2024

Statistically Principled Deep Learning for SAR Image Segmentation

Cassandra Goldberg
Bowdoin College

Statistically Principled Deep Learning for SAR Image Segmentation

An Honors Paper for the Department of Computer Science

By Cassandra Goldberg

Bowdoin College, 2024

BOWDOIN COLLEGE

Abstract

Department of Computer Science

Statistically Principled Deep Learning for SAR Image Segmentation

by Cassandra Goldberg

This project explores novel approaches for Synthetic Aperture Radar (SAR) image segmentation that integrate established statistical properties of SAR into deep learning models. First, Perlin Noise and G^0 sampling methods were utilized to generate a synthetic dataset that effectively captures the statistical attributes of SAR data. Subsequently, deep learning segmentation architectures were developed that utilize average pooling and 1x1 convolutions to perform statistical moment computations. Finally, supervised and unsupervised disparity-based losses were incorporated into model training. The experimental outcomes yielded promising results: the synthetic dataset effectively trained deep learning models for real SAR data segmentation, the statistically-informed architectures demonstrated comparable or superior performance to benchmark models, and the unsupervised disparity-based loss facilitated the delineation of regions within the SAR data. These findings indicate that employing statistically-informed deep learning methodologies could enhance SAR image analysis, with broader implications for various remote sensing applications and the general field of computer vision. The code developed for this project can be found here: `https://github.com/cgoldber/Statistically-Principled-SAR-Segmentation.git`.

Acknowledgements

I would like to express my deepest gratitude to my project advisor, Professor Jeová Farias, for his invaluable guidance, mentorship, and encouragement throughout this research endeavor. His expertise was instrumental in shaping this project, and I always genuinely looked forward to our weekly meetings, where I was continually inspired by his kindness and infectious passion for the subject matter.

Thank you to the Bowdoin Computer Science Department for providing me with the opportunity to undertake this research project. I also want to shout out DJ Merrill, Bowdoin's Director of High Performance Computing (HPC), for patiently fielding my endless barrage of emails and helping me to navigate all of the computational challenges I faced throughout this project.

I am incredibly appreciative of my friends and family for their constant support, whether it was through lending a patient ear, offering words of encouragement, or simply being a source of joy in my life. Sorry that I still don't know how to help you troubleshoot your day-to-day computer issues!

Finally, to my resilient computer: thank you for enduring the countless hours of code debugging, model training, and occasional frustration-induced keyboard-pounding temper tantrums. I truly could not have done it without you!

Contents

Abstract **iii**

Acknowledgements **v**

1 Introduction **1**
1.1 Motivation 1
1.2 Proposal 1

2 Background **3**
2.1 Synthetic Aperture Radar (SAR) 3
2.1.1 SAR Data 3
2.1.2 Applications 4
2.2 Relevant Statistics 5
2.2.1 Fundamental Statistics Concepts 5
2.2.2 Parameter Estimation 6
2.2.3 Stochastic Distances 7
2.2.4 Statistical Characteristics of SAR 7
2.3 Deep Learning and Computer Vision 8
2.3.1 Deep Learning Basics 8
2.3.2 Convolutional Neural Networks 10
2.3.3 Image Segmentation 12
2.4 Techniques for SAR Parameter Estimation and Segmentation 14
2.4.1 Statistical Techniques 14
2.4.2 Deep Learning Techniques 14

3 Methodology **17**
3.1 Synthetic Dataset Generation 17
3.2 Statistically-Motivated Segmentation Architectures 19
3.3 Disparity-Based Loss 24

4 Results **27**
4.1 Experimental Setup 27
4.1.1 Datasets 27
4.1.2 Data Preprocessing 28
4.1.3 Hyperparameters for Model Training 28
4.1.4 Evaluation Metrics 28
4.2 Generated Synthetic Dataset 29
4.3 Statistically-Motivated Architecture Performance 32
4.3.1 Architectures Trained on Synthetic Data: Quantitative 32
4.3.2 Architectures Trained on Synthetic Data: Qualitative 33
4.3.3 Architectures Trained and Tested on Optical SAR Imagery 37
4.3.4 Architectures Trained on Synthetic Data and Tested on Optical SAR Imagery 39
4.4 Disparity-Based Loss Performance 39

4.4.1 Supervised Approach: Combining Cross-Entropy and Disparity-Based Losses 39
4.4.2 Unsupervised Approach: Disparity-Based Loss 41

5 Discussion 43
5.1 Synthetic Dataset 43
5.2 Statistically-Motivated Segmentation Architectures 43
5.2.1 Benchmark Architectures 43
5.2.2 Architectures Employing Branches in Parallel with an Autoencoder 44
5.2.3 Architectures Employing Statistically-Motivated Computations in the Encoder of a U-Net 44
5.2.4 Architectures Employing Statistically-Motivated Computations in the Skip Connections of a U-Net 45
5.2.5 Architectures Employing Larger Kernels in the Encoder 45
5.2.6 Poorly Performing Models 45
5.2.7 Architecture Performance When Trained and Tested on Optical SAR Imagery 46
5.2.8 Synthetically-Trained Model Performance on Optical SAR Imagery 47
5.3 Disparity-Based Loss 47
5.3.1 Supervised Approach 47
5.3.2 Unsupervised Approach 48

6 Conclusion 49
6.1 Overview 49
6.2 Future Recommendations 50

A Architectures Tested 51
A.1 Helper Blocks 51
A.2 Experimental Encoder Blocks 54
A.3 High-Level Architectures 55

Bibliography 61

Chapter 1

Introduction

1.1 Motivation

Remote sensing offers a comprehensive perspective of the Earth's surface from distant vantage points, serving as an extremely valuable tool for environmental monitoring. Synthetic Aperture Radar (SAR), one form of remote sensing, operates by emitting wave signals from an airborne platform and capturing reflections to produce high-resolution images. Since this technique functions regardless of weather conditions and sunlight, it delivers extremely valuable remote sensing insights. Another noteworthy characteristic of SAR lies in its exhibition of established statistical properties that have been rigorously defined and explored in previous research endeavors.

Historically, SAR analysis has relied predominantly on statistical models, achieving notable success in this domain. Recent advancements in deep learning strategies have emerged as versatile and effective tools for analyzing remote sensing data. Such algorithms are much quicker than traditional models and can be applied to a variety of tasks. However, the current state-of-the-art in deep learning SAR image segmentation encounters challenges posed by the limited availability of labeled data. Another challenge arises from the intrinsic speckle characteristics of SAR data, which can degrade the quality of segmentation results and hinder model performance.

Given SAR's widespread usage, improving techniques for SAR analysis is crucial for advancing environmental monitoring, disaster management, urban planning, defense, security, and scientific research. Advancements in SAR image analysis also have broader implications for the field of computer vision. These advancements contribute to the development of more robust and versatile computer vision algorithms, which can then be applied to a wide range of domains beyond remote sensing, including medical imaging, autonomous vehicles, robotics, and augmented reality.

1.2 Proposal

This project aims to combine the strengths of both statistical and deep learning segmentation techniques by developing deep learning models for SAR image segmentation that incorporate known statistical principles of SAR. The methodology involves leveraging well-established probabilistic models to generate synthetic data, integrating moment-based features with 1x1 convolutions for a statistically sound network design, and employing losses that consider pixels in the context of their distribution. Beyond potentially improving model performance and efficiency, incorporating statistical knowledge into the architectures can also help enhance the crucial aspect of algorithmic explainability, a quality which is notoriously deficient in many deep learning models.

Chapter 2

Background

2.1 Synthetic Aperture Radar (SAR)

2.1.1 SAR Data

To generate SAR images, a sensor is mounted on a spacecraft (satellite, airplane, etc) and captures signals from Earth's surface remotely. As opposed to passive sensors, which simply receive a signal, SAR data is acquired through active radars that emit signals, which are then reflected back to gather insights into the dielectric and textural characteristics of the ground. Since the emitted electromagnetic signal is chosen and therefore known, terrain information can be deduced by comparing the emitted and received signals' amplitude and phase. In addition, because SAR uses active sensors to illuminate the area under observation with signals, they do not rely on sunlight and can operate with low visibility. The wavelength of the emitted signal can also be deliberately chosen to enable penetration through various materials such as clouds, allowing SAR data to be collected in any weather conditions. In addition, since SAR sensors travel while mounted to a moving spacecraft, they can emulate a large antenna aperture by capturing data from various angles and points in time, combining them to provide multiple perspectives of the same surface. The number of perspectives concatenated together to generate a single data point is referred to as the number of *looks*. Moreover, polarizations in SAR systems denote different orientations of the radar's transmitting and receiving antennas relative to the ground. These orientations, such as horizontal (H), vertical (V), and dual (HH, VV, HV), offer varied insights into the observed area's surface properties. Since SAR sensors can be attached to small satellites and provide high-resolution images at any time of day regardless of the weather conditions, they have gained prominence for remote sensing applications (Frery, Wu, and Deniz, 2022).

