Deep Learning Techniques Applied to the Remote Sensing of Soil Moisture and Sea Ice Type

by

Ryan Kruk

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfillment of the requirements of the degree of

Master of Science

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg, Manitoba, Canada

Abstract

This thesis presents deep learning techniques applied to the modeling of geophysical quantities with remote sensing data to estimate soil moisture and classify sea ice type. The objective of the thesis is to determine if it is possible to estimate soil moisture and classify sea ice type from satellite data using deep learning. Soil moisture is an important metric used for agriculture, hydrology, and meteorology, while accurate maps of sea ice type are needed to address increased interest in commercial marine transportation through the Arctic. Satellites equipped with Synthetic Aperture Radar (SAR) sensors operating in the microwave band provide the remote sensing data used to predict these two geophysical quantities since SAR is capable of measuring the Earth's surface in all weather conditions and in darkness. Herein, the primary deep learning technique used to model soil moisture and sea ice types from SAR data is the Convolutional Neural Network (CNN). The best performing experiment for estimating soil moisture was achieved by U-Net on a dataset of 1,034 images to achieve a testing mean square error (MSE) of 3.82e-03 and correlation of 79.2%. For classifying sea ice type, DenseNet achieved the highest overall classification accuracy of 94.0% including water and the highest ice classification accuracy of 91.8% on a three class dataset using a fusion of HH and HV SAR polarizations for the input data. These results are contextualized and are shown to be comparable to other similar studies thereby supporting that the objective of the thesis has been achieved.

Contributions

Provided below is a list of published works that have resulted from research conducted during my M.Sc. program.

Journal Papers

1. R. Kruk, M. C. Fuller, A. S. Komarov, D. Isleifson, and I. Jeffrey, "Proof of concept for sea ice stage of development classification using deep learning," *Remote Sensing*, vol. 12, no. 15, p. 2486, 2020
2. K. Edwards, N. Geddert, K. Krakalovich, R. Kruk, M. Asefi, J. Lovetri, C. Gilmore, and I. Jeffrey, "Stored grain inventory management using neural-network-based parametric electromagnetic inversion," *IEEE Access*, vol. 8, pp. 207 182–207 192, 2020. DOI: 10.1109/ACCESS.2020.3038312

Conference Papers

1. K. Edwards, K. Krakalovich, R. Kruk, V. Khoshdel, J. LoVetri, C. Gilmore, and I. Jeffrey, "The implementation of neural networks for phaseless parametric inversion," in *2020 XXXIIIrd General Assembly and Scientific Symposium of the International Union of Radio Science*, IEEE, 2020, pp. 1–3

Acknowledgements

I would first like to thank my advisors, Dr. Ian Jeffrey and Dr. Dustin Isleifson, for providing me the opportunity to pursue my Master's degree in my field of interest of deep learning and supplying me with interesting research problems for which to apply the knowledge base I have developed in deep learning over the past two years. I would also like to thank them for their time and efforts keeping me on schedule, helping me develop the methodologies used in this work, and supporting me throughout the publication process. Without their help, I would not have been able to accomplish as much as I did within this time frame.

I thank the team at Sightline Innovation for providing me with the technical and infrastructure support needed to program and execute experiments in this work. I learned many valuable skills from this team and I appreciate their time and patience in teaching me those skills. I also appreciate many of the interesting conversations that were had and relationships that were built during company lunches and outings.

I also thank Dr. M. Christopher Fuller and Dr. Alexander Komarov for their time and expertise in remote sensing of Arctic sea ice. They provided the means with which to conduct sea ice research and their support was crucial in developing the methods used in my first published paper, which has been included here as a major component of this thesis.

I would like to thank my M.Sc. reviewing committee, Dr. Ahmed Ashraf and Dr. Christopher Henry, for reviewing and evaluating my thesis as well as providing their feedback.

Finally, I would like to express my gratitude for the financial support provided during my Master's by Sightline Innovation, Mitacs, and 151 Research as well as the computing infrastructure enabling the research conducted herein provided by the Canadian Foundation for Innovation and Research Manitoba.

Dedication

I would like to dedicate this work to my friends and family. To my friends, you have been a source of relief, encouragement, fun and laughter through the difficult times that accompanied graduate studies and for that, I thank you. To my parents, thank you for the love and support you have given me and I am forever grateful for the countless actions and sacrifices you have made to support me throughout my life and education.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis applies deep learning to two remote sensing applications. The first consists of using satellite imagery to predict soil moisture content, and the second to classify sea ice by type in the Arctic. Soil moisture content (SMC) is measured as a volumetric or gravimetric (mass) ratio of water contained in soil. Many sea ice types exist, and the definitions are both qualitative and quantitative [1], but the sea ice types used herein are based on the stage of development feature as defined in the ice manual (MANICE) [2] produced by the Canadian Ice Service (CIS). Both applications use remote sensing techniques to predict geophysical quantities that are impacted by the climate. Remote sensing is the science of acquiring information about an object or area at a distance without direct physical measurement [3]. Remote sensing techniques typically use satellite or aircraft to measure reflected and emitted radiation from Earth's surface [3]. It is often used to measure physical characteristics of difficult-to-access areas across large expanses. Remote sensing is used in many fields such as hydrology, meteorology, oceanography, glaciology, and geology as well as humanitarian fields such as military, intelligence, commercial, economic, and planning [3]. Soil moisture estimates are used for agricultural purposes and can also serve as an important parameter in hydrological and meteorological models. Classifying sea ice by type on the other hand is important for naval transportation planning as well as climate monitoring and modeling. Developing models that can predict these two geophysical quantities using remote sensing is important given these application areas.

Satellites are the primary remote sensing technique used to obtain data to train the

neural network models used for both applications in this work. The sensors aboard satellites are typically categorized as one of two types. Passive sensors measure naturally occurring electromagnetic (EM) radiation being emitted by Earth's surface as well as reflected sunlight [4], [5]. Radiometers and spectrometers are two types of passive sensors that measure the intensity and spectral content of electromagnetic radiation respectively [4], [6]. Active sensors on the other hand use an on-board energy source to emit radiation towards Earth's surface and measure the reflected backscatter [4], [5]. Radar is an active sensor type that measures the distance of surfaces as well as the magnitude of the reflected energy [4], [6]. A scatterometer is another active sensor type that measures the backscattered radiation caused by diffusion in a medium or scattering from a surface [6], [7]. These remote sensing sensors typically operate in the radar, microwave, or visible light frequency bands of the EM spectrum [5].

Of the types of sensors presented, Synthetic Aperture Radar (SAR) sensors operating in the microwave band are typically used for soil moisture estimation and sea ice detection. Active sensors such as SAR benefit from measurements that can be taken of Earth's surface in all weather conditions as well as in darkness [7]. These types of sensors are called "synthetic" because the radar antenna's effective length is synthetically enlarged via the satellite's motion in order to capture larger areas of the reflected energy at higher resolutions [3]. A diagram of the SAR satellite setup is shown in Figure 1.1. SAR scatterometers are used to measure the backscatter energy in terms of σ^0, the Normalized Radar Cross Section (NRCS). The NRCS is the average radar cross section normalized per unit area. The radar cross section is an equivalent representation of the targeted object as a perfectly reflecting sphere that would produce the same reflection strength [8]. Arriving to σ^0 is accomplished by transmitting a pulse of microwave energy and measuring the reflected backscatter signal

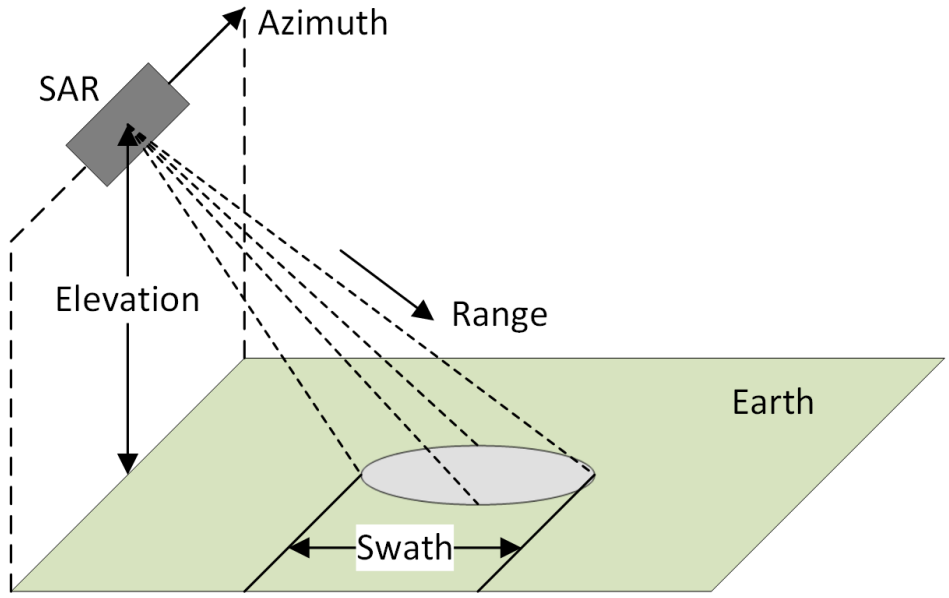


Figure 1.1: SAR Satellite Diagram

power. With the ratio of the transmitted and received signal power, σ^0 is calculated using the distributed target radar equation and normalizing the result per unit area [9]. Certain satellite systems produce these images of σ^0 for various polarizations as well, such as σ^0_{HH}, σ^0_{HV}, and σ^0_{VV} among others, where the pair of subscripts denote the polarization configuration. For example, the first letter of "HV" denotes that a horizontally polarized signal was transmitted and the second letter denotes that the vertical polarization of the received signal is measured [10]. Different polarizations are sensitive to different physical characteristics and can therefore be used to distinguish various quantities such as soil moisture or sea ice type.

1.1 Motivation – Deep Learning

The initial personal motivation for undertaking this thesis was to learn about deep learning and develop the skills to implement various neural networks and experiment in different ap-

plications. Deep learning is a field that greatly interests me because of all the advancements that are achieving state-of-the-art results in many different application areas. It is amazing that an algorithm is able to detect and correctly identify objects in a photo with better precision than humans [11], [12]. It is also impressive that an algorithm is able to not only do this for images, but to also detect actions in videos. For example, these capabilities are key components in the development of the auto pilot system used by the electric vehicle manufacturer Tesla and a short video of this can be seen on their website [13]. Another application area where remarkable achievements have been obtained by an algorithm is in Natural Language Processing (NLP) where for example the newly released GPT-3 [14] is able to generate meaningful language when prompted and carry out human-like conversations. This model not only needs to generate its own words and sentences, but also needs to accurately recognize human speech and understand all the semantics of human language for it to be able to respond. These capabilities are not only limited to ordinary conversations as the algorithm can also generate computer code, such as HTML, to create common website user interfaces. Examples of these can be found at the following blog post [15]. Deep learning algorithms have also been applied to games and it is stunning how well an algorithm can learn to play a game by learning through a reward and punishment system and come up with solutions not even considered by the researchers that created it [16]. As of now, the capabilities of deep learning have exemplified common human traits, such as visual recognition, language, and play, but deep learning has also been applied to scientific endeavors. I am fascinated that algorithms like Deep Mind's AlphaFold is able to predict the 3D structure of complex proteins with better accuracy than previous methods [17]. It is equally incredible to me that an algorithm is able to discover physical laws based on the change in position of stars in a dataset of images [18]. All these algorithms are based in deep

learning theory and consist of neural networks. It is inspiring that neural networks were formulated to model neurons in the brain, just like how planes were inspired by birds, how wind turbine blades are designed based on whale fins, or how high speed trains are designed based on Kingfishers beaks. Given all the advances that neural networks have made in all these different application areas, I am hopeful that there is much more room to grow in the field as well as new discoveries to make.

1.2 Thesis Goals

The primary objective of this work aims to determine if it is possible to predict geophysical quantities, such as soil moisture and sea ice type, from remote sensing data using deep learning techniques and previously labelled data from expert sources to make useful predictions on unseen data. Beyond this applied goal, my personal goals are to develop an understanding of deep learning theory, implement deep learning techniques and apply them towards practical applications in satellite remote sensing. While the two applications considered in this work are different, it is notable that the inputs used to arrive at two different outputs are the same. Both applications use SAR data as the input so it is interesting that neural networks can use the same input to arrive at two different outputs, the only difference being the labels it is given. Neural networks therefore serve as a versatile mapping tool between inputs and outputs. The two projects serve as good introductions to deep learning modeling since soil moisture estimation is a regression problem, whereas sea ice classification by type is a classification problem. Regression and classification are the two main types of tasks that supervised machine learning can perform thereby providing breadth to the scope of this work. There are many different types of neural networks to choose from. There are convolutional neural networks (CNNs), recurrent neural networks (RNNs), generative

adversarial networks (GANs), long short term memory networks (LSTMs), gated recurrent unit networks (GRUs), echo state networks (ESNs), and variational auto encoders (VAEs) to name a few. Among these, the main type of network that will be explored in this work is the convolutional neural network (CNN). CNNs are applicable to these two applications since physical characteristics are spatially dependent on features in the surrounding areas and CNNs are adept at extracting spatially dependent features.

1.3 Overview

Chapter 1 An introduction of the goals and application domains of the thesis are presented.

Chapter 2 A background of the theoretical concepts and components of neural networks is presented, along with background information for the two application domains considered in this work: soil moisture estimation and sea ice classification.

Chapter 3 The methodology used to apply neural networks to soil moisture estimation from satellite imagery is presented. The results achieved by several experiments are presented and are then followed by a discussion and concluding remarks.

Chapter 4 Similar to the chapter on the soil moisture application, this chapter presents the methodology used to classify sea ice by type from satellite imagery, results are shown and discussed, and concluding remarks are given.

Chapter 5 The conclusion assess the goals of the thesis and summarizes the findings of applying neural networks to soil moisture estimation and sea ice classification.

Chapter 2

Background

2.1 A Brief History of Deep Learning

It is important to have an awareness of the historical progression of deep learning in order to gain an appreciation of its capabilities that we enjoy today. Deep learning is a subfield of machine learning and is based on neural networks, networks that were inspired to model the brain in biological systems. The field of deep learning acquires its name based on "deep" neural networks that contain many hidden layers. The earliest implementation of a multi-layered neural network can be traced back to 1959 at Stanford University where Bernard Widrow and Marcian Hoff developed models they called ADALINE and MADALINE, an acronym of their use of Multiple ADAptive LINear Elements. Neural network research stagnated after Marvin Minsky and Seymour Papert published their research [19] in 1969 which demonstrated limitations of neural networks concerning the computation of some predicates, such as the XOR function, and the connectedness predicate. After some time, the next notable development in deep learning research was the backpropagation algorithm proposed by Werbos in 1974 [20], which went relatively unnoticed until a later publication was written by Rumelhart et. al. in 1986 [21]. The first convolutional neural network (CNN) was introduced by Kunihiko Fukushima in 1980 [22], which aimed to detect patterns within an input by producing the same output response regardless of translations of the pattern in the input. Another key contribution to the development towards modern CNNs was made by Yann LeCun in 1989 [23] where he applied a backpropagation learning

strategy to Fukushima's CNN, which used a reinforcement learning strategy instead. A second period of stagnation in neural network research took place during the 90s where only a few advancements were made in the late 90s and early 2000s. One of the major roadblocks for neural networks at this time was due to computer resource limitations in terms of both speed and capacity. The performance of neural networks improve significantly with larger datasets, but computer memory and runtime limitations made training these networks infeasible at the time. It wasn't until recently in 2012, after computer technology had made significant advancements, that convolutional neural networks regained popularity after the winning results achieved by AlexNet [24] in the ImageNet competition showed significant improvement in error rates. Since then, many different convolution-based neural network architectures have been designed for numerous applications. These applications can consist of computer vision and object recognition applications [25], natural language processing [26], anomaly detection [27], and time series forecasting [28] to name a few.

2.2 Deep Learning Theory

2.2.1 Neural Network Representations

As the name implies, neural networks consist of a network of neurons and can be represented in three different ways. The most publicly-presented illustrious form is the graphical representation shown in Figure 2.1. In this form, "neurons" are portrayed as nodes in a graph, and learnable weights w_{ij} and biases b_j are represented as edges, where subscripts i and j denote from and to respectively. The superscripts in Figure 2.1 denote the layer l. The neurons of the hidden layer are denoted by the letter H and symbolize an arbitrary function referred to as an activation function in the deep learning nomenclature. These neurons apply a pre-

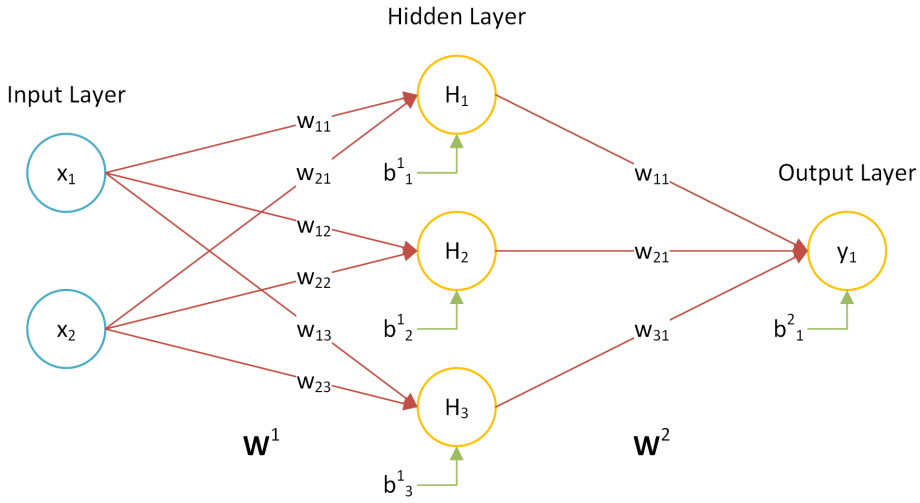
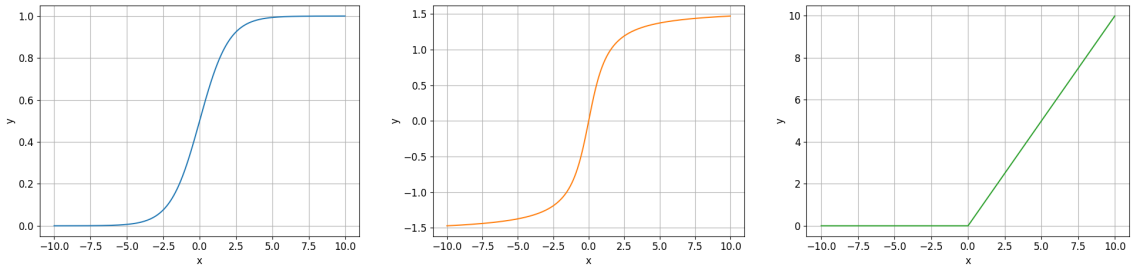


Figure 2.1: Neural Network Diagram.

determined activation function H to given inputs that are modulated by a learned set of weights $\mathbf{W}$. Examples of activation functions include Sigmoid, Tanh, or the Rectified Linear unit (ReLu) as shown in Figure 2.2. The graphical representation of a neural network shown



(a) Sigmoid (b) Tanh (c) ReLu

Figure 2.2: Three activation functions.

in Figure 2.1 aids in the understanding of what neural networks are, but this network can be expressed in a more compact form with matrices for computational purposes as shown in Figure 2.3. This form lends itself nicely to fast computations as matrix operations are easily parallelizable and can be executed on a GPU. According to Figures 2.1 and 2.3, the matrices

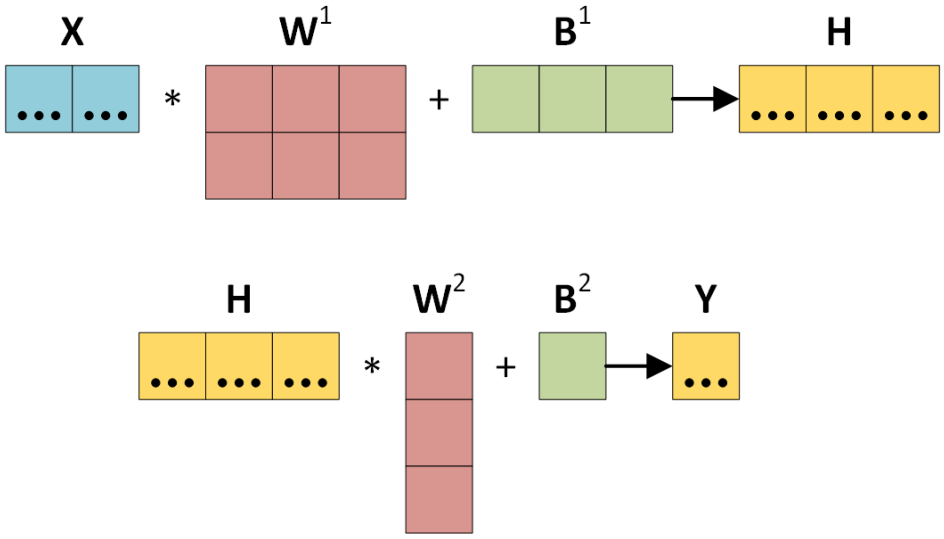


Figure 2.3: Neural Network in Matrix Form.

representing a fully connected layer can be written in compact mathematical notation given by Equation 2.1, where T denotes the transpose operator.

$$y = H(\mathbf{W}^T\mathbf{x} + \mathbf{b}). \tag{2.1}$$

2.2.2 Neural Network Layers – Fully Connected Layer

The network shown in Figure 2.1 depicts a hidden layer (column of neurons denoted by H) known in deep learning as a fully connected layer. This layer is fully connected since each hidden neuron is connected to all inputs thereby giving the layer full knowledge of the input. However, fully connected layers are not the only type of layer that compose CNNs. Convolution layers are another type of layer that can be viewed as a regularized form of fully connected layers since convolution layers are only locally connected (i.e. many connections are dropped out). Therefore, some neurons only have access to local parts of the input. For this reason, convolution layers are important in computer vision and recognition applications

such as the two applications that are the subject of this work.

2.2.3 Neural Network Layers – Convolution Layer

The foundational operation in convolutional neural networks is the convolution. Convolution is a measure of similarity of one signal on another. Given discrete signals $f(n)$ and $g(n)$, their convolution is defined as

$$y(n) = (f * g)(n) = \sum_{m=-\infty}^{\infty} f(m)g(n-m). \tag{2.2}$$

In words, the n-th entry in the convolution is calculated by flipping the filter g about the g-axis, translating the filter g by n positions left for negative n or right for positive n, and computing the weighted sum of f, where the weights are given by g. In other words, the signal $f(m)$ and the transformed filter $g(n-m)$ are multiplied element-wise and are summed to give the value of $y(n)$ at a given n. This step is simply a dot product and the process is repeated for all n. The net effect is to sweep the transformed filter $g(n-m)$ along the signal $f(m)$ to produce the signal $y(n)$. The degree to which the number of samples in $f(m)$ are combined into the sum depends on the width of the filter as well as the overlap between $f(m)$ and the transformed filter $g(n-m)$. The dot product is key to understanding that the signal $y(n)$ represents a measure of similarity between the signal $f(m)$ and the filter $g(n-m)$ at different steps n. The maximum value in $y(n)$ is where $f(m)$ and $g(n-m)$ are most similar.

Another way to understand convolutions is by observing the difference between convolution and correlation. Correlation, which is a measure of the similarity between two

functions, is mathematically defined as:

$$y(n) = (f * g)(n) = \sum_{m=-\infty}^{\infty} f(m)g(n+m). \tag{2.3}$$

The only difference between the convolution and correlation operations is that the filter is flipped for the convolution. Given that the filter weights are learnable parameters in convolutional neural networks, this difference is irrelevant and the main takeaway is that a convolution, like a correlation, is a measure of similarity.

Applying the concept of convolution to images requires extending the definition from 1D functions to 2D functions. Convolutions in 2D are a natural extension of the 1D case. For a 2D image or video frame $f(i,j)$ and a 2D filter $g(i,j)$ the convolution is defined as:

$$y(i,j) = (f * g)(i,j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f(m,n)g(i-m, j-n). \tag{2.4}$$

The filter $g(i,j)$ is two dimensional and the convolution involves flipping the filter about the m- and n-axis. Similar to the 1D case, the 2D convolution involves sweeping the flipped filter across both the m- and n-axis and doing a dot product between the area of the image that is overlapped by the filter and the weights in the filter. The 2D convolution over images can be viewed as a weighted sum of neighbouring pixels with a filter. A larger filter will encapsulate larger neighbourhoods of pixels. The results of a 2D convolution is itself an image that can represent many different things depending on the weights of the filter. For example, in pattern recognition the filter weights can be set to extract regions of an image that are most similar to the weights in the filter. This is an example of template matching. Another configuration of weights can be used for edge detection. Other filters can be used to enhance, smoothen, or denoise an image. The common aspect of all of these possibilities are that the weights are randomly initiated in CNNs and are learnable parameters that are

free to become any type of filter that will best fit the task needed at each layer.

CNNs often extend the idea of 2D convolution and use 3D filters, where the depth of the filter equals the number of channels in the input. The result of a 2D convolution on a c-channel input with a c-depth filter will still result in a single channel because the dot product is taken with the elements along the channel dimension as well. This is not to be confused with 3D convolutions where the filter would sweep across i, j, and k directions. A 2D convolutional layer in a CNN, with D filters, stacks the output of each filter along the channel dimension to create what is known as a feature map. Take for example a 100 $\times$ 100 image with 3 channels (RGB) and a 2D convolution layer with 50 filters and a user specified filter width and height of 3. Each of the 50 filters will be of size 3 $\times$ 3 $\times$ 3 and the outputted feature map of the 2D convolution layer will be 100 $\times$ 100 $\times$ 50. This assumes that padding is added around the image boundaries and a stride of 1 is used to maintain the original width and height. The stride is the number of steps taken by the filter in the i and j directions between each operation. For example, a stride of 2 would skip every other pixel. Calculating the output dimensions of the feature map subsequent to a convolution operation can be accomplished using Equations 2.5 - 2.7, where the subscript i denotes the input, subscript o denotes the output, W, H, and C are the width, height and number of channels, F is the filter size, P is the padding, S is the stride, and D is the number of filters.

$$W_o = \frac{W_i - F + 2P}{S} + 1. \tag{2.5}$$

$$H_o = \frac{H_i - F + 2P}{S} + 1. \tag{2.6}$$

$$C_o = D. \tag{2.7}$$

Another aspect to convolution layers in CNNs is the concept of receptive fields. A receptive field is a hyperparameter that indicates the part of the input space that is visible to a filter. The receptive field is impacted by the chosen size of the filter as well as the number of consecutive convolutional layers since CNNs contain multiple layers of convolutions. If one convolution layer is present, the receptive field is equivalent to the size of the filter. However, as is often the case in deep neural networks where multiple convolution layers appear one after another, the effective receptive field for filters in the last layer relative to the input space is larger than the size of the filters in that layer. This concept is illustrated in Figure 2.4. As shown in Figure 2.4, one feature in layer 3 has an effective receptive field of 5 relative

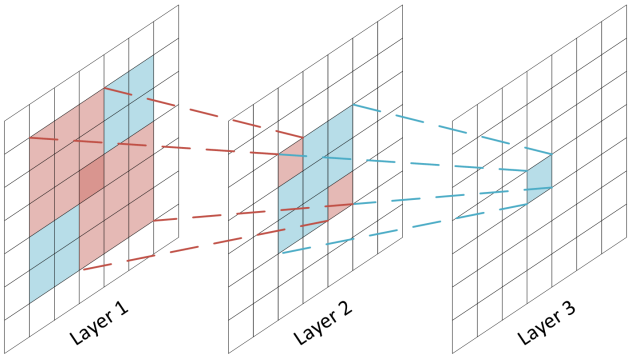


Figure 2.4: The feature in Layer 3 has a 5×5 receptive field relative to Layer 1 due to the cascading of 3 convolution layers, each having a 3×3 filter.

to layer 1. This is because that one feature in layer 3 is composed of a 3×3 convolution in layer 2 (highlighted in blue) and the extents of this section (i.e the upper left and bottom right feature highlighted in red in layer 2) is composed of the two red 3×3 boxes of layer 1 in the upper left and bottom right. Repeating this cascading process for each corner pixel highlighted in the blue section of layer 2 extends to the colored 5×5 receptive field in layer 1. Knowledge of the receptive field size of a neural network is important because a larger receptive field is able to capture larger features or objects in an image through the

contextualization of its parts. A logarithmic tendency between the classification accuracy and receptive field size of various modern networks was shown in [29], which supports that larger receptive fields give better classification accuracies, but with diminishing returns. It is noted however that the receptive field size is not the only factor that contributes to an increase in accuracy between the networks evaluated. Other factors include the network depth and width, residual connections, and batch normalization to name a few.

2.2.4 Neural Network Layers – Pooling Layer

In CNNs, pooling layers are used to reduce the dimensionality of the feature space and encode information into a smaller representation. There are different types of pooling layers such as max-pooling, or average-pooling that exist to perform different tasks, but they all work in the same way. Using a max-pooling layer as an example, the input is divided into typically non-overlapping 2D sub-regions of user defined size and outputs the maximum value of the section of the image contained in the 2D sub-region. This type of layer is typically used to reduce the feature space dimensionality and propagate the strongest features to smaller scales. The reduction in the width and height of the input is done by setting the stride, meaning the spacing between every 2D sub-region along the i and j directions, to be of the same size as the 2D sub-region itself. Doing so will cause no overlap between each 2D sub-region. Since the next layer is a down-sampled version of the previous, certain features will persist across different scales within the network. Persistence of features across scales using pooling layers provides CNNs with the important property of scale invariance [30]. Unlike a convolution layer, a pooling layer does not create trainable parameters. An example of a 3×3 max-pooling operation is shown in Fig. 2.5. The pooling filter sweeps across the feature map with a stride of three and the impacted features are highlighted by different colors for

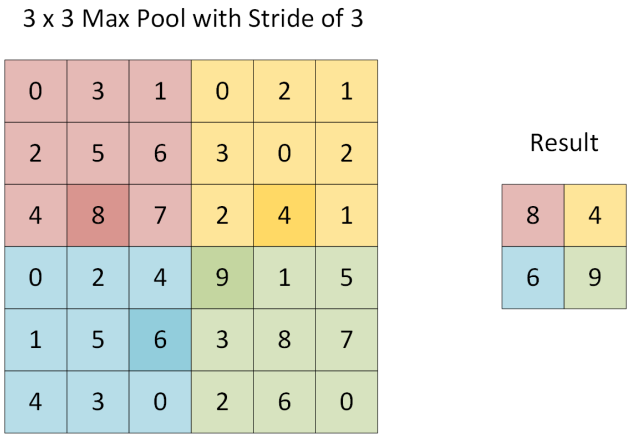


Figure 2.5: Max pool of a 6 × 6 feature map using a 3 × 3 filter with a stride of 3.

each pooling operation. The maximum feature of each distinct region is highlighted by a darker color and persists to the output shown on the right-hand-side.

2.2.5 Neural Networks – Purpose of Activation Functions

Returning to the conceptualization of neural networks, we can now elaborate on the purpose of activation functions. Activation functions serve to introduce non-linearities between layers. Non-linearities are needed to increase the representational power of deep networks since fully connected layers and convolutions are linear operators. Chaining multiple convolution or fully connected layers can be effectively performed by single a operation even though multiple layers were used. To illustrate, suppose we have a two layer neural network where $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ denote the weights and bias in the l^{th} layer. Let $\mathbf{x}$ be the input to the first layer and let $\mathbf{y}^{(l)}$ be the output of the l^{th} layer. The output of the first layer is

$$\mathbf{y}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}, \tag{2.8}$$

and is fed into the input of the second layer such that

$$\mathbf{y}^{(2)} = \mathbf{W}^{(2)}\mathbf{y}^{(1)} + \mathbf{b}^{(2)}. \tag{2.9}$$

By substitution and rearrangement, it can be seen that the representational power of the two layer neural network can be no better than a one layer neural network:

$$\begin{aligned} \mathbf{y}^{(2)} &= \mathbf{W}^{(2)}[\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}] + \mathbf{b}^{(2)} \\ &= \mathbf{y}^{(2)} = \mathbf{W}^{(2)}\mathbf{W}^{(1)}\mathbf{x} + [\mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)}] \\ &= \mathbf{y}^{(2)} = \mathbf{W}_{Effective}\mathbf{x} + \mathbf{b}_{Effective}. \end{aligned} \tag{2.10}$$

For this reason, a non-linear vector function $\boldsymbol{H}$ is applied between layers so that intermediary outputs $\mathbf{y}^{(l)} = \boldsymbol{H}(\mathbf{W}^{(l)}\mathbf{y}^{l-1} + \mathbf{b}^{(l)})$ are not linearly dependent in order to increase the representational power of the neural network. Such an analysis is useful in understanding that neural networks can be represented as a composition of functions. Extending the previous example to L layers, a neural network can be written as follows:

$$\mathbf{y}^{(L)} = \boldsymbol{H}^{(L)}(\boldsymbol{H}^{(L-1)}(\cdots \boldsymbol{H}^{(1)}(\mathbf{X}; \mathbf{W}^{(1)}); \mathbf{W}^{(L-1)}); \mathbf{W}^{(L)}), \tag{2.11}$$

where $\mathbf{b}^{(l)}$ has been added as a row into $\mathbf{W}^{(l)}$ and consequently, a row of ones is added to $\mathbf{x}$, now denoted as $\mathbf{X}$. This representation is key to understanding the algorithms used by neural networks for "learning", which are presented in the following sections. These algorithms consist of a forward pass to calculate the output of each neuron, a backward pass to calculate the gradients of a loss function with respect to the weights, and an optimizer to update all the weights in the network using the gradients calculated in the backward pass.

2.2.6 Learning – Loss Functions

Now that a representation of neural networks and its components has been discussed, the role and function of neural networks are presented next. The objective of neural networks, like any other machine learning algorithm, is to find the best "curve" to fit the data in a regression problem, or to find the best "curve" to separate and classify data into groups in a classification problem. A metric is needed in order to measure how well the neural network can do these tasks. These metrics are known as objective functions, cost functions, loss functions, or error functions in deep learning. A common loss function used for regression problems is the Mean Squared Error (MSE):

$$E = \frac{||\mathbf{y}_{pred} - \mathbf{y}_{true}||^2}{N}. \tag{2.12}$$

Here $\mathbf{y}_{pred}$ is the output predicted by the neural network, $\mathbf{y}_{true}$ is the true labelled output that the network is trying to recreate, and N is the number of samples used in the calculation of the loss. For classification problems, Cross-Entropy is typically used instead to measure the classification accuracy:

$$E = -\sum_{n=1}^{N}\sum_{k=1}^{K} \delta_{nk} \log(p(y^{(n)} = k|\mathbf{x}^{(n)}; \mathbf{W})). \tag{2.13}$$

Here, δ_{nk} is the Kronecker delta and $p(y^{(n)} = k|\mathbf{x}^{(n)}; \mathbf{W})$ is the probability of producing a prediction $y^{(n)}$ for sample n equal to the labelled class k for given inputs $\mathbf{x}^{(n)}$ and model weights $\mathbf{W}$.

2.2.7 Learning – Optimizers

In order to find the best fit or classifier given the data, the appropriate error function is minimized through a gradient-based optimization algorithm. The most basic optimizer outlined by Algorithm 1 is called gradient descent, which is usually attributed to L. Cauchy who presented the method in [31] and can be found in several machine learning textbooks, Goodfellow et. al. [32] is one of them. The gradient descent algorithm is a first-order iterative optimization strategy that uses the first derivative of a differentiable function to find its local or global minimum. Many alternative formulations of gradient descent exist to

Algorithm 1: Gradient Descent.

