount of data

There is little difference in runs with or without data augmentation as shown in Table 5.3. The discrepancies between classes is probably due to random weight initialization.

U-Net No Augmentation Ground Truth

Figure 5.3: Predictions with different amount of data

5.4 CapsNet

One of our research questions was *"Does a out-of-the-box implementation of CapsNet work for semantic segmentation?"*. A basic CapsNet model was implemented in Keras (Figure 5.4. This model was tested on MNIST with great results.

The model was then slightly modified to fit CIFAR-10 data (Figure 5.5. Input size was changed from 28x28x1 to 32x32x3 and the number of parameters was increased in order to process the increased number of input data.

The model was then modified to remove the classification part and only contain the decoder / segmentation part (Figure 5.7). The model got stuck

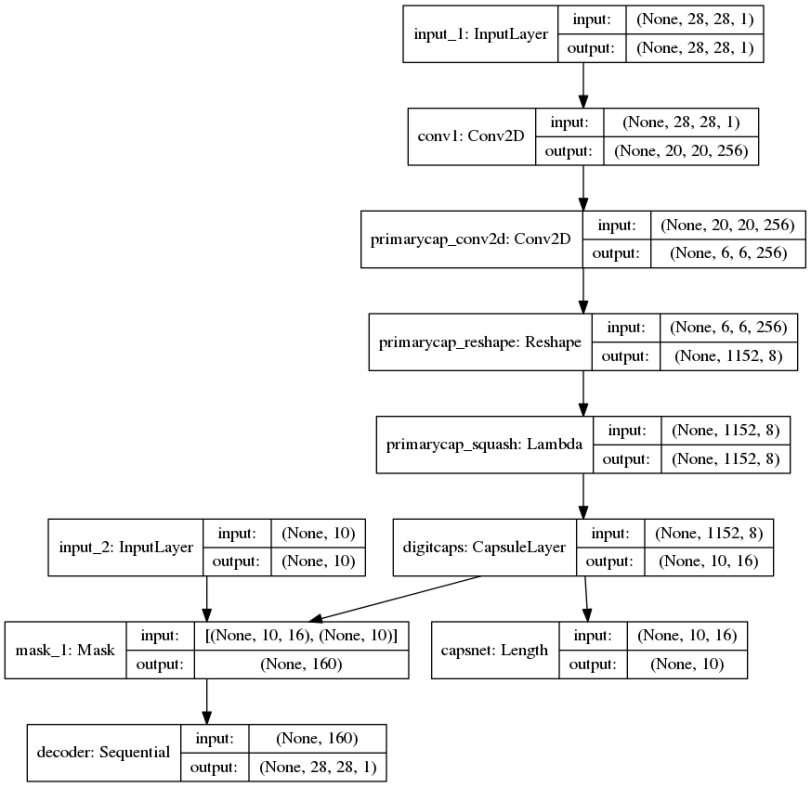


Figure 5.4: CapsNet Model

in a place where it predicts the same classes for every input image as can be seen in Figure 5.8.