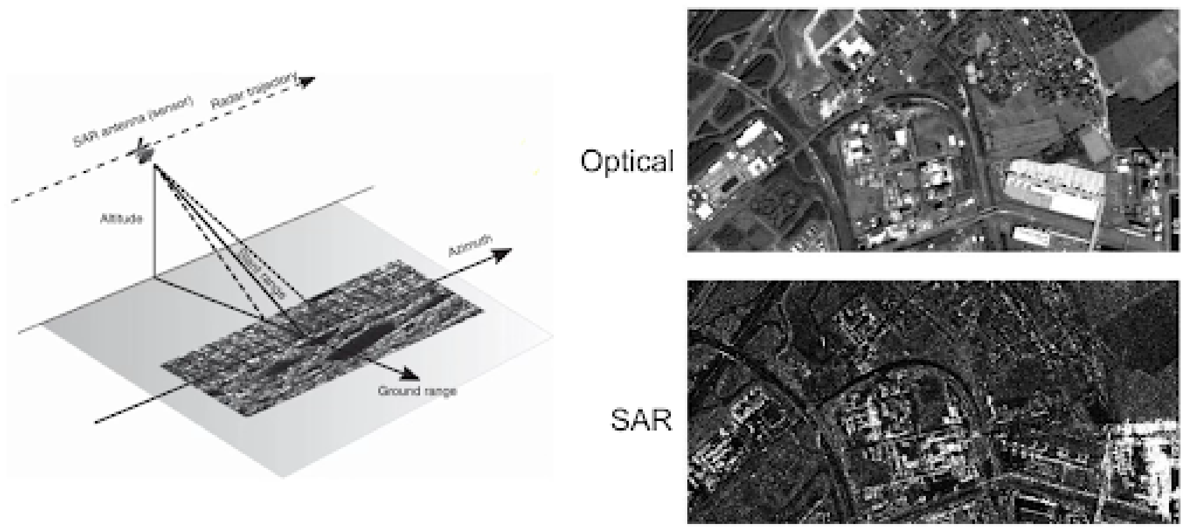


FIGURE 2.1: SAR sensor process and outputs (Frery et al., 1997).

Earth's surface is rough and uneven. Therefore, the received signal, or backscatter, will return with a varying degree of coherence based on the surface it reflected off of, as demonstrated by Figure 2.2. Therefore, the backscatter returns a fraction of the incident energy, which is described as a complex wave. These complex backscattered signals constructively and destructively interfere with each other, which results in pixels that 'should' be dark appearing lighter and vice versa. This granular quality is known as speckle. To reduce speckle, multiple images are often taken of the same area, either at different times or from different angles, and are averaged together. Due to this averaging process, the multi-look images may appear more blurry, but have less speckle (Frery, Wu, and Deniz, 2022).

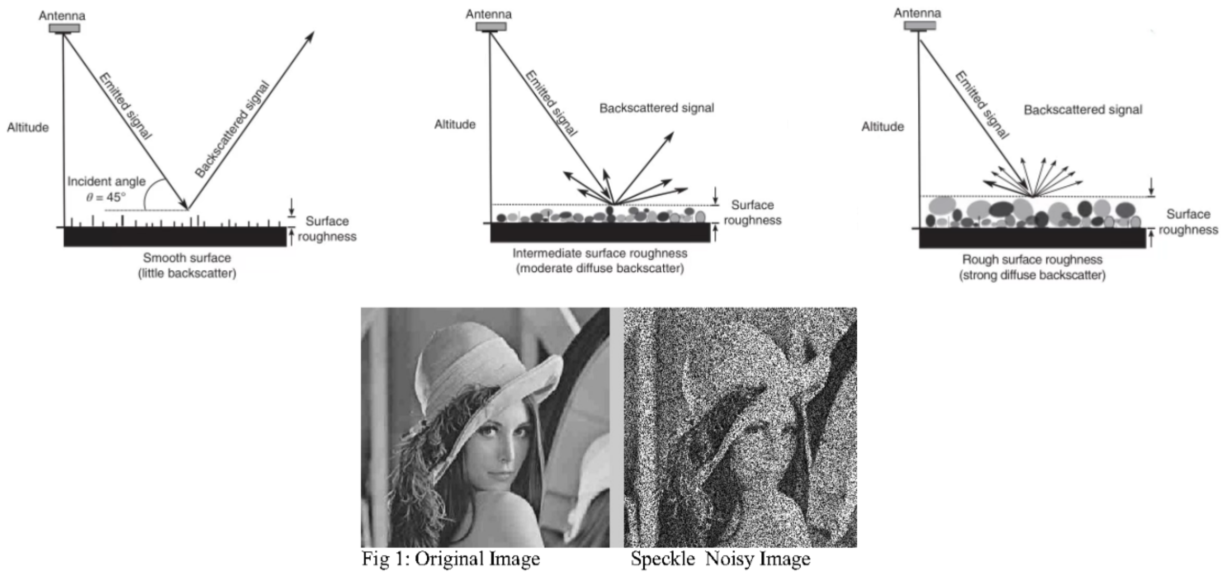


FIGURE 2.2: Backscatter and speckle noise (Frery et al., 1997; Maity et al., 2015).

2.1.2 Applications

SAR technology offers a powerful tool for monitoring and understanding the Earth's surface and its dynamic processes. Its ability to provide high-resolution, all-weather, day-and-night imaging makes it invaluable for a wide range of applications. Some examples include (Frery, Wu, and Deniz, 2022):

- Environmental Monitoring: SAR can be used to help track environmental changes like deforestation, land cover shifts, soil moisture, coastal erosion, and flooding.
- Agriculture and Forestry: SAR is utilized to aid farmers in crop monitoring, yield prediction, and forestry management by assessing vegetation health and land cover.
- Disaster Management: SAR can rapidly map disaster-affected areas, aiding in damage assessment, search and rescue, and recovery efforts.
- Urban Planning and Infrastructure Monitoring: SAR is used to monitor urban growth, land use changes, and infrastructure stability, detecting subsidence and deformation.
- Maritime Surveillance: SAR is used to detects ships, monitors maritime borders, and aids in search and rescue operations at sea.
- Defense and Security: SAR provides high-resolution imaging for surveillance, reconnaissance, and intelligence gathering.

2.2 Relevant Statistics

2.2.1 Fundamental Statistics Concepts

Random Variables:

Random variables (Goodfellow, Bengio, and Courville, 2016) represent the numerical outcome of some random process. They can be *discrete*, where their value can only be a countable number of possible outcomes or *continuous*, where they can take on any value within a given range.

Probability Distributions:

Probability distributions describe how likely the possible outcomes of a given random variable are. For discrete random variables, the probability distribution is described by a probability mass function (PMF), which gives the probability of each possible value. For continuous random variables, the probability distribution is described by a probability density function (PDF), which gives the relative likelihood of different outcomes within a range. The area of a curve under a PDF over a certain interval represents the likelihood that the random variable falls within that interval.

Moments:

Statistical moments are quantitative measures that provide valuable insights into the characteristics of a probability distribution (Larsen and Marx, 2005) . Moments describe various aspects of the distribution's shape, central tendency, spread, and symmetry.

The moment-generating function (MGF) is a fundamental tool in probability theory that provides a systematic way to generate moments of a random variable. The MGF of a random variable X, denoted as $M_X(t)$, is defined as the expectation of the exponential function e^{tX}:

$$M_X(t) = \mathbb{E}[e^{tX}] = \int_{-\infty}^{\infty} e^{tx} p(x;\theta)\, dx, \tag{2.1}$$

where $p(x;\theta)$ is the PDF or PMF of X, parameterized by θ.

The MGF provides a unified framework for understanding the moments of a distribution. Moments of X can be derived from the derivatives of $M_X(t)$ with respect to t:

$$\mu_n = \frac{d^n}{dt^n} M_X(t)\bigg|_{t=0}. \tag{2.2}$$

Specific Instances of Moments:

1. First Moment (Mean): The mean (μ) represents the central tendency of the data and is calculated by averaging all data points:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{2.3}$$

2. Second Moment (Variance): The variance (σ^2) measures the spread of the data around the mean and is calculated as the average of the squared differences between each data point and the mean:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N} (x_i - \mu)^2. \tag{2.4}$$

3. Third Moment (Skewness): Skewness (γ) quantifies the asymmetry of the probability distribution:

$$\gamma = \frac{1}{N}\sum_{i=1}^{N} \left(\frac{x_i - \mu}{\sigma}\right)^3. \tag{2.5}$$

4. Fourth Moment (Kurtosis): Kurtosis (κ) measures the peakedness or flatness of a distribution compared to a normal distribution:

$$\kappa = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i - \mu}{\sigma}\right)^4. \tag{2.6}$$

Log Cumulants:

Cumulants (Larsen and Marx, 2005) are statistical measures derived from moments. The cumulant-generating function (CGF) $K_X(t)$ is defined as the logarithm of the moment-generating function:

$$K_X(t) = \ln M_X(t), \tag{2.7}$$

where $M_X(t)$ is the moment-generating function of the random variable X.