Input: $\mathbf{x}$: Input data,
$\mathbf{w}_0$: Initialized weights,
η: Learning rate,
T: Number of iterations

for $t = 1 \ldots T$ **do**
  $\mathbf{g}_t = \nabla_w E(\mathbf{x}, \mathbf{w}_{t-1})$; // Calculate gradients
  $\mathbf{w}_t = \mathbf{w}_{t-1} - \eta * \mathbf{g}_t$; // Update weights
end

improve the downsides of gradient descent such as getting caught in local minimas, saddle points, or plateaus, but one of the most popular optimizers used in deep learning is the Adaptive Moment Estimation (ADAM) optimizer. ADAM uses adaptive moment parameters to create momentum in the gradient steps. This optimizer is used for all models presented in this work and the algorithm is reproduced here as Algorithm 2 from Kingma and Ba's paper [33].

An important calculation in either of these optimizers is $\nabla_w E(\mathbf{x}, \mathbf{w})$, the derivatives of the error function E with respect to the weights $\mathbf{w}$ in the network. The derivatives are calculated during backpropagation and the representation of neural networks as a composition

Algorithm 2: The ADAM optimizer adapted from [33].

Input: $\mathbf{x}$: Input data,
$\mathbf{w}_0$: Initialized weights,
η: Learning rate,
β_1, β_2: Exponential decay rates for the moment estimates,
T: Number of iterations

$m_0 = 0$; // Initialize 1st moment vector
$v_0 = 0$; // Initialize 2nd moment vector
for $t = 1 \ldots T$ **do**
  $\mathbf{g}_t = \nabla_w E(\mathbf{x}, \mathbf{w}_{t-1})$; // Calculate gradients
  $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$; // Update biased first moment
  $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$; // Update biased second moment
  $\hat{m}_t = m_t / (1 - \beta_1^t)$; // Compute bias-corrected first moment
  $\hat{v}_t = v_t / (1 - \beta_2^t)$; // Compute bias-corrected second moment
  $\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$; // Update weights
end

of function shown in Equation 2.11 can aid in understanding the backpropagation algorithm. Taking the derivative of Equation 2.11 with respect to $\mathbf{w}^{(l)}$ using the chain rule yields:

$$\frac{\partial \mathbf{y}^{(L)}}{\partial \mathbf{w}^{(l)}} = \frac{\partial \mathbf{y}^{(L)}}{\partial \mathbf{y}^{(L-1)}} \prod_{k=l}^{L-2} \frac{\partial \mathbf{y}^{(k+1)}}{\partial \mathbf{y}^{(k)}} \frac{\partial \mathbf{y}(l)}{\partial \mathbf{w}^{(l)}}. \tag{2.14}$$

It can be seen that Equation 2.14 requires the derivative of the output of each neuron with respect to its input. Equation 2.14 can also be simplified by observing that $\frac{\partial \mathbf{y}^{(L)}}{\partial \mathbf{w}^{(l)}}$ is a Jacobian, where $\mathbf{w}^{(l)} \in \mathbb{R}^n$ and $\mathbf{y}^{(L)} \in \mathbb{R}^m$:

$$J_{y^{(L)}}(\mathbf{w}^{(l)}) = \begin{bmatrix} \frac{\partial y_1^{(L)}}{\partial w_1^{(l)}} & \cdots & \frac{\partial y_1^{(L)}}{\partial w_n^{(l)}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m^{(L)}}{\partial w_1^{(l)}} & \cdots & \frac{\partial y_m^{(L)}}{\partial w_n^{(l)}} \end{bmatrix}. \tag{2.15}$$

Therefore, Equation 2.14 can be expressed in terms of Jacobians J:

$$J_{\mathbf{y}^{(L)}}(\mathbf{w}^{(l)}) = J_{\mathbf{y}^{(L)}}(\mathbf{y}^{(L-1)}) J_{\mathbf{y}^{(L-1)}}(\mathbf{y}^{(L-2)}) \cdots J_{\mathbf{y}^{(l+1)}}(\mathbf{y}^{(l)}) J_{\mathbf{y}^{(l)}}(\mathbf{w}^{(l)}). \tag{2.16}$$

A multiplication of the derivatives of each neuron in the network is required from back to front to calculate the gradients of neurons at a desired layer l. This is the core of backpropagation, which is shown in Algorithm 3 and reproduced from [34].

Algorithm 3: Forward and Backward Propagation.

Input: $\mathbf{x}$: Input data,
$\mathbf{w}$: Weights

$h[L+1]$; // initialize array for outputs of each layer
$g[L]$; // initialize array for gradients of each layer
$h[1] = \mathbf{x}$; // initialize input layer
$d = 1$; // initialize derivative for chain rule
for $l = 2, \ldots, L+1$ **do** // forward pass
  $h[l] = H_{l-1}(h[l-1])$; // store current layer output
end
for $l = L, \ldots, 1$ **do** // backward pass
  if *layer l has weights* **then**
    $g[l] = d \cdot J_{\mathbf{y}^{(l)}}(\mathbf{w}^{(l)})$; // calculate gradient of current layer
  end
  $d = d \cdot J_{\mathbf{y}^{(l)}}(\mathbf{y}^{(l-1)})$; // accumulate gradients in chain rule
end

To summarize this section, neural networks were presented in three different ways. First, as a graph diagram of a network of neurons, second as matrix multiplications, and third as compact mathematical notation. Within this context, neurons were shown to represent the evaluation of an activation function given weighted input. Subsequently, three different layers used in CNNs were presented, namely dense layers, convolutional layers, and pooling layers, where significant emphasis was made on the convolution operation. A composition of functions was then built up to represent a multi-layered neural network, which prompted a discussion about the role of non-linear activation function between each layer. The function composition representation also aided in the description of the learning process where loss functions, optimizers and backpropagation were presented. Fortunately, general frameworks

exist that make all these tools available to create and train neural network models in Python [35], which is the main programming language used as part of this work. All neural networks used in this work are defined and trained using Keras [36] and TensorFlow as the backend.

2.3 Remote Sensing of Soil Moisture

Estimation of soil moisture from remote sensing data is one of the two applications where it is sought to determine whether or not it is possible to estimate using deep learning. Soil moisture is an important metric that is used for applications such as agriculture, hydrology, and meteorology [37]. An accurate and detailed map is required for these applications since soil moisture is typically used as an initial condition for numerical simulation models and can impact the outcome of the models developed for such applications.

Soil moisture is measured primarily using two methods; the first is via in-situ ground measurements and the second is through the inversion of space-based remote sensing measurements [37]. In-situ soil moisture measurements are typically performed using Time Domain Reflectometry (TDR) probes or through gravimetric analysis [37]. The TDR method is faster, but less precise than the gravimetric method. The gravimetric method on the other hand is the most accurate of the two, but also the most labour and time intensive [37]. In-situ measurements are applicable to localized and controlled regions such as farming fields, but are not well-suited for monitoring large and remote areas for hydrological and meteorological applications. The space-based remote sensing method is the primary way to regularly monitor environmental conditions such as soil moisture at a large scale [37]. Microwave Synthetic Aperture Radar (SAR) sensors are typically used for these applications since operation is independent of meteorological conditions (cloud cover) and time of day [37]. Pertaining specifically to soil moisture estimation from SAR, sensors operating at the L-band of frequencies (1 - 2 GHz) have been shown to be sensitive to variations in SMC [38], [39], but the majority of modern SAR systems (Sentinel-1, RADARSAT-2, COSMO SkyMed, TerraSAR-X) [40] operate at C-band (4 - 8 GHz) and X-band (8 - 12 GHz). It has

been shown that frequencies above C-band are not optimal for measuring soil moisture since the backscatter response at these higher frequencies (compared to L-band) is also sensitive to surface roughness and vegetation cover [41], [42]. However, a variety of approaches have been researched to improve SMC estimates from sensors operating in the C-band to utilize the improved capabilities of modern SAR satellites. Generally, the approaches used to estimate SMC through the inversion of SAR data are categorized in two forms; the first is through theoretically derived backscattering models (physical model) and the second is through the formulation of semi-empirical models [37].

Physical models are formulated using backscattering theory which aims to describe the reflection between an interface and incident microwaves. These models typically use sensor configurations such as operation wavelength, polarization, and incidence angle along with surface characteristics such as SMC and roughness to estimate the radar backscattering that would be seen in practice [37]. Physical models, such as the small perturbation method (SPM), the physical optics (PO) model, and the geometric optics (GO) model are evaluated in [43] and show poor agreement with observed data, even within the limited valid regions of applicability. Another model called the Integral Equation Model (IEM) is the most common physical model used to estimate SMC and is generally more complex than the other models mentioned above, but does provide a greater range of valid surfaces [44].

Empirical models are constructed by inverting functions used to model the observed σ^0 backscatter measured by satellite which are dependent on soil characteristics such as SMC and roughness. Popular empirical models used to extract SMC from SAR backscatter include models put forward by Oh [43] and Dubois [45]. A simple, empirically derived, linear model was also studied in [37]. A known issue with such methods however is the extent with which

the models can be applied to regions that were not part of the dataset used to formulate the model. The strength of such models thus depends on the representativeness of the dataset [37]. A recent study was conducted to evaluate and compare the performance of the Oh, Dubois and IEM models in their ability to estimate soil moisture from SAR imagery by validating them against experimental soil measurements [46]. It was found that the Oh model gave the best fitting of the backscattering coefficients for HH and VV polarizations. However, with minor improvements to the IEM model made by Baghdadi, it was found that this model was the best in estimating soil moisture and roughness from SAR data.

Change detection is another example of a common empirical approach to estimating SMC through temporal changes between SAR acquisitions. The premise for the change detection approach is that vegetation cover and surface roughness are relatively stable and the main cause for change in SAR backscatter at C-band across time is SMC [40]. The claim is made in [40] that multi-temporal approaches can account for surface roughness and low vegetation cover and successfully extract soil moisture from SAR images acquired in the C-band frequency range based on previous research [47], [48]. The recently launched Sentinel-1 satellite provides a good source of recurring C-band SAR data needed for change detection algorithms due to its relatively quick revisit times that range between 6 to 12 days. One change detection algorithm that was developed to use Sentinel-1 data to predict SMC as part of the Global Monitoring for Environment and Security (GMES) campaign, now re-branded as the Copernicus program, is called the SMOSAR algorithm [49]. The "Soil MOisture retrieval from multi-temporal SAR data" (SMOSAR) algorithm is an algorithm that uses the change between multiple Sentinel-1 Interferometric Wave (IW) mode SAR images taken across time to determine SMC. The algorithm was assessed using SAR data from three well documented data sets collected during the AgriSAR 2006 campaign over the

Gormin farm at the DEMMIN site in Germany, the AgriSAR 2009 campaign over Flevoland in the Netherlands, and the Italian ENVISAT 2003 and 2005 campaigns over Matera in Italy. Ground truth data used for validating the algorithm was synthetically produced by a hydrological model because on-site ground measurements did not coincide temporally with the SAR acquisitions. An 84% correlation is achieved between the SMOSAR's SMC maps for both HH and VV polarizations and the hydrological model maps. It was noted however that the VV polarization manifested greater variability in RMSE ranging between 1-10%, than did the HH polarization, which ranged between 4-6%. The greater variability in RMSE for the predictions made based on the VV polarization may be due to difficulties in generalizing over different test sites or due to an insufficient number of images and test sites.

Deep learning techniques have also been used to develop empirical models that are learned from SAR data and in-situ measurements. Such approaches also allow for the combination of auxiliary data such as vegetation cover and temperature to improve results. A recent study investigates the utility of combining Sentinel-1 SAR data with Landsat-8 Operational Land Images (OLI) within an Artificial Neural Network (ANN) to estimate SMC [50]. Landsat-8 is used in this study because it can provide measurements for vegetation cover, expressed by the Normalized Difference Vegetation Index (NDVI), and temperature, measured by the Thermal Infrared Sensor (TIRS); two useful parameters for the estimation of SMC. NDVI estimates vegetation coverage in terms of height and density [50] and can be derived from the visible and near infrared bands of optical sensors aboard Landsat 8. As stated previously, SMC retrieval from C-band SAR imagery is made difficult due to the sensitivity to vegetation cover. Therefore the inclusion of NDVI in SMC models aims to account for this effect in the SAR backscatter to improve results. The ANN developed for the study in [50] consisted of four input neurons, one hidden layer consisting of ten neu-

rons, and an output layer giving the estimated SMC. The four inputs to the ANN were the corrected radar backscattering, NDVI, the incidence angle, and the thermal infrared temperature (TIR). The ANN model produced results with correlations of 0.90171, 0.721, and 0.50841 on the training, validation, and testing sets respectively, which are not great testing results and may be caused by overfitting. A sensitivity analysis done on their model through the leave-one-out method (one component of the set of components is left out one at a time) showed that the incidence angle was the least important input, while NDVI was the most important for accurate SMC estimates.

Another study using a deep learning model consisting of two hidden layers of ten neurons each investigated the performance of variably trained networks with various input polarizations and NDVI inclusion [40]. This study was also conducted for European Space Agency's (ESA) GMES program. In this study, dataset augmentation was used to synthetically generate backscattering coefficients for co- and cross-polarizations using the Advanced Integral Equation Model (AIEM) and Oh models due to insufficient data for training the neural network. It was noted in the study that ESA's download policy limited most SAR data to VV polarized images, which is unfortunate since HH and co-polarized images gave better results. Experiments using VV polarization data had the poorest performance with a coefficient of determination of 0.4047, but was improved to 0.6404 with the addition of NDVI. Inputs using the HH polarization performed better than the VV experiments for both cases where NDVI was not included and where NDVI was included by achieving a coefficient of determination of 0.86 and 0.8767 respectively. However, the best coefficient of determination of 0.8716 was achieved when co- and cross-polarizations (HH + HV) were used.

A large source of data is needed for the purposes of this thesis given that the perfor-

mance of empirical approaches such as deep learning is dependent on the representativeness of the dataset with which it is trained. Sentinel-1 is one of the most widely used source of SAR data due to the European Space Agency's (ESA) Copernicus program which has a free, full and open data policy [50]. Sentinel-1 operates in the C-band and the Interferometric Wave (IW) mode is typically used. Sentinel-1 SAR images can therefore serve as the input data to various deep learning models assessed as part of this thesis. The difficulty in training a deep learning model for SMC estimation is in the acquisition of sufficiently large and accurate SMC data to be used as the corresponding output to the input SAR image. Previous studies have used in-situ SMC measurements during different campaigns, but these consist of point sources that can only be related to a few pixels in a SAR image and are not easily accessible. Fortunately, there exists satellites that produce SMC images through various pre-validated algorithms. One of these is NASA's Soil Moisture Active-Passive (SMAP) satellite, which can measure microwave energy emitted from the Earth to produce soil moisture maps at a resolution of 1 km x 1 km pixel spacing [51]. The ESA's Soil Moisture Ocean Salinity (SMOS) satellite is another satellite that produces SMC maps using L-band microwave data, but was not chosen for this work as the spatial resolution of these maps are on average 40 km [52], much larger than NASA's SMAP satellite. The accuracy of NASA's original soil moisture algorithm used to convert microwave measurements to soil moisture achieved average correlation values of 0.717 and 0.829 for 3 km and 9 km range images respectively [53]. These correlation metrics were validated with in-situ soil moisture measurements at select regions and were achieved by NASA's original algorithm, which used data from the active and passive microwave sensors aboard the SMAP satellite. However, the active sensor malfunctioned and NASA resorted to the development of a new algorithm to produce soil moisture maps. This new algorithm combines data from the working passive radiometer sen-

sor on the SMAP satellite with SAR data from Sentinel-1. The problem with this solution is that the SMAP and Sentinel-1 satellites do not have the same orbit and therefore do not intersect with one another. The acquisitions taken from the SMAP and Sentinel-1 satellites are often taken 12 hours apart and the revisit time to the same geolocation occurs every 12 days. This is a problem because varying weather conditions can occur within this 12 hour span of time thereby causing two different soil conditions to be measured and combined.

The goal of this thesis is to gain experience using deep learning techniques in different application areas and demonstrate the versatility of deep learning. Despite the data acquisition lag between the Sentinel-1 and SMAP satellites, these two data sources are used to train a neural network to reproduce SMAP based soil moisture maps with SAR imagery from Sentinel-1 as input. These data sources are chosen based on the availability of large quantities of data and for Sentinel-1's high spatial resolution. Potential improvements to NASA's current process can include the elimination of the 12 hour discrepancy between data readings (since Sentinel-1 will be the only source of data at prediction time) and this work may also provide a means to increase the spatial resolution of the soil moisture maps. This can be accomplished by applying a trained model to full resolution 10 m $\times$ 10 m pixel spacing [54] Sentinel-1 Ground Range Detected (GRD) SAR images. For the purposes of maintaining the thesis of this work, generating high resolution soil moisture maps using full resolution SAR images is deemed unnecessary and infeasible due to the lack of readily available in-situ data needed to validate the high resolution estimations. For this reason, applying the neural networks to high resolution SAR images to estimate high resolution soil moisture maps was not pursued and the comparison of the performance of the neural networks trained in this work to NASA's and other researchers validation correlation metrics is sufficient to support the thesis.

2.4 Remote Sensing of Sea Ice

Material in this section has been adapted from [55][1], a paper accepted for publication in MDPI's Remote Sensing Journal and included in the Special Issue on Polar Sea Ice: Detection, Monitoring and Modeling.

The classification of sea ice type in the Arctic is the second application domain used to explore the capabilities of neural networks in this thesis. This application domain was chosen due to the growing interest in having accurate maps of sea ice conditions during summer melt periods for commercial naval transportation. Changing sea ice conditions during the summer melt period have led to changes in transportation and usage of Arctic waterways [56]. Knowledge of the thickness, extent, and type of sea ice is critical for operations planning performed by Environment and Climate Change Canada Regional Ice-Ocean Prediction System [57], shipping routes, and sustainable development of the North. Operational ice monitoring agencies, such as the Canadian Ice Service (CIS), produce sea ice maps describing the total and partial ice concentrations, the stage of development (or ice type), and the floe size of the ice type. All three of the aforementioned ice characteristics are summarized in the form of an "egg code" [2] to describe homogeneous regions in the sea ice maps produced by the CIS. Ice analysts at the CIS manually interpret RADARSAT-2 (R2) SAR imagery as their source of data to produce ice maps since it is the main satellite used for routine monitoring of the Canadian Arctic. The expert knowledge of ice analysts is needed for the manual interpretation of SAR imagery due to the complex nature of the sea ice backscatter measured and illustrated in SAR imagery. Due to the large dataset of SAR acquisitions however, an

[1]All articles published by MDPI are made available worldwide under an open access license, meaning no special permission is required to reuse all or parts of articles published by MDPI. Reprinted with permission from R. Kruk, M. C. Fuller, A. S. Komarov, D. Isleifson, and I. Jeffrey, "Proof of concept for sea ice stage of development classification using deep learning," *Remote Sensing*, vol. 12, no. 15, p. 2486, 2020.

automated process could help increase the data throughput of the CIS and accelerate expert analysis by applying the expert knowledge learned through machine learning techniques to the classification of sea ice in SAR scenes.

SAR satellites are used to monitor remote regions because they are capable of measuring the Earth's surface in all weather conditions and in darkness. This is particularly important for monitoring the Canadian Arctic since there are periods of 24 hour darkness due to the high latitude and optical sensors are incapable of making ground measurements in the dark or when significant cloud cover occurs during the spring and fall seasons. These systems are ideal for detailed mapping of a large region because of the high spatial resolution that can be achieved (below 100 m pixel spacing) and the high spatial coverage (up to 500 km by 500 km). The CIS uses RADARSAT-2 to produce ice maps, which operates in the microwave C-band at 5.4 GHz and has various acquisition polarization modes such as HH (horizontal transmit, horizontal receive) and HV (horizontal transmit, vertical receive). Different polarizations are sensitive to different sea ice types since the microwave backscatter is dependent on the physical state of the ice. The backscatter response of sea ice from a remote sensing lens are well known [7] since the physical state of the sea ice can be modelled as a collection of its dielectric properties. The dielectric properties of different ice types govern the microwave interactions for a given incidence angle, frequency, and polarization. The horizontal co-polarization (HH) is generally able to distinguish between the state of sea ice or open water, whereas the cross-polarization (HV) generally provides information about volume scattering within snow and sea ice surfaces which can be used to distinguish between sea ice types from open water [7]. Furthermore, the cross-polarized ratio (the ratio of HV to HH) can provide additional information about the composition of a surface cover to be used to improve classification. Classifying ice from SAR imagery however is a difficult task, par-

ticularly during the spring and summer melt, because of many-to-one scenarios where many different sea ice properties have the potential to appear as the same SAR backscatter. Also, SAR imagery is highly influenced by the presence of water on sea surfaces [58], [59] during the spring melt season. Difficult classification scenarios also exist during the fall freeze-up period due to high variability in conditions within the same region that can form different types of ice [60]. For these reasons, expert analysis has been required thus far to decipher the SAR backscatter under these difficult and conflicting conditions to produce meaningful and usable descriptions of sea ice coverage throughout the Arctic.

The goal of supporting ice analysts through the application of machine learning algorithms to classify sea ice has been a focus of research for at least the past twenty years. Automated Sea Ice Segmentation (ASIS), developed in 1999 [61], combined image processing, data mining, and machine learning to automate segmentation of SAR images (single polarization) from RADARSAT and European Remote Sensing (ERS) satellites. The authors used multi-resolution peak detection and aggregated population equalization spatial clustering to segment SAR images for pre-classification of sea ice for subsequent expert classification and analysis. The results worked well in identifying ice classes in the image, but encountered issues with melting ice conditions in the summer season. Another sea ice classification algorithm was developed in 2005 [62] using Freeman-Durden decomposition as a seed for an unsupervised Wishart-based segmentation algorithm on SAR images. The algorithm was applied to single and dual frequency polarimetric data (C- and L-band) with good results, but noted that the summer melt season sea ice presented the greatest challenge with regard to SAR based classification. In a more recent publication, land-fast sea ice in Antarctica was mapped using decision trees and random forest machine learning approaches [63]. These approaches were applied on a fusion of time-series AMSR-E radiometer sea ice

brightness temperature data, MODIS spectroradiometer ice surface temperature data, and SSM/I ice velocity data. All three studies noted that summer melt season sea ice presented the greatest challenge with regard to SAR based classification due to the increased occurrence of many-to-one scenarios.

More recently, state-of-the-art deep learning techniques have achieved successful results for challenging predictive tasks in other areas and there is also growing interest in applying deep learning techniques to predict sea ice characteristics. A recent study uses DenseNet [64] to estimate sea ice concentration from training on a dataset of 24 SAR images [65]. The authors of this paper used the Artist Sea Ice (ASI) algorithm [66] to label their dataset. This study experimented with a variety of factors and found that the accuracy of the model was dependent on the size of the patches used from the SAR images. The authors addressed over-fitting using data augmentation to increase the dataset size by adding varying degrees of Gaussian noise. The authors also developed a novel two-step training technique that consisted of first training a model with augmented data until convergence to a local minimum. Subsequently, a new model was initialized with the weights from the previous model and trained with a smaller learning rate on the dataset without augmentation. The authors hypothesized that noise injection into the SAR patches removed the relevance of texture information and led to degraded results. This training process was found to improve the results because the texture information relevance was recovered. However, there were difficulties predicting ice concentrations for thin new ice. Another study [67] used a 5-layer Convolutional Neural Network to estimate sea ice concentration from SAR imagery. The dataset used in this study consisted of 11 HH and HV SAR images that were sub-sampled and labelled using CIS charts by extracting 41×41 sub-regions centered about the image analysis sample location denoted by latitude and longitude. The study found that the num-

ber of water samples contained in their dataset was about 8 times greater than the next most common ice concentration class. Several approaches exist to work around such a disparity in the distribution of classes such as undersampling the majority, oversampling the minority or using a Bayesian cross-entropy cost function, but the authors chose to train with the skewed dataset for a larger number of epochs (500) to get out of a local minimum found at early epochs due to the over representation of water.

While the deep learning studies mentioned above attempted the estimation of ice concentration, the focus of the sea ice application domain of this thesis is in the classification of sea ice by type. The challenge in developing a deep learning classifier for sea ice type though is the availability of a large number of accurately labelled data as well as the labelling method itself since there are various methods to do so. A previous study used a Pulse-Coupled Neural Network (PCNN) with RADARSAT-1 ScanSAR Wide mode images over the Baltic Sea [68]. Therein, classes for supervised training were created by decomposing homogeneous regions in the SAR image into a mixture of Gaussian distributions using an Expectation-Maximization algorithm. These generated classes were then paired with the associated region in the SAR image and given to the PCNN for training. Another study [69] generated classes for their dataset per the World Meteorological Organization (WMO) terminology [1]. Their dataset consisted of SAR patches along the path of an icebreaker and the generated classes were based on *in-situ* observations and other sensor information.

As an alternative to the labelling methods above, the labelling method adopted in this thesis to create input-output pairs for the supervised training of a neural network uses the stage of development feature of the egg codes given by the expert ice analysts at the CIS ice charts. The labelling method is similar to the 2016 study done by Wang et al. [67], but is

applied to the estimation of ice type, as oppose to ice concentration. While ice concentration is an important factor for naval navigation in the Arctic, our goal in this work is to predict ice type given by the stage of development. Knowledge of current ice type conditions can aid in the decision making process during naval routing because ships can break through regions where the ice concentration may be high, but the ice type is thin. Conversely, it is also important to know where the ice type is thick in order to avoid those regions.

Chapter 3

Soil Moisture Application

The work presented in this chapter aims to determine whether or not soil moisture can be estimated from SAR data and replicate the results produced by NASA's SMAP satellite using deep learning techniques. The contents of this chapter includes a section concerning the methodology used to preprocess the satellite imagery and construct datasets of input-output pairs for the supervised learning of deep learning models. The methodology section is followed by a results section where eight different experiments are conducted in an aim to improve results and achieve similar correlations to those achieved by NASA and other studies found in the literature. A correlation of 79.2% on the entirety of the test set is the best result achieved by the experiments conducted for reproducing a SMAP soil moisture image. A discussion is then made to assess the results of the experiments and a comparison is made to findings obtained from other studies. It was found that the correlation of 79.2% achieved herein is comparable to results found in other studies. The discussion also includes points of improvements and limitations. A concluding section for the chapter follows the discussion, which summarizes the methodology, the results, points of improvement for future work, and an assessment of the goals of the thesis is made with respect to soil moisture estimation.

3.1 Methodology

Data acquisition is the first step in training any machine learning model. The search for data was carried out across the globe to increase the probability that the gathered dataset is representative of all SMC conditions. Specific regions were selected based on high density regions of in-situ ground measurement facilities outlined by the International Soil Moisture Network (ISMN). These regions are summarized in Table 3.1 and ESA's database was queried from the beginning of April 2018 to the end of June 2018 to have sufficient data across seasonal changes. The areas of interest were chosen to provide a potential source of in-situ

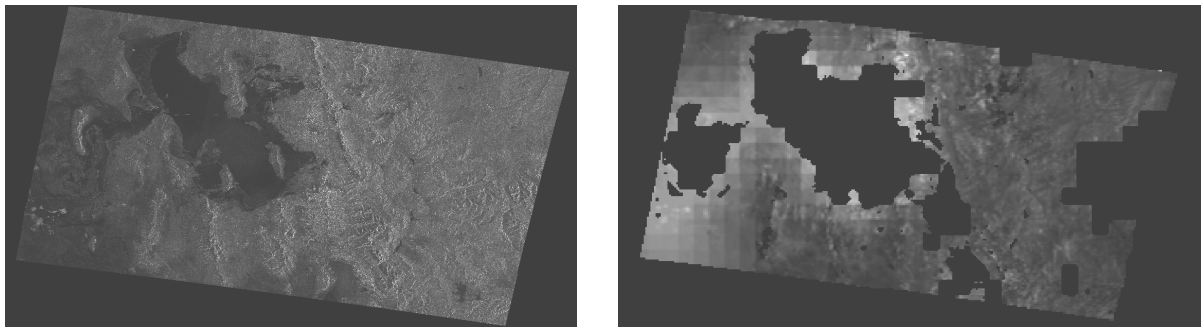
Table 3.1: Sentinel-1 Areas of Interest

	Lower Left Corner		Upper Right Corner	
Area of Interest	Lon	Lat	Lon	Lat
1	-120.775	36.890	-117.310	39.800
2	-113.315	37.250	-110.450	42.250
3	-107.050	35.725	-104.475	40.590
4	-102.100	40.000	-95.900	42.850
5	-99.400	34.720	-95.590	38.340
6	-1.105	42.940	5.450	45.100
7	-5.680	41.110	-5.030	41.505
8	8.815	55.910	9.120	56.040
9	15.685	46.850	16.180	47.050
10	91.690	30.835	93.440	32.190
11	100.230	38.760	100.465	38.905
12	101.700	33.635	102.655	34.010

ground data that could be used in the future to validate the predictions. However, the validation of the predictions made by the neural networks trained in this work with in-situ ground measurements was not conducted for logistical reasons. The first reason is due to unknowns concerning the treatment of in-situ ground measurements acquired by the ISMN.

This leads to questions on how to relate the ground data to the predicted output. ISMN data could be treated as point-like observations, where the ground measurement location would only be related to a single pixel in the predicted output, or they could represent an average over multiple measurement sites across a small region that could be related to a small grouping of pixels. The second reason is due to time limitations. Several steps are needed to link the geo-location of the ground measurements to validate the predictions. Code is needed to not only download the ground measurements, which come in the form of a graph with SMC measurements taken across time, but to also find the measurement that is most closely related to the SAR image both in time and geographically. Nonetheless, these areas of interest serve as a good starting point since they span the northern hemisphere and therefore span many different soil characteristics and climates.

Sentinel-1 images were queried and downloaded from ESA's Copernicus Open Access Hub [70] according to the timeline and regions of interest given in Table 3.1. The corresponding SMAP images were downloaded from NASA's Earth Data repository [71]. A sample of each satellite images are shown in Figure 3.1a and Figure 3.1b respectively.



(a) High resolution SAR image acquired by Sentinel-1

(b) Corresponding soil moisture image given by the SMAP satellite

Figure 3.1: Raw sample of input (a) and output (b) pair of images used to learn how to estimate soil moisture from SAR data.

The input to the neural networks trained herein are the Sentinel-1 SAR images, while the labelled target output are the SMAP soil moisture images. Sentinel-1 provides C-band Synthetic Aperture Radar (SAR) data in a variety of modes, but the interferometric wide swath (IW) was chosen because it is Sentinel-1's primary operational mode over land [72]. The radar can be configured in a variety of polarizations [73], but the single VV polarization is typically the most abundantly available type of data due to ESA's download policy. In addition to those options, the Level-1 Ground Range Detected (GRD) images are used because the pixel values of the image represent the detected amplitude, whereas the Single Look Complex (SLC) product contains a real and imaginary part [74]. The GRD SAR images have a pixel spacing [75] of 10 m x 10 m per pixel [54]. Sentinel-1 data can come in various processed forms such as Level-0, which is raw data, Level-1, which is calibrated pixel data by the Instrument Processing Facility (IPF) using a variety of algorithms, and Level-2, which adds geolocations to geophysical products. The Level-1 GRD images can be verified by checking the manifest.safe file in the downloaded SAR samples for the IPF version per [76], which indicates the IPF version to be >2.90. SAR products with IPF >2.90 do not require any additional processing for masking purposes as the no-value pixels are set to 0.

The SMAP satellite derives SMC of the top 5 cm of soil from brightness temperatures and measurements of the backscatter coefficient σ^0 and is given as a volumetric or gravimetric ratio. SMC estimates given by SMAP are derived using an algorithm developed by NASA that uses SMAP's L-band radiometer to measure four brightness temperature Stokes parameters in combination with the SAR backscatter coefficients given by Sentinel-1. The soil moisture image is projected to the EASE-Grid 2.0 projection, which is defined with the World Geodetic System (WGS 84 in particular) and has a resolution of 1 km x 1 km per pixel. It is important to note that the 1 km resolution SMAP images are meant for research

purposes since the soil moisture estimates found in those images have not been validated at that spatial scale (3 km and 9 km pixel spacing resolution images have been validated) [51].

3.1.1 Preprocessing

The SAR images downloaded from ESA's Copernicus Open Access Hub are GeoTIFF files that are georeferenced but not geocoded [77]. This means the SAR image file contains geographical information, but has not been mapped to a specific projection of the Earth. The process of geocoding therefore transforms the SAR geometry into a map projection. Geocoding is needed because the SAR and SMAP images need to be projected to the same coordinate system for the construction of a pixel-to-pixel relationship. Typical geographic information system (GIS) applications such as ArcGIS [78] or QGIS [79] geocode images on the fly when the image is opened for it to be displayed properly within the context of the other images in the GIS project. However, these applications are not efficient in processing large datasets of images that can be upwards of 1 GB in size each due to memory constraints and operation complexity. A popular utility used in the GIS community for processing GeoTIFFs is GDAL [80], a Geospatial Data Abstraction Library. GDAL was used to geocode and preprocess the SAR data for the purposes of this thesis due to the large quantities of data being processed. Another benefit to using GDAL is that it is both fast and can be called from a variety of programming languages, such as Python [35], the main programming language used in this work. The SAR images are geocoded to the same projection as the SMAP images given that the SMAP images downloaded from NASA's Earth Data repository are already geocoded to the EPSG 4326 coordinate system as given by the GDAL command `gdalinfo`.

Obtaining the SMAP image projection information can only be done once the 1 km

x 1 km resolution layer has been extracted from the HDF5 file downloaded from the Earth Data repository. The HDF5 file contains many layers [81]; in this work the soil_moisture_1km layer is used. HEG [82] is a HDF-EOS to GeoTiff conversion tool that was used to extract this layer to a GeoTiff file and a batch of HDF5 files are processed as outlined in [83].

Once the SMAP images have been extracted and the projections have been established, geocoding the SAR image with the same projection as the corresponding SMAP image can be accomplished with `gdalwarp` [84] per the instructions outlined in [77]. Opening the resulting geocoded SAR image with the corresponding SMAP image in a GIS application such as QGIS will result in the image shown in Figure 3.2. Visual inspection of Figure 3.2

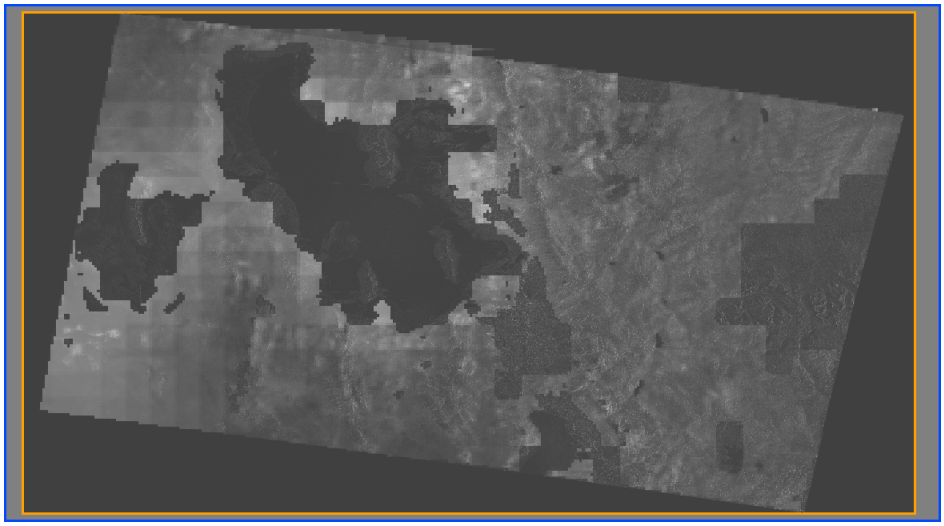


Figure 3.2: Overlap of the SMAP image shown in Figure 3.1b (foreground enclosed in the blue bounding box) on top of the SAR image shown in Figure 3.1a (background enclosed in the orange bounding box).

unveils challenges that must be addressed:

1. The SAR image extents (orange bounding box) cover a smaller geographic region than the SMAP image (blue bounding box). This can be perceived since the two images have

been overlapped based on their geographic information in QGIS and each have been set to a 50% opacity. The light grey region between the blue and orange bounding boxes shows the area not covered by the SAR image.