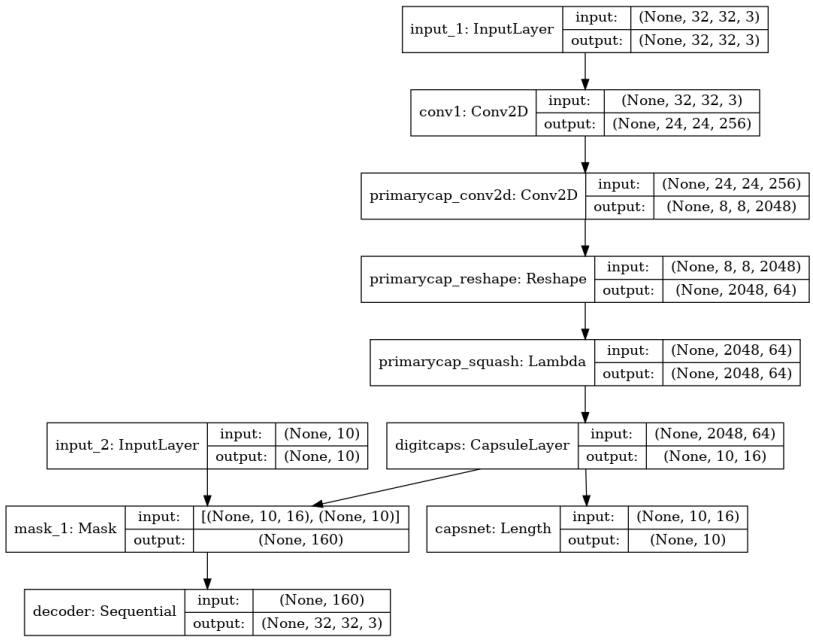


Figure 5.5: CapsNet CIFAR Model

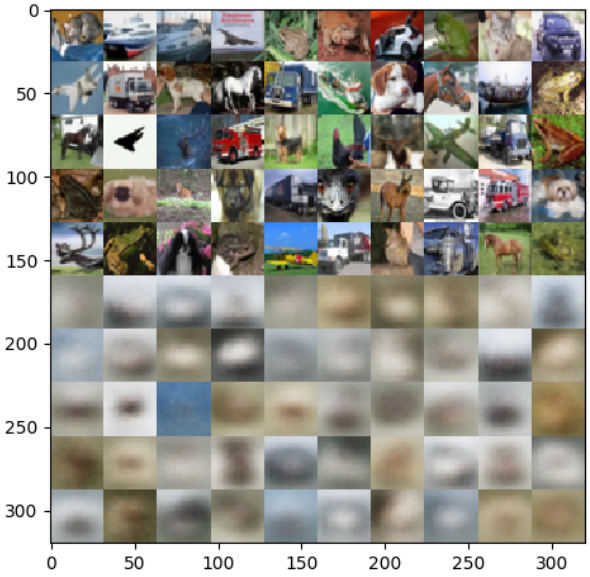


Figure 5.6: CapsNet CIFAR Model Predictions

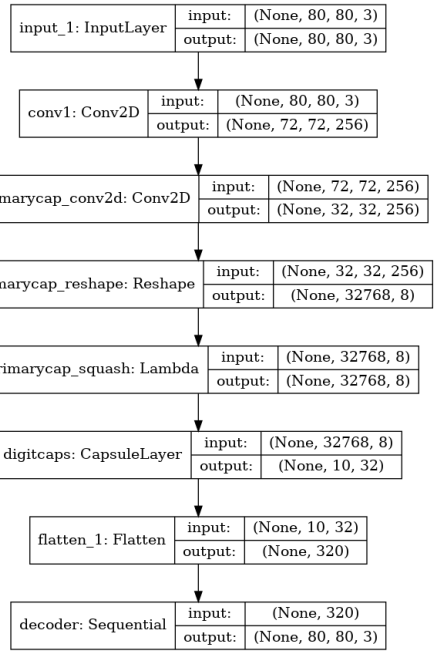


Figure 5.7: CapsNet Segmentation Model

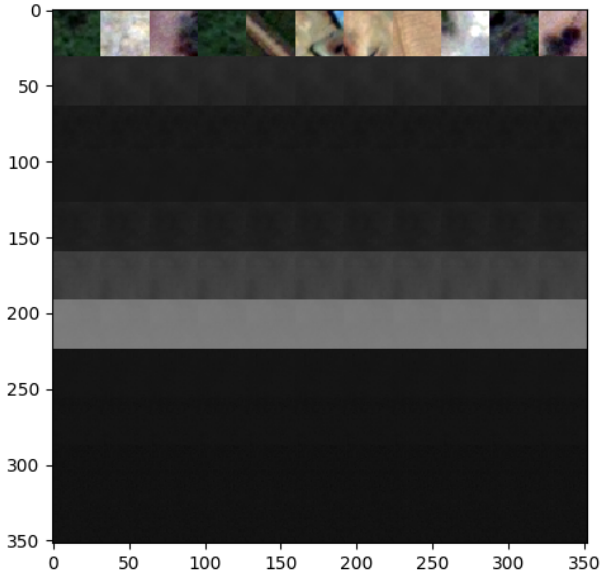


Figure 5.8: CapsNet Segmentation Model Predictions

Chapter 6

Conclusion and Future Work

In the conclusion there will be mentions of results and research questions. What the findings are and how this is relevant. The last sub-chapter include the future work and what could be done in the future for better performance.