Once the CGF is obtained, log cumulants can be computed directly from its derivatives with respect to the parameter t. Specifically, the nth log cumulant κ_n is given by:

$$\kappa_n = \frac{d^n}{dt^n}K_X(t)\bigg|_{t=0}. \tag{2.8}$$

2.2.2 Parameter Estimation

Parameter estimation, a core task in statistics and machine learning, entails determining the values of unknown parameters within a statistical model (Larsen and Marx, 2005). This process aims to find the most suitable values for these parameters to ensure that the model aligns with the observed data.

Maximum Likelihood Estimation:

Maximum Likelihood Estimation (MLE) (Larsen and Marx, 2005) is a statistical method used to estimate the parameters of a probability distribution by maximizing the likelihood function. Often, it actually involves maximizing a log-likelihood function to simplify the calculations and ensure numerical stability. By constructing a likelihood function representing the probability of the data for different parameter values, MLE estimates are obtained by maximizing this function, typically using optimization techniques. The likelihood function may not always have a closed-form expression, requiring numerical optimization methods for estimation. The equation for MLE is as follows:

$$\hat{\theta}_{\text{MLE}} = \arg\max_{\theta} \mathcal{L}(\theta \mid \text{data}), \tag{2.9}$$

where $\hat{\theta}_{\text{MLE}}$ is the maximum likelihood estimate of the parameter θ and $\mathcal{L}(\theta \mid \text{data})$ is the likelihood function, representing the probability of observing the given data under the parameter θ.

Method of Moments:

The Method of Moments (MoM) is a technique used in statistics to estimate the parameters of a statistical model by equating sample moments with theoretical moments (Larsen and Marx, 2005). While *sample moments* are statistics calculated from the observed data, *theoretical moments* are quantities calculated from the assumed distribution and are defined in terms of the parameters of the distribution. To estimate parameters using the MoM, if a distribution has k parameters, then the first k sample moments are equated to the first k theoretical moments. This creates a system of equations that can be solved for the parameters.

For example, for a normal distribution with parameters μ (mean) and σ^2 (variance), the first and second theoretical moments are:

$$\begin{aligned} \text{Theoretical mean (first moment):} &\quad \mu \\ \text{Theoretical variance (second central moment):} &\quad \mu^2 + \sigma^2 \end{aligned}$$

To estimate the parameters using the method of moments, these theoretical moments are equated to their corresponding sample moments:

$$\begin{aligned}\text{Sample mean (first moment):}\quad & \bar{X}\\ \text{Sample variance (second central moment):}\quad & S^2\end{aligned}$$

Equating these provides the method of moments estimators for μ and σ^2:

$$\begin{aligned}\hat{\mu} &= \bar{X}\\ \hat{\sigma}^2 &= S^2 - \bar{X}^2.\end{aligned} \tag{2.10}$$

MoM is advantageous due to its simplicity and ease of implementation, as it directly matches sample moments to theoretical moments, providing intuitive parameter estimates without the need for complex optimization algorithms.

Method of Log Cumulants:

The method of log cumulants (MoLC) is another moment-based procedure for estimating parameters (Larsen and Marx, 2005). Similar to the process of MoM, it involves calculating the cumulants of a distribution, taking their natural logarithms, and using those transformed cumulants to estimate the parameters of the distributions. In certain distributions, leveraging log cumulants can streamline moment equations, enhancing the efficiency of variables and simplifying the process of solving the equation system. Past work has shown a close relation between log cumulants and the log of statistical moments of the same order, as well as their capacity for efficient SAR processing (Rodrigues et al., 2016). Therefore, a common practice is to take the natural log of the SAR data prior to computing its moments.

2.2.3 Stochastic Distances

Stochastic distances (Nascimento, Cintra, and Frery, 2010) are metrics used to quantify the dissimilarity or similarity between probability distributions. Unlike deterministic distances, which measure the difference between fixed points, stochastic distances consider entire distributions, accounting for uncertainty and randomness inherent in probabilistic models. Within the SAR context, these distances have found applications in parameter estimation (Gambini et al., 2015), segmentation (Marques, Medeiros, and Nobre, 2012), and change detection (Nascimento, Frery, and Cintra, 2019).

2.2.4 Statistical Characteristics of SAR

The data generated from SAR images are often modeled using the Generalized Gamma (G^0) distribution (Frery, Wu, and Deniz, 2022; Frery et al., 1997). This distribution depends on roughness (α), scale (γ), and look (L) parameters. It has gained prominence due to its simplicity in sampling (Frery, Wu, and Deniz, 2022) and its ability to generate realistic synthetic SAR imagery (Rodrigues et al., 2016, Nobre et al., 2016, Fan and Neto, 2023).

The choice of the G^0 distribution stems from the multiplicative relationship between backscatter (Z), reflected signal (X), and speckle (Y) in SAR imagery (Frery, Wu, and Deniz, 2022):

$$Z = X \times Y. \tag{2.11}$$

The signal, X, is modeled by the $\Gamma(1, L)$ distribution, reflecting the sum of independent random processes inherent in SAR data acquisition. The speckle, Y, is typically modeled by the $\Gamma^{-1}(1, L)$ distribution. Because the backscatter, Z, is the product of X and Y, its distribution is often represented by the G^0 distribution, $G^0(\alpha, \gamma, L)$.

In practical terms, each pixel in SAR imagery can be thought of as sampled from a distinct G^0 distribution defined by some α, γ, and L parameters. For effective synthetic SAR generation, different roughness (α) parameters are utilized to convey different regions within the imagery, as illustrated in Figure 2.3.

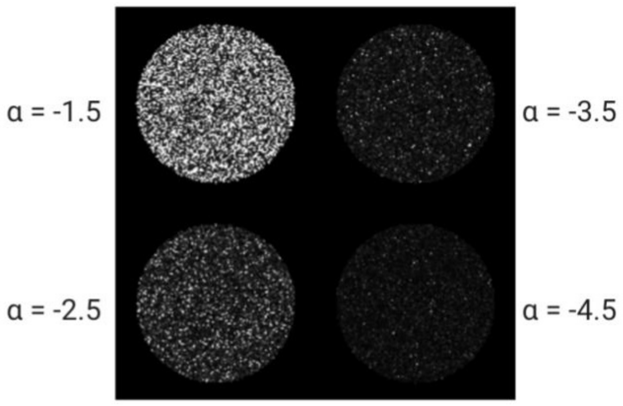


FIGURE 2.3: G^0 distribution sampling with various roughness (α) parameters ($\gamma = -\alpha + 1$ by convention and $L = 1$).

For SAR images, typical parameter values are $\alpha \in [-1.5, -15]$, $\gamma = -\alpha - 1$, and $L = 1, 3, 8$ (Gambini et al., 2015).

Furthermore, SAR image data has exhibited a notable capacity for efficient processing through direct application of computational methods to its moments (Neto et al., 2019; Rodrigues et al., 2016). Prior research indicates that neural network models can effectively exploit these moment-based features, particularly for tasks like parameter estimation (Fan and Neto, 2023). These techniques are discussed in more detail in Section 2.4.

2.3 Deep Learning and Computer Vision

2.3.1 Deep Learning Basics

Deep learning, a subset of machine learning within the broader field of Artificial Intelligence (AI), uses multi-layered neural network models to process input data x, and produce an optimal output, y (Goodfellow, Bengio, and Courville, 2016). Each neural network comprises of *perceptrons*, which are fundamental building blocks that receive input signals either from the external world or previous layers in the network, perform computations, and produce an output.

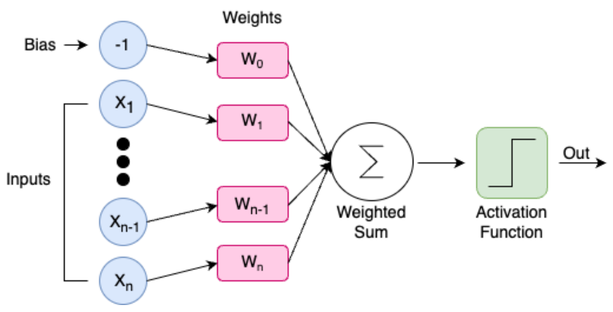


FIGURE 2.4: Diagram illustrating the computation carried out by a perceptron.

Perceptrons, or units, utilize weights to gauge the significance of input signals and compute a weighted sum of the inputs plus a bias term. The output of this weighted sum is then constrained through an activation function, enabling non-linear relationships in the network. One commonly used activation function is the *Rectified Linear Unit (ReLU)*, which returns 0 for any negative input and the input value itself for any positive input. The output of a single perceptron, obtained through a *forward pass* is represented by the equation:

$$y = a(w_1x_1 + w_2x_2 + \ldots + w_nx_n + w_0), \tag{2.12}$$

where x is an input vector of size n, w is the perceptron's weight vector of size n, w_0 is the bias term, and a is the activation function.

A neural network (Goodfellow, Bengio, and Courville, 2016) is constructed out of multiple layers of perceptrons, with each layer potentially containing multiple units. A *feed-forward network*, which is common in deep learning, consists of connections in only one direction and is typically *fully-connected*, meaning that each output of the previous layer is used as the input for every unit in the next layer. The *input layer* receives external signals, the *output layer* generates final predictions, and the layers in between are *hidden layers*. An example of a typical feed-forward neural network is shown in Figure 2.5.