2. The SAR image is at a much higher resolution than the SMAP image. The difference in resolutions can be seen more clearly in Figure 3.1.

These are problems because each pixel is accompanied by a latitude and longitude coordinate and a pixel-to-pixel mapping between the two images cannot be made if $I_{x,y}$ (I denotes an arbitrary image) in the SAR image does not correspond to the same geographic location as $I_{x,y}$ in the SMAP image.

To address the differences in extents, the SAR image must be extended such that the corners align to the same latitude and longitude coordinates as the SMAP image. This is done by finding the extents of the SMAP image using GDAL and adding them to the `-te` flag of `gdalwarp` where the result is shown in Figure 3.3. The SAR and SMAP images now

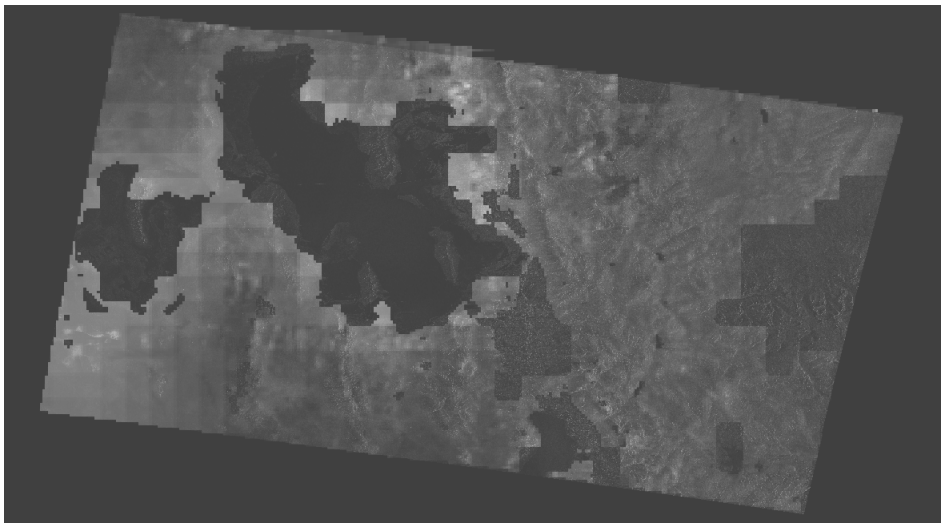
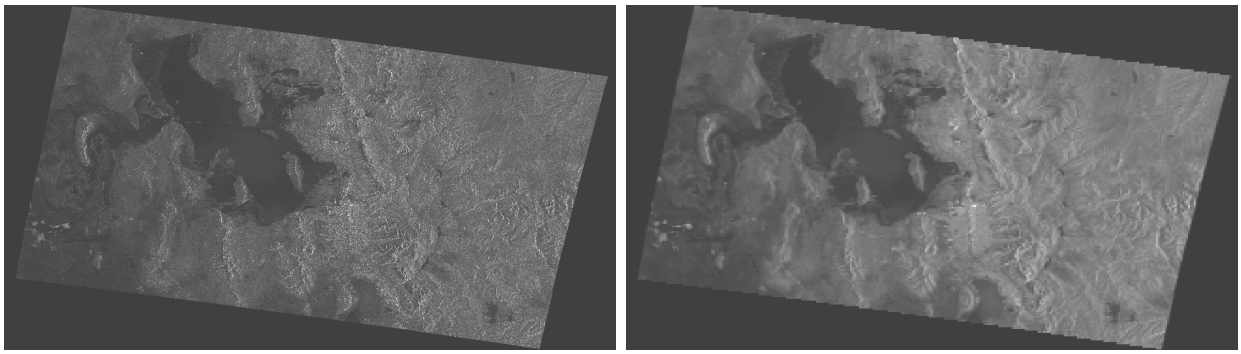


Figure 3.3: Overlap of the SMAP image shown in Figure 3.1b (foreground) on top of the newly extended SAR image (background).

cover the same spatial regions thereby eliminating the area between the blue and orange bounding boxes shown in Figure 3.2 since both images are now completely overlapping.

To resolve the differences in resolution, the SAR image is downsampled using an averaging algorithm. Downsampling can be done by adding the `-ts` flag to `gdalwarp` to specify the output dimensions along with the `-r` flag to specify the algorithm to be used. Combining all three transformations (`-te` sets the geographical extents, `-ts` sets the output resolution, and `-r` sets the downsampling algorithm) yields the final command used to produce input-output pairs for the neural network. The result of downsampling is shown in Figure 3.4.



(a) High resolution SAR image (b) Downsampled SAR image

Figure 3.4: Comparison of the raw high resolution SAR image and its downsampled variant.

A pixel-to-pixel model can be learned now that the the same geographical extents and resolutions have been given to the two images. Doing so ensures that indices x and y in the SAR and SMAP images represent the same geolocation and it also ensures that there are the same number of x and y indices in both images thereby creating a pixel-to-pixel relationship between both images. The result of preprocessing the SAR images to coincide with the SMAP images is shown by a sample input-output pair shown in Figure 3.5.

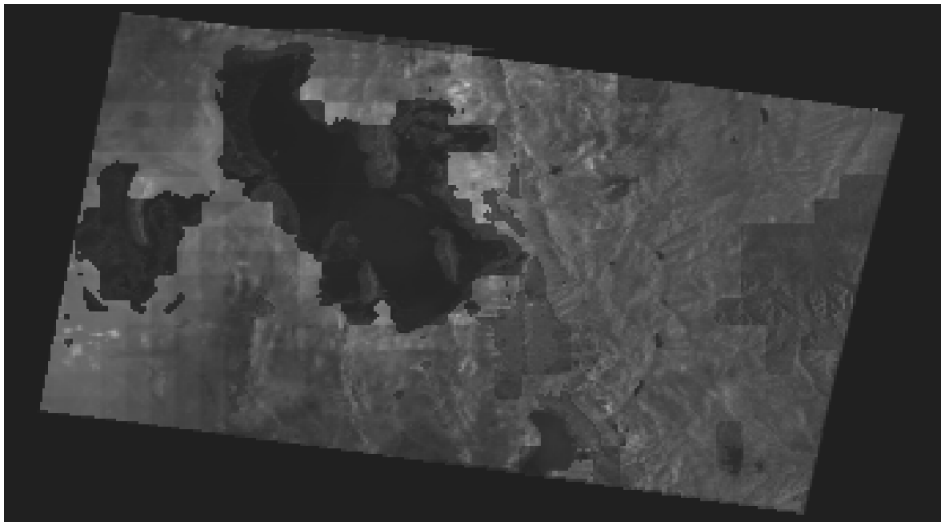


Figure 3.5: Overlap of a processed SAR image (background) having the same geographic extents and resolution as its corresponding SMAP image (foreground) for a pixel-to-pixel model.

3.1.2 Dataset Description

Several configurations of datasets and models were tested to find the parameters that gave the best results. A smaller subset of the SAR and SMAP images returned by the queried regions of Table 3.1 was first used during the development phases of this work. This subset was hand-picked and the images primarily belong to the second area of interest defined in Table 3.1, which is located in the Salt Lake City area of Utah in the United States. Some images were selected outside the areas of interest defined in Table 3.1, but were located in between areas 2 and 3. Images in the Salt Lake area were chosen to constitute the initial dataset because they overlapped with what has been visually determined to be the highest density region of in-situ ground measurement facilities shown by the ISMN [85]. This initial dataset contained 26 full resolution SAR-SMAP image pairs. The dataset appears to be relatively small for deep learning requirements, but this is due to the low download speeds in obtaining the SAR images that are each nearly 1 GB in size.

Once all the images were downloaded and preprocessed as detailed in Section 3.1.1, data augmentation was used in an attempt to solve the problem of having such a limited dataset. The data augmentation strategy adopted consisted of tiling the full resolution images into 32 × 32 pixel subsections. Tiling the geocoded images presents its own set of challenges, mainly due to the many fill value pixels located on the edges of the images. A naive approach to tiling from the corner of the image would yield many 32 × 32 subsections being completely composed of fill values, which is useless for learning. Therefore, both SAR and SMAP images are cropped such that most of the fill values located around the edges are discarded. Cropping the SAR and SMAP images can be accomplished using the GDAL command `gdal_translate` and specifying the size of the crop with the `-srcwin` flag. A crop of the SMAP image is shown in Figure 3.6.

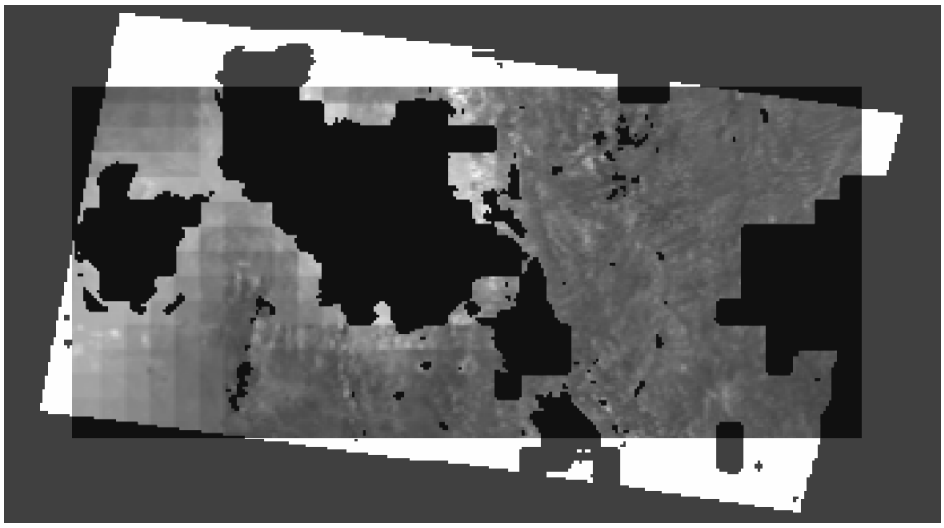


Figure 3.6: Cropped SMAP image is displayed on the foreground of its uncropped version that is set to a black and white image.

The cropped image resolution was chosen to have a width of $32 \times 9 = 288$ and a height of $32 \times 4 = 128$ so it could be tiled evenly into $9 \times 4 = 36$ tiles. The cropped image is then

tiled to yield 32 × 32 pixel subsections with GDAL's `gdal_retile` script using the `-ps` and `-overlap` flags, which set the subsection dimensions and overlap between tiles respectively. The retiling process spans the entire cropped image to produce 36 tiles for each of the 26 images thereby producing $36 \times 26 = 936$ tiles, which is referred herein as Batch 0. A few tiles were chosen for illustration purposes in Figure 3.7 and are highlighted in white.

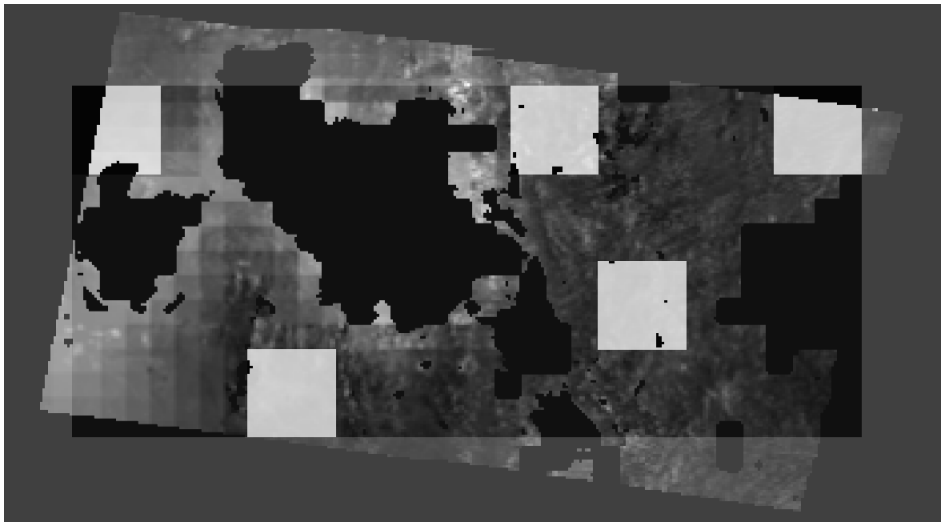


Figure 3.7: Sample tiles highlighted in black and white are extracted from the cropped SMAP image for data augmentation.

One issue that leads to two different dataset configurations is the presence of fill value pixels. These fill value pixels represent areas where no data, or noisy or unusable data, was obtained by the Sentinel-1 and SMAP satellites. From a deep learning perspective, it is unclear how these fill value pixels should be treated. Including samples that contain fill values raises issues with how the loss, and therefore the gradients, are calculated. Fill value pixels, if included in the dataset, will contribute to the loss, but should not as they represent areas of no data. However, using a masking strategy when the loss is calculated proves to be difficult as the position of the fill values from image to image are not consistent and will

therefore impact the gradients. For these reasons, two dataset configurations are used in this work, one where no samples in the dataset include fill values, and the other where fill values are included and the neural network is left to learn how to identify and predict fill values. Removing the samples that contain fill values from the initial dataset of 936 tiles leaves only 160 samples remaining that do not contain fill values, which is a considerable reduction in the number of samples in the dataset. This dataset of 160 samples is referred to as Batch 1.

Careful treatment is needed for the dataset of 936 tiles that contains samples with fill values (Batch 0) since fill values must not to be considered in the performance metrics as they are not valid data. Treatment of datasets containing samples with fill values can be accomplished with classification terms such as true positives, true negatives, false negatives, and true negatives. Negative pixel values identify fill values in the SMAP images and these can be used to classify pixels into the four groups, where positive pixel values represent good data. True positives (TP) refer to true pixels being positive and the predicted pixels being positive. False positives (FP) refer to true pixels being negative (fill value) and the predicted pixels being positive. False negatives (FN) refer to true pixels being positive and the predicted pixels being negative. Lastly, true negatives (TN) refer to true pixels being negative and the predicted pixels being negative. These results can be summarized using a confusion matrix [86], a common way to show the performance of classification problems.

In addition to the initial dataset of 26 images that was used for testing, significant effort went towards the further development of a script provided by the Copernicus program to automatically download SAR images to obtain a larger dataset. This larger dataset spans the areas of interest summarized in Table 3.1 and consists of 1,034 SAR and SMAP images, which is referred to as Batch 4 (Batch 2 and 3 are skipped for reasons that are

explained later). Tiling this dataset into 32 × 32 subsections produces a total of 45,622 tiles. Alternatively, another configuration can be created by varying the size of the subsections. It was found that the largest crop size that could be made among all SAR and SMAP images was 144 x 288. One of the benefits to increasing the subsection size to cover nearly the entire image is that it allows for the use of larger neural networks. A single 144 x 288 sample is taken from each image, therefore there are 1,034 samples for this maximal crop size dataset.

Throughout the work done for the soil moisture application, a 90%-10% split is used for the training and testing sets respectively. A summary of the datasets that have been described so far along with their training and testing split are shown in Table 3.2.

Table 3.2: Dataset Summary.

Data Batch	Images	Tiles	Training Samples	Testing samples
Batch 0	26	936	842	94
Batch 1	26	160	144	16
Batch 4	1,034	45,622	41,059	4,563
Batch 4*	1,034	1,034	930	104

* Maximal crop sizes.

In order to better organize the datasets and models used in the experiments conducted for this thesis, a total of 7 configurable parameters were formulated. These 7 parameters are described and itemized below and their values are summarized in Table 3.3.

- **Data Batch** – indicates the SAR and SMAP images to use as part of the dataset as well as whether or not the samples include fill values or not. Batch 0 refers to the initial dataset of 26 images that have been tiled to generate 936 samples. Batch 1 refers to same 26 images as Batch 0, except 776 samples have been removed from the 936 samples, thereby leaving 160 samples that do not contain any fill values. Batches

2 and 3 refer to the exact same datasets as Batches 0 and 1 respectively, but Batch 2 and 3 were stored on a different machine to utilize a GPU for training, while Batch 0 and 1 were store on a machine that was only setup to train on a CPU. Batch 4 refers to the dataset of 1,034 images whose samples include fill values.

- **Tile Size** – height and width of the samples used in the dataset. Values of 32, 24, 48 were experimented with and a max crop of 144 × 288 is denoted by "M".

- **Overlap** – indicates the percentage of overlap between subsequent tiles. The overlap values that were experimented with was 0%, 25%, 50%, and 75%, however overlaps of 25% - 75% did not produce better results than experiments with non-overlapping tiles and are therefore not shown in the results.

- **Model** – several models were tested. In the beginning of the project, ad hoc models were created to learn about various layers and occupied values 0 - 3 which are not shown here. Then a simple model consisting of 3 dense layers and several U-Net derivations were evaluated (U-Net$_{0-3}$) These U-Net derivations begin by adapting some features of U-Net, one at a time, until the full U-Net architecture is used.

- **Loss** – several variations of the absolute error and the squared error loss functions were studied, which included masked and normalized versions. However, the mean squared error loss function is the sole loss function used in the reported results.

- **Training Controls** – the training controls (T_c) identifies the values used for the batch size, the number of epochs, the early stopping patience, and the patience used to determine when to reduce the learning rate. There is no uniform set of values for this parameter as they were determined on a case by case basis, therefore there is no

consistent value for T_c that is used across experiment identifiers.

- **Normalized** – indicates whether or not the data is z-score normalized or not. Z-score normalization consists of setting the mean of the features in the dataset to 0 and the variance to 1.

A summary of the various configurations can be found in Table 3.3 in order to organize and identify experimental results based on the configuration of the dataset and models used to produce them. For example, the seven digit code 4MMA411 decodes to an experiment

Table 3.3: Dataset configuration naming map.

	1 Data Batch	2 Tile Size	3 Overlap	4 Model	5 Loss	6 T_c	7 Normalized
0	–	32	0	–	–	–	No
1	Batch 1	24	25	–	–	–	Yes
2	Batch 2	48	50	–	–	–	–
3	–	–	75	–	–	–	–
4	Batch 4	–	–	Dense	MSE	–	–
5	–	–	–	U-Net$_0$	–	–	–
6	–	–	–	U-Net$_1$	–	–	–
7	–	–	–	U-Net$_2$	–	–	–
8	–	–	–	U-Net$_3$	–	–	–
9	–	–	–	–	–	–	–
A	–	–	–	U-Net	–	–	–
M	–	Max	Max	–	–	–	–

with data from Batch 4 (4) with maximally (M) cropped samples of 144 $\times$ 288 and since there is only 1 sample per image, the concept of an overlap other samples does not exist and therefore is denoted by an (M) as well. The model used in this example is U-Net (A) with MSE loss (4) and the second set of training controls are used (1). Lastly, the data for this experiment is z-score normalized (1). Unpopulated entries in Table 3.3 denote parameter values that are either not shown in the results or are unoccupied.

3.2 Results

In this section, results from 8 different experiments are shown. These experiments appear in order and are identified as follows: 1004411, 1005411, 1006411, 1007411, 2007411, 4007411, 4MM7411, 4MMA411. While the subsequent sections 3.2.1 - 3.2.8 are repetitive in nature, these experiments are shown to illustrate the improvements that are made by increasing the number of samples in the dataset with which to train the neural networks as well as the effects of increasing the image sizes and using deeper and more complex networks to increase the receptive field. An outline of the expected metrics and figures to expect for each experiment in the following sections is made here.

A diagram of the network used for each experiment is shown along with a description of its features and a comparison to previous networks. Each model in the experiments are trained with the same loss function, which is the Mean Squared Error between each pixel in the target soil moisture image and the predicted soil moisture image. The MSE was introduced in Section 2.2.6 and is reproduced here for convenience:

$$E = \frac{||\mathbf{y}_{pred} - \mathbf{y}_{true}||^2}{N}. \tag{3.1}$$

A historical plot of the training loss and testing loss across epochs for each experiment is also given in the subsequent sections.

Each model is also evaluated with the correlation coefficient shown in Equation 3.2. In this equation, x represents y_{pred} and y represents y_{true} for simplicity, and μ_x and μ_y are the means of x and y respectively. The covariance of x and y is given by $cov(x, y)$, $\mathbb{E}$ is the expectation, and σ_x and σ_y is the standard deviation of x and y. The loss and correlation

for both the training and testing set are summarized in a table for each experiment.

$$\begin{aligned} R_{x,y} &= \frac{cov(x,y)}{\sigma_x \sigma_y} \\ R_{x,y} &= \frac{\mathbb{E}[(x-\mu_x)(y-\mu_y)]}{\sigma_x \sigma_y} \\ R_{x,y} &= \frac{\sum_{i=1}^{n}(x_i-\mu_x)(y_i-\mu_y)}{\sqrt{\sum_{i=1}^{n}(x_i-\mu_x)^2}\sqrt{\sum_{i=1}^{n}(y_i-\mu_y)^2}}. \end{aligned} \tag{3.2}$$

A visual way to represent the correlation is through a scatter plot, where an example is shown in Figure 3.8. This plot illustrates random data and an ideal line where all points

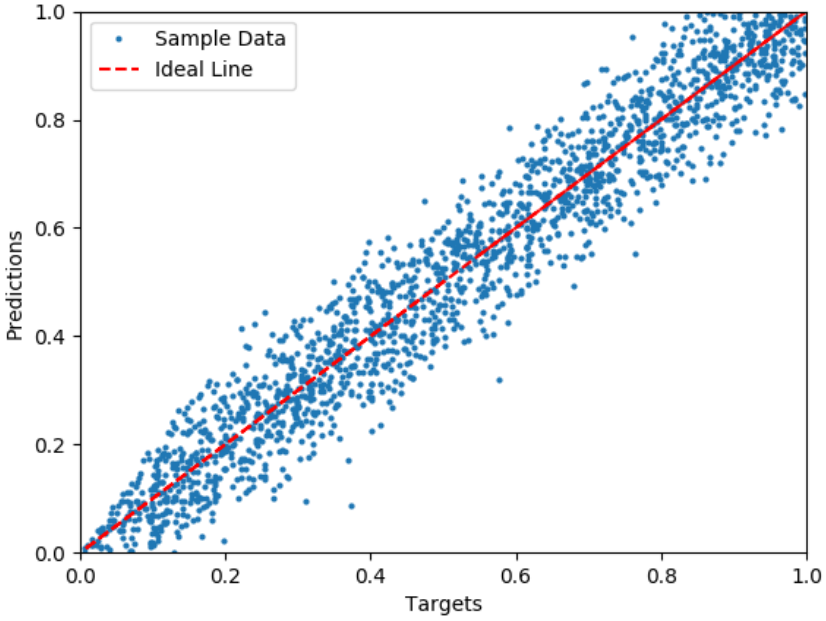


Figure 3.8: Sample scatter plot of random data

must reside to achieve a correlation of 1. Each point in the scatter plots presented in the subsequent sections are based on each pixel's intensity value from the target SMAP image as the independent variable and the corresponding pixel intensity value from the predicted SMAP image as the dependent variable. If all predicted pixels are equal to the target pixels,

then the error is zero and the points on the scatter plot will form a straight 45° line as shown by the red ideal line in Figure 3.8. Two scatter plots are shown to determine how well the model performs on the training set in comparison to the testing set. The plots consider every pixel from every image in the training or testing set as individual data point.

Another way to view the pixel-to-pixel error is through a tolerance plot. A sample tolerance plot is shown in Figure 3.9 for the same random data used to produce the scatter plot shown in Figure 3.8. The tolerance plot is created by first finding the relative error ε_r

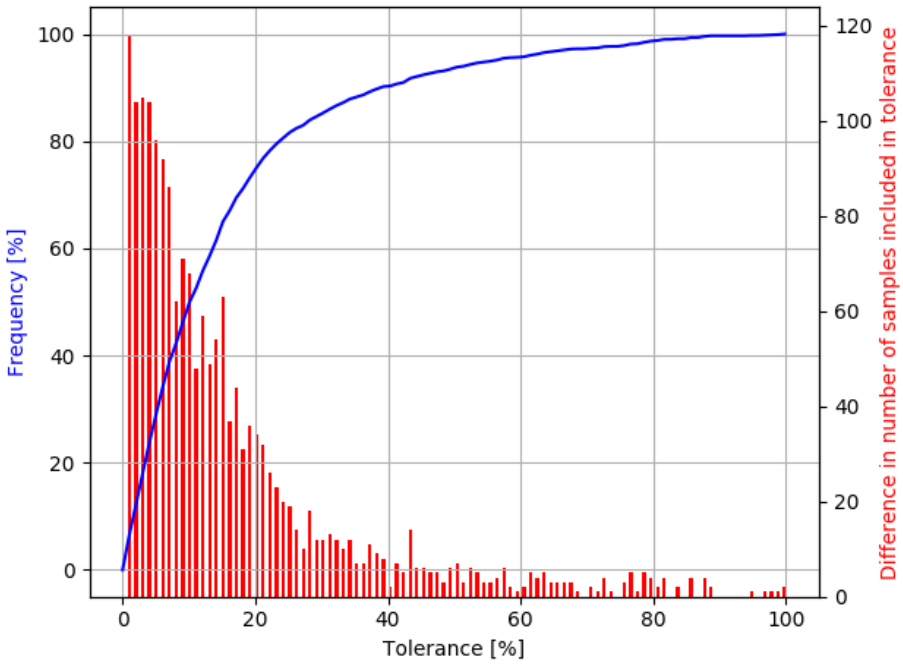


Figure 3.9: Sample tolerance plot of the same random data used in Figure 3.8

between each target pixel intensity and predicted pixel intensity. Given the array of ε_r, a count of the number of ε_r instances that are less than a varying tolerance level is acquired. The term "tolerance" is used here to represent a gauge of the relative error. The varying tolerance levels are determined simply with a linearly spaced array of 100 elements that range

between 0 and the maximum ε_r. The tolerance levels are plotted as the independent variable and the frequency of ε_r elements appearing below a specific tolerance level is plotted as the dependent variable. The frequency is reported as a percentage of the number of ε_r elements below a tolerance level relative to the entire set of ε_r elements (total number of pixels in the dataset). 100% of the ε_r elements will be captured once the tolerance level increases to be equal to the maximum ε_r. Additionally, a bar chart is superimposed on the tolerance plot to show the difference in the count of ε_r appearing below subsequent tolerance levels. Using the tolerance plot shown in Figure 3.9 as an example while observing the blue curve, it can seen that 75% of the predicted pixels have a tolerance <20%, 90% of the predicted pixels have a tolerance <40%, 95% of the predicted pixels have a tolerance <60% and so on. Interpreting the red bar chart in Figure 3.9 can be accomplished by first looking at the first bar to the left. This bar signifies that there are 118 ε_r elements that have been included once the tolerance increases from 0 to the next level. The next bar to the right shows that 104 additional ε_r elements have been included when the tolerance increases again. This process continues until there are only 2 additional ε_r elements that have been included when the tolerance increases to 100%. In other words, using a fictitious example unrelated to Figure 3.9, if 100 ε_r elements appear below a 10% tolerance and 150 ε_r elements appear below a 15% tolerance, there will therefore be $150 - 100 = 50$ additional ε_r elements appearing between the 10-15% tolerance and the bar describing this range would have a height of 50 units.

Finally, a mosaic of four samples are displayed in four quadrants, one from the training set in the upper-left corner, and three from the testing set in the remaining corners. For each sample, three images are displayed in the respective quadrant. The first is the input SAR sample displayed on the left, the second is the target SMAP image displayed on the top right, and the third is the predicted SMAP image on the bottom right.

3.2.1 Experiment 1004411

The first experiment consists of the small dataset of 160 tiles of size 32 × 32 that do not include any samples containing fill values. Of these 160 tiles, 144 were used to train the first network presented here, which is a simple fully connected neural network with three layers as shown in Figure 3.10. There are a total of 3,148,800 trainable parameters in this model. The shape of the input is flattened to a 1,024 sized vector given that all inputs connect to all

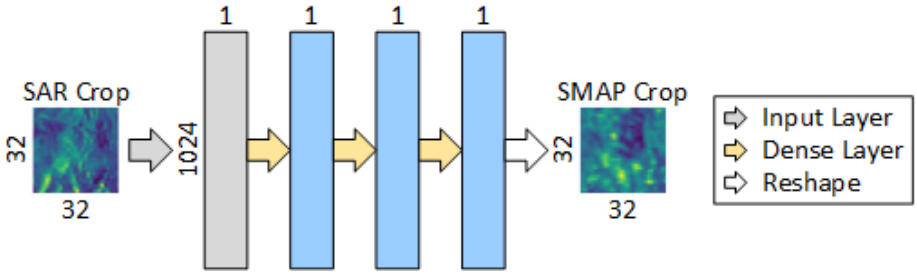


Figure 3.10: Densely Connected Network Architecture

neurons in the dense layer thereby making the locality of the pixels in an image irrelevant. This is indicated by the 1,024 input shape of a 32 × 32 sized tile, which contains 1,024 pixels, being flattened into a one dimensional vector. For this reason, 1,024 neurons are chosen to compose each dense layer. For an image-to-image model to work, the last layer outputs a vector having a shape of 1,024 so it can be reshaped into a 32 × 32 image.

The history of the loss during training of this densely connected network is shown in Figure 3.11. It can be seen that the training loss converges rapidly after approximately 15 epochs, as does the testing loss, then both curves flatten out and remain constant for the remainder of the training. This experiment was conducted without any dynamic control over the early stopping patience or the reduction in the learning rate, thereby showing that 600 epochs is more than sufficient to train this model on the given dataset.

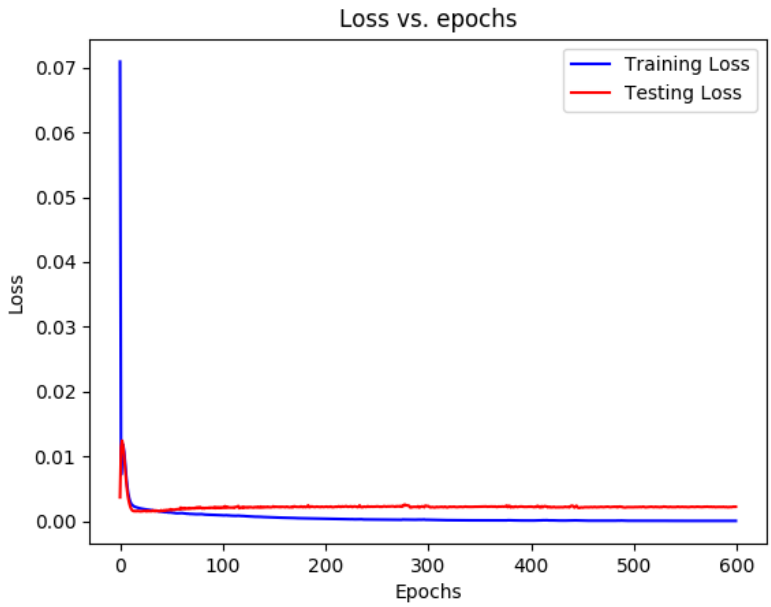


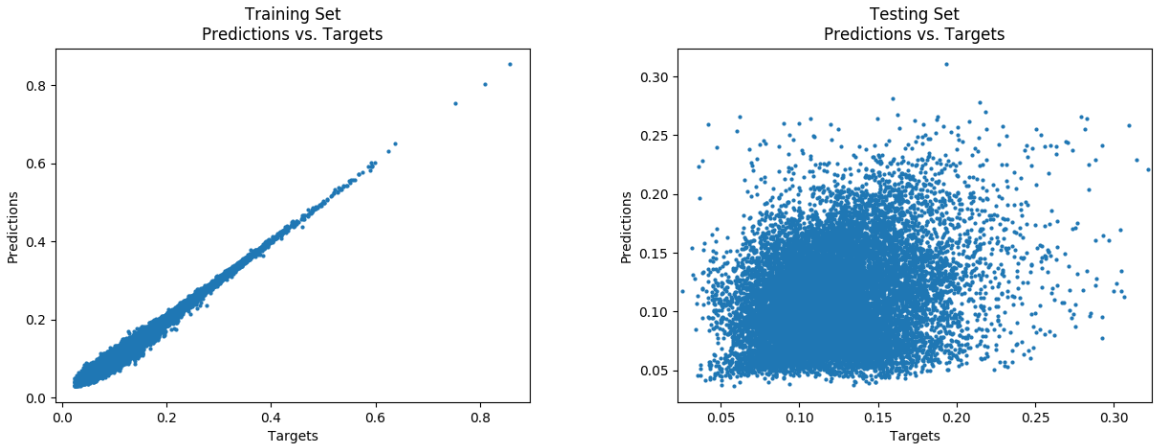
Figure 3.11: Densely Connected Network Training Loss

A summary of the loss and correlation results for the Dense Model is shown in Table 3.4. The loss remains low for both the training and testing sets, which leads one to believe that the network did well in reproducing the SMAP images. However, the correlation is near perfect on the training set, but does not translate at all to the testing set. This is indicative of overfitting and regularization is needed for better generalization to the testing set.

Table 3.4: Dense Model Results

	Training	Testing
Loss	3.77e-05	2.22e-03
R	0.992	0.267

The scatter plots of the target output pixel intensity in relation to the predicted pixel intensity for both the training and testing sets are shown in 3.12. The high correlation in the training set scatter plot is clearly visible, but unfortunately the scatter plot generated by the

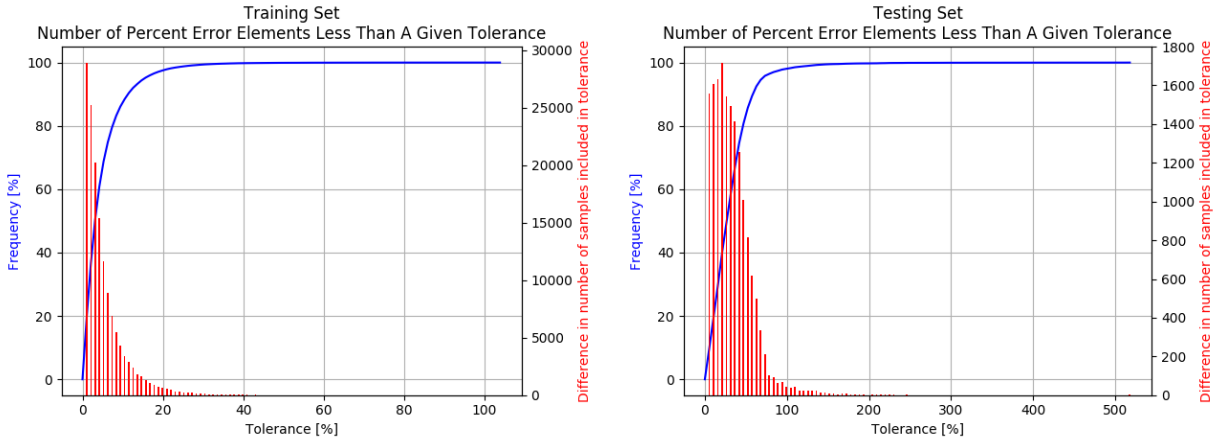


(a) Training Set Scatter Plot (b) Testing Set Scatter Plot

Figure 3.12: Scatter Plots for Experiment 1004411

Dense Model in Figure 3.12b appears to do no better than a random normal distribution.