6.1 Conclusion

We present an overview of semantic segmentation algorithms and their results tested on satellite images. Further we explored the effects of increasing the number of spectrum bands 3(RGB) to 8 bands. The leaderboard in the Kaggle competition that delivered the dataset we used, all of top ten used U-Net as their algorithm. Experiments shows that U-Net is still the overall most accurate algorithm dispite being 1-2 years old than the other state-of-the-art

6.2 Future Work

The following sections proposes a number of ways in which our current implementations may be improved.

6.2.1 U-Net with Capsnet

Segmentation using the standard convolution methods shows very promising results. They do however suffer from no rotation variance among other shortcomings. This means the network will have to be trained for many different rotations in order to generalize better. A basic attempt to use U-Net with CapsNet[36] has been implemented by someone else and it shows very promising results for further research.

6.2.2 Dataset

As discussed in Chapter 4 the used dataset delivered by DSTL contains multi-band satellite images. They are all square with a size of 1000m x 1000m. Every images is taken in Uganda and has a very distinct look based on farmland, small roads and brick houses. This means that the model will be better suited to predict images from this specific region. Training the algorithms using a dataset with different climates would give positive results. Our dataset is also very limited in size and increasing the data amount will help significantly.

Some of the tested methods are proven to be good in urban environments. Testing with an urban dataset and a different perspective than satellite images would be something to look at. A dataset such as CamVid are already tested on some of the algorithms and a comparison could most likely give a different result than in this thesis with top-down perspective.

6.2.3 Hyperparameters

Many of the network architectures tested have a lot of different hyperparameters which can be tuned. All of our experiments are run with the default parameters and architectures specified in the respective papers. Performance will probably be improved if these are tuned for the specific dataset at hand.

References

[1] *You Only Look Twice — Multi-Scale Object Detection in Satellite Imagery With Convolutional Neural Network*. Accessed: 8. Feb 2018. URL: https://medium.com/the-downlinq/you-only-look-twice-multi-scale-object-detection-in-satellite-imagery-with-convolutional-neural-38dad1cf7571.

[2] *Dstl Satellite Imagery Feature Detection*. Accessed: 15. Dec 2017. URL: https://www.kaggle.com/c/dstl-satellite-imagery-feature-detection.

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *CoRR* abs/1604.01685 (2016). arXiv: 1604.01685. URL: http://arxiv.org/abs/1604.01685.

[4] M. Everingham, L. Van Gool, C. K. I. Williams, et al. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. URL: http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[5] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. "Semantic Object Classes in Video: A High-Definition Ground Truth Database". In: *Pattern Recognition Letters* xx.x (2008), pp. xx–xx.

[6] Seymour Papert. "The Summer Vision Project". In: (Oct. 2004).

[7] Araujo Santos, Leonardo. *Object Localization and Detection*. URL: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/object_localization_and_detection.html.

[8] Y. Lecun, L. Bottou, Y. Bengio, et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. ISSN: 0018-9219. DOI: 10.1109/5.726791.

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[11] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2014). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556.

[12] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *CoRR* abs/1602.07261 (2016). arXiv: 1602.07261. URL: http://arxiv.org/abs/1602.07261.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[14] Prakash, Jay. *Understanding and Implementing Architectures of ResNet and ResNeXt for state-of-the-art Image Classification: From Microsoft to Facebook*. URL: https://cdn-images-1.medium.com/max/1600/1*1y9hueMSZAeo1Hbp9KYKiw.png.

[15] Prakash, Jay. *Understanding and Implementing Architectures of ResNet and ResNeXt for state-of-the-art Image Classification: From Microsoft to Facebook*. URL: https://cdn-images-1.medium.com/max/1600/1*WVs9ywVLLKjSUBZ_mnfFrw.png.

[16] Shaoqing Ren, Kaiming He, Ross B. Girshick, et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[17] CVPR. *CVPR 2017 Best Paper Awards*. URL: http://cvpr2017.thecvf.com/program/main_conference#cvpr2017_awards.

[18] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *CoRR* abs/1608.06993 (2016). arXiv: 1608.06993. URL: http://arxiv.org/abs/1608.06993.

[19] C.-Y. Lee, S. Xie, P. Gallagher, et al. "Deeply-Supervised Nets". In: *ArXiv e-prints* (Sept. 2014). arXiv: 1409.5185 [stat.ML].