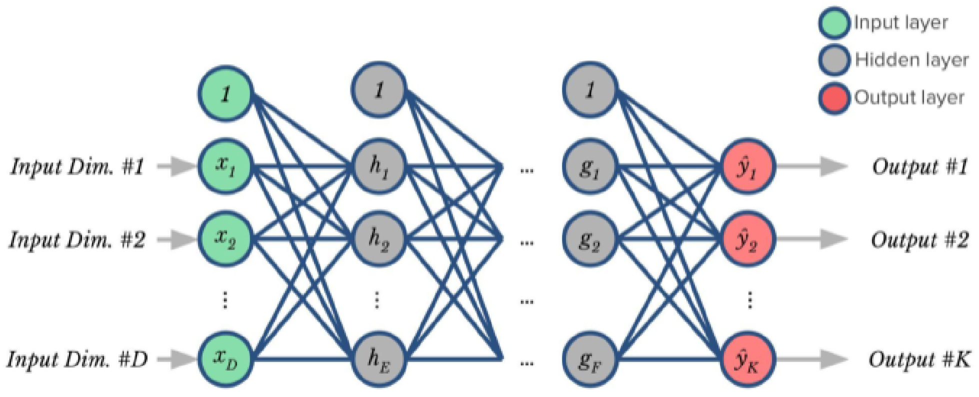


FIGURE 2.5: Feed-forward neural network.

The output of one forward pass through a typical feed-forward neural network with L hidden layers is represented as:

$$y = a_L(W_La_{L-1}(W_{L-1}\ldots a_0(W_0x))). \tag{2.13}$$

The network learns to optimize its predictions by adjusting its weights, or parameters, through a process called *backpropagation*. Initially, the weights are randomly assigned. Then, for each known (X, y) tuple, X is sent through the network and the model outputs a corresponding y prediction. The y prediction is evaluated against the true y, or *ground truth*, using a *loss function*. This loss is then propagated back through the network, which involves computing the gradients of the loss function with respect to the weights in the network. In standard gradient descent, the network then 'learns' to predict better next time by adjusting each weight in the opposite direction to minimize the loss after iterating through the entire training dataset, or over one *epoch*. This algorithm is often repeated over multiple epochs.

This process is also typically enhanced with optimization algorithms such as *stochastic gradient descent*, which updates the model's parameters using gradients estimated from mini-batches rather than the entire training dataset at each iteration, or *Adaptive Moment Estimation* (ADAM), which uses exponentially decaying learning rates and momentum to enhance training (Kingma and Ba, 2017). Additional common deep learning techniques are *batch normalization* and *dropout*. Batch normalization involves normalizing the input of each layer across a mini-batch to improve training stability. Dropout involves randomly shutting off individual neurons at a set probability to prevent *overfitting*,which occurs when the model memorizes the training data.

Following training, the model is typically evaluated on a *validation dataset* sourced from the same data as the *training dataset*, yet remains unseen by the model during training. These tests can be used to help detect overfitting and to tune a model's hyperparameters (number of units, number of epochs, etc). Often, the model is then tested on a distinct *test dataset* to assess its generalization capabilities.

Deep learning commonly uses *tensors* as the fundamental data structure for representing and processing data. Tensors are multi-dimensional arrays, and they are well-suited for representing the input data, weights, biases, and activations in neural networks.

2.3.2 Convolutional Neural Networks

Convolutional Neural Networks (Goodfellow, Bengio, and Courville, 2016), or CNNs, are a type of deep learning network that are primarily used for processing visual data. In CNNs, *convolutional layers* are able to learn spatial information by sliding learnable filters, or kernels, across the input data. By computing dot products at each position, the network is able to detect local patterns or features. The *kernel size* of a convolutional operation refers to the height and width of the kernel, and the *stride* refers to how many spatial location the kernel slides before making another computation. Figure 2.6 displays an example of a convolutional operation being applied to a two-dimensional input:

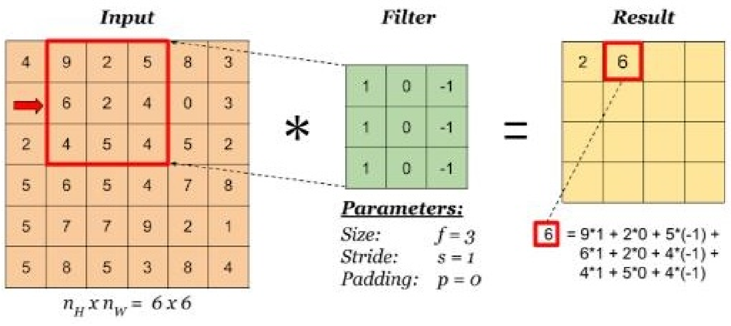
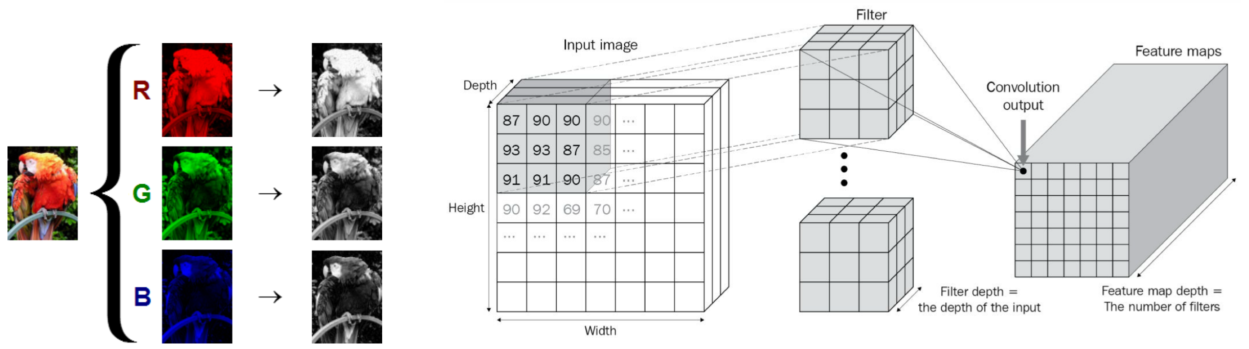


FIGURE 2.6: Convolution operation example (Kumar, 2020).

Convolutional layers may learn multiple filters in order to obtain various, distinct feature information. The programmer may decide how many filters they want to learn, and consequently, the number of *channels* outputted by the operation. Channels in image processing refers to the different dimensions or layers of information contained within an image. A simple example of the concept of channels is RGB color images, where each channel corresponds to red, green, or blue, with varying intensities defining the overall color of each pixel. Below, Figure 2.7a provides an example of how RGB images can be represented by channels, and Figure 2.7b depicts how convolutional operations work when the input and/or the kernel has more than two channels:



(A) RBG image (Niemietz, 2008). (B) Example of convolutional operation with multiple channel dimensions.

FIGURE 2.7: Channels in deep learning.

Conceptually, CNNs differ from fully connected networks in their local connectivity. Rather than each unit receiving inputs from all units in the previous layer, CNN units have limited connectivity, allowing them to focus on local regions of the input. This design, along with parameter sharing—where the same set of filter parameters is used across different spatial locations—makes CNNs computational efficient and capable of learning intricate features.

In addition to convolutional layers, CNN architectures often incorporate pooling layers to downscale spatial inputs while preserving essential information. Common pooling methods include *average pooling* and *max pooling*, which reduce the dimensionality of feature maps, aiding in computation and mitigating overfitting. There are no learnable parameters within average and max pooling layers. An example of average and max pooling being applied to a two-dimensional input matrix is displayed in Figure 2.8.

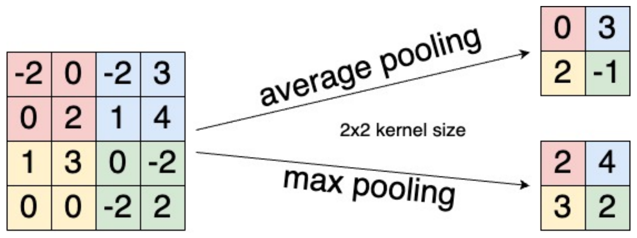


FIGURE 2.8: Average and max pooling operation examples.

Another common tool employed in convolutional deep learning is applying *padding* (Dumoulin and Visin, 2018). Padding refers to the addition of extra pixels of value 0 around the input image or feature map. This enables the programmer to control the output dimensions by adjusting the amount of padding added around the input. It also helps in enabling deeper network architectures and preventing information loss at the edges of the input. Figure 2.9 provides a visualization of applying padding to a convolutional operation:

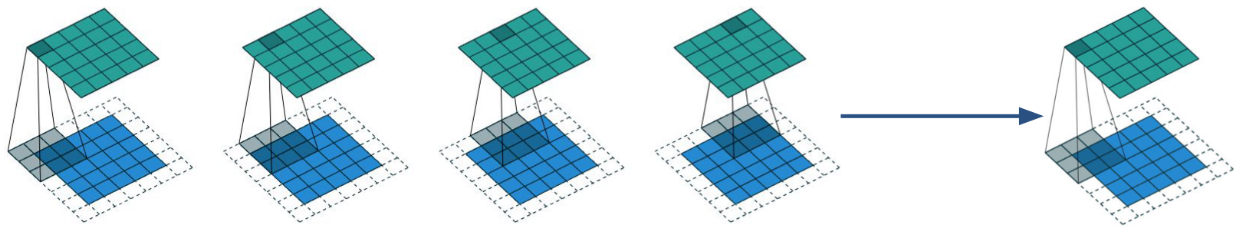


FIGURE 2.9: Example of convolutional operation that employs padding (Dumoulin and Visin, 2018).