The tolerance plots for the frequency of relative error elements produced by the Dense Model that occur below a given tolerance level is shown for both training and testing sets in Figure 3.13. The tolerance plot for the Dense Model in Figure 3.13b shows that

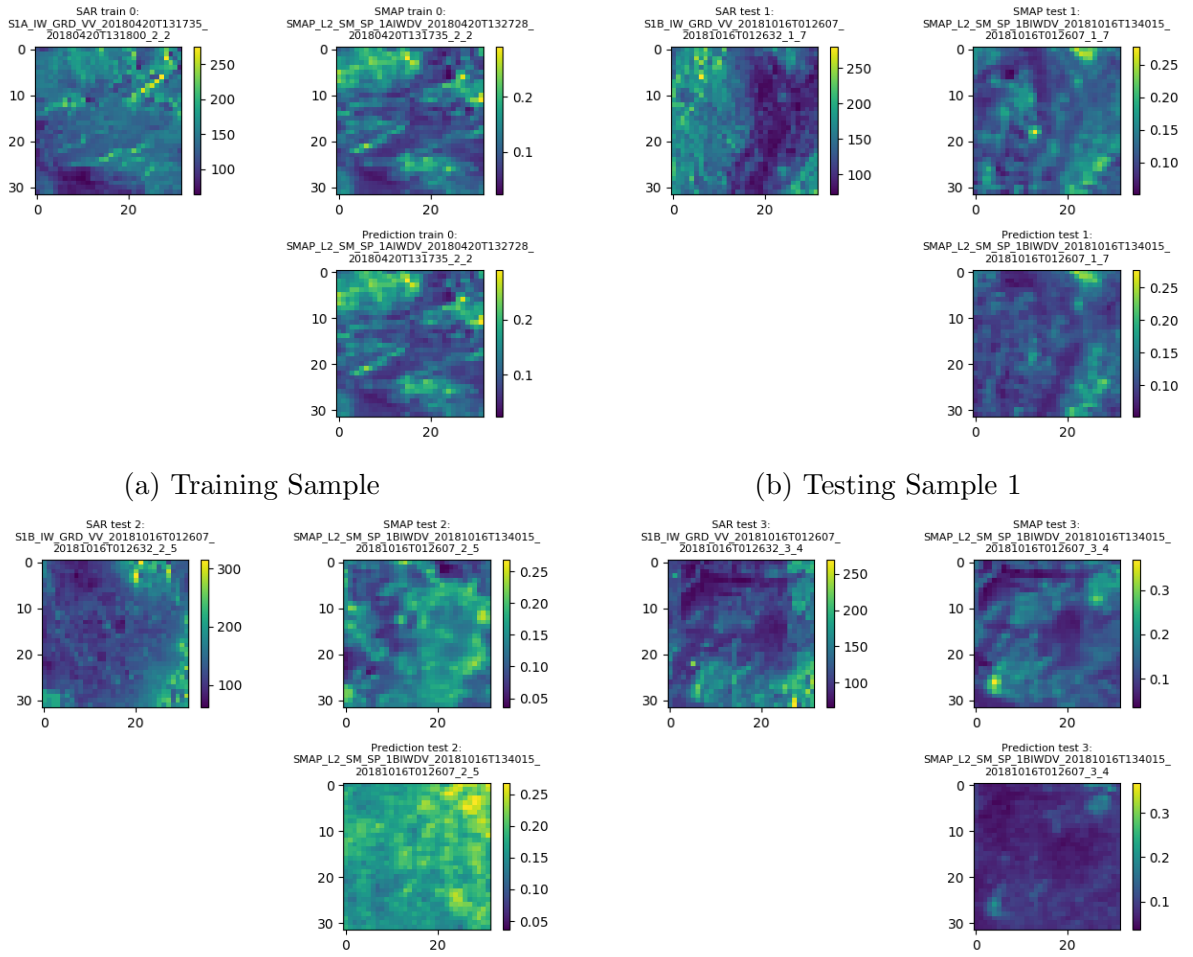


(a) Training Set Tolerance Plot (b) Testing Set Tolerance Plot

Figure 3.13: Tolerance Plots for Experiment 1004411

approximately 80% of the relative error elements have a relative error <50% while the largest relative error is approximately 500%.

A few predictions from experiment 1004411 are shown in Figure 3.14 for visualization.



(a) Training Sample

(b) Testing Sample 1

(c) Testing Sample 2

(d) Testing Sample 3

Figure 3.14: Four predictions conducted during experiment 1004411

3.2.2 Experiment 1005411

The results achieved by experiment 1004411 suggest that some form of regularization is needed. The second network that was built is an adaptation of the popular U-Net [87] architecture and is based on convolution layers. As previously mentioned in Section 2.2, convolution layers can be thought of as a regularized form of a densely connected layer since many connections are dropped out. The adapted architecture is shown in Figure 3.15 and is denoted as U-Net$_0$. The number of total trainable parameters in this model is 184,961, which is significantly less than the 3 million parameters of the Dense model seen previously. The major feature of U-Net that is maintained in this model is the downsampling and re-

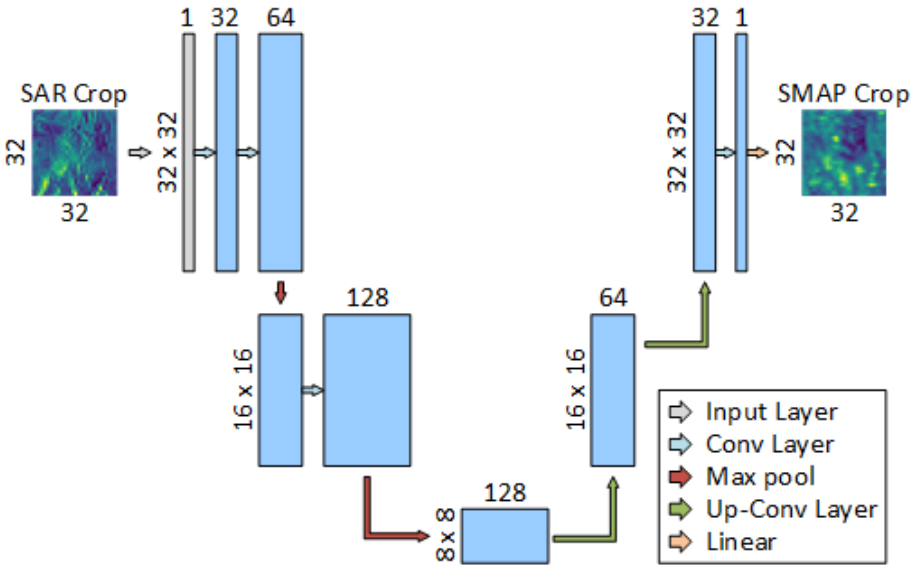


Figure 3.15: U-Net$_0$ Architecture

upsampling methodology that gives the network its "U" shape. Max pooling layers are used for downsampling, and upsampling is accomplished with transpose convolution layers. Another feature that is maintained from U-Net is that the number of filters for each layer doubles after every maxpooling layer and halves after every deconvolutional layer.

The history of the loss during training of U-Net$_0$ is shown in Figure 3.16. Here the training loss does not converge as rapidly as before with the Dense model as it takes approximately 100 epochs before the training loss begins to settle to a constant. The testing loss however quickly finds a bottom and remains constant for 40 epochs then slowly increases for the remainder of the training. Once again, this experiment did not use any dynamic controls over the training procedure and 200 or 300 epochs would have been sufficient.

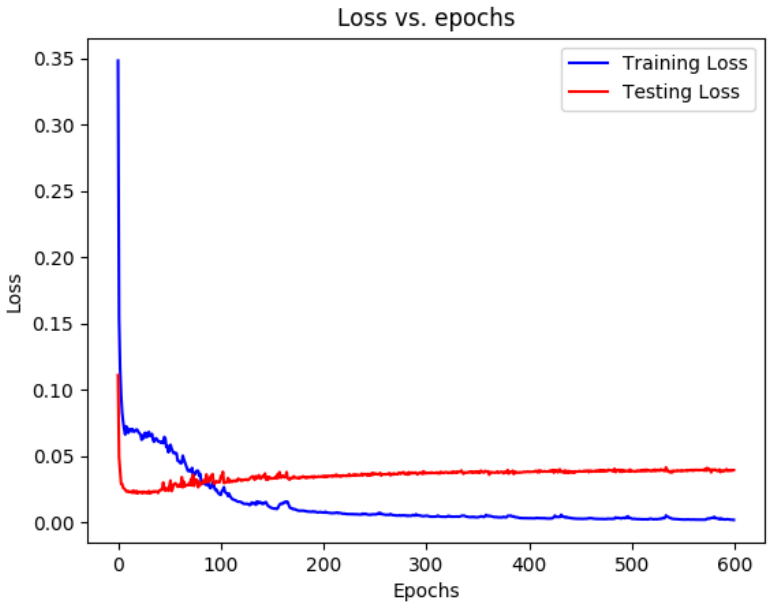


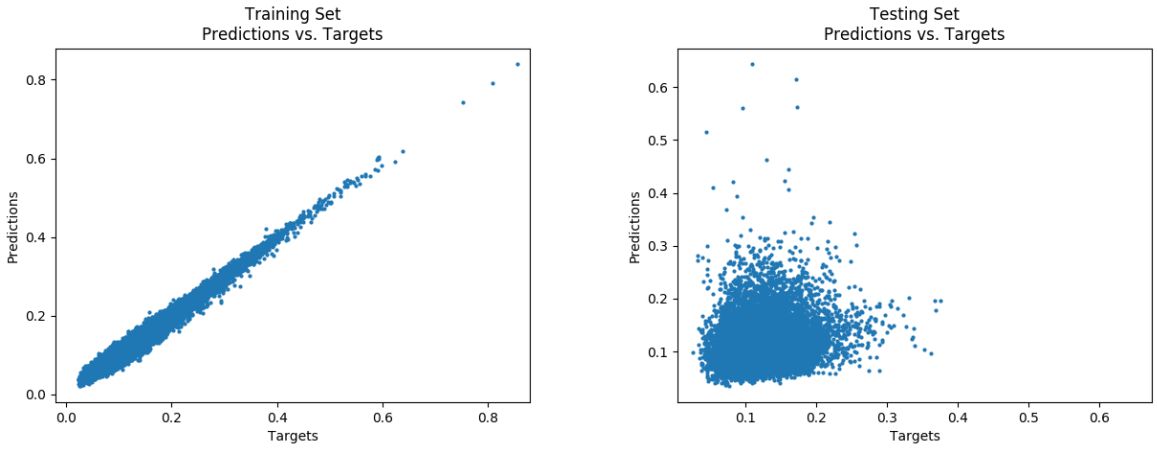
Figure 3.16: U-Net$_0$ Training Loss

A summary of the loss and correlation results for U-Net$_0$ is shown in Table 3.5. Comparatively to the Dense model, U-Net$_0$ has a higher training and testing loss, both are still relatively good though, but the correlation suffers on the testing set. This is demonstrative of substantial overfitting, and the discrepancy in the testing correlations between the two models could also be explained by the large difference between the number of training parameters between the two models.

Table 3.5: UNet-Like Model Results

	Training	Testing
Loss	2.00e-03	3.94e-02
R	0.986	0.162

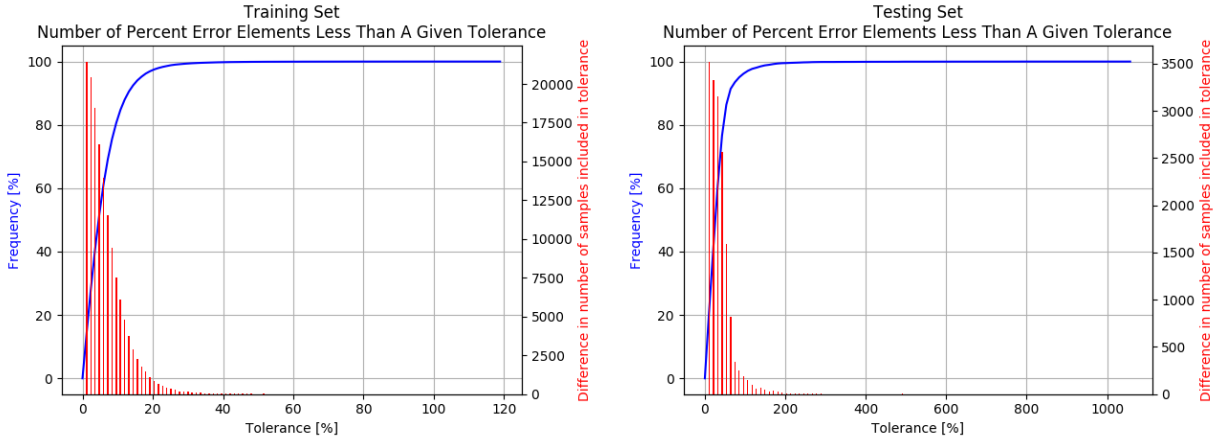
The scatter plots of the target output pixel intensity in relation to the predicted pixel intensity made by U-Net$_0$ for both the training and testing set are shown in 3.17. The testing scatter plot generated by U-Net$_0$ appears to do worse than a random normal distribution because some predictions are >0.5 while the target values are within the 0.05 - 0.2 range. This is demonstrative of large variance in the predictions.



(a) Training Set Scatter Plot (b) Testing Set Scatter Plot

Figure 3.17: Scatter Plots for Experiment 1005411

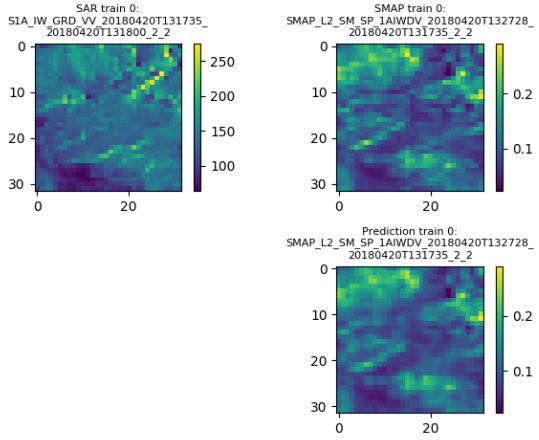
The tolerance plots for the frequency of relative error elements produced by U-Net$_0$ that occur below a given tolerance level is shown for both training and testing sets in Figure 3.18. Large errors are reflected in the tolerance plot in Figure 3.18b with the highest tolerance reaching approximately 1000%. This is double the largest error produced by the Dense model of experiment 1004411. However, approximately 80% of the relative error elements are <50% indicating that the bulk of the estimates are in line with the Dense Model predictions.



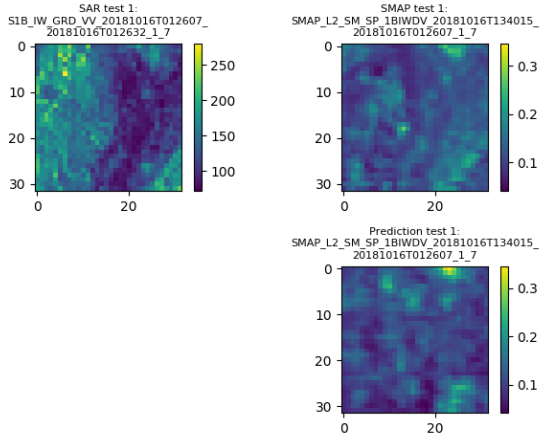
(a) Training Set Tolerance Plot (b) Testing Set Tolerance Plot

Figure 3.18: Tolerance Plots for Experiment 1005411

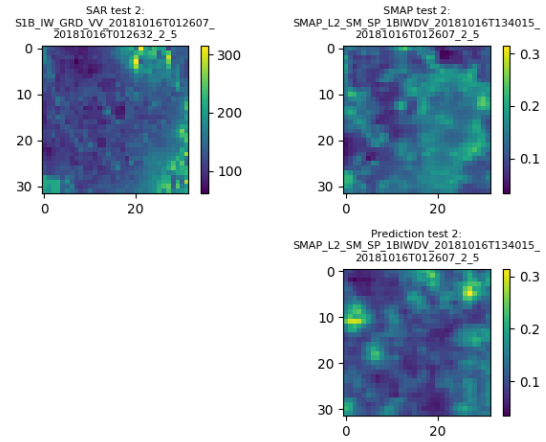
A few predictions from experiment 1005411 are shown in Figure 3.19 for visualization.



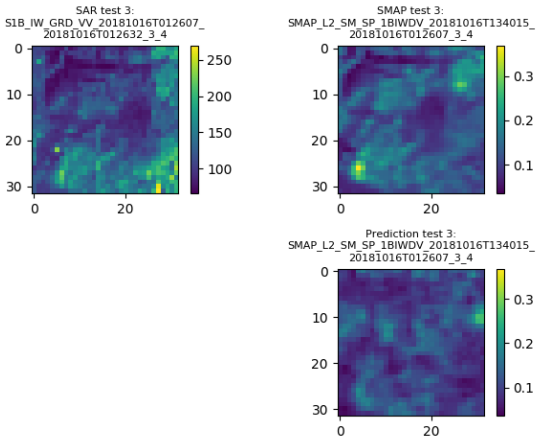
(a) Training Sample



(b) Testing Sample 1



(c) Testing Sample 2



(d) Testing Sample 3

Figure 3.19: Four predictions conducted during experiment 1005411

3.2.3 Experiment 1006411

In an effort to increase the correlation on the testing set, experiment 1006411 attempts to build on U-Net$_0$ by simply adding regularization as shown in Figure 3.20. The regularization technique used here is via the addition of dropout layers (weights in the layer are randomly set to 0) subsequent to convolutional layers that occur before max pooling layers or after transpose convolutions. This change is reflected in Figure 3.20 with purple arrows and this model is denoted as U-Net$_1$. Since dropout layers do not contain trainable parameters, U-Net$_1$ has the same 184,961 trainable parameters as U-Net$_0$.

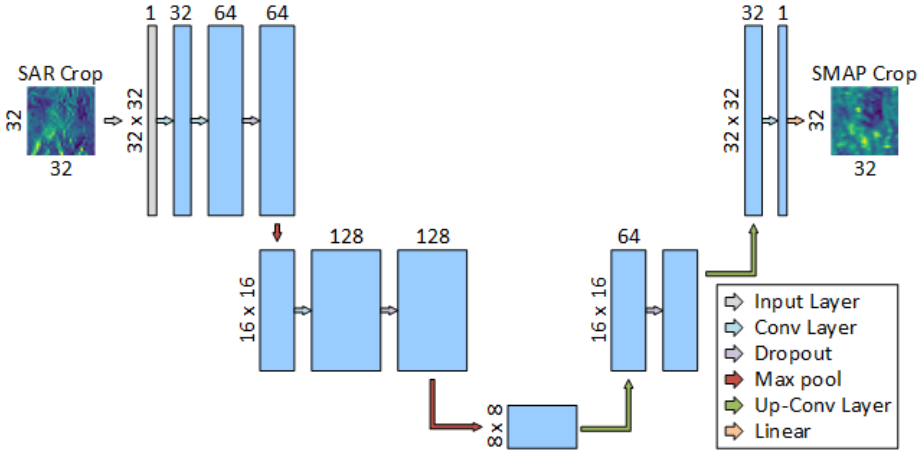


Figure 3.20: U-Net$_1$ Architecture

The history of the loss during training of U-Net$_1$ is shown in Figure 3.21. The training history is very similar to experiment 1005411 as would be expected since both models are the same except for the addition of dropout layers. One visual difference that can be seen by comparing to the loss plot produced by experiment 1005411 is in the amount of jitter experienced by the testing loss. This can be attributed to the dropout layer since connections are randomly dropped at evaluation time and the links that are dropped are different at each

evaluation. The training procedure was kept the same as in experiment 1005411, but 200 or 300 epochs would have been sufficient since the testing loss remains steady at that point.

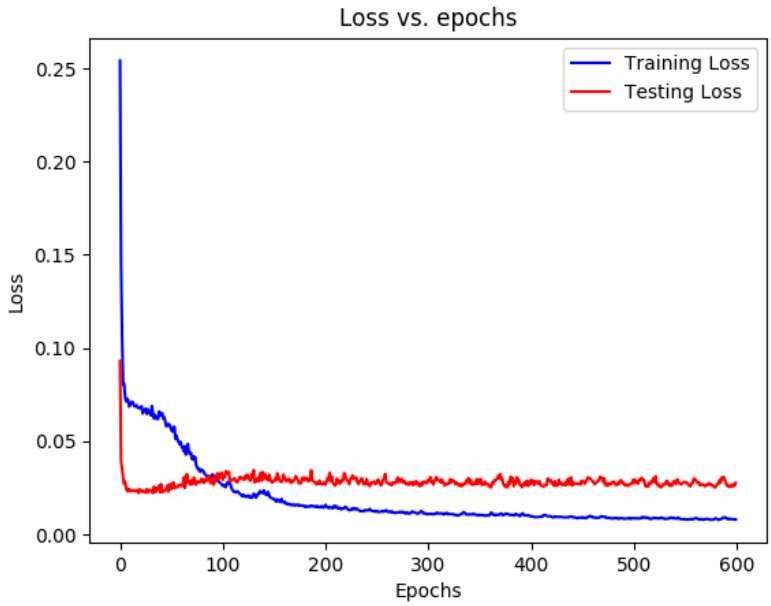


Figure 3.21: U-Net$_1$ Training Loss

A summary of the loss and correlation results for U-Net$_1$ is shown in Table 3.6 and as one would expect, there is a slight improvement to the loss of the testing set over experiment 1005411 even though both results are in the same order of magnitude. The testing correlation

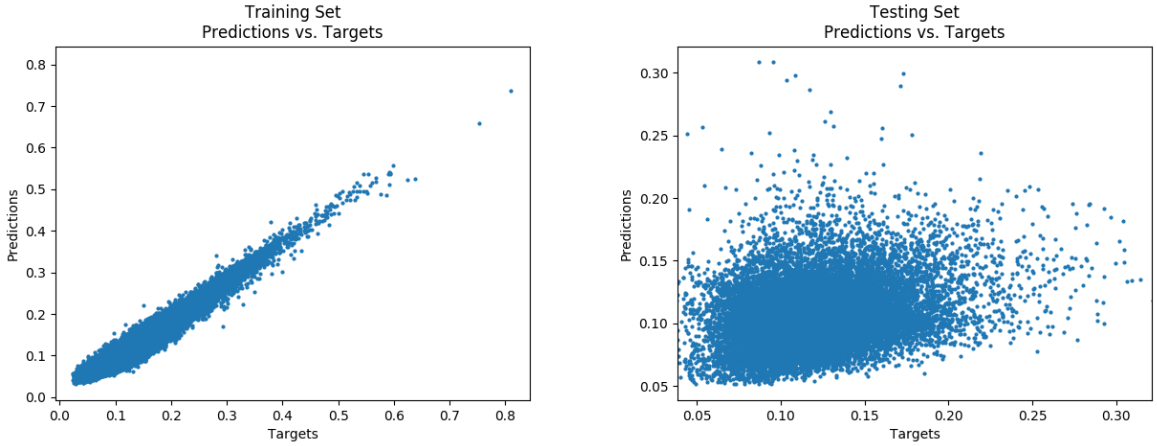
Table 3.6: U-Net-Like Model with Regularization Results

	Training	Testing
Loss	6.19e-03	2.76e-02
R	0.971	0.271

has also recovered from experiment 1005411 and U-Net$_1$ reaches similar correlation values to experiment 1004411. This suggests that the regularization technique did in fact reduce

the overfitting that was exhibited by U-Net$_0$ and U-Net$_1$ is able to achieve similar results to the Dense model of experiment 1004411 with a fraction of the trainable parameters.

The scatter plots of the target output pixel intensity in relation to the predicted pixel intensity made by U-Net$_1$ for both the training and testing set are shown in 3.22. Visually

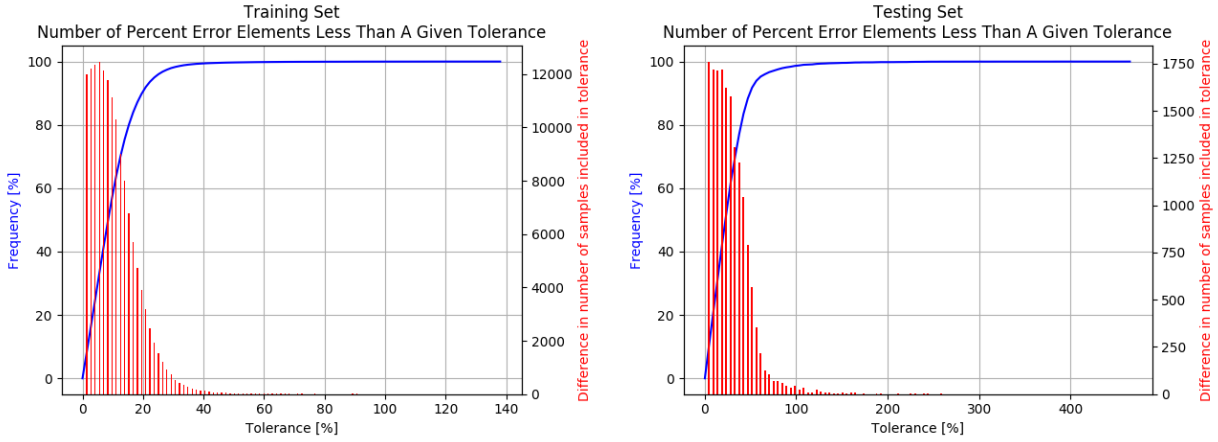


(a) Training Set Scatter Plot (b) Testing Set Scatter Plot

Figure 3.22: Scatter Plots for Experiment 1006411

inspecting Figure 3.22b shows that U-Net$_1$ tends to estimate the predicted pixel intensity to be lower than than the target pixel value. However, the variance in the scatter plot has decreased compared to experiment 1005411. This is supported by the tolerance plot.

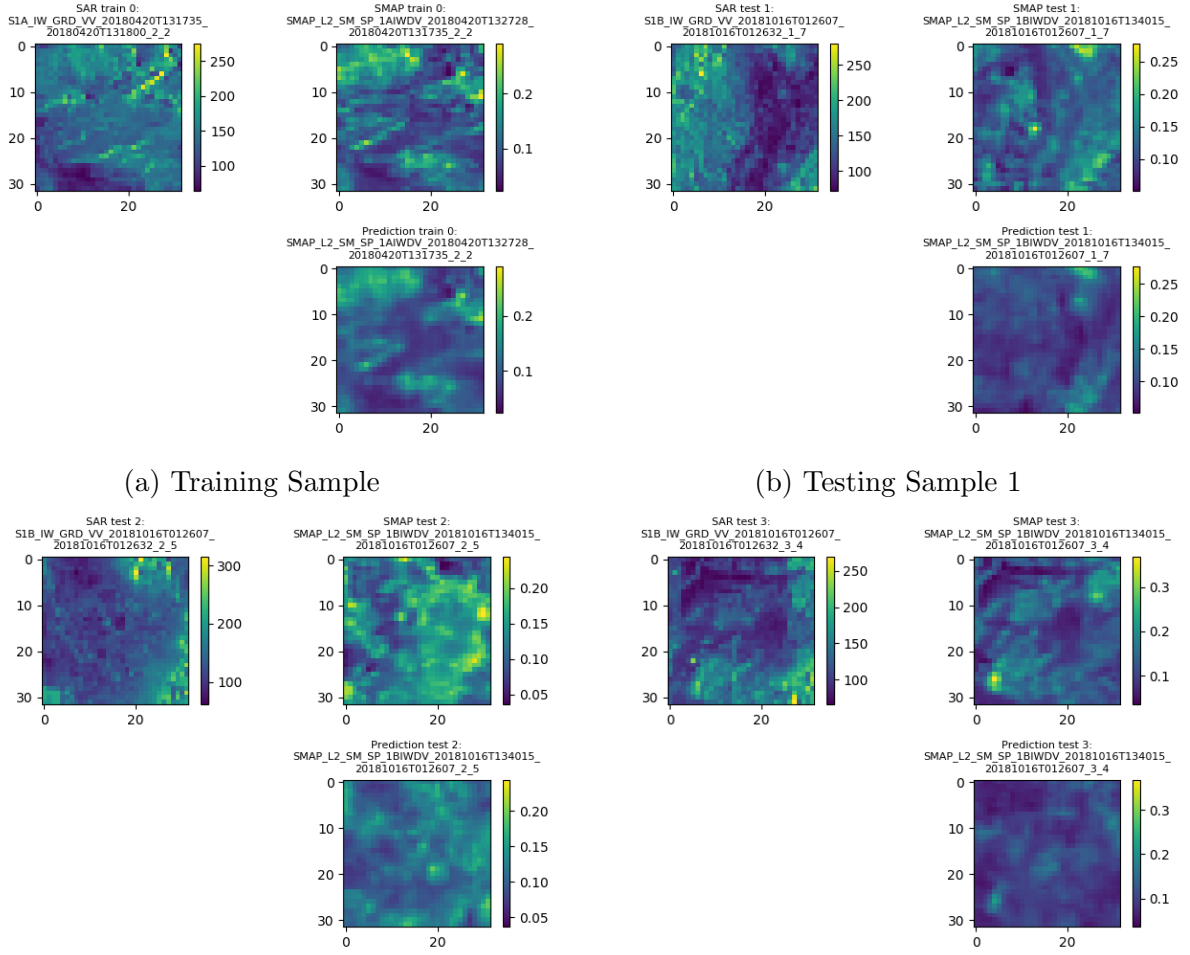
The tolerance plots for the frequency of relative error elements produced by U-Net$_1$ that occur below a given tolerance level is shown for both training and testing sets in Figure 3.23. This model seems to control the variance of the predictions better than U-Net$_0$ because the largest tolerance per Figure 3.23b is approximately 450%, less than half that of the U-Net$_0$, and comparable to the Dense model. Once again, U-Net$_1$ continues the trend of having approximately 80% of the relative error elements fall below a 50% relative error.



(a) Training Set Tolerance Plot (b) Testing Set Tolerance Plot

Figure 3.23: Tolerance Plots for Experiment 1006411

A few predictions from experiment 1006411 are shown in Figure 3.24 for visualization.



(a) Training Sample

(b) Testing Sample 1

(c) Testing Sample 2

(d) Testing Sample 3

Figure 3.24: Four predictions conducted during experiment 1006411

3.2.4 Experiment 1007411

The U-Net$_0$ and U-Net$_1$ derivations omit an important feature of U-Net called skip connections, which are introduced here in the U-Net$_2$ derivation illustrated in Figure 3.25. Two convolution layers are also used at each level of the architecture to increase the receptive field of each level, just as in the original U-Net architecture. This is different from the single convolution layer used at each level in the architectures seen in experiments 1005411 and 1006411. Additionally, no dropout layers are used here in order to compare the effects of the skip connections to experiment 1005411, which had none. These modifications result in an increase to 2,066,497 trainable parameters found in U-Net$_2$.

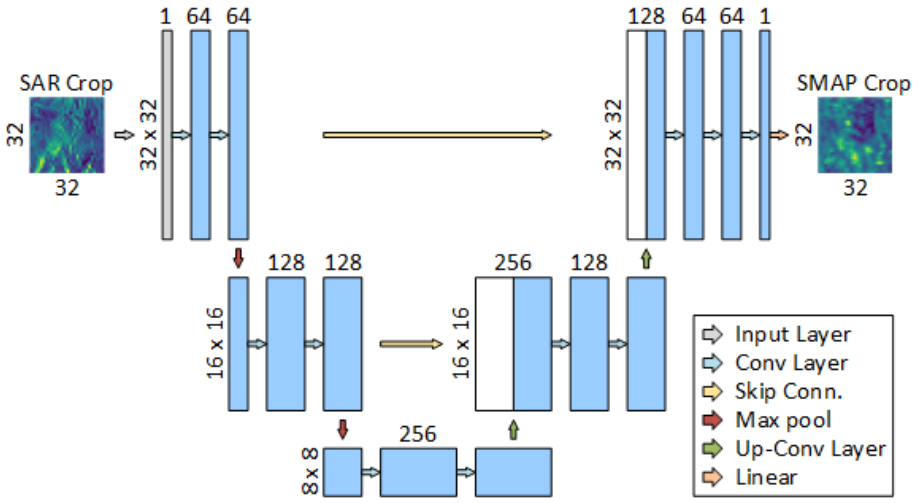


Figure 3.25: U-Net$_2$ Architecture

The history of the loss during training of U-Net$_2$ is shown in Figure 3.26. One factor to notice is that the loss history for this experiment is on a comparable order of magnitude as experiment 1004411, which is 10x less than experiments 1005411 and 1006411 when the testing loss is compared. This is interesting because the number of parameters in U-Net$_2$

is also on the same order of magnitude as the Dense model, which suggests that the loss is dependent on the number of parameters. As the network grows in capacity, it seems more capable in fitting the data. Another factor to notice is that training controls are used in this experiment and the learning rate is reduced once from 1e-3 to 1e-4 at epoch 305, at which point the loss remains relatively constant until the end of training.

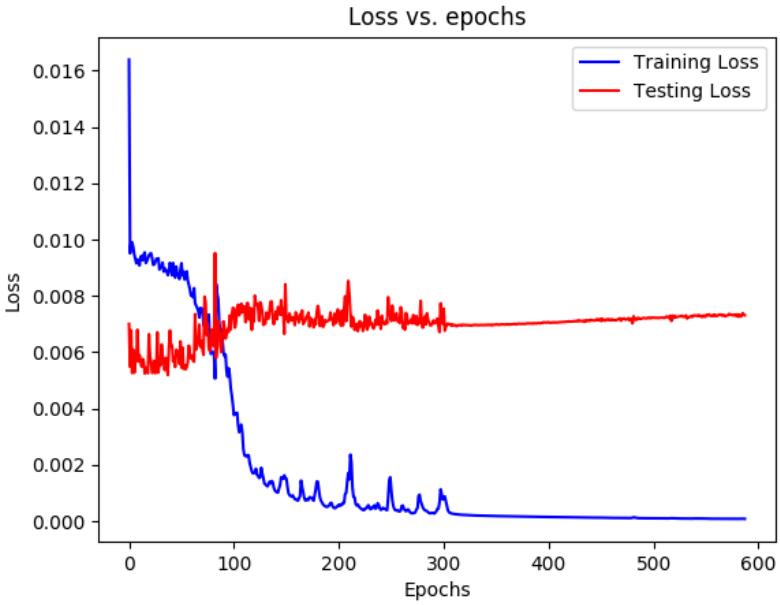


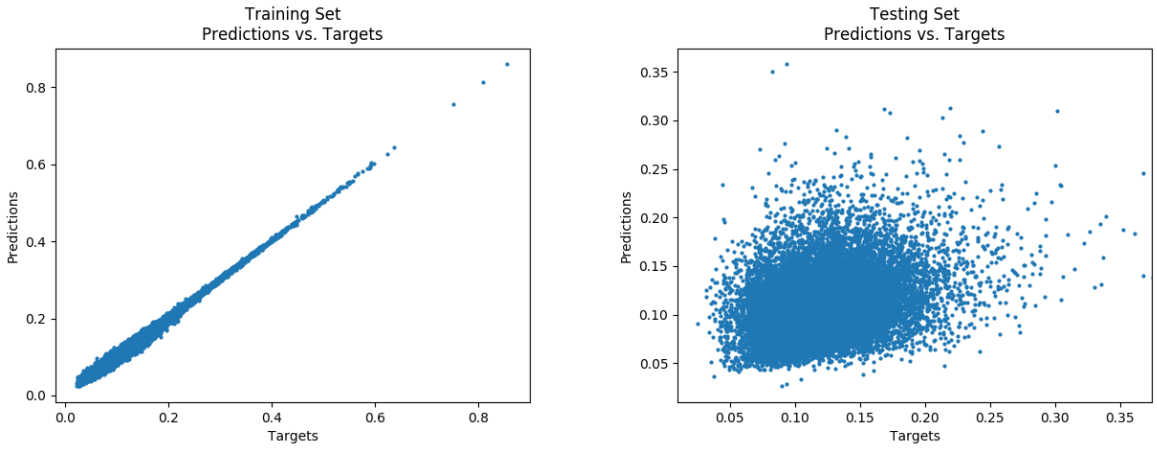
Figure 3.26: U-Net$_2$ Training Loss

A summary of the loss and correlation results for U-Net$_2$ is shown in Table 3.7. The addition of skip connections does not seem to improve results in any significant way. The 0.294 correlation achieved by U-Net$_2$ is a slight improvement in comparison to experiments 1004411 and 1006411, which achieved correlations of 0.267 and 0.271 respectively. This network produces similar results to experiments 1004411 and 1006411 in terms of correlation, but without the need for dropout, which leaves a few options open for improvement.