[20] Christian Szegedy, Wei Liu, Yangqing Jia, et al. "Going Deeper with Convolutions". In: *CoRR* abs/1409.4842 (2014). arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842.

[21] Shaoqing Ren, Kaiming He, Ross B. Girshick, et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[22] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *CoRR* abs/1411.4038 (2014). arXiv: 1411.4038. URL: http://arxiv.org/abs/1411.4038.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[24] URL: https://lmb.informatik.uni-freiburg.de/people/ronneber/isbi2015/.

[25] Simon Jégou, Michal Drozdzal, David Vázquez, et al. "The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation". In: *CoRR* abs/1611.09326 (2016). arXiv: 1611.09326. URL: http://arxiv.org/abs/1611.09326.

[26] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, et al. "Pyramid Scene Parsing Network". In: *CoRR* abs/1612.01105 (2016). arXiv: 1612.01105. URL: http://arxiv.org/abs/1612.01105.

[27] François Chollet. "Xception: Deep Learning with Depthwise Separable Convolutions". In: *CoRR* abs/1610.02357 (2016). arXiv: 1610.02357. URL: http://arxiv.org/abs/1610.02357.

[28] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, et al. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *CoRR* abs/1606.00915 (2016). arXiv: 1606.00915. URL: http://arxiv.org/abs/1606.00915.

[29] Liang-Chieh Chen, Yukun Zhu, George Papandreou, et al. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: *CoRR* abs/1802.02611 (2018). arXiv: 1802.02611. URL: http://arxiv.org/abs/1802.02611.

[30] Fisher Yu and Vladlen Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions". In: *CoRR* abs/1511.07122 (2015). arXiv: 1511.07122. URL: http://arxiv.org/abs/1511.07122.

[31] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic Routing Between Capsules". In: *CoRR* abs/1710.09829 (2017). arXiv: 1710.09829. URL: http://arxiv.org/abs/1710.09829.

[32] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. "Transforming Auto-Encoders". In: *Artificial Neural Networks and Machine Learning – ICANN 2011*. Ed. by Timo Honkela, Włodzisław Duch, Mark Girolami, et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 44–51. ISBN: 978-3-642-21735-7.

[33] Geoffrey Hinton, Sara Sabour, and Nicholas Frosst. "Matrix capsules with EM routing". In: 2018.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[35] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[36] Rodney LaLonde and Ulas Bagci. "Capsules for Object Segmentation". In: (2018). URL: https://arxiv.org/abs/1804.04241.

[37] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "CIFAR-10 (Canadian Institute for Advanced Research)". In: (). URL: http://www.cs.toronto.edu/~kriz/cifar.html.

[38] Favyen Bastani, Songtao He, Sofiane Abbar, et al. "Unthule: An Incremental Graph Construction Process for Robust Road Map Extraction from Aerial Images". In: *CoRR* abs/1802.03680 (2018). arXiv: 1802.03680. URL: http://arxiv.org/abs/1802.03680.

[39] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[40] Matija Radovic, Offei Adarkwa, and Qiaosong Wang. "Object Recognition in Aerial Images Using Convolutional Neural Networks". In: *Journal of Imaging* 3.2 (2017). DOI: 10.3390/jimaging3020021. URL: http://www.mdpi.com/2313-433X/3/2/21.

[41] DigitalGlobe. *WorldView-3 Data Sheet*. URL: http://satimagingcorp.s3.amazonaws.com/site/pdf/WorldView3-DS-WV3-Web.pdf.

[42] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[43] J. Kiefer and J. Wolfowitz. "Stochastic Estimation of the Maximum of a Regression Function". In: *Ann. Math. Statist.* 23.3 (Sept. 1952), pp. 462–466. DOI: 10.1214/aoms/1177729392. URL: https://doi.org/10.1214/aoms/1177729392.

[44] Augustus Odena, Vincent Dumoulin, and Chris Olah. "Deconvolution and Checkerboard Artifacts". In: *Distill* (2016). DOI: 10.23915/distill.00003. URL: http://distill.pub/2016/deconv-checkerboard.

[45] Paul Jaccard. "Etude de la distribution florale dans une portion des Alpes et du Jura". In: 37 (Jan. 1901), pp. 547–579.

UiA University of Agder
Master's thesis
Faculty of Engineering and Science
Department of ICT