Unlike typical convolutional layers, as well as average and max pooling layers, *transpose convolutions* are used to increase the spatial dimensions of an input. They involve sliding a kernel over the input feature map with padding to produce an output feature map with higher spatial dimensions. An example of a transpose operation is shown in Figure 2.10, where A is the input, B is the kernel, and the operation uses a stride of 1:

$$A = \begin{bmatrix} 2 & 4 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad A \otimes^{\mathsf{T}} B = \begin{bmatrix} 2 & 4 & \\ 6 & 8 & \\ & & \end{bmatrix} + \begin{bmatrix} & 4 & 8 \\ & 12 & 16 \\ & & \end{bmatrix} + \begin{bmatrix} & & \\ 0 & 0 & \\ 0 & 0 & \end{bmatrix} + \begin{bmatrix} & & \\ & 1 & 2 \\ & 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 8 & 8 \\ 6 & 21 & 18 \\ 0 & 3 & 4 \end{bmatrix}$$

FIGURE 2.10: Transpose convolution operation example.

*1x1 convolution*s, standard convolutional layers with a kernel size and stride of 1, have unique properties. A 1x1 convolutional layer is functionally similar to a fully-connected layer, since it operates on the entire spatial extent of its input feature map, just like a fully-connected layer operates on the entire input vector. Moreover, the 1x1 convolutional layer preserves the weight sharing properties of convolutional layers, which are important for capturing spatial hierarchies and reducing the number of parameters in the network. Therefore, despite its small size, the 1x1 convolutional layer can perform complex transformations and feature extraction similar to fully-connected layers, making it a powerful tool in convolutional neural network architectures.

2.3.3 Image Segmentation

Image segmentation (Goodfellow, Bengio, and Courville, 2016) is a computer vision task where an image is divided into distinct segments, with each pixel assigned a specific label. It aims to identify and delineate different regions within an image, providing a representation for further analysis. In the context of SAR, image segmentation analysis can clearly delineate boundaries that may be unclear from far away, such as bodies of water, forests, oil plumes, and more.

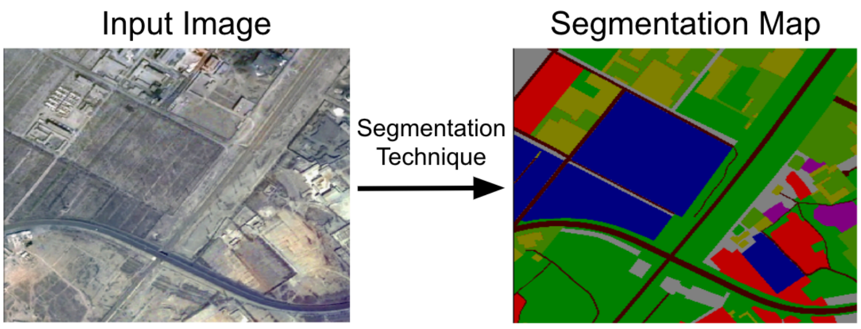


FIGURE 2.11: Image Segmentation (Liu et al., 2018)

For image segmentation, the most commonly used deep learning architecture is the *U-Net* (Ronneberger, Fischer, and Brox, 2015), which has also been broadly applied to other computer vision tasks on SAR data (Zhu et al., 2021). As pictured in Figure 2.12, the U-Net architecture contains an encoder (the left side of the U-Shape), a bottleneck (the bottom of the U-Shape), a decoder (the right side of the U-Shape), and a final convolutional layer.

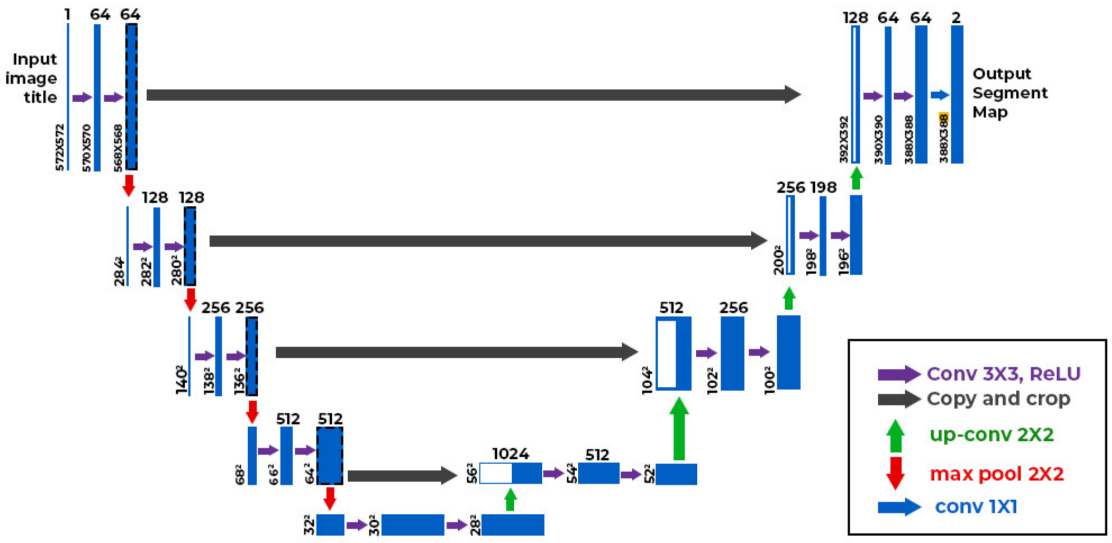
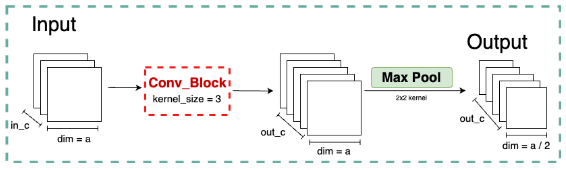
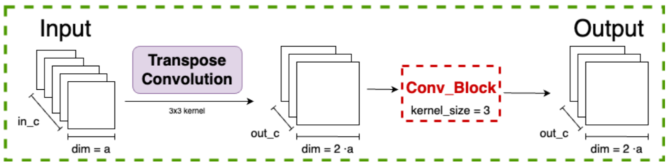


FIGURE 2.12: Typical U-Net architecture used for image segmentation tasks (Ronneberger, Fischer, and Brox, 2015).

As displayed in Figure 2.13a, the encoder utilizes convolutional and max-pooling layers to gradually reduce the spatial dimensions of the input while increasing the number of channels. Each convolutional layer applies learnable filters to capture local patterns and features in the input. At the center of the U-Net is the bottleneck. This layer captures the most abstract features of the input image at a minimal spatial resolution. It typically contains multiple convolutional layers to extract high-level representations. The decoder blocks of the U-Net, shown in Figure 2.13b, utilize a transpose convolution to up-sample the feature maps from the bottleneck layer, gradually increasing the spatial dimensions while decreasing the number of channels. The goal of the decoder is to reconstruct the original spatial resolution of the input image.



(A) Encoder Block



(B) Decoder Block

FIGURE 2.13: Fundamental Blocks in Typical U-Net

The final convolution of a U-Net, usually also followed by a softmax activation function, produces the segmentation map. This map consists of one channel per region, with each value indicates the probability that the pixel in the corresponding position belongs to that region. A typical U-Net also employs skip connections, directly concatenating feature maps from the encoder to the corresponding decoder at the same spatial resolution. While the U-Shape enables the network to learn global features and establish relationships between distant parts of the image, the skip connections help alleviate the problem of information loss during down-sampling and provide spatial context for the final segmentation. They also help mitigate the vanishing gradient problem, which occurs when the gradients of the loss function become extremely small as they are backpropagated through many layers in a network.

Typical U-Nets are *fully-convolutional*, meaning they are composed entirely of convolutional layers. Since convolutional layers share weights across spatial locations and therefore require less parameters than fully connected layers, fully-convolutional networks train extremely quickly.

The *Autoencoder* is almost identical to the U-Net, and can also be used for segmentation tasks. Its primary distinction is that it does not contain skip connections and in many implementations, the data is flattened into a one-dimensional latent representation at the bottleneck.

2.4 Techniques for SAR Parameter Estimation and Segmentation

2.4.1 Statistical Techniques

Historically, traditional statistical techniques were used to perform segmentation on SAR data. Non-parametric methods such as Maximum Likelihood Estimation (MLE), Method of Moments (MoM), and Method of Log-Cumulants (MoLC), were used to approximate parameters. These parameters were then fed in as inputs into segmentation algorithms, such as thresholding-based techniques, to generate segmentation maps. These techniques found success in some domains, especially for data that exhibited extremely homogeneous regions. However, they had trouble analyzing SAR data that contained more heterogeneous regions (Frery, Wu, and Deniz, 2022).