Table 3.7: U-Net-Mini Model Results

	Training	Testing
Loss	8.57e-05	7.32e-03
R	0.996	0.294

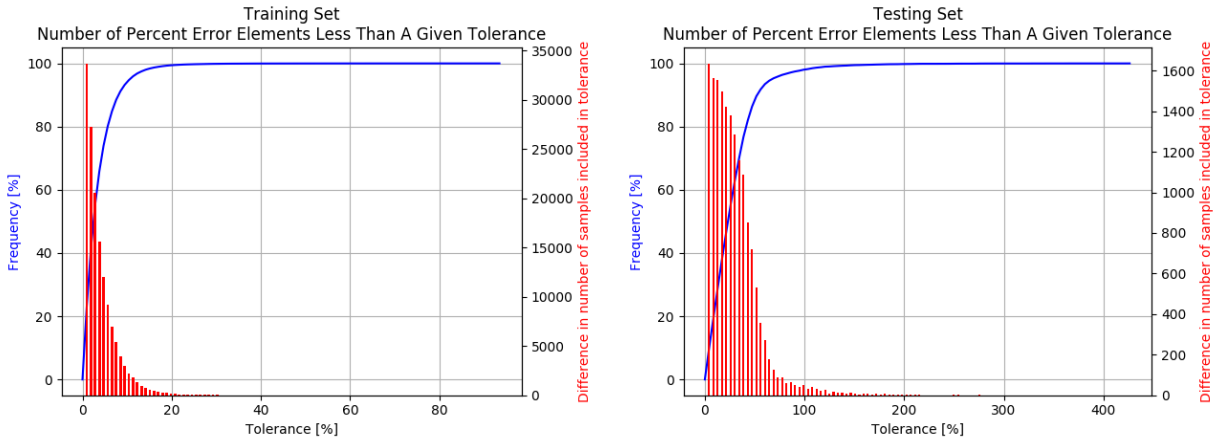
The scatter plots of the target output pixel intensity in relation to the predicted pixel intensity made by U-Net$_2$ for both the training and testing set are shown in 3.27. As expected with a correlation of 0.294, the scatter plot in Figure 3.27b looks similar to previous experiments and U-Net$_2$ does not do much better at producing a 45 degree line on the test set than any of the other experiments conducted so far.



(a) Training Set Scatter Plot (b) Testing Set Scatter Plot

Figure 3.27: Scatter Plots for Experiment 1007411

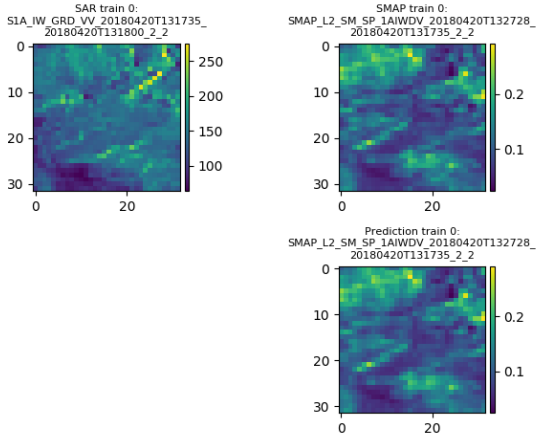
The tolerance plots for the frequency of relative error elements produced by the UNet-Mini Model that occur below a given tolerance level is shown for both training and testing sets in Figure 3.28. The tolerance plot shown in Figure 3.28b produced by U-Net$_2$ is also very similar to the others. It has a maximum relative error of approximately 425% and 80% of the relative error elements fall below a relative error of 50%.



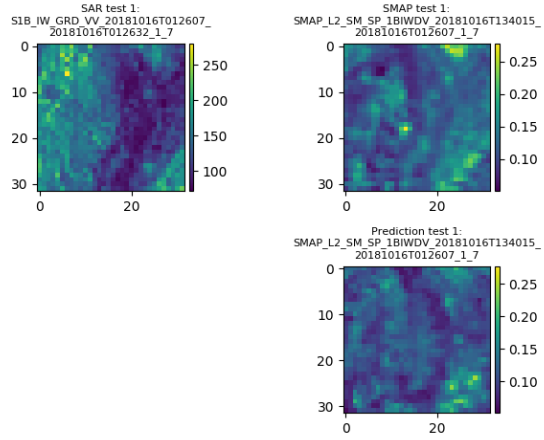
(a) Training Set Tolerance Plot (b) Testing Set Tolerance Plot

Figure 3.28: Tolerance Plots for Experiment 1007411

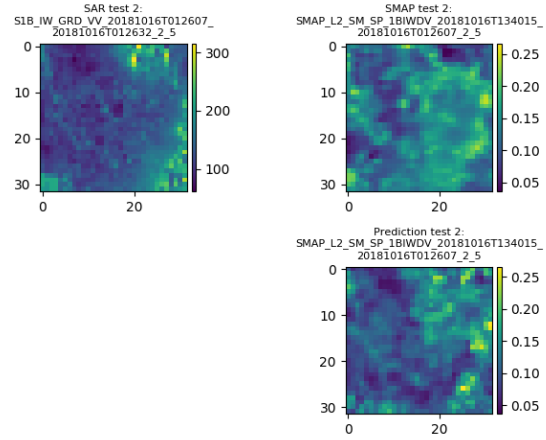
A few predictions from experiment 1007411 are shown in Figure 3.29 for visualization.



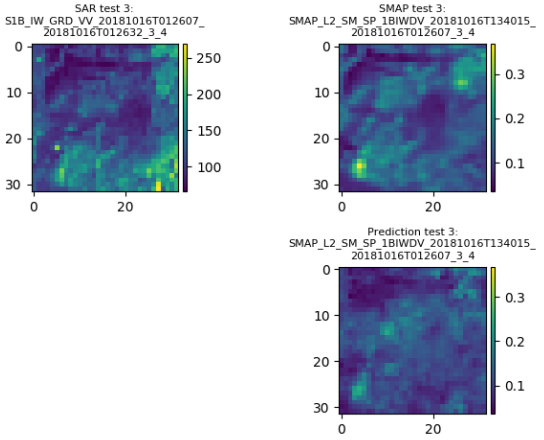
(a) Training Sample



(b) Testing Sample 1



(c) Testing Sample 2



(d) Testing Sample 3

Figure 3.29: Four predictions conducted during experiment 1007411

3.2.5 Experiment 2007411

Experiment 2007411 uses the same network seen in Figure 3.25, which is not reproduced here. Instead of making changes to the network architecture in an attempt to improve results, this experiment seeks to improve results by increasing the number of samples in the dataset. This is accomplished by including samples containing fill values, which broadens the initial dataset (Batch 1) to 936 samples (Batch 2), and the network is left to learn how to predict fill values in SMAP images, which are set to -1, from fill values in SAR images that are set to 0. As discussed in Section 3.1.2, doing so adds a classification perspective to the soil moisture project and results can be described in terms of TPs, FPs, FNs, and TNs.

The history of the loss during training of U-Net$_2$ on Batch 2 is shown in Figure 3.30. In this experiment, an early stopping training control is used in addition to a learning

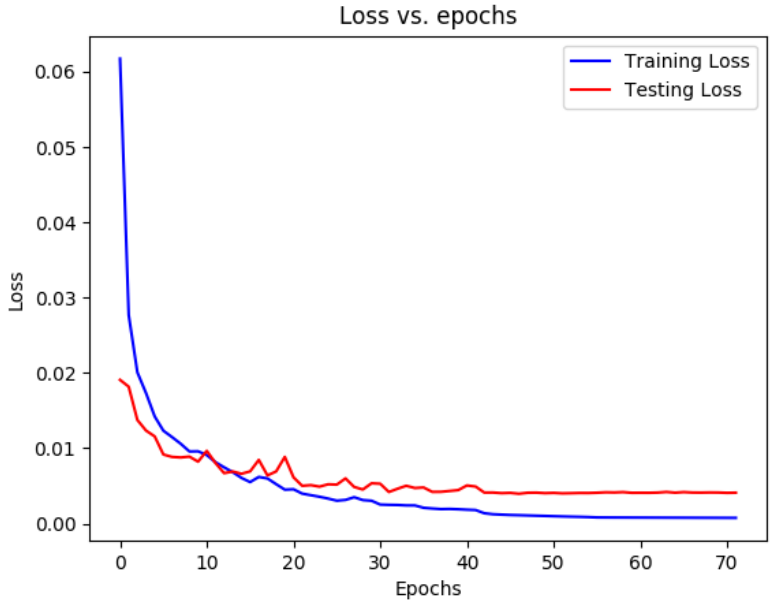


Figure 3.30: Experiment 2007411 Training Loss

rate reduction schedule. This explains why training only required 70 epochs compared to previous experiment. The addition of more data also seems to have reduced the jitter in the testing loss. A potential explanation for this can be attributed to the fact that more gradient updates occur in every epoch since the number of samples have increased and the batch size has remained the same.

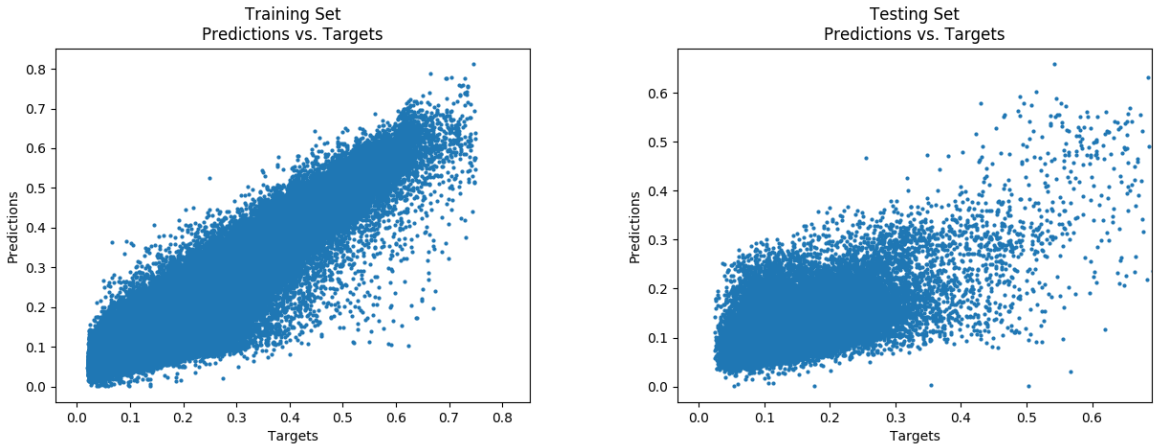
The loss and correlation for U-Net$_2$ on Batch 2 with all samples included is shown in Table 3.8. As can be seen, the testing loss is the same order of magnitude as experiments 1004411 and 1007411, which suggests that the network has the capacity to fit the added data. However, it is important to note that the loss reported for this experiment includes the error contributions from fill value pixels, but the correlation is only for the true positives. With

Table 3.8: UNet-Mini Model Results

	Training	Testing
Loss	7.48e-04	4.10e-03
R	0.927	0.596

the addition of 776 (936 - 160) samples, the correlation has increased to 0.596, which is a significant improvement of 0.30 compared to the previous experiments that were trained on the small dataset of 160 samples. Out of all the experiments conducted thus far, which made improvements through changes in the network architecture, adding more data has been the most successful.

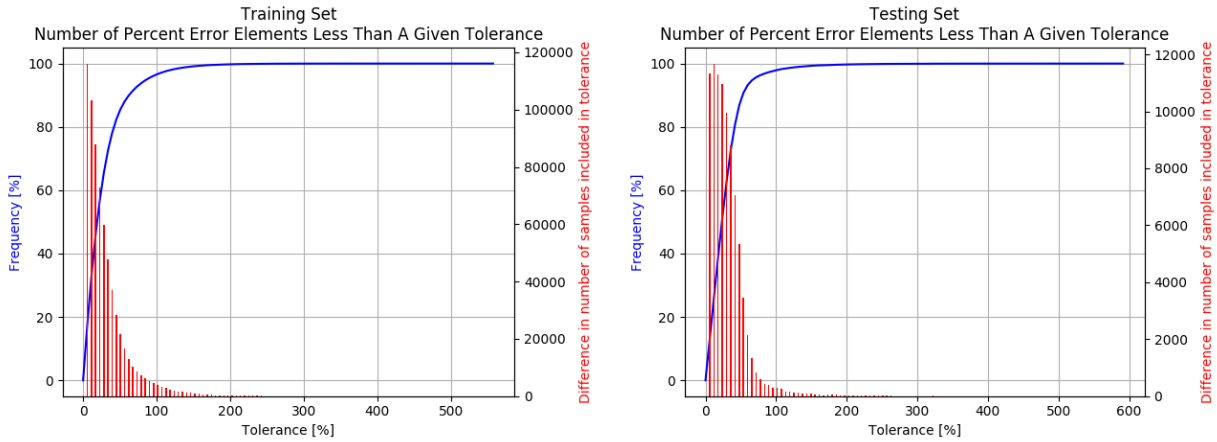
The scatter plots of the target output pixel intensity in relation to the predicted pixel intensity made by U-Net$_2$ on Batch 2 with all samples included for both the training and testing set are shown in Figure 3.31. With the increase in the correlation to 0.596, a linear tendency becomes visible in the testing scatter plot.



(a) Training Set Scatter Plot (b) Testing Set Scatter Plot

Figure 3.31: Scatter Plots for Experiment 2007411

The tolerance plots for the frequency of relative error elements produced by U-Net$_2$ on Batch 2 that occur below a given tolerance level is shown for both training and testing sets in Figure 3.32. Even with the addition of 776 samples to the dataset, this configuration



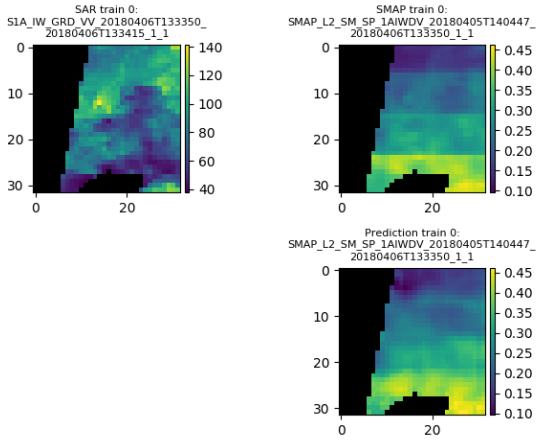
(a) Training Set Tolerance Plot (b) Testing Set Tolerance Plot

Figure 3.32: Tolerance Plots for Experiment 2007411

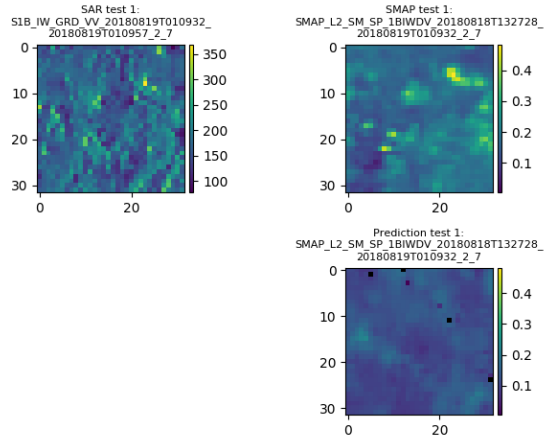
is still able to maintain 80% of the relative error elements below the same 50% tolerance as the other configurations as shown in Figure 3.32b. The maximum relative error did however

increase to approximately 600% from the previous best of 425% achieved by experiment 1007411.

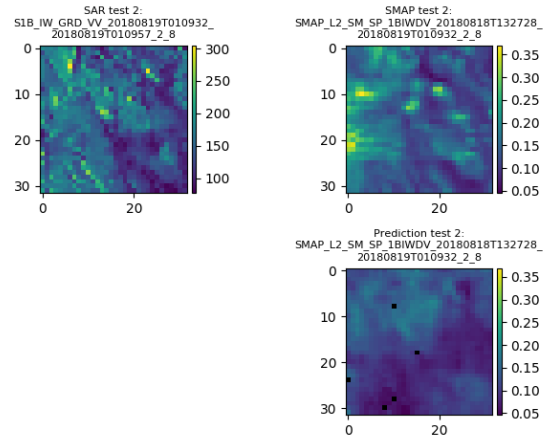
A few predictions from experiment 2007411 are shown in Figure 3.33 for visualization.



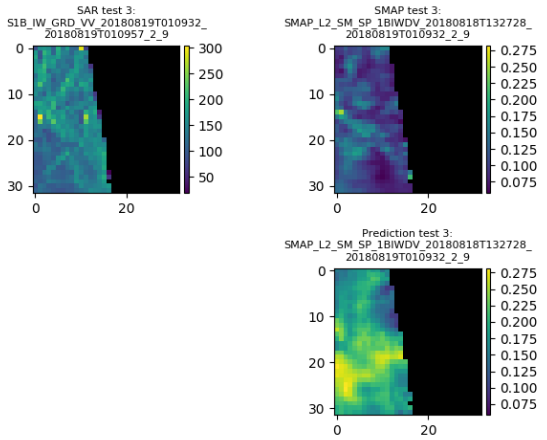
(a) Training Sample



(b) Testing Sample 1



(c) Testing Sample 2



(d) Testing Sample 3

Figure 3.33: Four predictions conducted during experiment 2007411

3.2.6 Experiment 4007411

Experiment 4007411 once again uses the same U-Net$_2$ shown in Figure 3.25. To keep in line with the course of action that improved the correlation the most, significant effort and time went into downloading more SAR and SMAP images. More data is therefore used in this experiment and consists of 1,034 images (Batch 4) that are tiled to produce 45,622 samples. As in the previous experiment, these samples contain fill value pixels for the network to learn as well.

The history of the loss during training of U-Net$_2$ on Batch 4 is shown in Figure 3.34. Similar to the previous experiment 2007411, training controls are used and the loss curves exhibit a smooth decay until they reach a steady state at which point training is terminated.

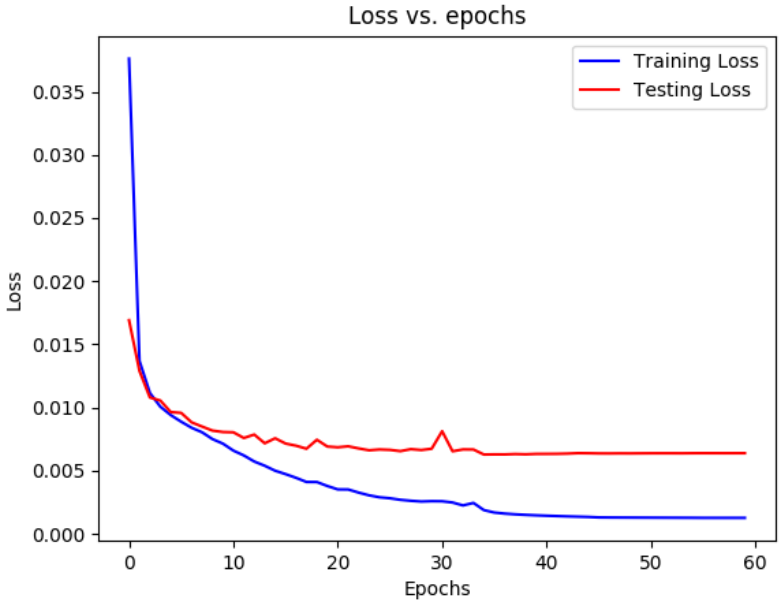


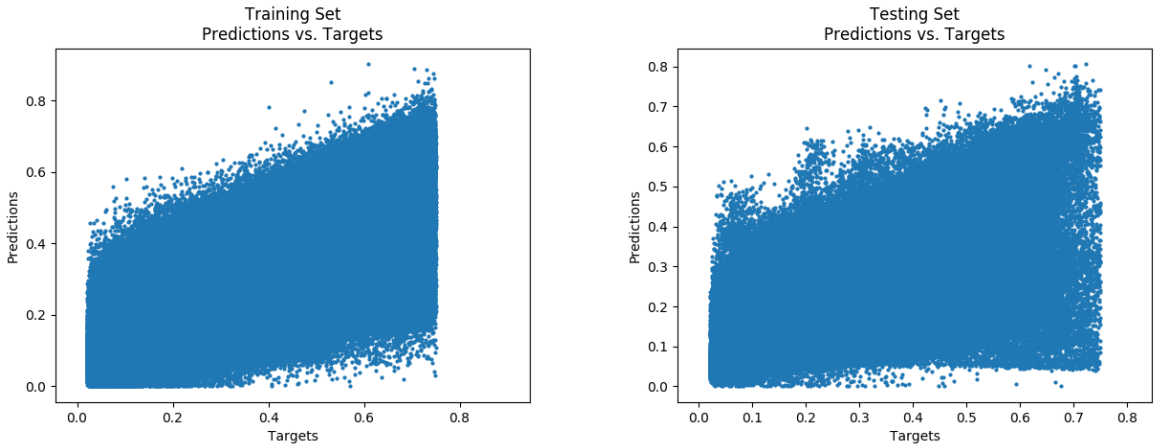
Figure 3.34: Experiment 4007411 Training Loss

A summary of the loss and metric results for U-Net$_2$ on Batch 4 is shown in Table 3.9. The loss on the testing set is still on the same order of magnitude as in previous experiments, which is a good sign that more training examples are not inhibiting or confusing the network in fitting the data. The correlation on the testing set for the TPs improved to 0.72, a 0.12 increase from experiment 2007411. Another interesting trend that happened with the increase in data (that is also visible from experiment 1007411 to 2007411) is the decrease in the correlation of the training set all while an increase occurred to the correlation for the testing set. This is a sign that the network is overfitting the training set less and less and is generalizing better and better to the test set.

Table 3.9: UNet-Mini Model Results

	Training	Testing
Loss	1.26e-03	6.38e-03
R	0.908	0.720

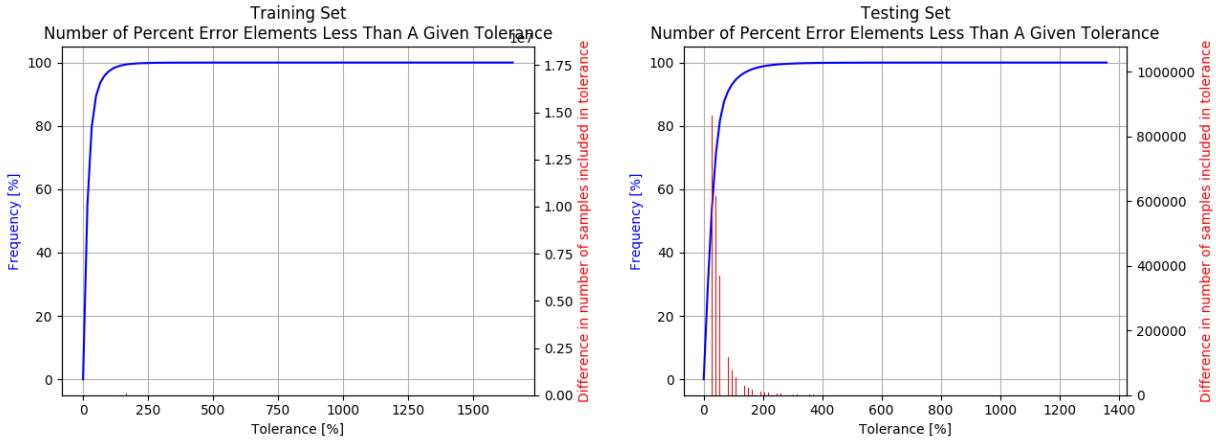
The scatter plots of the target output pixel intensity in relation to the predicted pixel intensity made by U-Net$_2$ on Batch 4 for both the training and testing set are shown in Figure 3.35. The scatter plot shown in Figure 3.35b is very densely populated because of the increase in size of the dataset, however, a linear trend is still visible. The scatter plot is far from being a 45 degree line though as the length between the highest data point and the lowest data point in the upper range of the target axis is approximately 0.75. Such a range nearly covers the entire valid range (0 - 0.80) of SMAP pixel intensities and this shows that the network has difficulty in predicting wetter conditions from SAR imagery.



(a) Training Set Scatter Plot (b) Testing Set Scatter Plot

Figure 3.35: Scatter Plots for Experiment 4007411

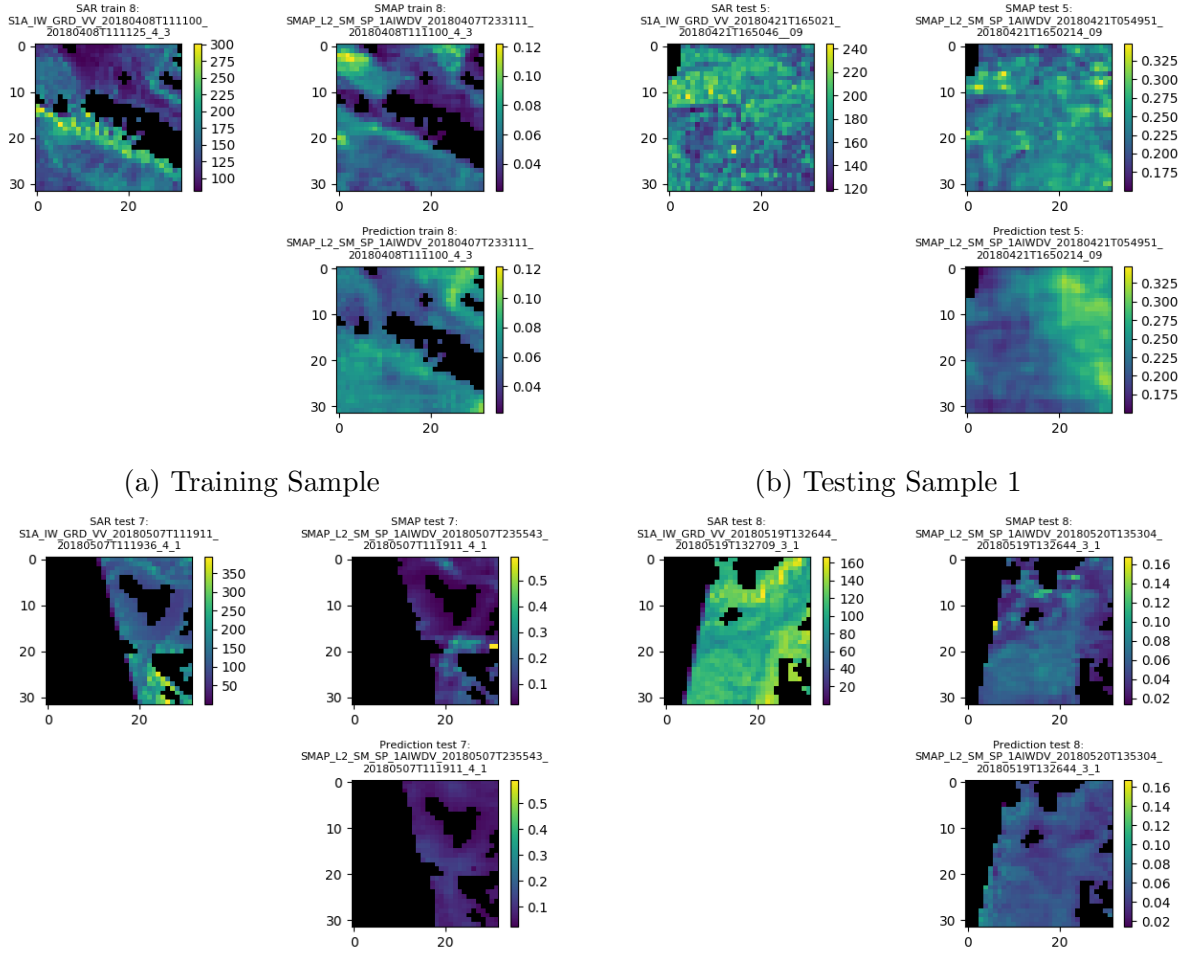
The tolerance plots for the frequency of relative error elements produced by the U-Net$_2$ on Batch 4 that occur below a given tolerance level is shown for both training and testing sets in Figure 3.36. The tolerance plot shown in Figure 3.36b does suffer from the growth in the dataset as the maximum relative error is approximately 1350%, yet 80% of the relative error elements do still fall below the same 50% tolerance.



(a) Training Set Tolerance Plot (b) Testing Set Tolerance Plot

Figure 3.36: Tolerance Plots for Experiment 4007411

A few predictions from experiment 4007411 are shown in Figure 3.37 for visualization.



(a) Training Sample

(b) Testing Sample 1

(c) Testing Sample 2

(d) Testing Sample 3

Figure 3.37: Four predictions conducted during experiment 4007411

3.2.7 Experiment 4MM7411

Experiment 4MM7411 uses the same U-Net$_2$ architecture, but uses maximally sized crops as samples instead of the 32 × 32 sized tiles that have been used in all previous experiments. This experiment tests whether or not the size of the samples has an impact on the results since a larger image would allow for larger effective receptive fields as discussed in Section 2.2. Larger receptive fields may be able to better predict local soil moisture levels depending on the homogeneity of the soil in the neighbouring region. With this change, the dimensionality of the feature maps in the network change slightly and this is represented in Figure 3.38. Despite having larger feature maps, there are still 2,066,497 trainable parameters as in the other experiments that use U-Net$_2$. This is because the convolution filters have not changed in size and neither has the number of filters in each layer. While the size of the samples have changed, the source data are the same 1,034 images (Batch 4) as before, but only 1 maximally cropped sample is taken from each thereby generating 1,034 samples. These samples also include fill value pixels.

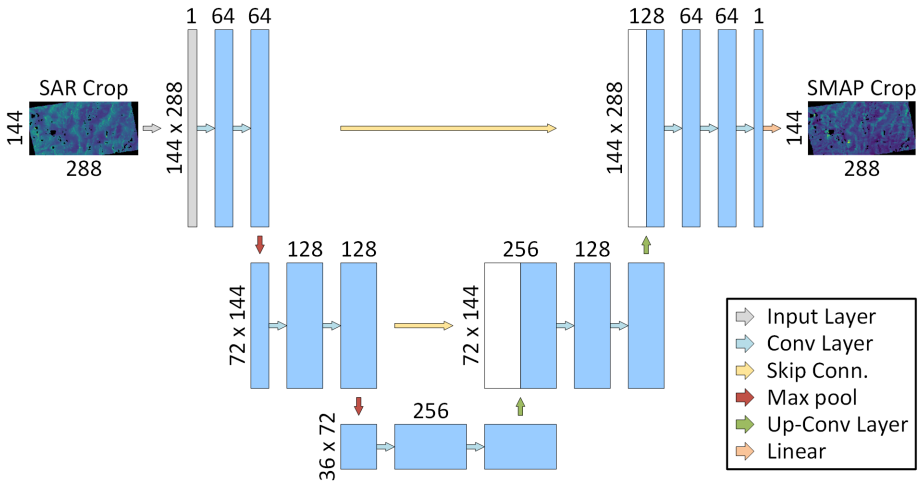


Figure 3.38: U-Net$_2$ Architecture with Increased Feature Map Dimensions

The history of the loss during training of U-Net$_2$ on Batch 4 with larger samples is shown in Figure 3.39. Training controls are used once again as in experiment 1007411 to 4007411. All the loss plots for these experiments bare resemblance to each other as would be expected since the same network architecture is used.

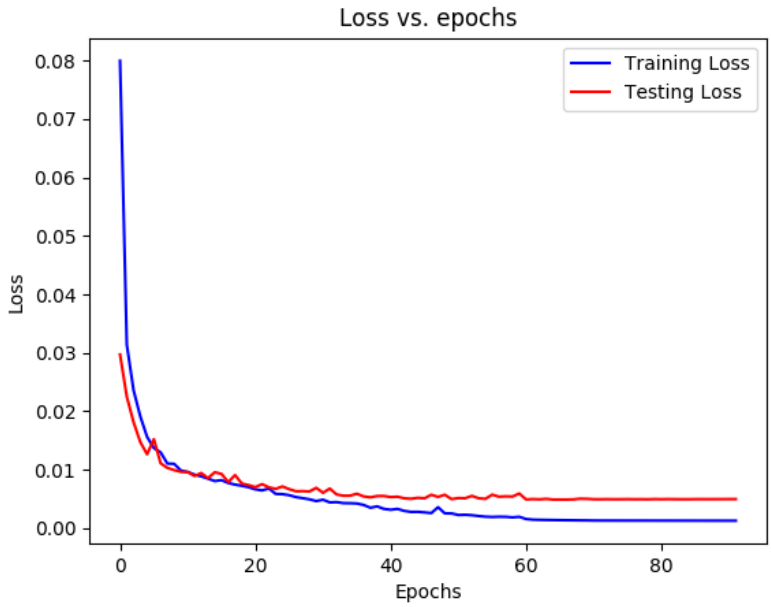


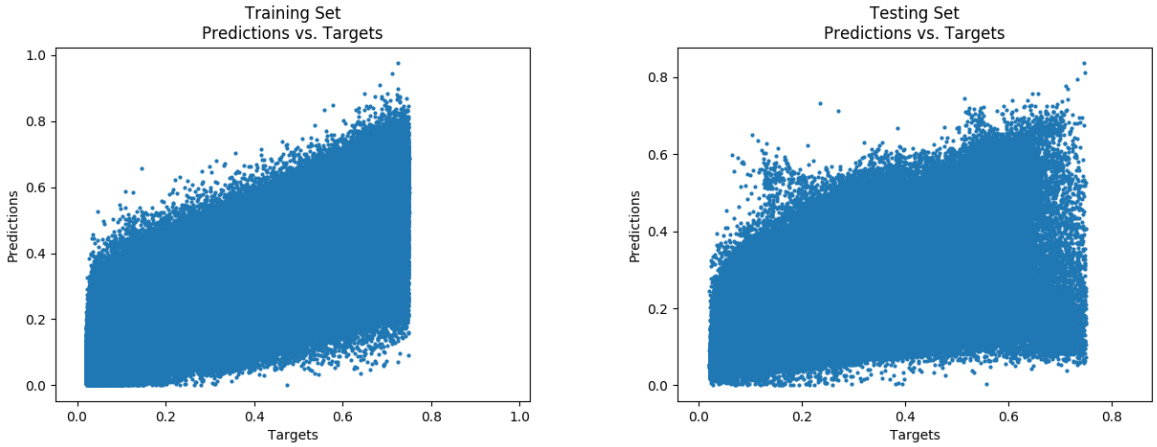
Figure 3.39: Experiment 4MM7411 Training Loss

A summary of the loss and correlation results for U-Net$_2$ on Batch 4 with maximally sized samples is shown in Table 3.10. Again, the testing loss is on the same order of magnitude as the other experiments, but the testing correlation does not improve from experiment 4007411 as they both achieve correlations of 0.72. This suggests that larger sample sizes does not enable this network to learn from larger regions of neighbouring pixels, which may also suggest that soil moisture is dependent more on local conditions than regional climate. Having said that, one other experiment remains to test this hypothesis.