Cheng et al. (2013) was able to address these issues by combining MoLC and MoM. His approach yielded algebraic expressions that enabled more accurate parameter estimation, which were particularly crucial in scenarios where conventional methods failed. Subsequently, Rodrigues et al. (2016) applied MoLC estimation specifically to SAR image segmentation tasks. Their methodology involved computing the roughness of individual pixels, and combining the results to generate a roughness map for the entire SAR image. By inputting these roughness maps into statistical segmentation algorithms, they were able to achieve improved segmentation outcomes.

2.4.2 Deep Learning Techniques

Amid the recent advancements in deep learning, parametric neural network models have emerged as effective tools for a wide variety of SAR analysis tasks. A significant advantage of deep learning methods over traditional statistical techniques is their capacity to process substantial volumes of data efficiently. This efficiency is largely facilitated by the parallel processing capabilities offered by modern GPUs, enabling deep learning algorithms to analyze SAR images with unparalleled speed and scalability. Some example application areas are terrain classification, despeckling, parameter estimation, and image segmentation. The interested reader can consult Zhu et al. (2021), a comprehensive survey of deep learning applications in the context of SAR. Notably, due to the U-Net's ability to capture intricate spatial dependencies within SAR imagery, it has been widely adopted for SAR image segmentation tasks (Nava et al., 2022; Hartmann et al., 2021; Ren et al., 2021; Mazza et al., 2019).

Additionally, recent advancements by Fan and Neto (2023) have showcased the potential of neural network-based algorithms in estimating roughness parameters when trained on G^0-modeled synthetic SAR data. Their architecture is depicted in Figure 2.14:

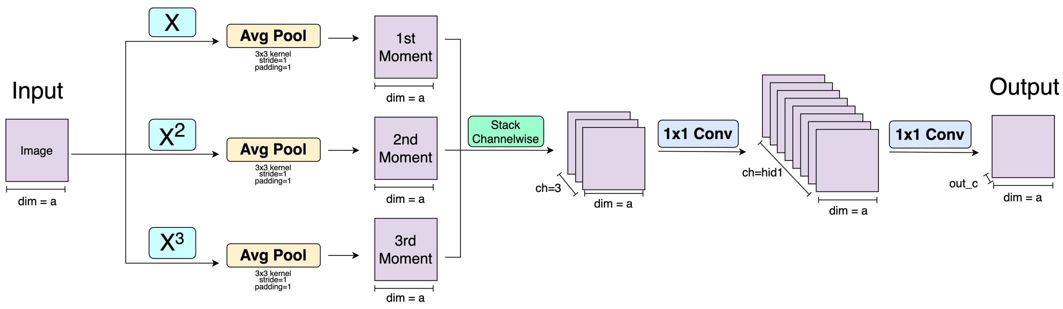


FIGURE 2.14: Architecture introduced in Fan and Neto (2023) that uses method of moments-inspired computations to estimate SAR roughness parameters.

Their network drew inspiration from the statistical method of moments computation process outlined in Section 2.2.1 and was implemented as follows: the input undergoes a replication process resulting in three identical copies. One copy remains unchanged, while another has all pixel values squared, and the third has all pixel values cubed. Subsequently, these copies are each subjected to separate average pooling layers, employing a kernel size and stride of 3, along with padding to maintain consistent width and height dimensions. Mirroring the numerical moment computation procedure, the first copy is meant to capture the mean of the pixel values (first moment), the second is meant to capture the variance (second moment), and the third is meant to capture the skewness (third moment). These moment representations were then sent through three 1x1 convolution layers, taking advantage of the fact that 1x1 convolutions functionally act like fully connected layers, yet still allow the network to be fully-convolutional. The output of these convolutional layers were the roughness parameters of the input image, which could then be transformed into roughness maps and input into segmentation algorithms to generate segmentation maps. This algorithm was shown to outperform traditional estimation methods in terms of accuracy, speed, and reliability. The efficiency of the methodology introduced in Fan and Neto (2023), only requiring synthetic data for training, is particularly advantageous in the context of SAR, where large, high-quality datasets are scarce. In instances where they are available, they tend to be either highly specialized to a particular domain or lack segmentation labels due to the labor-intensive nature of creating these maps.

Chapter 3

Methodology

The fundamental framework common to all machine learning problems involves crafting a model capable of translating inputs, denoted as X, into optimal outputs, represented by y. Within the domain of deep learning, this typically entails constructing a sophisticated neural network model. This project is organized into three primary sections, each corresponding to a distinct aspect of machine learning algorithms: input, model, and output. The goal of the input section is to generate synthetic data that faithfully replicates the statistical characteristics observed in real SAR data. In the model section, the objective is to devise models that accurately map inputs to their respective outputs while integrating considerations of statistical moments. Finally, the output section aims to develop a loss function that takes advantage of the known statistical properties of SAR to guide the model more effectively toward precise predictions of y. Collectively, the overarching objective of all project sections is to enhance SAR image segmentation capabilities by leveraging its well-established statistical properties.

3.1 Synthetic Dataset Generation

This portion of the project focused on generating high-quality synthetic SAR data. To generate images with shapes that looks like the terrains in real SAR data, PyTorch's pyperlin library (Paszke et al., 2019) was employed to generate Perlin Noise. Perlin Noise is a type of fractal noise that is widely used for natural-looking textures and terrains (Perlin, 1985). Perlin noise is based on random processes, so an infinite amount of unique synthetic data could theoretically be generated. After producing a noise map, a threshold was imposed to create a segmentation map with two distinct regions. The scope of this project was limited to data with only two regions, a deliberate simplification aimed at enabling a more focused investigation into other aspects of the project.

Various hyperparameters were experimented with to alter the segmentation maps' smoothness and amount of detail. Figure 3.1 shows the effect of altering the Perlin Noise's persistance parameter, which controls the smoothness of the boundary and is usually between 0 and 1. A random seed was set to ensure consistency and facilitate comparison across different hyperparameter settings.

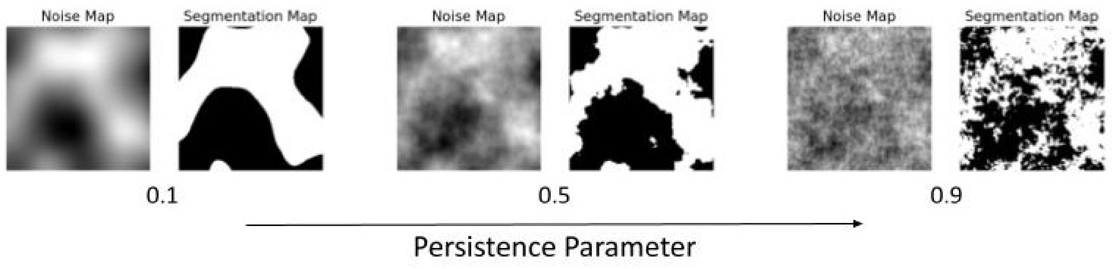


FIGURE 3.1: Perlin Noise generated from varying persistence parameter values and the corresponding segmentation maps yielded from applying thresholding.

For the synthetic dataset, two hundred segmentation maps were generated using Perlin Noise with a persistence parameter of 0.5. Some examples are displayed in Figure 3.2.

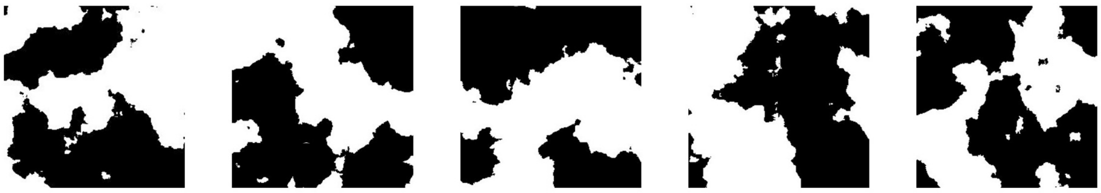


FIGURE 3.2: Examples of ground truth segmentation maps generated for synthetic dataset.

Since the Generalized Gamma (G^0) is known to accurately model SAR data and its inherent speckle noise (see Section 2.2.4), pixel values were sampled from G^0 to generate the synthetic data. To produce synthetic data associated with a ground truth segmentation map, pixels corresponding to region 1 of the ground truth were sampled using one roughness parameter (α_1), while pixels corresponding to region 2 of the ground truth were sampled using a different roughness parameter (α_2). Figure 3.3 shows three examples of synthetic data generated from the same segmentation map with different roughness parameter combinations. When the α values are closer together, the regions are less visually discernible.

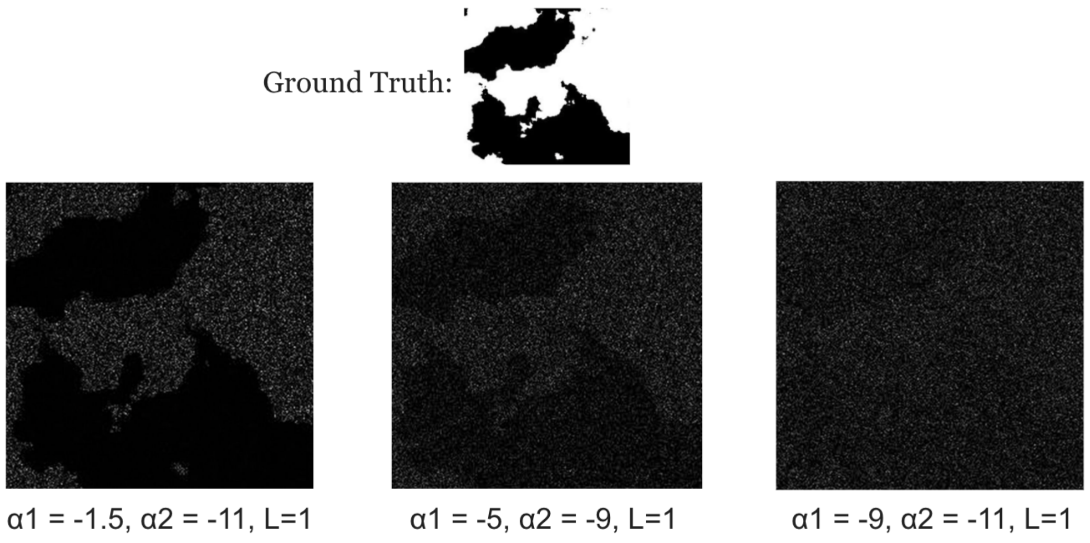


FIGURE 3.3: Examples of synthetic data generated from the same ground truth label with different roughness parameter combinations.

Various roughness combinations were experimented with in an attempt to create a dataset that accurately represents the diversity of SAR data, yet is still interpretable enough that a segmentation model can discern the distinct regions. Another hyperparameter that was experimented with was the number of looks, which, as discussed in Section 2.1.1, is the number of radar pulses that are integrated to form a single image pixel.

To assess the synthetic dataset's diversity and capacity to train SAR segmentation models, a standard U-Net was trained on the synthetic dataset and subsequently tested on synthetic and real test SAR images.

3.2 Statistically-Motivated Segmentation Architectures

Building upon the work in Fan and Neto (2023) that used a network architecture to emulate statistical moment computation and compute roughness parameters, this part of the project focused on designing and experimenting with similarly statistically-principled architectures that can directly transform input SAR data into binary segmentation maps.

The models were first all trained on the synthetic dataset. Subsequently, their performance was quantitatively assessed using synthetic validation data, followed by qualitative evaluation using both synthetic validation data and real SAR test data. Next, the models were trained and tested on the SARBuD Dataset, which, as discussed in more detail in Section 4.1.1, contains labeled segmentation data derived from real SAR images, but with notable limitations. Finally, the models trained on synthetic data underwent quantitative testing on the SARBuD Dataset.

All eighteen of the architectures developed for this project can be found in Appendix A. For modularity, Section A.1 of the Appendix contains representations of blocks of code that were used in various architectures. Section A.2 of the Appendix contains representations of the various encoder blocks that were experimented with as encoders for U-Shape architectures. Section A.3 of the Appendix contains representations of the high-level architectures that transform images into a corresponding segmentation.

The current state-of-the-art for segmentation tasks, the `Unet` (Architecture A.3.1), was used as a benchmark for the other architectures that were experimented with. Another benchmark architecture utilized was the `Autoencoder` (Architecture A.3.2), which in this implementation is identical to the typical `Unet`, but does not employ skip connections.

The `three_moments` block, depicted in Figure 3.4, served as a foundational element across numerous architectures, incorporating the key statistical principles that this project aims to integrate into the designs.

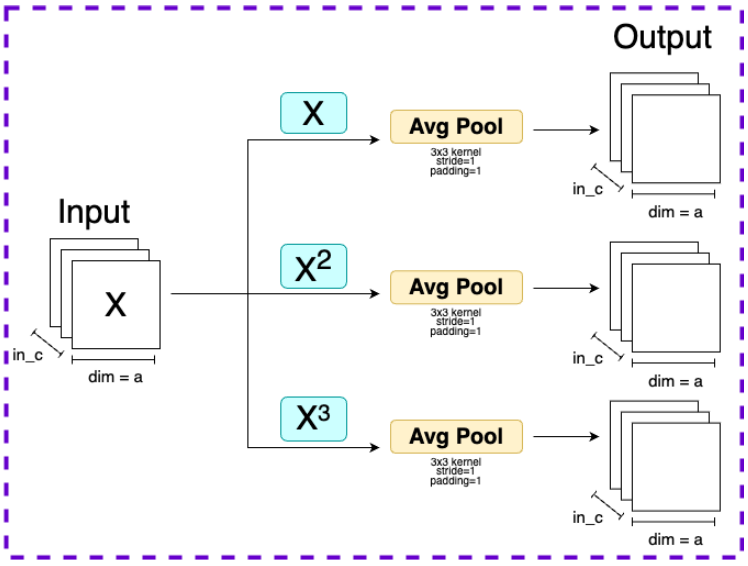


FIGURE 3.4: `Three_Moments` block.

Given that statistical moments can be utilized for parameter estimation (see Section 2.2.2), the rationale is that the outputs of the `three_moments` block represent the first three statistical moments of the input and therefore contain the necessary information for generating precise segmentations of the input image. In Fan and Neto (2023), a similar architecture, along with 1x1 convolutions, was employed to estimate distribution parameters within SAR data. Subsequently, these parameters were processed by traditional (non-deep learning) segmentation algorithms.

This project aimed to employ the same architectural techniques to train a model capable of implicitly estimating the parameters and translating them into meaningful segmentations, all within a fully-convolutional network. The elimination of post-processing segmentation algorithms could potentially result in a significantly faster segmentation process.

The `3Moms` model, depicted in Figure 3.5, closely resembles the model introduced in Fan and Neto (2023), with some adjustments made to the channel dimensions. These modifications were aimed at producing a segmentation map that matches the input dimension, rather than a single roughness parameter.

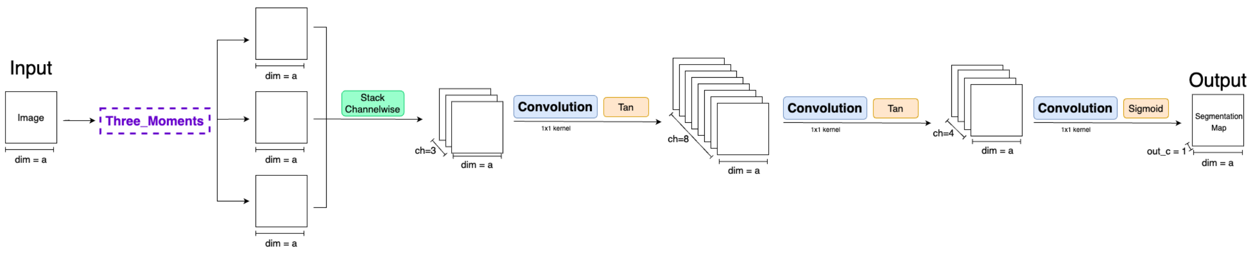


FIGURE 3.5: `3Moms` segmentation architecture.

Given the effectiveness of the U-Net in various segmentation contexts, many of the architectures primarily focused on integrating statistical moment computation within the framework of the standard U-Net. The general `U-shape` block, illustrated in Figure 3.6, features encoder blocks that reduce the height and width dimensions of the input, decoder blocks that restore the height and width dimensions of the latent representation to their original size, and skip connections that allow the decoder blocks to access high-resolution feature maps from earlier stages of the network. A U-Shape architecture proves effective for segmentation tasks due to its ability to compress the input into a smaller spatial dimension, facilitating the learning of relationships between spatially distant pixels. The statistical computations employed in this project operate locally within their window, and 1x1 convolutions, while enacting global learning, can not inherently capture spatial relationships between pixels. Therefore, compression within the architecture likely remains crucial to ensure that the model can accurately classify pixels from disparate regions of the image into the same region.