Table 3.10: UNet-Mini Model Results

	Training	Testing
Loss	1.29e-03	4.99e-03
R	0.920	0.720

The scatter plots of the target output pixel intensity in relation to the predicted pixel intensity made by U-Net$_2$ on Batch 4 with maximally sized samples for both the training and testing set are shown in 3.40. Visually inspecting Figure 3.40b and comparing it to Figure 3.35b, it seems there is slight improvement using larger sample sizes. On the low end of the target axis, the length between the highest data point and the lowest data point was reduced from 0.5 in Figure 3.35b to 0.4 in 3.40b, however the same 0.75 range still remains on the high end of the target axis.

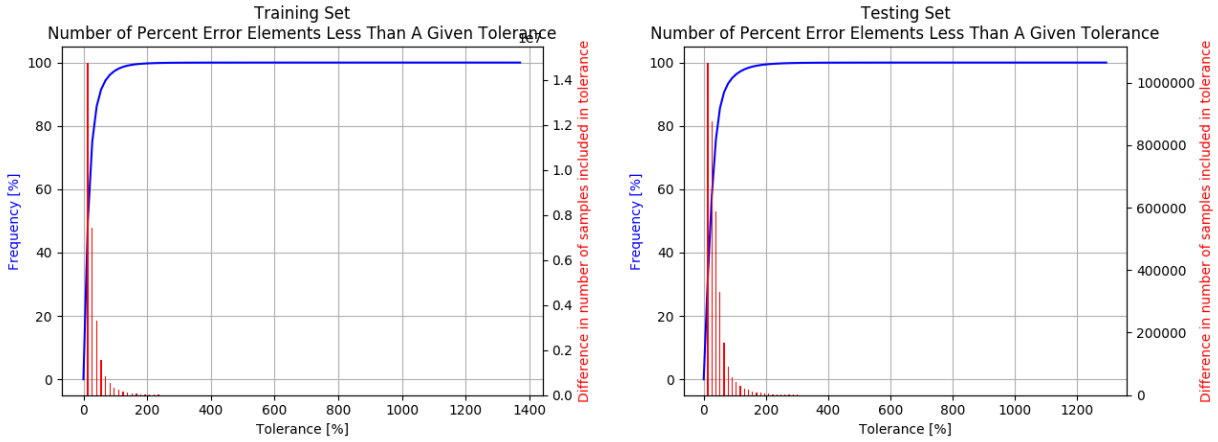


(a) Training Set Scatter Plot (b) Testing Set Scatter Plot

Figure 3.40: Scatter Plots for Experiment 4MM7411

The tolerance plots for the frequency of relative error elements produced by U-Net$_2$ on Batch 4 with maximally sized samples that occur below a given tolerance level is shown for both training and testing sets in Figure 3.41. Roughly speaking, the testing tolerance

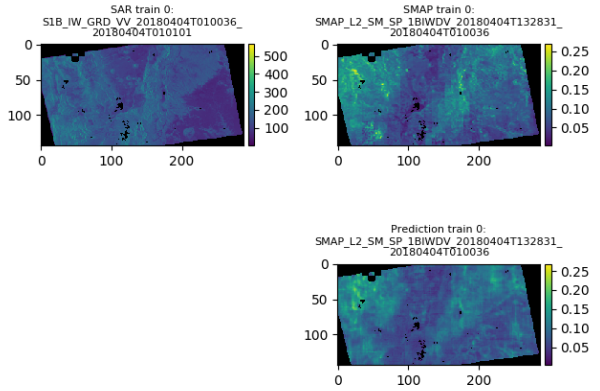
plot still show that 80% of the relative errors are below 50%, but the maximum relative error reduced from 1350% in Figure 3.36b to approximately 1300% in Figure 3.41b. This reduction in the maximum relative error is still small comparatively to the size of the error though and is therefore mostly meaningless.



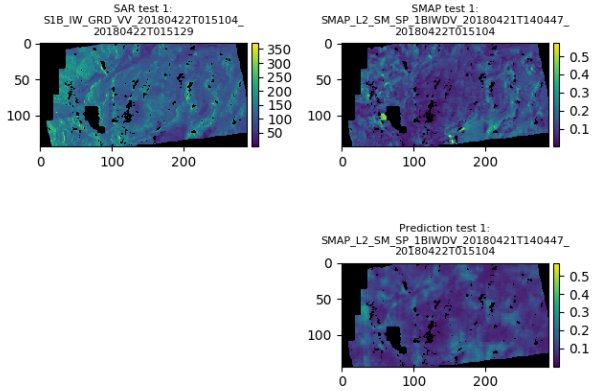
(a) Training Set Tolerance Plot (b) Testing Set Tolerance Plot

Figure 3.41: Tolerance Plots for Experiment 4MM7411

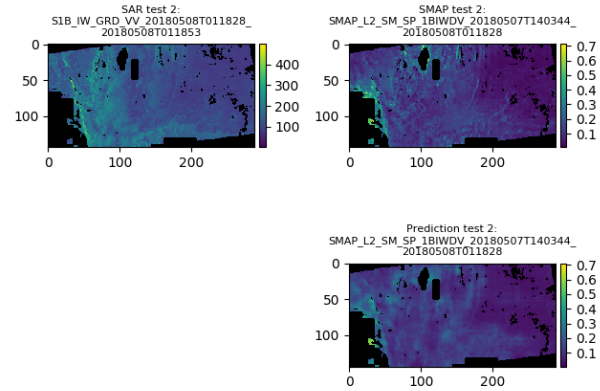
A few predictions from experiment 4MM7411 are shown in Figure 3.42 for visualization.



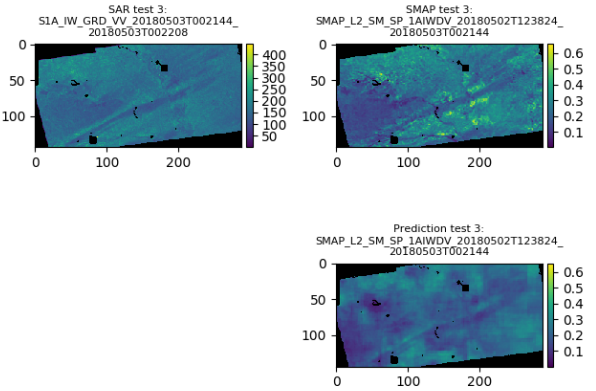
(a) Training Sample



(b) Testing Sample 1



(c) Testing Sample 2



(d) Testing Sample 3

Figure 3.42: Four predictions conducted during experiment 4MM7411

3.2.8 Experiment 4MMA411

As alluded to in the previous section, this experiment tests the hypothesis that larger sample sizes do not enable the network to improve results by potentially combining larger regional features into smaller local features. This was deduced based on the testing correlation that did not improve between experiment 4007411 and 4MM7411, where the only difference between the two was the size of the samples. In order to test this, the full scale U-Net architecture as originally proposed in [87] was used as it contains two additional levels, thereby making the network deeper and the effective receptive field larger. The main limitation to using the full U-Net architecture in previous experiments was due to the small size of the tile samples, but now that the samples are sized as large as possible, the full U-Net architecture can be used. The U-Net architecture is reproduced here as Figure 3.43 and contains 8,629,921 trainable parameters, which is 4x greater than the U-Net$_2$ model and 2.7x greater than the Dense model used in experiment 1004411.

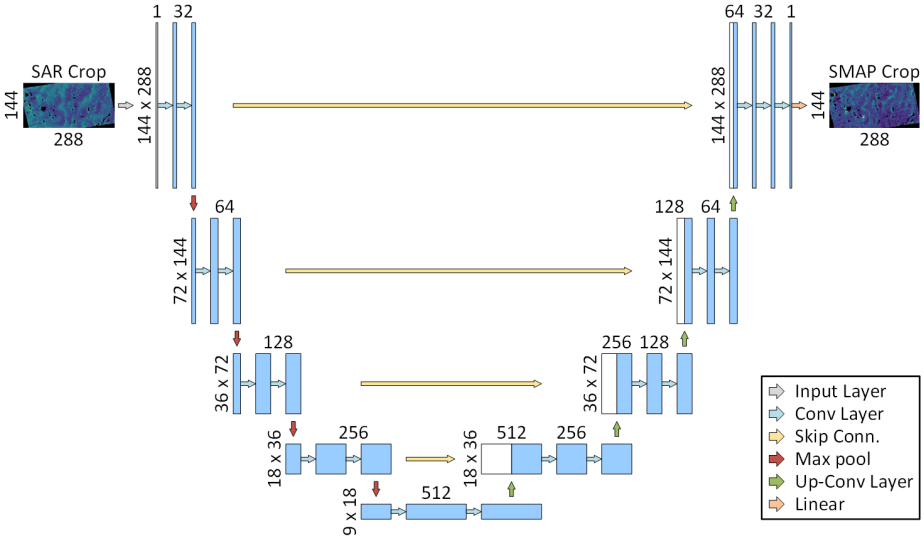


Figure 3.43: U-Net Architecture

The history of the loss during training of U-Net on Batch 4 with larger samples is shown in Figure 3.44 and the same training controls are used here as in experiment 4MMA411. The loss plot shows similar convergence as previous experiments and even though there are more parameters to train, U-Net requires approximately the same number of epochs as experiment 4MM7411 to complete.

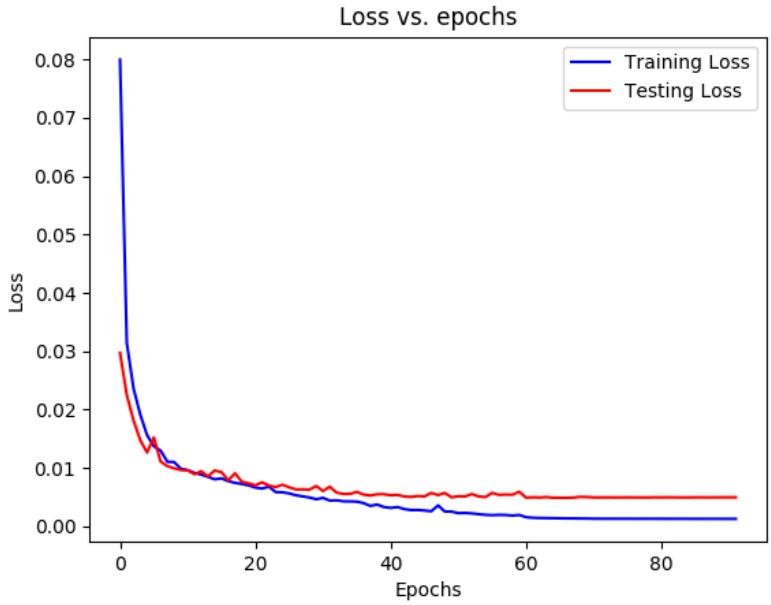


Figure 3.44: Experiment 4MM7411 Training Loss

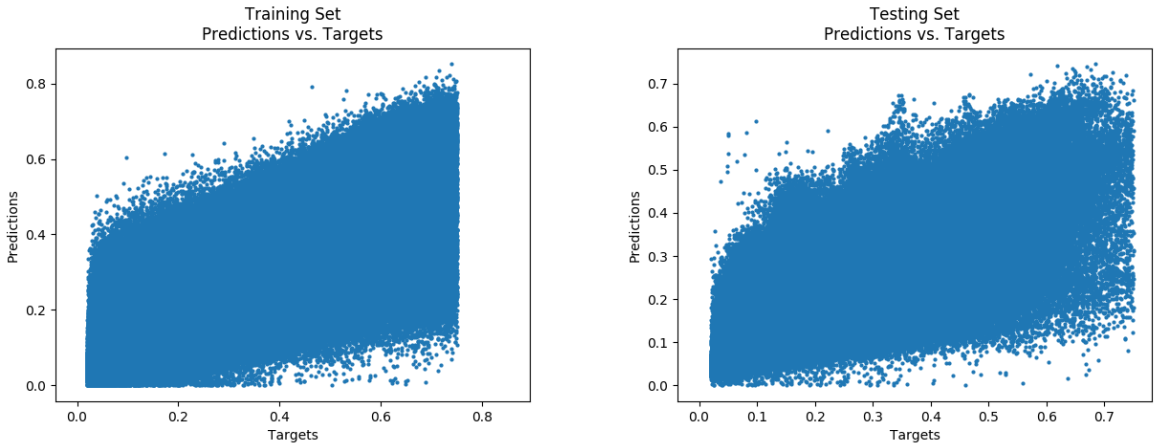
A summary of the loss and correlation results for U-Net on Batch 4 with maximally sized samples is shown in Table 3.11. Despite the larger neural network and number of trainable parameters, the testing loss is still on the same order of magnitude as experiment 4MM7411, which has the same number of samples as this experiment. This is a good sign since a general rule of thumb for deeper networks is that more samples are needed to improve results, but here, the same number of samples are used and similar losses are obtained. Additionally, improvements are made in the testing correlation, increasing it to

0.792 compared to 0.72 for experiment 4MM7411. This improvement suggests that the hypothesis was wrong and the limiting factor in experiment 4MM7411 was the network, and not the sample sizes. The larger receptive field of U-Net compared to U-Net$_2$ does indeed use the larger sample sizes to improve results by combining larger regional features into smaller local features

Table 3.11: U-Net Model Results

	Training	Testing
Loss	1.40e-03	3.82e-03
R	0.924	0.792

The scatter plots of the target output pixel intensity in relation to the predicted pixel intensity made by the U-Net Model on Batch 4 with the maximum sample size for both the training and testing set are shown in 3.45. With the improved testing correlation, a slight



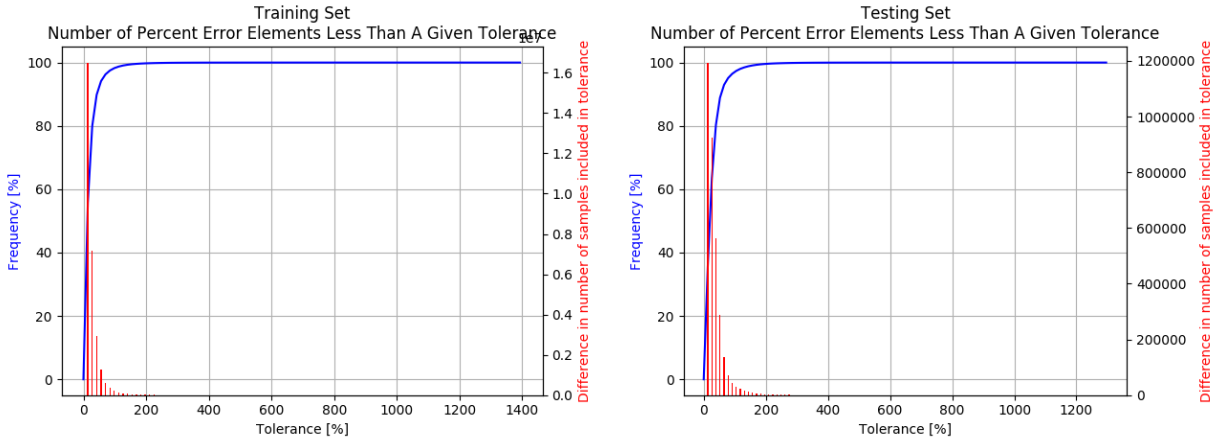
(a) Training Set Scatter Plot (b) Testing Set Scatter Plot

Figure 3.45: Scatter Plots for Experiment 4MMA411

improvement to the scatter plot shown in Figure 3.45b can also be seen. The length between the highest data point and the lowest data point within the bulk of the data on the low end

of the target axis further reduced from 0.4 in Figure 3.40b to 0.3 in Figure 3.45b. At the high end of the target axis, the length reduced from 0.75 in 3.35b to 0.6 in Figure 3.45b.

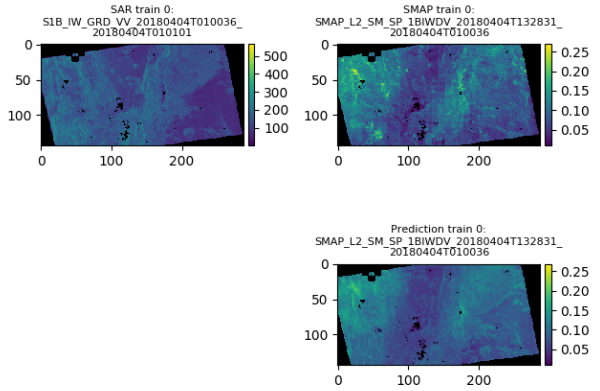
The tolerance plots for the frequency of relative error elements produced by the U-Net Model on Batch 4 with the maximum sample size that occur below a given tolerance level is shown for both training and testing sets in Figure 3.46. No apparent improvement was made in the tolerance plot in Figure 3.46b as the same 80% of relative error elements fall beneath a 50% tolerance level and the largest tolerance level is approximately 1300% and is comparable to experiment 4MM7411.



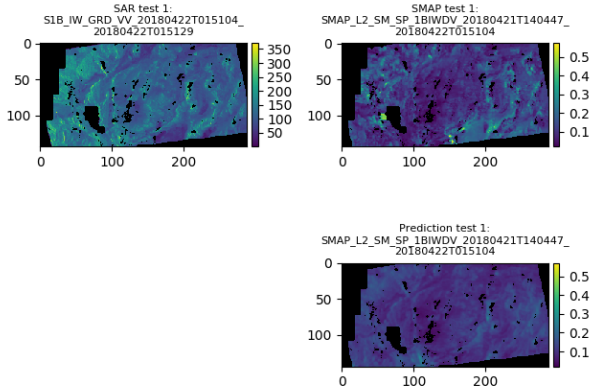
(a) Training Set Tolerance Plot (b) Testing Set Tolerance Plot

Figure 3.46: Tolerance Plots for Experiment 4MMA411

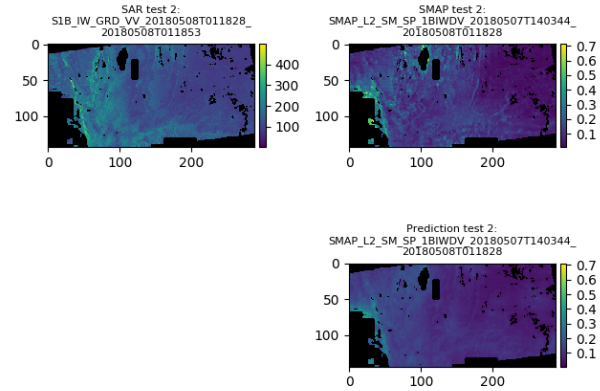
A few predictions from experiment 4MMA411 are shown in Figure 3.47 for visualization.



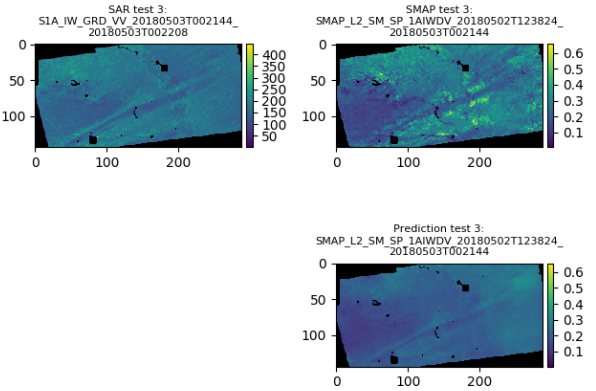
(a) Training Sample



(b) Testing Sample 1



(c) Testing Sample 2



(d) Testing Sample 3

Figure 3.47: Four predictions conducted during experiment 4MMA411

3.3 Discussion

A total of eight different experiments were conducted and presented to gain insights into the ability of neural networks to estimate soil moisture from Sentinel-1 SAR imagery using SMAP images as the labelled data for learning. The results noted in Tables 3.4 - 3.11 of these experiments have been summarized here in Table 3.12 for convenience.

Table 3.12: Soil Moisture Results Summary

	Training		Testing	
Experiment[1]	Loss	R	Loss	R
1004411	3.77e-05	0.992	2.22e-03	0.267
1005411	2.00e-03	0.986	3.94e-02	0.162
1006411	6.19e-03	0.971	2.76e-02	0.271
1007411	8.57e-05	0.996	7.32e-03	0.294
2007411	7.48e-04	0.927	4.10e-03	0.596
4007411	1.26e-03	0.908	6.38e-03	0.720
4MM7411	1.29e-03	0.920	4.99e-03	0.720
4MMA411	1.40e-03	0.924	3.82e-03	0.792

To summarize the findings made in the results section for the soil moisture application, the experiments can be categorized in two groups. The first group consists of experiments that were designed to test different network architectures. These experiments include 1004411, 1005411, 1006411, and 1007411, which all use the Batch 1 dataset that consists of 160 samples with no fill value pixels. Each of the various models used in these preliminary experiments perform well on the training set and achieve correlations >97%. Such high correlations are a good indication that the networks are able to easily fit the given data via the reduction in the mean squared error loss function. However, the great results achieved

[1]1004411 = Section 3.2.1, 1005411 = Section 3.2.2, 1006411 = Section 3.2.3, 1007411 = Section 3.2.4, 2007411 = Section 3.2.5, 4007411 = Section 3.2.6, 4MM7411 = Section 3.2.7, 4MMA411 = Section 3.2.8

by each model on the training set are meaningless due to the poor performance that each achieve on the unseen test data. It is apparent that the models overfit the training data and fail to generalize to the testing set. The best performer of these four experiments is the U-Net$_2$ Model having a correlation of 29.4%. For this reason, the U-Net$_2$ architecture was chosen as the best contender for further experimentation.

The second group consists of experiments where the effects of the datasets are evaluated. Batch 1 was expanded to include samples that contain fill values, which augmented Batch 1 into a 936 sample dataset denoted as Batch 2. Experiment 2007411, using Batch 2, let the model learn the transformation needed to predict fill values in SMAP images based on the fill values contained in SAR images. As previously discussed, this action results in predictions that are best described using terms such as TPs, FPs, FNs, and TNs and are represented in a confusion matrix. It is important to note that the correlation results stated here are of the true positives. As shown by the results achieved by experiment 2007411 in Table 3.12, increasing the number of data-points has shown to be the strongest factor in improving results. It increased the correlation on the test set by 0.30 up to 0.596 on Batch 2, from the 0.294 correlation obtained by U-Net$_2$ on Batch 1. For this reason, significant effort and time went into downloading a larger number of SAR and SMAP images.

Subsequently, a dataset consisting of 1,034 images, denoted as Batch 4, was used in experiments 4007411, 4MM7411, and 4MMA411. The same U-Net$_2$ model was used in experiment 4007411 and it was trained on a 32×32 tiled representation of Batch 4, which further improved results by achieving a 72% correlation on the test set. Experimentation not shown here was performed to produce different datasets of varied sample sizes and overlap but these experiments did not produce any meaningful improvements. Even experiment

4MM7411 with samples of maximally allowable dimensions achieved the same 72% correlation on the test set. It seems as though adding more of the same pixel data points through various sample sizes and overlaps does not produce any significant improvements. However, an improvement to the correlation was achieved using the deeper and full scale U-Net architecture in experiment 4MMA411. Using the full U-Net was only made possible with the maximally sized samples because the image size could now support the 4 max pooling layers in U-Net. This improvement can be attributed to U-Net's increased number of trainable parameters and increased receptive field over the U-Net$_2$ derivation. Therefore, the best result is achieved by experiment 4MMA411 with a 79.2% correlation on the test set.

It is essential to compare the results achieved here to other published works on soil moisture to gauge the relative goodness of the best performing experiment. As discussed in the soil moisture background in Section 2.3, NASA's algorithm achieved correlation values of 0.717 and 0.829 for 3 km and 9 km range images respectively [53]. Their algorithm was validated with in-situ ground measurements in select regions and is therefore not representative of their entire dataset with which the algorithm was formulated since it is infeasible to do so. However, NASA's findings provide a gauge that can be used to compare the results achieved here. Despite using finer resolution, research quality 1 km $\times$ 1 km soil moisture maps, experiment 4MMA411 achieves a correlation of 0.792 across the entire dataset of 1,034 images, which is within the range of NASA's findings. This means that the error within the predictions made by the U-Net model trained here is within the error of the labelled data itself. The method used to achieve this result is therefore a reasonable method to estimate soil moisture from satellite data.

Comparing correlations to results achieved by other researchers is also important. The

ANN formulated in [50] produced results with correlations of 0.90171, 0.721, and 0.50841 on the training, validation, and testing sets respectively. While a validation set was not used during the training of the U-Net model herein, the testing results attained by experiment 4MMA411 are on a similar level to the correlation achieved on the validation set by [50]. Also, similar findings are reached on the training set where experiment 4MMA411 achieved a correlation of 0.924. An important distinction in the methodologies exists however. In [50], the input to the ANN consists of four inputs: 1) the corrected radar backscattering, 2) NDVI, 3) the incidence angle, and 4) the thermal infrared temperature (TIR). Soil moisture estimates are treated as point sources therein, whereas in this work it is treated as an image. Their results were also dependent on auxiliary data such as NDVI, incidence angle, and TIR, where NDVI has been noted in this study and in others to be the most sensitive parameter in improving SMC estimates using C-band SAR data. It is therefore a good indication that while experiment 4MMA411 achieved comparable results to this study, the results were based solely on SAR data and can most likely be improved with the inclusion of NDVI.

Similar findings are reached in [40] where VV polarized SAR images performed the most poorly, but was improved with NDVI. Their findings were that the VV polarization performed the worse with a coefficient of determination of 0.4047, but was improved to 0.6404 with the addition of NDVI. This equates to a correlation of 0.636 and 0.80 respectively and the results with NDVI are comparable to the findings of experiment 4MMA411, which only used VV polarized SAR images and did not include NDVI. This further shows that NDVI could be used as a second channel to improve the results of this work. Alternatively, [40] found that the best performance was achieved when co- and cross-polarizations were available. This suggests that there may be value in using other polarizations as additional channels to the input images used in this work.

3.3.1 Future Work for the Soil Moisture Application

Within the framework of the existing configurable parameters used to identify different experiments, future work can be done to improve results in a couple of ways. The most readily available improvement, and the one that has shown to produce the largest impact on results, is to download more data. Images contained in Batch 4 were sourced from the second quarter (Q2) of 2018. More images can be acquired by running the same scripts used to download Batch 4 and querying ESA's and NASA's servers for Q3 of 2018 or other years. It is unclear however how much more data to obtain as a saturation point could exist where adding more and more data would only make marginal improvements. The second is to tune the hyperparameters of the U-Net architecture. The U-Net architecture used herein only had half the number of filters used in each level compared to the original architecture used in [87]. Further increasing the number of trainable parameters in the model could also lead to better results. Alternatively, other known networks for semantic segmentation could be used such as DenseNet or Fully Convolutional Networks (FCN) [88].

A limitation of the methodology presented is in the lack of in-situ ground measurements for validation purposes. The methodology also contains a type of circular reasoning since the SMAP data used as labelled data in these experiments are originally formulated as a synergy of Sentinel-1's SAR data with SMAP's passive data. Therefore, the labelled image data is already dependent on the input SAR data. However, this presents an interesting opportunity to use transfer learning [89], a concept in deep learning where a network is trained on a large corpus and refined to a specific application with a smaller dataset. The number of SAR and SMAP images available is very large and could be used to train a network to learn a general relationship between SAR data and soil moisture. This trained network could

then be refined through transfer learning with in-situ ground measurements, which are more limited in geographical scope and expanse. Such an approach may produce better results than a network that was just trained on a limited dataset of only in-situ data as is the case in numerous other studies.

As previously discussed and seen in other studies, the inclusion of NDVI data has been shown to greatly improve results. The methodology presented in this work may be improved by including NDVI as a second channel to the SAR images. Convolutions would then combine these channels in some way to best fit the labelled data. However, there is one main challenge associated with this approach and that challenge is to find NDVI data that corresponds to the data used herein. While NDVI can be calculated from the bands provided by Landsat-8 or Sentinel-2 for example, it is unknown at this time whether or not these NDVI maps would coincide temporally and geographically with the SAR and SMAP images. Studying the effects of other auxiliary data such as incidence angle on the results is another avenue that could be pursued. The question is how to include such data in an image-to-image model. Some techniques have been proposed in [37] and [48] to calibrate the σ^0 values in the SAR images using the incidence angle. Otherwise, the incidence angle could be included as an independent input parameter or channel to a given model. Along the similar route of adding auxiliary data, adding other polarizations as channels to the input image could also improve results. Unfortunately, this is not as easy as it could be since the ESA's download policy is mostly limited to VV polarizations despite Sentinel-1 having sensors for HH single polarization, and HV or VH cross polarizations. One solution to this problem is to simulate these co- and cross-polar images using the AIEM or Oh's model as in [40]. Any or all three of these auxiliary data augmentation schemes could be used to improve the soil moisture estimates found as part of this thesis.

3.4 Conclusions

A summary of the methods and results presented for the soil moisture application is given to conclude this chapter. The main goals of the chapter were to gain experience using deep learning tools and to test whether or not a neural network could be used to effectively estimate soil moisture from satellite data. To setup this supervised learning process, the input data used was VV polarized SAR data acquired by the Sentinel-1 satellite operating in the C-band and the labelled output data was 1 km $\times$ 1 km resolution soil moisture images produced by the SMAP satellite. The SAR data was then preprocessed using GDAL to geocode, downsample, and geographically correct the image extents to create a pixel-wise image-to-image equivalence with the SMAP data. Both the SAR and SMAP images were then cropped to remove bordering fill value pixels and tiled into sub-regions to increase the number of samples in the dataset. Eight different experiments were formulated to observe the effects of certain parameters on the results, which were quantified with the mean squared error loss function and the correlation between the true and predicted SMAP images. These parameters consisted of 1) the data batch of source files used to create a dataset of input-output pairs, which also distinguished between datasets containing samples with fill values or no fill values, 2) the tile size, 3) the overlap between tiles, 4) the neural network model, 5) the loss function, 6) training controls, and 7) normalization. Of these experiments, experiment 4MMA411 achieved the best results, which used the U-Net architecture to reduce the MSE on a dataset of 1,034 images with maximally sized samples that were z-score normalized. This experiment achieved a testing MSE and correlation of 3.82e-03 and 0.792 respectively. These results are comparable to other findings in the literature, which also show that results can be improved with the addition of auxiliary data, NDVI being the most important, as well

as the addition of other polarization channels and incidence angle calibration. Given that the results are similar to those achieved by other researchers suggests that the methodology formulated as part of this work is a reasonable approach to estimating soil moisture using deep learning techniques. Therefore, the goals of this thesis in regards to the soil moisture application are fulfilled since experience using different deep learning tools was acquired and it was shown that neural networks were capable of estimating soil moisture on unseen test data with comparable results to other studies.

Chapter 4

Sea Ice Application

This chapter presents the work that was conducted to test if sea ice types could be classified based on SAR imagery using deep learning models. The sections that follow include an introduction to the study region where data is acquired as well as a description of the data products and methodology used to create input-output pairs for the deep learning models. Following is a presentation of the models used and the results achieved for different experiments. The best results achieved consist of an overal classification accuracy of 94.02% and an ice accuracy of 91.75% on a three class classification problem. An analysis of the results of several experiments is then made to understand which factors contribute the most to improved results. A discussion follows where the results are compared to other studies, the utility of the method is presented, and avenues for future work are put forward. The chapter is then concluded with a brief summary of the methodology, the results, and the discussion presented for the sea ice application, where it is determined that it is possible to predict sea ice types from satellite data using deep learning.

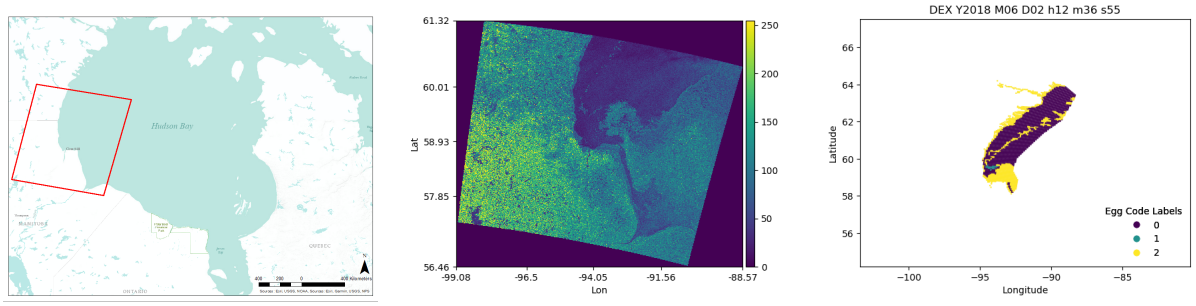
Material in this chapter has been adapted from [55][1], a paper accepted for publication in MDPI's Remote Sensing Journal and included in the Special Issue on Polar Sea Ice: Detection, Monitoring and Modeling.

[1]All articles published by MDPI are made available worldwide under an open access license, meaning no special permission is required to reuse all or parts of articles published by MDPI. Reprinted with permission from R. Kruk, M. C. Fuller, A. S. Komarov, D. Isleifson, and I. Jeffrey, "Proof of concept for sea ice stage of development classification using deep learning," *Remote Sensing*, vol. 12, no. 15, p. 2486, 2020.

4.1 Materials and Methods

4.1.1 Study Region

The Hudson Bay (the Bay) region seen in Figure 4.1a is important for shipping and other maritime activities [90]. The Bay (coupled with James Bay) is the largest inland sea in North America at over 1,300,000 km^2 [91], with Arctic water entering through Roes Welcome Sound and Atlantic water entering through channels connecting Hudson Strait. Winters are characterized by seasonally varying ice cover between October and August, and the Bay generally exhibits colder temperatures than those at similar latitudes due to ice lasting into the late summer [91]. During summer, the Bay exhibits temperatures that are 5 to 10 °C cooler than the surrounding land, while during the winter, sea ice cover coupled with cooling Arctic Oscillation flow effects can result in temperatures reaching −45 to −50 °C. Seasonal snowfall generally occurs in October and November, with totals reaching 120 to 150 cm in the northern part of the Bay. Due to the seasonal melt regime, the Bay contains first-year ice (FYI) [92] with a high concentration (10/10 coverage) and typical ice thickness reaching a maximum of ∼2 m with an average ice thickness of ∼1.6 m peaking in April and May [91].



(a) (b) (c)

Figure 4.1: (**a**) Hudson Bay region. (**b**) Input RADARSAT-2 ScanSAR Wide image. (**c**) Labelled image analysis ice chart used for outputs according to Table 4.1.

Recent studies have shown that freeze-up is occurring later in the year and melt is occurring earlier in the year [92], and contemporary studies continue to investigate the linkage between field based observations and satellite measurements [93] during these critical times in the seasonal evolution. Although the inter-annual variability of sea ice extent is constrained by the surrounding land mass, ice extent in Hudson Bay decreased between 1978 and 1996 at a rate of 1400 km^2/yr. [94]. Beginning in late May or early June, coastal leads (potentially navigable open water between shore and the remaining contiguous ice-pack) extend and broaden, encircling the Bay, with complete melt occurring during the following 4 to 5 weeks [91].

4.1.2 Description of Data Products

To create a dataset appropriate for deep learning classification of ice types, a dataset of 350 RADARSAT-2 (R2) ScanSAR Wide (SCW) images and 172 image analysis charts is gathered, herein referred to as Batch 4. Batches 1 - 3 are not considered here as they were subsets of Batch 4 used to test the correctness of the methodology, software and models in a similar approach to the build up of datasets and experiments conducted for the soil moisture application. The chosen dataset spans from June to December 2018 to obtain a distribution of labelled data representing the melt and freezing periods. The SCW images have a swath width of 500 km with a nominal pixel spacing of 50 m $\times$ 50 m and range between 20 degrees for the near incidence angle and 49 degrees for the far incidence angle [95]. The CIS sea ice charts serve as expert approximations to in situ conditions and are produced by ice analysts using R2 SAR imagery, as it is the main satellite data source for routine monitoring of the Canadian Arctic. The CIS ice analysts manually identify a set of spatial polygons for a given R2 SAR image, summarize the attributes of each polygon in the form of an egg code [2]

shown in Figure 4.2, and publish these results in image analysis charts.

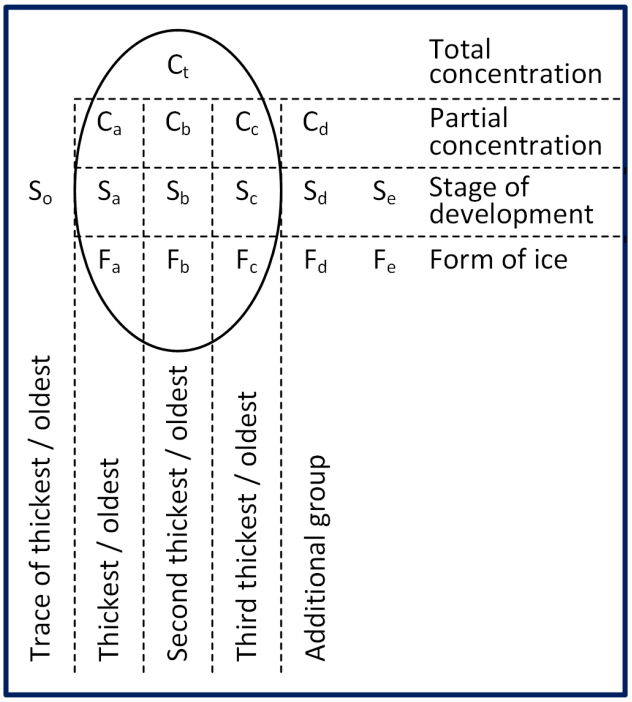


Figure 4.2: Canadian Ice Service egg code reproduced from [2].

The main attributes of the egg are the total and partial ice concentrations, the stage of development (or ice type), and the predominant floe size range (or the form) of the ice. The CIS produces various types of satellite based ice charts, including image analysis charts and daily charts. Image analysis charts are used as they correspond most closely to specific SAR images. An example SCW image is shown in Figure 4.1b.

4.1.3 Dataset Labelling

Image analysis ice charts are parsed to extract the stage of development feature (interchangeably referred to as ice type) from the egg codes in the CIS image analysis ice charts to provide labels for each SAR image in Batch 4. For the purposes of this study, egg codes having only

a total sea ice concentration (C_t) $\geq$9+ are used. The egg codes are then simplified into a smaller set of ice type categories per Table 4.1. The result of categorizing the eggs in this way is shown in Figure 4.1c.

Table 4.1: Simplified labelling scheme reducing CIS egg codes to three classes.

Category	Label
Ice-free	0
New ice corresponds to the stage of development codes 1 – 5	1
First-year ice corresponds to the stage of development codes 6 – 4 •	2

Labels 1 and 2 for ice types encapsulate the stage of development codes consistent with Table 3.1 in the CIS manual, while label 0 for water is consistent with the ice-free diagram shown in Figure 2.2 in the CIS manual [2]. Old ice was not included in this study because it is not an ice type typically found in the Hudson Bay. This labelling scheme is referred to as Label Type 4 in the remainder of the chapter to follow the historical development of the software framework used to conduct this study. This means three different labelling schemes were evaluated prior to Label Type 4 and were not suitable for the results presented in this thesis. Furthermore, egg code samples are included in the dataset only if all sub-categories of the stage of development (S_a, S_b, S_c) shown in Figure 4.2 map to the same label, which are referred to as pure samples, otherwise the sample is discarded. For example, suppose that S_a is given a code of 6 and S_b is given a code of 1 in the egg code produced by the CIS. S_a would map to a label of 2 and S_b would map to a label of 1 per Label Type 4 defined in Table 4.1. Since the sub-categories of the stage of development do not map to the same label in this example, the egg code is not included in the dataset.

4.1.4 SAR Sub-Region Extraction

The chosen labels are paired with 100 × 100 sub-regions in the corresponding SAR image, where the sub-regions are centered about the latitude and longitude given by the labels in the image analysis ice chart. This dimensionality is chosen to be consistent with the ice charts, which are rasterized by the CIS experts with a 5 km × 5 km pixel resolution, wherein the pixel size of a SAR image represents a 50 m × 50 m area. The R2 products use a rigorous projection model to describe the mapping between image coordinates (x, y) and ground coordinates $(lat, lon, height)$ [96]. The approximation used for the rigorous projection model is given by a set of rational polynomials, and normalized polynomial coefficients are given in the Product Information File (PIF) of the R2 SAR product. The image coordinates of the centre of each sub-region are calculated using the coefficients and the rational polynomials detailed in [97] for the given latitude and longitude associated to the corresponding egg code. Example sub-regions for one of the SAR images used in the dataset are shown in Figure 4.3.

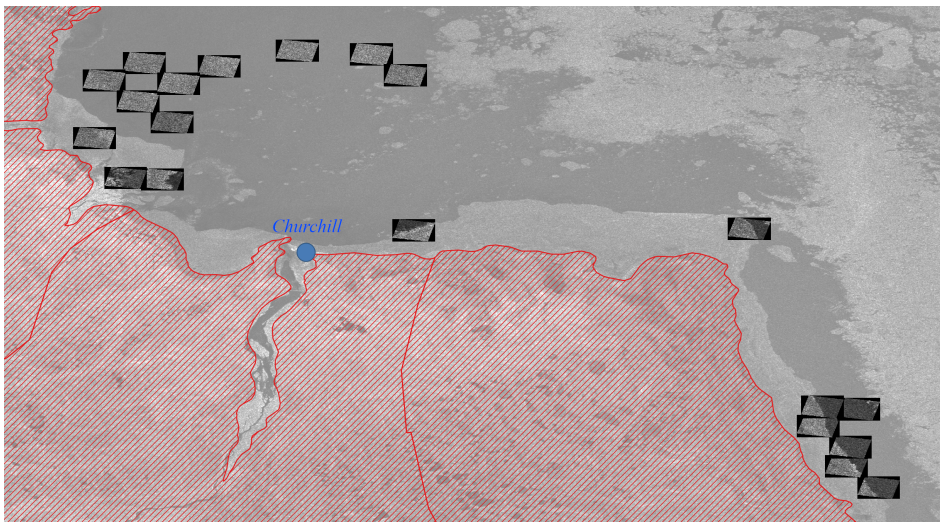


Figure 4.3: Candidate dataset samples consist of SAR image sub-regions centred about ice chart image analysis sample coordinates. Background SAR image showing the southwestern region of the Hudson Bay and Churchill, Manitoba. The red hatched area denotes land.

Note that the sub-regions shown in Figure 4.3 do not appear to be square because the WGS 84 projection causes warping of the SAR image and the sub-regions. The correctness of the sub-regions can be verified by measuring the length of the edges of the sub-regions in a GIS application. The same sub-regions in both HH and HV polarizations of SCW images are used to form the first two channels of an input sample to be used in the machine learning classification. The cross-polarization (cross-pol) ratio of HV/HH is appended to the sample as a third channel to provide additional information as the ratio is sensitive to volume scattering within a medium. The input sample is therefore a 3D object with a shape of (3, 100, 100), and each is labelled per Table 4.1. Once labelled, the sub-region and the label are treated as an independent sample. We acknowledge that calibration and noise removal information is provided with the SAR products and can be used to preprocess the images, but for this proof of concept, a streamlined framework is sought such that it could be used in an operational capacity for sea ice service entities while still providing accurate results.

4.1.5 Model Setup

Two different convolutional neural networks (CNN) are used for training. The first neural network is modelled after the encoder section of U-Net [87] and is shown in Figure 4.4. Sea ice classification is naturally an image segmentation problem, and U-Net has been shown to be effective for this particular type of application, as in [98], which applied a U-Net based architecture on the Cityscapes dataset, and in [99], which applied U-Net to segment surface ice concentration on images of two Alberta rivers. Only the encoder section is used because the methodology chosen to create our dataset assigns class labels that are single-valued. The encoder section of U-Net consists of four convolution blocks, where each block is formed by two convolution layers followed by a max-pooling layer. The number of convolution filters

after each convolution block doubles.

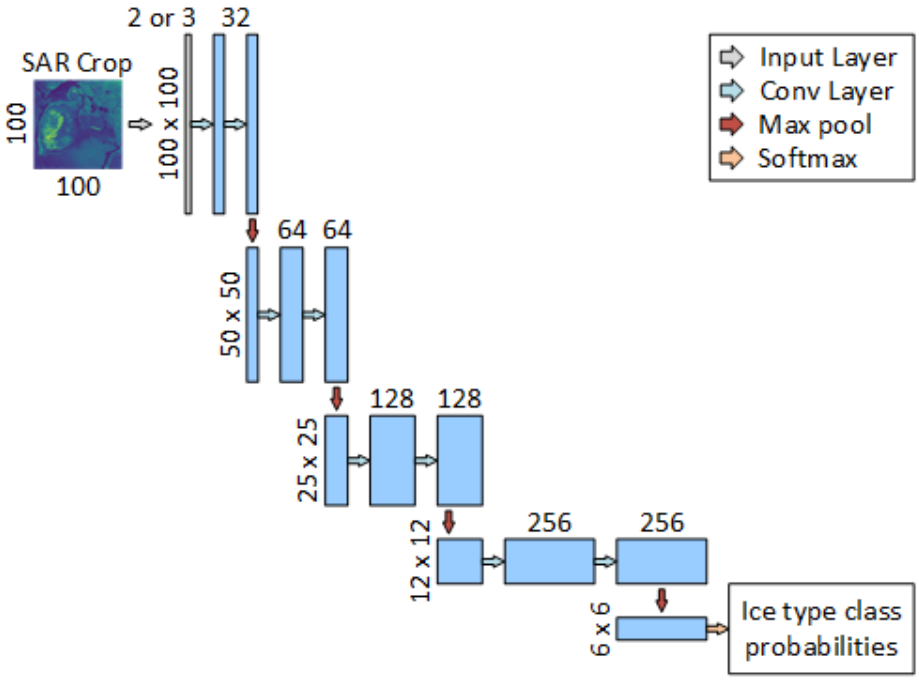


Figure 4.4: U-Net Encoder: The model consists of the encoder (down-sampling) portion of a U-Net model that maps a SAR sub-image to a prediction label. A representative model used in this work consists of approximately 1.2 million parameters.

The second CNN used for training is DenseNet [64]. DenseNet was chosen as it has been shown to achieve state-of-the-art results on challenging datasets such as ImageNet [64]; it was also used in a previous study to estimate sea ice concentration with promising results [65]. A general block diagram of the DenseNet used in this work is shown in Figure 4.5. One of DenseNet's properties is its capability to reuse features at different scales throughout the network, which applies nicely to SAR images where important features are recurrent at different scales. Feature reuse is achieved within the dense blocks shown in Figure 4.5. A dense block is composed of a series of convolution blocks whose output is concatenated with the input of every subsequent convolution block via skip connections. DenseNet's au-

thors showed that output concatenation (as opposed to output summation found in residual networks) enables the model to more efficiently reuse features [64]. The DenseNet model consists of a series of these dense blocks and transition blocks, the latter of which consist of a bottlenecking convolution layer and an average pooling layer.

The DenseNet configuration used in this study is DenseNet-121, indicating that there are 121 layers in the model. This is achieved by having four dense blocks, each having 6, 12, 24, and 16 convolution blocks, respectively, where each convolution block consists of 2 convolution layers, contributing 116 layers. One transition layer is used between each dense block, thereby adding another 3 layers to the total. The two remaining layers come from an initial convolution layer and the classification layer. Note that the initial convolution layer is typically used to down-sample larger inputs, but is not necessary for our 100×100 pixel input.

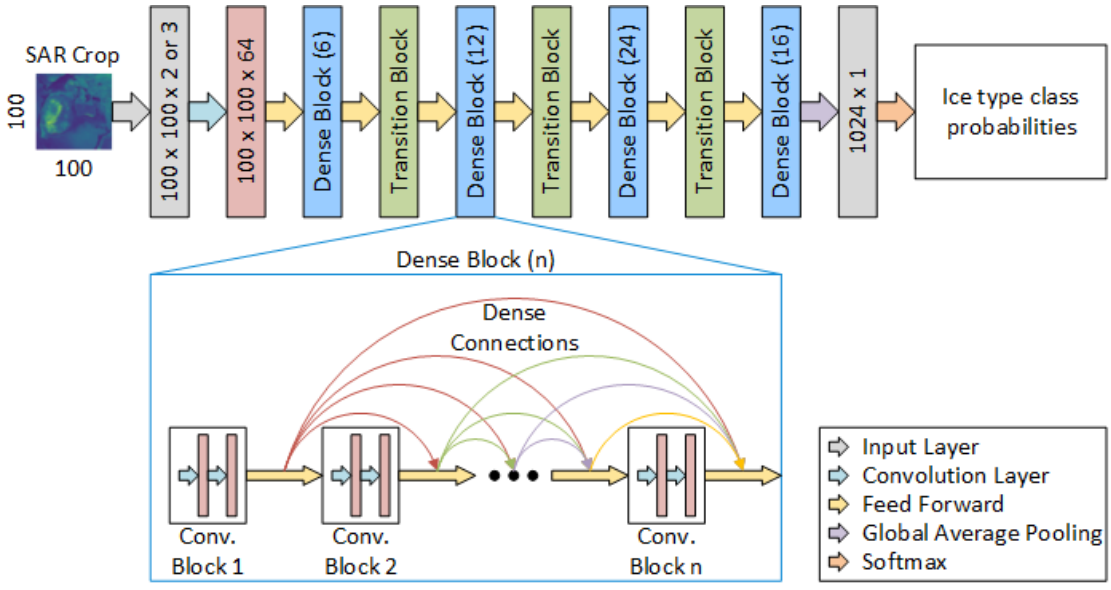


Figure 4.5: DenseNet model: The model consists of a number of dense blocks connected by transition blocks that maps a SAR sub-image to a predication label. A representative model used in this work consists of approximately 7 million parameters.

Both the U-Net encoder and DenseNet models use softmax as the classification layer to assign an ice type class probability to a given input. The softmax probabilities are given by Equation 4.1, where $y^{(n)}$ is the output class label for the nth sample, $\mathbf{x}^{(n)}$ are the outputs of the last layer, and $\mathbf{W}^T = [\mathbf{w}_1\ \mathbf{w}_2 \dots \mathbf{w}_K]$ are the weights of the last layer.

$$p(y^{(n)} = k|\mathbf{x}^{(n)}; \mathbf{W}) = \frac{e^{\mathbf{w}_k^T \mathbf{x}^{(n)}}}{\sum_{k'=1}^{K} e^{\mathbf{w}_{k'}^T \mathbf{x}^{(n)}}}. \tag{4.1}$$

These probabilities are optimized through a categorical cross-entropy loss function L, as described in Section 2.2.6, which is reproduced here for convenience:

$$L = -\sum_{n=1}^{N} \sum_{k=1}^{K} \delta_{nk} \log(p(y^{(n)} = k|\mathbf{x}^{(n)}; \mathbf{W})). \tag{4.2}$$

4.1.6 Training Setup

The dataset created per the methodology in this section generated 186,673 water samples, 14,539 new ice samples, and 63,265 first-year ice samples. The dataset was balanced such that the loss is not over- or under-represented for any particular class during training [100], and this was accomplished by undersampling the majority and randomly keeping 14,539 samples in each class. These samples were randomly shuffled and split into an 80%-10%-10% training, validation, and testing set. These subsets were z-score normalized by first calculating the mean and standard deviation of each channel across all training samples. This set the mean and variance of the subsets to 0 and 1, respectively, to provide better gradient flow during training. The gradients were calculated with backpropagation, and the weights were updated with the Adam optimizer. Additionally, training was conducted in batch sizes of 1000 for the adapted model of U-Net and 32 for DenseNet; the DenseNet model had smaller batch sizes due to GPU memory constraints and the increasingly large array

sizes arising from concatenation in the model. Methods for improving memory efficiencies in DenseNet models can be found in the literature [101], but are not considered here.

4.2 Results

In order to prove the concept of using CIS labels and deep learning to predict ice type, several experiments were conducted by changing the dataset in a variety of ways. The performances of the U-Net encoder model and DenseNet model were quantified using the percentage of correctly classified samples referred to as the accuracy. Additional insight was obtained by a second metric aligned with the goal of this proof of concept to predict ice types. The ice accuracy metric is defined as the sum of the true positive samples for Ice Classes 1 and 2 based on Table 4.1 divided by the total number of samples in those ice classes. The ice accuracy was formulated to better compare experiments with unbalanced datasets since certain experiments had a large number of water samples compared to ice samples (greater than a 7:1 ratio before undersampling the majority). The best experiment achieved an overall accuracy of 94.02% and an ice accuracy of 91.75% on the test set.

4.2.1 Experimental Configurations

The software framework developed for this work was created to flexibly support six configurable parameters used to uniquely identify different experimental datasets. These parameters are described and itemized below and their values are summarized in Table 4.2.

1. **Data batch** — indicates the SAR scenes and CIS data source files to use. In this work, only Batch 4 is considered, whereas previous iterations of data batches consisted of different subsets of source files leading up to Batch 4.

2. **DEX type** — determines the type of CIS data to use, DEXA being a code for daily ice charts and DEXI being a code for image analysis charts. Only DEXI is used herein.

3. **Label type** — specifies the egg code labelling type used. Only Label Types 4 and 5 are used in this work. Label Type 4 refers to the set of labels defined in Table 4.1. Label Type 5 modifies the labels in Table 4.1 to group new ice (label 1) and first-year ice (label 2) into a single class, producing a two class experiment used to benchmark the Label Type 4 experiments.

4. **Polarization** — specifies the SAR polarizations included as input channels in the dataset. Only two channel input samples with HH and HV polarizations and three channel input samples with HH, HV, and the HV/HH ratio are used in this study.

5. **Total concentration ($\mathbf{C_t}$)** — indicates the minimum C_t required from the egg code for a sample to be included in the dataset. Only samples with concentrations $\geq$ 9+ were considered in this study.

6. **Preprocessing** — determines the preprocessing steps to use on the SAR samples. No preprocessing was considered in this study.

Table 4.2: Dataset configuration naming map.

	1 Data Batch	2 Dex Type	3 Label Type	4 Pol	5 C_t	6 Preprocessing
A	-	-	-	-	-	Raw
B	-	DEXI	-	-	-	-
C	-	-	-	-	$\geq$ 9+	-
D	Batch 4	-	Type 4	HH, HV	-	-
E	-	-	Type 5	-	-	-
F	-	-	-	-	-	-
G	-	-	-	HH, HV, HV/HH	-	-

For example, the experimental configuration DBDDCA decodes to the configuration parameter values: Batch 4 (D), DEXI charts (B), Label Type 4 (D), HH and HV polarization channels (D), ice concentration ≥9+ (C), and no preprocessing (A). For the purposes of this study, the parameters for data batch, DEX type, total ice concentration, and preprocessing (i.e., the first two and last two letters in the dataset identification code) can be ignored by the reader. Only the label type, which denotes two and three class experiments, and the polarization parameters were varied (i.e., the middle two letters of the dataset identification code). Unpopulated entries in the table refer to parameter values not used in this study, but are maintained for historical and future continuity.

4.2.2 Experimental Results – Summary and Interpretation

The resulting accuracy metrics of the U-Net encoder model and the DenseNet model are summarized in Tables 4.3 and 4.4, respectively. The experiments that were performed for the purposes of this proof of concept sought to compare the performance of training on datasets with two channel inputs (HH, HV) and three channel inputs (HH, HV, HV/HH). The experiments conducted for the three class dataset (Label Type 4 with water, new ice, and first-year ice classes) were also replicated with the corresponding two class dataset (Label Type 5 with water and ice classes) for benchmarking purposes.

There are a few patterns that emerged based on the experimental results shown in these tables. The first is that, as expected, none of the three class classifiers were able to achieve the same level of accuracy as the two class classifiers. The obvious explanation for this difference is that samples of two different ice types are more difficult to distinguish than water and ice. Another possibility for the discrepancy is the number of samples with which the models were trained. The two class experiments had 81,928 samples for each

Table 4.3: U-Net encoder classifier results.

			Training			Testing		
Config.	Classes	Ch.	Loss	Acc.	Ice Acc.	Loss	Acc.	Ice Acc.
DBEDCA [1]	2	2	0.0190	99.37	99.32	0.0337	99.05	98.99
DBEGCA [1]	2	3	0.0128	99.59	99.45	0.0356	99.01	98.90
DBDDCA	3	2	0.1378	94.76	92.62	0.1742	92.99	90.51
DBDGCA	3	3	0.0741	97.49	96.51	0.2371	91.75	88.48

[1] Two class classifier benchmark experiments.

Table 4.4: DenseNet based classifier results.

			Training			Testing		
Config.	Classes	Ch.	Loss	Acc.	Ice Acc.	Loss	Acc.	Ice Acc.
DBEDCA [1]	2	2	0.0244	99.37	99.13	0.0329	99.07	98.65
DBEGCA [1]	2	3	0.0225	99.42	99.17	0.0309	99.10	98.65
DBDDCA	3	2	0.1414	95.95	94.23	0.1872	94.02	91.75
DBDGCA	3	3	0.1129	97.21	96.02	0.2008	93.12	90.34

[1] Two class classifier benchmark experiments.

class, while the three class experiments had only 14,539 samples. The choice to balance the datasets by undersampling the majority contributed the most to this difference. The effects of undersampling the majority on the number of samples in each class are outlined in Table 4.5. Undersampling the majority in the three class case can be seen to have eliminated 172,134 samples of water and 48,726 samples of first-year ice, compared to the two class case that only eliminated 104,745 water samples. The other factor that caused such a large difference in the number of samples available between the two and three class datasets was the choice to reject impure egg code samples whose sub-categories of the stage of development feature did not all map to the same ice type. Rejected impure samples in the three class case accounted for a relative increase of 4,124 pure samples in the two class case, widening the gap even further. This occurred because the sub-categories for the stage of development feature (S_a,

S_b, S_c) mapped to the same label in the two class case (resulting in pure samples), but mapped to different labels in the three class case (resulting in rejected impure samples).

Table 4.5: Effects of under-sampling the majority demonstrating the differences between the full dataset and balanced samples.

	Before			After		
Label Type	Class 0	Class 1	Class 2	Class 0	Class 1	Class 2
2 Class	186,673	81,928	-	81,928	81,928	-
3 Class	186,673	14,539	63,265	14,539	14,539	14,539

The second pattern to notice from the accuracy results is that DenseNet consistently outperformed the U-Net encoder model. This was not a surprising result since the U-Net encoder model was adapted to fit the needs of the dataset formulation, while the DenseNet architecture is more naturally suited to the problem. The modified U-Net architecture only has 1.2 million trainable parameters compared to DenseNet's seven million, which allows DenseNet to have more representational power over the modified U-Net, which could explain the differences in the results. The size of DenseNet came at the cost of a 6.58 h training time (for the DBDDCA experiment with two polarization channels and three classes), compared to the 19 min required to train the U-Net encoder model for the same experiment. DenseNet is also a memory-hungry model, and even with a modern GPU (16 GB Tesla V100 GPU), only batch sizes of 32 could be used to train this model compared to the batch size of 1000 used to train the U-Net encoder model. According to the creators of DenseNet, improvements can be made to improve memory efficiency, but their implementation shown in [101] was not immediately compatible with the deep learning framework used to conduct the experiments herein. For this reason, an investigation into the use of memory efficient implementations is left for future work. While the differences in batch size clearly impact the training time, it is

unclear if they affect performance. A recent study showed that models trained with smaller batch sizes achieve better testing results than models trained with larger batch sizes [102]. Training the U-Net model with the same batch size of 32 would clarify whether or not this was the cause for the discrepancy in the results. Nonetheless, an approximate 1% difference in accuracy is not a bad trade-off to be made for a fraction of the training time required.

The third pattern is that experiments that used three polarizations channels (HH, HV, HV/HH) in the input sample performed markedly better than the two channel experiments (HH, HV) on the training set, but performed slightly worse on the test set. A possible explanation for this behaviour is that the HV/HH ratio channel is not independent of the HV and HH channels and that supplying this additional information alleviates the model from learning that the ratio between the two channels could provide beneficial features. As a consequence, the model could overfit the training set, degrading the model's ability to generalize to the unseen test data. Further experimentation in the form of model regularization is needed to assess whether or not overfitting is occurring and will be the subject of a future study.

4.2.3 Experimental Conclusions

While the analysis of the differences between numerical results was necessary to gain insights into the possible variables that caused them, a visual representation of the predictions can serve as a reminder that the goal of this work is to provide a proof of concept that this methodology could be sufficient to automate classification of ice types prior to the analysis of a sea ice expert. Three SAR scenes were selected to showcase the predictive quality of the experiment that achieved the highest test set results, which was a DenseNet model trained on the DBDDCA (three classes, two channels) experiment dataset. Figure 4.6 shows samples

extracted from the SAR images that were dedicated to the training set and the testing set, as well as the corresponding predictions made from each sample. The plots in Figure 4.6 show that some of the erroneous predictions may be identified by an ice analyst based on the proximity to other samples.

The capacity of a neural network to accurately predict ice types during the summer melt season was the other research question to be answered by this proof of concept. A monthly breakdown of the samples in the training and testing set and their class for the DBDDCA experiment is given in Tables 4.6 and 4.7.

Table 4.6: DenseNet DBDDCA (3 class, 2 channel) monthly count breakdown.

	Training				Testing			
Mon.	Class 0	Class 1	Class 2	% of Total	Class 0	Class 1	Class 2	% of Total
Jun.	0	0	11600	33.24	0	0	1435	32.89
Oct.	5017	489	0	15.78	644	64	0	16.23
Nov.	6606	3946	0	30.24	811	474	0	29.45
Dec.	0	7177	58	20.73	0	932	3	21.43
Total	11623	11612	11658	100.00	1455	1470	1438	100.00

Table 4.7: DenseNet DBDDCA (3 class, 2 channel) monthly accuracy breakdown.

	Training				Testing			
Mon.	Class 0	Class 1	Class 2	Accuracy	Class 0	Class 1	Class 2	Accuracy
Jun.	-	-	95.54	95.54	-	-	92.96	92.96
Oct.	98.70	87.12	-	97.68	97.20	79.69	-	95.62
Nov.	99.89	93.84	-	97.63	99.63	90.72	-	96.34
Dec.	-	93.44	17.24	92.83	-	91.52	0.00	91.23
Total	99.38	93.31	95.15	97.95	98.56	90.75	92.77	94.02

It can be seen from this analysis that classes 0 and 1 were not represented during the month of June, which was during the summer melt season. This was due to the sampling

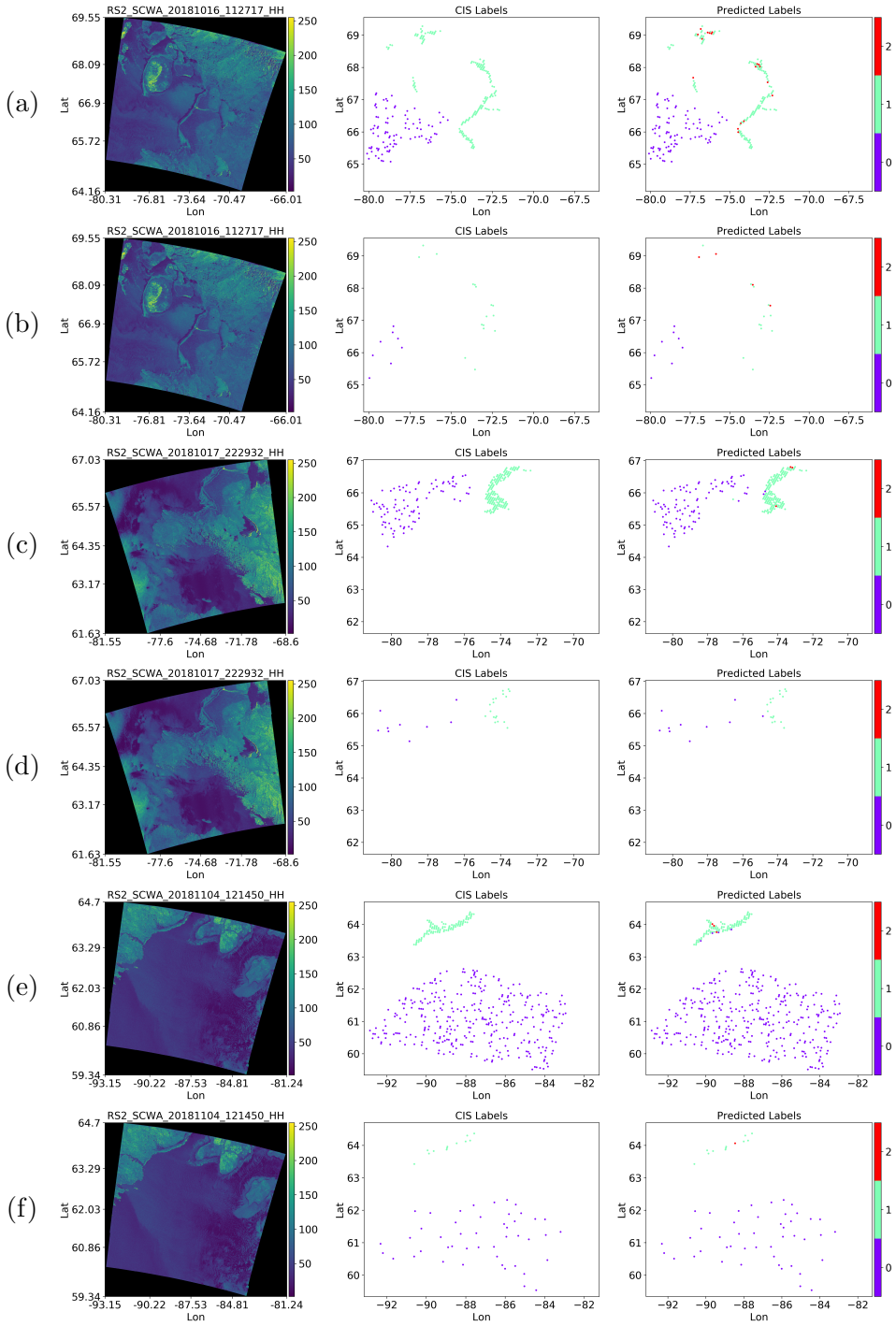


Figure 4.6: Training and testing samples from 3 SAR scenes and DenseNet predictions. (**a**, **b**) Training & testing samples from 2018/10/16 11:27:17, (**c**, **d**) Training & testing samples from 2018/10/17 22:29:32, (**e**, **f**) Training & testing samples from 2018/11/04 12:14:50.

restrictions of using "ice-free" only image analysis labels to represent Class 0 and choosing to only use egg codes with ice concentrations ≥9+ to represent Classes 1 and 2. Rationally, the ice concentration should decrease during the summer melt season as ice breaks up and a given area is occupied by more water. While this limits the conclusions that can be made about summer melt accuracy, it can be seen that all three classes had a testing accuracy over 90%, where the new ice class was the lowest at 90.75%. This suggests that the neural network had more difficulty recognizing features that can distinguish between ice types than it did in distinguishing between water and ice, and this was consistent with the analysis in Section 4.2.2 that compared results between the two class and three class experiments. This conclusion is also supported by the confusion matrices for the training and test sets shown in Table 4.8.

Table 4.8: DenseNet DBDDCA (3 class, 2 channel) confusion matrices.

		True Condition					
		Training			Testing		
		Class 0	Class 1	Class 2	Class 0	Class 1	Class 2
Prediction Condition	Class 0	11551	30	76	1434	5	17
	Class 1	8	10835	489	2	1334	87
	Class 2	64	747	11093	19	131	1334

It is clear that the neural network misclassified ice samples as other ice classes more than it misclassified them as water. As for the outliers of 17.24% and 0% for the training and testing accuracies that can be seen for Class 2 in the month of December in Table 4.7, these cases consisted of 58 and three samples, respectively, which was too small of a sample size for any conclusion to be made. Generally speaking however, the 91.75% ice accuracy achieved by DenseNet showed that neural networks can be used to effectively classify between new and first-year ice.

4.3 Discussion

It was shown that DenseNet accurately learns from expertly labelled data to classify between water and ice for the two class case, as well as water, new ice, and first-year ice for the three class case, using only uncalibrated HH and HV SAR images. The two class classifier provides a good baseline for the three class classifier as it sets upper-limit performance expectations. DenseNet consistently outperformed the U-Net encoder model, and it was observed that three channels in the input sample performed better than two channel experiments on the training set, but performed slightly worse on the test set. The visual representation of the predictions demonstrated that erroneous predictions could be rectified by sea ice experts based on the bulk of correctly classified samples that are in proximity to the erroneous predictions. An analysis of the class accuracy showed that the neural network would have a greater chance of producing erroneous predictions between ice types than it would between water and ice. No conclusions could be made concerning the accuracy of the model during the summer melt season due to the lack of representational data. Sea ice classification during the summer melt period has been attempted by others with little success. Our inability to assess the predictive quality of sea ice classification during the summer melt season using the tested neural networks herein was primarily due to the lack of representational data of all three classes during the month of June.

With the goal of producing accurate, high-resolution maps of Arctic marine conditions for industrial and marine transportation, it is also important to compare the results found in this study with existing systems and results from other published work, beyond the two class experiments that were conducted here to benchmark our own result. The study completed by [68] generated six distributions using the expectation-maximization algorithm, which were

used to label their dataset for training. While [68] used a different labelling scheme than the one presented in this chapter, a relatively similar behaviour was observed. The PCNN model used in their study to segment ice types also misclassified ice with other ice types, fast ice in particular. Unfortunately, no metrics were provided by the authors to assess the classification accuracy; therefore, no further comparison can be made with our results. The study completed by [69] used the WMO terminology [1] to label their dataset into six classes based on in situ observation. The six classes were smooth, medium, deformed first-year ice, young ice, nilas, and open water. The total accuracy achieved among these six classes was 91.2% with a neural network using a fusion of three types of data (ERS, RADARSAT SAR, and Meteor Visible) and noted that the addition of images in the visible spectrum helped improve classification between nilas and open water. While our results cannot be compared directly, due to the differences in the labelling methods and number of classes, the results found in [69] showed similar levels of accuracy.

With limited available studies using deep learning and CIS labelled data for comparison of ice type classification, comparisons with similar work for sea ice concentration are made next. The study presented in [65] used a DenseNet model to predict sea ice concentration from SAR imagery. The training labels used in that study were obtained from passive microwave data and determined by the ASI algorithm; the results were also compared to CIS charts. While this does not directly compare to the results presented herein, the error metrics used to evaluate performance achieved a similar error rate. Specifically, the test set error in [65] was 7.87%, while the overall error found in our results was 5.98%. This shows that DenseNet can be used effectively to achieve good results in two different applications relating to the same input data with different outputs, a testament to the flexibility and generalizability of the deep learning model. Similarly, the study presented in [67] also used

a convolutional neural network (though not the same as DenseNet or U-Net) to predict sea ice concentration from SAR imagery. Therein, the methodology used to generate and label the dataset was similar to the methodology presented here, the only difference being in the resulting output. Given that estimating sea ice concentration is a regression problem, the L1 mean error metric was used in [67] to report a 0.08 average E_{L1} score. While this metric is not comparable to classification accuracy, the low error score achieved supports accurate prediction of ice characteristics with deep neural networks using SAR imagery, which is a similar experimental finding of the work herein.

4.3.1 The Utility of DenseNet for Sea Ice Mapping

The methodology presented in this chapter can be applied towards the mapping of sea ice type in the Arctic by following a few steps. An incoming SAR image would be uniformly subdivided into 100 $\times$ 100 sub-regions without overlap as a first step. The collection of sub-regions would then be filtered such that sub-regions containing known land masses are removed from the collection. For accuracy purposes, it may also be desirable to exclude sub-regions having <80% of their pixels represent valid SAR data in order to imitate the samples used in the training set. The remaining sub-regions would then be processed and classified by the trained DenseNet model from this study into water, new ice, or first-year ice and assembled to form a segmented image. The time required for DenseNet to completely label a typical 10,000 $\times$ 10,000 pixel SAR image that has been pre-subdivided was approximately 40 s on a modern high-end GPU. This shows that DenseNet can automate the prediction of sea ice by type from SAR images to generate image analysis ice charts in a timely manner.

4.3.2 Future Work

The presented three class models for predicting CIS ice types from SAR imagery can be easily extended within the scope of the configurable parameters used to identify different datasets. As a first step to furthering the research to make it useful to operational sea ice services, the labelling type must be expanded to increase the number of ice classes. There are a total of 14 classes that CIS ice experts use to classify ice based on its stage of development [2]. Given the success that both neural networks used in this study achieved on the experiments conducted, in particular the three class experiments, a natural next step would be to assess the performance of these neural networks in a similar manner for incremental class definitions from three to 14 class setups. To increase the chances of obtaining a sufficient number of samples for each of the 14 classes, it may be necessary to relax the ice concentration criteria such that more egg codes are included. The effects of doing so should also be studied since this action will come at the cost of having impure samples in the sense that they will be a mix of water and ice. In addition to reducing purity requirements, class samples for each of the 14 ice classes can also be increased by expanding the region of interest and time of year. By expanding the space and time query options, a better assessment could be made about the ability for the proposed networks to accurately classify ice type during seasonal melt and fall freeze-up.

As supported ice type classes are expanded, it is expected that improvements to the dataset and model can also be made to improve classification performance. Staying within the scope of the configurable parameters used to identify datasets, further testing should be conducted to assess the effects of calibration and noise removal on the accuracy of ice classification. This can be accomplished by using additional information provided with the

SAR products detailing corrective gain values and noise signals [96]. Furthermore, including a third channel for the HV/HH polarization ratio improved the three class classification accuracy between ice types, at least on the training set. Therefore, future work should also include the effects of regularization techniques on the models, in an attempt to translate the improved training set performance to unseen data.

To strengthen the case that this proof of concept is a viable method to classify ice by type, future studies with more ice types should include K-cross-validation for each experiment to verify the predictive quality and generalizability of the models. The difference in accuracy between each experiment in this proof of concept was within the range caused by random initialization of the models. K-cross-validation would also reduce the uncertainty that the differences in accuracies are caused by random initialization.

Another avenue for future research goes outside the confines of the current framework by constructing a dataset better suited for the complete encoder-decoder U-Net architecture. Although the results for the three class experiments conducted were good, the models treated each sample sub-region as an independent sample. However, sub-regions that are close in proximity to each other will be logically dependent on one another, with similarities in ice conditions. This is where the U-Net architecture would be able to extract the spatial context of nearby samples to hopefully provide better segmentation results. U-Net was designed with this aspect in mind and uses skip connections between the encoder and decoder in order to localize and contextualize features from the macro-resolution of the encoder with the generated micro-resolution features from the decoder side. U-Net also generates feature maps at various scales with different contexts, which is important for SAR imagery as features at large scales tend to appear at finer resolutions as well.

While this is a lengthy list of extensions that can be made to this work, they were not applied; however, the proof of concept was sufficiently validated to show that there is merit in pursuing these changes/improvements for broader ice type classification.

4.4 Conclusions

The study presented in this chapter consisted of a proof of concept that neural networks could be used to classify ice types effectively using SAR imagery and CIS image analysis ice charts as labelled data. The SAR images were cropped into sub-regions per the latitude and longitude coordinates given for each ice sample egg code in the CIS image analysis ice chart, and each sub-region was treated as an independent sample. Experiments were conducted on datasets that had input samples consisting of a fusion of both HH and HV polarizations for each sub-region, and other experiments also added a third channel using the ratio of HV/HH. These datasets were labelled based on a simplification of the egg code samples from the CIS image analysis ice charts in two ways. The first labelling type was a simple water and ice categorization, which was used to benchmark experiments of the second labelling type, which consisted of three categories that described water, new ice, and first-year ice. Two neural networks were trained on these datasets, one of which was a modified U-Net architecture, and the other was a DenseNet. The DenseNet architecture achieved the highest overall accuracy of 94.02% and the highest ice accuracy of 91.75% on the three class dataset with the dual-pol HH and HV configuration. An analysis of the prediction results produced by DenseNet showed that the neural network experienced some more difficulty distinguishing between ice samples of different types than it did distinguishing between water and ice samples; however, the majority (>90%) of ice samples were still correctly classified. The lack of representative data for the three classes for the months of June,

October, November, and December, important to seasonal classification of sea ice melt and formation, resulted in an inability to draw conclusions about the performance of the model with respect to seasonality. To resolve this issue, further experimentation is necessary to test if the classification of ice type in the summer and fall season can be established through a relaxation of the ice concentration criteria by including impure samples (in the sense that they will be a mix of water and ice). The results presented in this work validate the proof of concept that a neural network can be used to effectively automate the pre-categorization of different ice types based on SAR imagery and segment a typical 10,000 $\times$ 10,000 SAR image in a timely manner for further expert analysis.

Chapter 5

Conclusions

To conclude this thesis, the objectives set out in the introduction are evaluated. The scientific objective was to determine whether or not neural networks are capable of predicting geophysical quantities using remote sensing and expertly labelled data. Both applications chosen relate to the remote sensing of geophysical quantities that are impacted by the climate, namely soil moisture and sea ice type. A summary of the findings regarding the two applications evaluated as part of this work are used to support that the scientific objective was also achieved. Additionally, my personal objectives were to develop an understanding of deep learning theory, implement deep learning techniques and apply them towards real world problems. A summary of the concepts used in deep learning that were introduced in the background chapter is reviewed to show that this objective was achieved. This work is then concluded with a summary of the avenues where future work and improvements can be pursued in the applications presented, which were also highlighted and discussed more in depth in Sections 3.3.1 and 4.3.2.

5.1 Evaluation of Thesis Goals

Two different applications were used to achieve the scientific objective of this thesis. The first application presented was the estimation of soil moisture from satellite imagery. The chosen method used to perform this task consisted of an image-to-image model where SAR data from the Sentinel-1 satellite was used as the input data and the labelled data used to train

the model was soil moisture maps from the SMAP satellite. It was found that increasing the size of the dataset and the number of trainable parameters in the neural network led to the greatest gains in correlation leading up to the best performing experiment. The best performing experiment used U-Net on a dataset of 1,034 images to achieve a testing MSE of 3.82e-03 and a testing correlation of 0.792 respectively. These results are good as they are inline with the correlations achieved on the soil moisture products validated by NASA, which were produced by the SMAP satellite and used herein as labelled data. These results are also comparable to other findings in the soil moisture literature as discussed in Section 3.3. With these comparisons in mind, this thesis has shown that it is possible to estimate soil moisture from satellite data using deep learning techniques and achieve comparable results to different empirical techniques used in other studies.

The second application aimed to classify sea ice by type, also from satellite imagery. Here, RADARSAT-2 SAR data was used as the input and the stage of development feature from sea ice charts produced by the CIS were used as the labelled data. The best performing experiment for this application used DenseNet and achieved an overall classification accuracy of 94.0% and a 91.8% ice accuracy on a dataset of 43,617 samples where the three classes of ice type (water, new ice, and first-year ice) each had 14,539 samples. Due to there being few studies that used a similar methodology and classes as presented herein, a two class experiment, which consisted of water and ice, was conducted to benchmark the results of the three class experiment conducted. The benchmark achieved an overall accuracy of 99.07% and an ice accuracy of 98.65%, which demonstrated an upper limit of accuracy to expect as the problem becomes more complex with the addition of more classes of sea ice. The results of the three class experiment are therefore found to be good considering the benchmark only achieved a few more percentage points. Further analysis using a confusion matrix showed

that the source of error in the three class experiment was tied more closely to distinguishing between ice types than it did between water and ice, but overall more than 90% of the samples were still classified correctly. In terms of other studies, one similar study was made by Bogdanov et. al. [69] where six classes were used and a 91.2% accuracy was achieved using a neural network. This shows that the results achieved herein are comparable in nature, even though a direct comparison cannot be made due to the differences in the number of classes used. Given that there are not many other comparable studies for predicting sea ice type using CIS labels, the two class experiment and high level comparison to the study conducted by Bogdanov et. al. [69] has been used to show that the results achieved herein are good results. As such, predicting sea ice type using satellite data and neural networks has also been shown to be possible using the methodology presented in this thesis.

An interesting perspective on the work done in this thesis can be extracted by analyzing the differences and similarities in the methodologies used to predict soil moisture and sea ice types using deep learning models. The method used in the sea ice application differs from the soil moisture application due to the absence of readily available segmented ice chart images. For this reason, an image-to-image model was not trained as it was in the soil moisture application, but instead sub-regions in the RADARSAT-2 images were used as the input and geolocated points with a given ice type were used as the labelled output. Despite this difference, both applications use satellite SAR data to predict two different physical characteristics by simply providing the networks with different labelled data. The fact that the neural networks tested in this work achieved good results on two different problems and be given similar input data, but learn different meanings based on the labelled data is a testament to the versatility of neural networks. Through this observation and the results presented, the evidence needed to support the scientific objective of the thesis has

been further reinforced.

Regarding the personal objective of the thesis, many neural networks were implemented in Keras and applied to two real world problems; the estimation of soil moisture and the classification of sea ice by type. Through the implementation of these networks, an understanding of deep learning and its constitutive components, the neural network and the training process, was gained. Deep learning can be described as a neural network that learns a transformation between two sets of data, an input and labelled output, by minimizing a loss function using a gradient descent based optimizer. As described in the background chapter of this thesis, the optimizer requires gradients of the loss function with respect to the model parameters in order to learn and update the parameters to best fit the data. This is accomplished by first calculating the loss through a forward pass of the neural network, and then calculating the gradients during a backward pass, also called backpropagation, which is derived from the chain rule since neural networks can be represented as a composition of functions. Once the forward and backward passes are complete, the optimizer updates the model parameters and this is continued until no meaningful improvement is made to the loss. While there are many loss functions to choose from based on the application, this work was exposed to loss functions belonging to the two main categories of machine learning, which are regression and classification. Additionally, there are many types of neural networks, but the two applications used as part of this thesis motivated the choice to use CNNs due to the image based nature of the data as well as the spatial dependence of neighbouring pixels in the SAR images. CNNs are typically composed of convolution layers, dense layers, and pooling layers, all of which were further discussed in the background chapter. My personal objectives for this thesis have therefore also been satisfied given the comprehensive description of CNNs and the training process.

5.2 Summary of Future Work

Future work for the soil moisture application is described first in order to summarize the discussion on this topic that was made in Sections 3.3.1 and 4.3.2. As stated earlier, the two most significant improvements to the results achieved for the soil moisture application were obtained by increasing the size of the dataset as well as the number of trainable parameters in the neural networks used. The logical next step towards improvement is therefore to continue with these two options until a point of saturation is achieved. Once this has been reached, further improvements, which were highlighted in other studies, include the addition of auxiliary data, such as NDVI as well as the addition of other polarization channels and incidence angle calibration. Another point of improvement discussed was the validation or refinement of the models using in-situ SMC measurements from the ISMN, but this could involve significant data processing challenges.

In terms of the sea ice application, the next step towards improvement is to increase the number of classes with which to distinguish types of ice for the application to be useful for operational purposes. Doing so would extend the number of ice types to 14 if using all stage of development features defined by the CIS. Other improvements discussed include the inclusion of calibration and noise removal from the raw SAR data used herein. It is also important for the proof of concept of being able to predict sea ice type from satellite data using deep learning that this can also be accomplished during summer melt periods, where traditional methods have experienced difficulties in doing so. Points of discussion were put forward about the lack of representative data for the three classes used during the months of June, October, November, and December, which are important to seasonal classification of sea ice melt and formation. Therefore obtaining more data of the three classes during these

months to determine whether or not distinguishing ice types during summer melt is possible is also an important step for future work in this application.

General deep learning related experimentation can also be pursued for future work. Examples of deep learning parameters to experiment with include model hyper-parameters, such as the number of filters or neurons to use in a specific layer or another consists of varying the probability of dropout. Others options can be the use of different models, loss functions, and optimizers to fine tune results. Different types of regularization techniques besides dropout can also be explored. Another way to make the work presented herein more robust is to conduct K-cross validation experiments once it is determined that the dataset size is sufficient and that adding more data only leads to marginal improvements. While it is unclear how to vary many of these hyper-parameters and model configurations, therein lies the fun and mystery of deep learning and its potential.

Bibliography

[1] World Meteorological Organization, *WMO Sea Ice Nomenclature (WMO No. 259, Volume I - Terminology and Codes, Volume II - Illustrated Glossary, Volume III - International, System of Sea Ice Symbols*. Secretariat of the World Meteorological Organization, 2014.

[2] MANICE, CIS, *Manual of standard procedures for observing and reporting ice conditions*, 9th, Ottawa, ON, Canada: Canadian Ice Service (CIS), Meteorological Service of Canada, Jun. 2005.

[3] Natural Resources Canada, *Fundamentals of remote sensing*, 2007.

[4] J. B. Campbell and R. H. Wynne, *Introduction to remote sensing*. Guilford Press, 2011.

[5] NASA, *What is remote sensing?* Nov. 2020. [Online]. Available: https://earthdata.nasa.gov/learn/remote-sensing.

[6] NASA, *Remote sensors*, Nov. 2020. [Online]. Available: https://earthdata.nasa.gov/learn/remote-sensors.

[7] F. Ulaby and D. Long, *Microwave radar and radiometric remote sensing*. Artech House, 2015.

[8] E. F. Knott, *Radar cross section measurements*. Springer Science & Business Media, 2012.

[9] W. Emery and A. Camps, *Introduction to Satellite Remote Sensing - Atmosphere, Ocean, Cryosphere and Land Applications*. Elsevier, 2017, pp. 291–453, ISBN: 978-0-12-809254-5. DOI: https://doi.org/10.1016/B978-0-12-809254-5.00005-1.

[10] ESA, European Space Agency, "User Guides - Definitions," [Online]. Available: https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/definitions.

[11] K. He, X. Zhang, S. Ren, and J. Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, 2015. arXiv: 1502.01852 [cs.CV].

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, *Imagenet large scale visual recognition challenge*, 2015. arXiv: 1409.0575 [cs.CV].

[13] Tesla, *Autopilot ai*. [Online]. Available: https://www.tesla.com/en_CA/autopilotAI?redirect=no.

[14] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, *Language models are few-shot learners*, 2020. arXiv: 2005.14165 [cs.CL].

[15] J. Zornoza, *The mighty gpt-3*, Sep. 2020. [Online]. Available: https://towardsdatascience.com/the-mighty-gpt-3-1cb6ee477932.

[16] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, *Emergent tool use from multi-agent autocurricula*, 2020. arXiv: 1909.07528 [cs.LG].

[17] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. R. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, P. Kohli, D. T. Jones, D. Silver, K. Kavukcuoglu, and D. Hassabis, "Improved protein structure prediction using potentials from deep learning," *Nature*, vol. 577, no. 7792, pp. 706–710, 2020, ISSN: 0028-0836. DOI: 10.1038/s41586-019-1923-7.

[18] R. Iten, T. Metger, H. Wilming, L. del Rio, and R. Renner, "Discovering physical concepts with neural networks," *Physical Review Letters*, vol. 124, no. 1, Jan. 2020, ISSN: 1079-7114. DOI: 10.1103/physrevlett.124.010508. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.124.010508.

[19] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969.

[20] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," *Ph. D. dissertation, Harvard University*, 1974.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," eng, *Nature (London)*, vol. 323, no. 6088, pp. 533–536, 1986, ISSN: 1476-4687.

[22] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[23] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[25] A. Karpathy and L. Fei-Fei, *Deep visual-semantic alignments for generating image descriptions*, 2015. arXiv: 1412.2306 [cs.CV].

[26] T. Young, D. Hazarika, S. Poria, and E. Cambria, *Recent trends in deep learning based natural language processing*, 2018. arXiv: 1708.02709 [cs.CL].

[27] D. Kwon, K. Natarajan, S. C. Suh, H. Kim, and J. Kim, "An empirical study on network anomaly detection using convolutional neural networks," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 2018, pp. 1595–1598. DOI: 10.1109/ICDCS.2018.00178.

[28] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: A review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, Mar. 2019, ISSN: 1573-756X. DOI: 10.1007/s10618-019-00619-1. [Online]. Available: http://dx.doi.org/10.1007/s10618-019-00619-1.

[29] A. Araujo, W. Norris, and J. Sim, "Computing receptive fields of convolutional neural networks," *Distill*, vol. 4, no. 11, e21, 2019. [Online]. Available: https://distill.pub/2019/computing-receptive-fields/.

[30] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Artificial Neural Networks – ICANN 2010*, K. Diamantaras, W. Duch, and L. S. Iliadis, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 92–101, ISBN: 978-3-642-15825-4.

[31] A. Cauchy, "Methode generale pour la resolution des systemes d'equations simultanees," *C.R. Acad. Sci. Paris*, vol. 25, pp. 536–538, 1847. [Online]. Available: https://ci.nii.ac.jp/naid/10026863174/en/.

[32] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[33] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[34] C. Gaudreau, "Fast detection of bees using deep learning and bayesian optimization," *M.Sc. thesis, University of Manitoba*, 2018.

[35] Python Software Foundation, *Python*, version 3.6. [Online]. Available: https://www.python.org/.

[36] F. Chollet *et al.*, *Keras*, https://keras.io, 2015.

[37] N. Baghdadi, M. Aubert, O. Cerdan, L. Franchistéguy, C. Viel, M. Eric, M. Zribi, and J. F. Desprats, "Operational mapping of soil moisture using synthetic aperture radar data: Application to the touch basin (france)," eng, *Sensors (Basel, Switzerland)*, vol. 7, no. 10, pp. 2458–2483, 2007, ISSN: 1424-8220.

[38] J. Shi, J. Van Zyl, J. V. Soares, and E. T. Engman, "Development of soil moisture retrieval algorithm for l-band sar measurements," in *Proceedings of the Geoscience and Remote Sensing Symposium IGARSS*, vol. 1992, 1992, pp. 495–497.

[39] J. Shi, J. Wang, A. Y. Hsu, P. E. O'Neill, and E. T. Engman, "Estimation of bare surface soil moisture and surface roughness parameter using l-band sar image data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 35, no. 5, pp. 1254–1266, 1997.

[40] S. Paloscia, S. Pettinato, E. Santi, C. Notarnicola, L. Pasolli, and A. Reppucci, "Soil moisture mapping using sentinel-1 images: Algorithm and preliminary validation," eng, *Remote sensing of environment*, vol. 134, pp. 234–248, 2013, ISSN: 0034-4257.

[41] G. Macelloni, S. Paloscia, P. Pampaloni, S. Sigismondi, P. De Matthaeis, P. Ferrazzoli, G. Schiavon, and D. Solimini, "The sir-c/x-sar experiment on montespertoli: Sensitivity to hydrological parameters," eng, *International Journal of Remote Sensing*, vol. 20, no. 13, pp. 2597–2612, 1999, ISSN: 0143-1161.

[42] S. Paloscia, P. Pampaloni, S. Pettinato, and E. Santi, "A comparison of algorithms for retrieving soil moisture from envisat/asar images," eng, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 10, pp. 3274–3284, 2008, ISSN: 0196-2892.

[43] Y. Oh, K. Sarabandi, and F. Ulaby, "An empirical model and an inversion technique for radar scattering from bare soil surfaces," eng, *IEEE transactions on geoscience and remote sensing*, vol. 30, no. 2, pp. 370–381, 1992, ISSN: 0196-2892.

[44] A. K. Fung, Z. Li, and K. S. Chen, "Backscattering from a randomly rough dielectric surface," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 30, no. 2, pp. 356–369, 1992.

[45] P. C. Dubois, J. van Zyl, and T. Engman, "Measuring soil moisture with imaging radars," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 33, no. 4, pp. 915–926, 1995.

[46] M. Choker, N. Baghdadi, M. Zribi, M. El Hajj, S. Paloscia, N. Verhoest, H. Lievens, and F. Mattia, "Evaluation of the oh, dubois and iem backscatter models using a large dataset of sar data and experimental soil measurements," eng, *Water (Basel)*, vol. 9, no. 1, pp. 38–, 2017, ISSN: 2073-4441.

[47] W. Wagner, G. Lemoine, M. Borgeaud, and H. Rott, "A study of vegetation cover effects on ers scatterometer data," eng, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, no. 2, pp. 938–948, 1999, ISSN: 0196-2892.

[48] M. Zribi, A. Chahbi, M. Shabou, Z. Lili-Chabaane, B. Duchemin, N. Baghdadi, R. Amri, and A. Chehbouni, "Soil surface moisture estimation over a semi-arid region using envisat asar radar data for soil evaporation evaluation," eng, *Hydrology and earth system sciences*, vol. 15, no. 1, pp. 345–358, 2011, ISSN: 1607-7938.

[49] A. Balenzano, F. Mattia, G. Satalino, V. Pauwels, and P. Snoeij, "Smosar algorithm for soil moisture retrieval using sentinel-1 data," in *2012 IEEE International Geoscience and Remote Sensing Symposium*, IEEE, 2012, pp. 1200–1203.

[50] D. Alexakis, F.-D. Mexis, A.-E. Vozinaki, I. Daliakopoulos, and I. Tsanis, "Soil moisture content estimation based on sentinel-1 and auxiliary earth observation products. a hydrological approach," *Sensors*, vol. 17, no. 6, p. 1455, 2017.

[51] N. Das, D. Entekhabi, R. S. Dunbar, S. Kim, S. Yueh, A. Colliander, P. E. O'Neill, and T. Jackson, National Snow and Ice Data Center, "SMAP/Sentinel-1 L2 Radiometer/Radar 30-Second Scene 3 km EASE-Grid Soil Moisture, Version 2," Note: Information under the User Guide tab, Mar. 31, 2015, [Online]. Available: https://nsidc.org/data/SPL2SMAP_S/versions/2.

[52] Y. H. Kerr, P. Waldteufel, P. Richaume, J. P. Wigneron, P. Ferrazzoli, A. Mahmoodi, A. Al Bitar, F. Cabot, C. Gruhier, S. E. Juglea, *et al.*, "The smos soil moisture retrieval algorithm," *IEEE transactions on geoscience and remote sensing*, vol. 50, no. 5, pp. 1384–1403, 2012.

[53] N. N. Das, D. Entekhabi, S. Dunbar, S. Kim, S. Yueh, A. Colliander, T. Jackson, P. O'Neill, M. Cosh, T. Caldwell, *et al.*, "Assessment report for the l2_sm_sp beta release data products," Nov. 1, 2017, [Online]. Available: https://nsidc.org/sites/nsidc.org/files/technical-references/SMAPSPBeta%2BReleaseAssessmentReport_01-18-2019_final.pdf.

[54] ESA, European Space Agency, "User Guides - Sentinel-1 SAR - Level-1 Ground Range Detected - Sentinel Online," [Online]. Available: https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/resolutions/level-1-ground-range-detected.

[55] R. Kruk, M. C. Fuller, A. S. Komarov, D. Isleifson, and I. Jeffrey, "Proof of concept for sea ice stage of development classification using deep learning," *Remote Sensing*, vol. 12, no. 15, p. 2486, 2020.

[56] N. A. Carter, J. Dawson, J. Joyce, and A. Ogilvie, *Arctic corridors and northern voices: Governing marine transportation in the canadian arctic (arviat, nunavut com-*

munity report), 2017. DOI: 10.20381/RUOR36924. [Online]. Available: https://ruor.uottawa.ca/handle/10393/36924.

[57] A. S. Komarov, A. Caya, M. Buehner, and L. Pogson, "Assimilation of sar ice and open water retrievals in environment and climate change canada regional ice-ocean prediction system," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 6, pp. 4290–4303, 2020.

[58] S. E. Howell, D. Small, C. Rohner, M. S. Mahmud, J. J. Yackel, and M. Brady, "Estimating melt onset over arctic sea ice from time series multi-sensor sentinel-1 and radarsat-2 backscatter," *Remote Sensing of Environment*, vol. 229, pp. 48–59, 2019.

[59] J. J. Yackel, J. P. Gill, T. Geldsetzer, M. C. Fuller, and V. Nandan, "Diurnal scale controls on c-band microwave backscatter from snow-covered first-year sea ice during the transition from late winter to early melt," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 7, pp. 3860–3874, 2017.

[60] D. Isleifson, B. Hwang, D. G. Barber, R. K. Scharien, and L. Shafai, "C-band polarimetric backscattering signatures of newly formed sea ice during fall freeze-up," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 8, pp. 3256–3267, Aug. 2010, ISSN: 0196-2892. DOI: 10.1109/TGRS.2010.2043954.

[61] L. Soh and C. Tsatsoulis, "Unsupervised segmentation of ers and radarsat sea ice images using multiresolution peak detection and aggregated population equalization," English (US), *International Joural of Remote Sensing*, vol. 20, no. 15-16, pp. 3087–3109, Jan. 1, 1999, ISSN: 0143-1161. DOI: 10.1080/014311699211633.

[62] B. Scheuchl, I. Cumming, and I. Hajnsek, "Classification of fully polarimetric single- and dual-frequency sar data of sea ice using the wishart statistics," *Canadian Journal of Remote Sensing*, vol. 31, no. 1, pp. 61–72, 2005. DOI: 10.5589/m04-060. eprint: https://doi.org/10.5589/m04-060. [Online]. Available: https://doi.org/10.5589/m04-060.

[63] M. Kim, J. Im, H. Han, J. Kim, S. Lee, M. Shin, and H.-C. Kim, "Landfast sea ice monitoring using multisensor fusion in the antarctic," *GIScience & Remote Sensing*, vol. 52, no. 2, pp. 239–256, 2015. DOI: 10.1080/15481603.2015.1026050. eprint: https://doi.org/10.1080/15481603.2015.1026050. [Online]. Available: https://doi.org/10.1080/15481603.2015.1026050.

[64] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017. DOI: 10.1109/cvpr.2017.243. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2017.243.

[65] C. L. V. Cooke and K. A. Scott, "Estimating sea ice concentration from sar: Training convolutional neural networks with passive microwave data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 7, pp. 4735–4747, 2019.

[66] G. Spreen, L. Kaleschke, and G. Heygster, "Sea ice remote sensing using amsr-e 89-ghz channels," eng, *Journal of Geophysical Research - Oceans*, vol. 113, no. C2, C02S03–n/a, 2008, ISSN: 0148-0227.

[67] L. Wang, K. A. Scott, L. Xu, and D. A. Clausi, "Sea ice concentration estimation during melt from dual-pol sar scenes using deep convolutional neural networks: A case study," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 8, pp. 4524–4533, 2016.

[68] J. A. Karvonen, "Baltic sea ice sar segmentation and classification using modified pulse-coupled neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 42, no. 7, pp. 1566–1574, 2004.

[69] A. V. Bogdanov, S. Sandven, O. M. Johannessen, V. Y. Alexandrov, and L. P. Bobylev, "Multisensor approach to automated classification of sea ice image data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 7, pp. 1648–1664, 2005.

[70] ESA, European Space Agency, "Copernicus Open Access Hub," [Online]. Available: https://scihub.copernicus.eu/.

[71] N. Das, D. Entekhabi, R. S. Dunbar, S. Kim, S. Yueh, A. Colliander, P. E. O'Neill, and T. Jackson, National Snow and Ice Data Center, "SMAP/Sentinel-1 L2 Radiometer/Radar 30-Second Scene 3 km EASE-Grid Soil Moisture, Version 2," Note: Information under the User Guide tab, Mar. 31, 2015, [Online]. Available: https://doi.org/10.5067/KE1CSVXMI95Y.

[72] ESA, European Space Agency, "User Guides - Overview," [Online]. Available: https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/overview.

[73] ESA, European Space Agency, "User Guides - Polarimetry," [Online]. Available: https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/product-overview/polarimetry.

[74] ESA, European Space Agency, "User Guides - Definitions," [Online]. Available: https://sentinel.esa.int/web/sentinel/user-guides/sentinel-1-sar/product-types-processing-levels/level-1.

[75] ESA, European Space Agency, "Sentinel-1 Product Definition," [Online]. Available: https://sentinel.esa.int/documents/247904/1877131/Sentinel-1-Product-Definition.

[76] ESA, European Space Agency, "Masking "No-value" Pixels on GRD Products generated by the Sentinel-1 ESA IPF," [Online]. Available: https://sentinels.copernicus.eu/documents/247904/2142675/Sentinel-1-masking-no-value-pixels-grd-products-note.

[77] A. S. Facility, Alaska Satellite Facility, "Geocoding sentinel-1 grd products using gdal utilities," Aug. 30, 2017, [Online]. Available: https://media.asf.alaska.edu/uploads/pdf/grd_geocoding-v4.pdf.

[78] Environmental Systems Research Institute, *Arcgis*. [Online]. Available: https://www.arcgis.com.

[79] Open Source Geospatial Foundation Project, *Qgis*. [Online]. Available: http://qgis.org/.

[80] GDAL/OGR contributors, *GDAL/OGR geospatial data abstraction software library*, Open Source Geospatial Foundation, 2018. [Online]. Available: http://gdal.org.

[81] N. Das, D. Entekhabi, R. S. Dunbar, S. Kim, S. Yueh, A. Colliander, P. E. O'Neill, and T. Jackson, National Snow and Ice Data Center, "Data Fields - SMAP/Sentinel-1 L2 Radiometer/Radar 3 km EASE-Grid Soil Moisture, Version 2," Mar. 31, 2015, [Online]. Available: https://nsidc.org/data/smap/spl2smap_s/data-fields/.

[82] T. L. Zeiler, NASA, "Heghome," Nov. 5, 2018, [Online]. Available: https://newsroom.gsfc.nasa.gov/sdptoolkit/HEG/HEGHome.html.

[83] T. L. Zeiler, NASA, "Batch processing," Nov. 5, 2018, [Online]. Available: https://newsroom.gsfc.nasa.gov/sdptoolkit/HEG/HEGBatchProcessing.html.

[84] GDAL/OGR contributors, *GDAL/OGR geospatial data abstraction software library*, Open Source Geospatial Foundation, 2018. [Online]. Available: https://www.gdal.org/gdalwarp.html.

[85] W. Dorigo, W. Wagner, R. Hohensinn, S. Hahn, C. Paulik, A. Xaver, A. Gruber, M. Drusch, S. Mecklenburg, P. Oevelen, A. Robock, and T. Jackson, "The international soil moisture network: A data hosting facility for global in situ soil moisture measurements," en, *Hydrology and Earth System Science*, vol. 15, pp. 1675–1698, 2011. DOI: 10.5194/hess-15-1675-2011. [Online]. Available: https://ismn.geo.tuwien.ac.at/en/.

[86] K. M. Ting, "Encyclopedia of machine learning and data mining," in, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2017, pp. 260–260, ISBN: 978-1-4899-7687-1. DOI: 10.1007/978-1-4899-7687-1_50. [Online]. Available: https://doi.org/10.1007/978-1-4899-7687-1_50.

[87] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing*

and Computer-Assisted Intervention, Springer, 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28. [Online]. Available: http://arxiv.org/abs/1505.04597.

[88] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, Apr. 2017, ISSN: 2160-9292. DOI: 10.1109/tpami.2016.2572683. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2016.2572683.

[89] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, *A Comprehensive Survey on Transfer Learning*, 2020. arXiv: 1911.02685 [cs.LG].

[90] J. Andrews, D. Babb, and D. G. Barber, "Climate change and sea ice: Shipping accessibility on the marine transportation corridor through hudson bay and hudson strait (1980–2014)," *Elem Sci Anth*, vol. 5, 2017.

[91] I. Martini, "Coastal features of canadian inland seas," in *Elsevier oceanography series*, vol. 44, Elsevier, 1986, pp. 117–142.

[92] K. P. Hochheim and D. G. Barber, "An update on the ice climatology of the hudson bay system," *Arctic, antarctic, and alpine research*, vol. 46, no. 1, pp. 66–83, 2014.

[93] M. L. Harasyn, D. Isleifson, W. Chan, and D. G. Barber, "Multi-scale observations of the co-evolution of sea ice thermophysical properties and microwave brightness temperatures during the summer melt period in hudson bay," *Elem Sci Anth*, vol. 8, no. 1, 2020.

[94] C. L. Parkinson, D. J. Cavalieri, P. Gloersen, H. J. Zwally, and J. C. Comiso, "Arctic sea ice extents, areas, and trends, 1978–1996," *Journal of Geophysical Research: Oceans*, vol. 104, no. C9, pp. 20 837–20 856, 1999.

[95] B. Slade, "Radarsat-2 product description," *RN-SP-52-1238*, no. 1/6, 2018.

[96] D. MacDonald *et al.*, "Ltd.,"radarsat-2 product format definition,"," *RN-RP-51-2713*, no. 1/7, 2008.

[97] D. MacDonald *et al.*, "Geolocation of RADARSAT-2 georeferenced products," Tech. Rep. RN-TN-53-0076, MDA Corporation, Richmond, BC, Canada, Tech. Rep., 2010.

[98] M. Siam, M. Gamal, M. Abdel-Razek, S. Yogamani, and M. Jagersand, *RTSeg: Real-time Semantic Segmentation Comparative Study*, 2018. arXiv: 1803.02758 [cs.CV].

[99] A. Singh, H. Kalke, M. Loewen, and N. Ray, *River ice segmentation with deep learning*, 2019. arXiv: 1901.04412 [cs.CV].

[100] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, and P. J. Kennedy, "Training deep neural networks on imbalanced data sets," eng, in *2016 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2016, pp. 4368–4374, ISBN: 1509006206.

[101] G. Pleiss, D. Chen, G. Huang, T. Li, L. van der Maaten, and K. Q. Weinberger, *Memory-Efficient Implementation of DenseNets*, 2017. arXiv: 1707.06990 [cs.CV].

[102] D. Masters and C. Luschi, *Revisiting Small Batch Training for Deep Neural Networks*, 2018. arXiv: 1804.07612 [cs.LG].