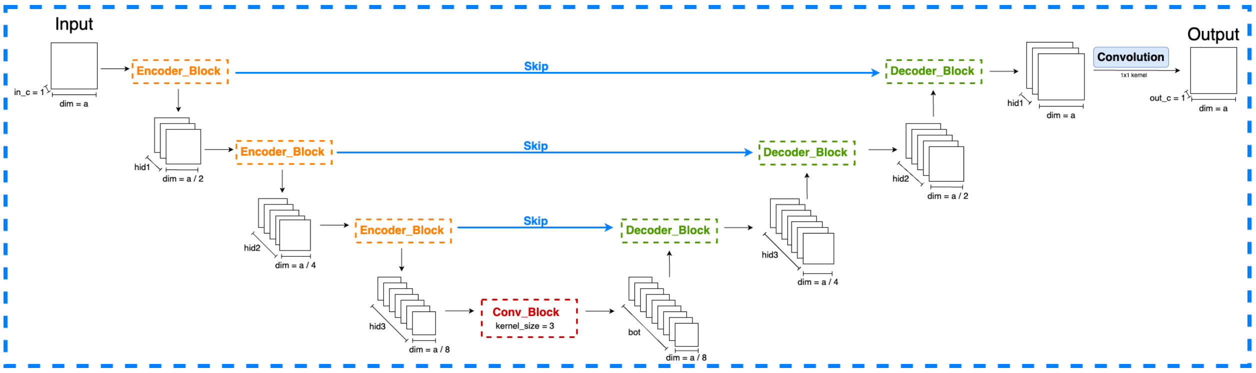
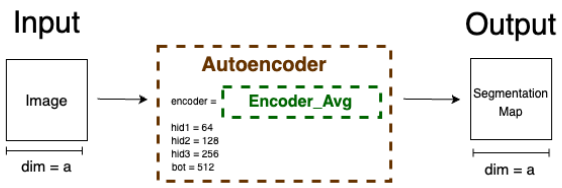


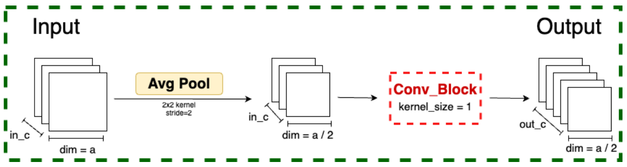
FIGURE 3.6: `UShape block.`

All of the architectures that incorporate the `UShape` block use the same decoder block, illustrated in Architecture A.1.3, which consists of a transpose convolution and two 1x1 convolutions. The statistical architecture experiments were often implemented in the encoder blocks.

For example, the `AE_Avg` model is an autoencoder that utilizes `Encoder_Avg` as its encoder (Figure 3.7). `Encoder_Avg` consists of an average pooling layer, reducing the height and width dimensions by half, and then two 1x1 convolutions. The intuition behind this design is that the encoders will be able to learn a representation of the first moment, and then pass that representation along to eventually be interpreted in the segmentation.



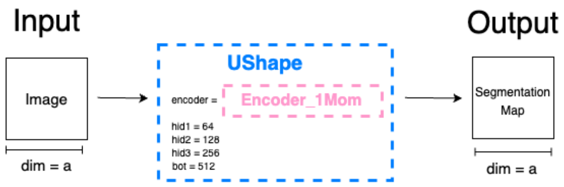
(A) AE_Avg



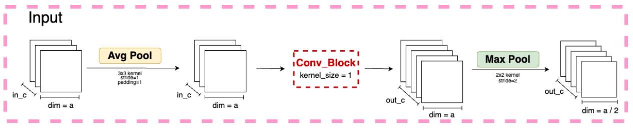
(B) Encoder_Avg

FIGURE 3.7: AE_Avg segmentation architecture and its corresponding encoder.

As depicted in Figure 3.8, the Unet_1Mom model, which uses Encoder_1Mom, is very similar to the AE_Avg model. In contrast to Encoder_Block_Avg, the average pooling layer in Encoder_1Mom applies padding to maintain the same height and width dimensions. Subsequently, the data undergoes 1x1 convolutions before passing through a max pooling layer, resulting in an output with halved height and width dimensions. This approach mirrors the encoder structure of the encoder in a typical U-Net (Architecture A.1.2), where convolutions precede dimension reduction through max pooling. Despite this alteration, the statistical rationale aligns with that of Encoder_Avg.



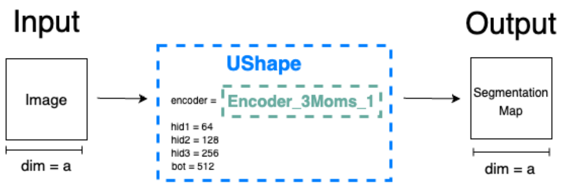
(A) Unet_1Mom



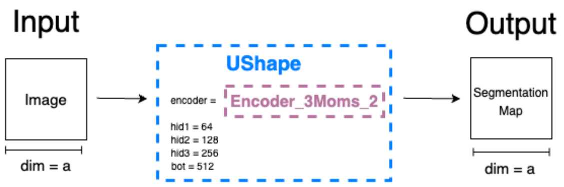
(B) Encoder_1Mom

FIGURE 3.8: Unet_1Mom segmentation architecture and its corresponding encoder.

The Unet_3Moms_1 architecture (Figure 3.9a), and its corresponding Encoder_3Moms_1 (Architecture A.2.3), is very similar to the Unet_3Moms_2 architecture (Figure 3.9), and its corresponding Encoder_3Moms_2 (Architecture A.2.4).



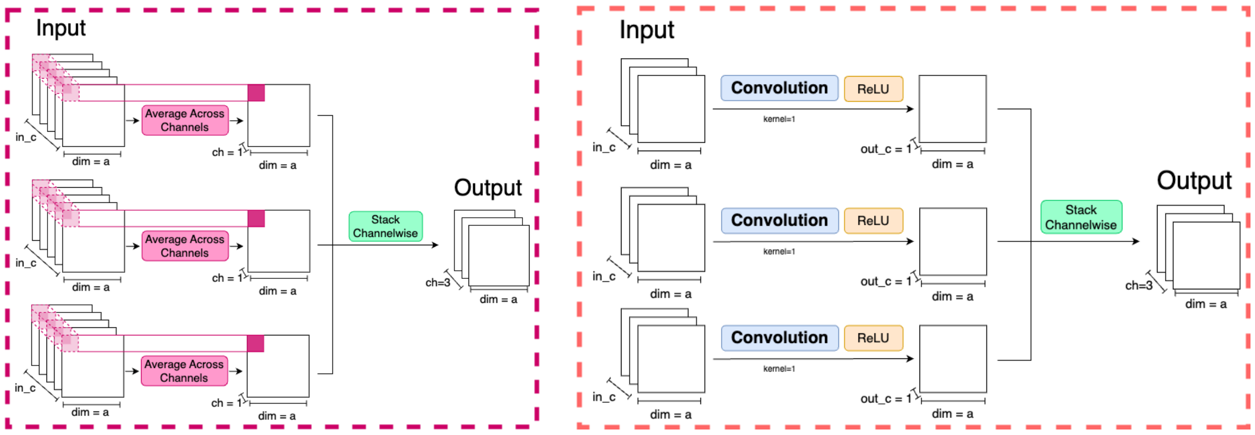
(A) Unet_3Moms_1



(B) Unet_3Moms_2

FIGURE 3.9: Unet_3Moms_1 and Unet_3Moms_2 segmentation architectures.

In both Encoder_3Moms_1 and Encoder_3Moms_2, the input is fed through the Three_Moments block. Since the input to an encoder block may have any number of *in_c* channels, the corresponding output tensors will also have *in_c* channels. Instead of stacking $3 * in\_c$ channels, the next step is to transform the tensor so that it only contains 3 channels, one for each moment. Encoder_3Moms_1 and Encoder_3Moms_2 accomplish this in different ways. Encoder_3Moms_1 utilizes the Average_Across_Channels block, illustrated in Figure 3.10a, averaging all of the pixels that are in the same spatial location across channels for each moment tensor and then stacking them. Encoder_3Moms_2 utilizes the Convolve_To_1Chan block, illustrated in Figure 3.10b, sending each moment tensor through a 1x1 convolution with an output channel dimension of 1, and then stacking the outputs.



(A) `Average_Across_Channels` (B) `Convolve_to_1Chan`

FIGURE 3.10: Blocks for transforming three moment representations into one tensor with three channels.

Obtaining a tensor with three channels representing the three moments, both `Encoder_3Moms_1` and `Encoder_3Moms_2` then employ the same architecture as in `Encoder_1Mom`, sending the data through two 1x1 convolutions and a max pooling layer. Ideally, these encoders would be able to preserve statistical information about the input data that, when combined with the relative spatial information gained from compressing the image in the bottleneck, could subsequently be leveraged for more effective reconstruction of a segmentation map later on.

Models `Unet_3moms_3chans_1` (Architecture A.3.8) and `Unet_3moms_3chans_2` (Architecture A.3.9) are identical to models `Unet_3moms_1` and `Unet_3moms_2` respectively, with the only difference being the channel dimensions throughout the U-Shape. Emulating the standard U-Net architecture, both `Unet_3moms_1` and `Unet_3moms_2` expand the data's channel dimensions within the encoders, peaking at 512 channels at the bottleneck, before subsequently reducing the channel dimensions within the decoders. In models `Unet_3moms_3chans_1` and `Unet_3moms_3chans_2`, however, the input and output tensors of the encoder and decoder blocks are always three. The rationale behind this decision is that if the encoders are only expected to learn about the first three statistical moments of the input, only three channels might be necessary to retain this information. With much fewer weights to train, this approach could greatly enhance training speed.

In the subsequent experimental architectures, statistical moment computations were integrated into the skip connections of the standard U-Net architecture, rather than in the encoders. Given that typical skip connections preserve low-level feature information, replacing them with moment computations that encapsulate broader window-scope statistical properties may aid in minimizing redundancy and promoting homogeneity within the outputted segmentation maps.

One example of such architectures is model `Unet_3Moms_Skip_1`, as depicted in Figure 3.11:

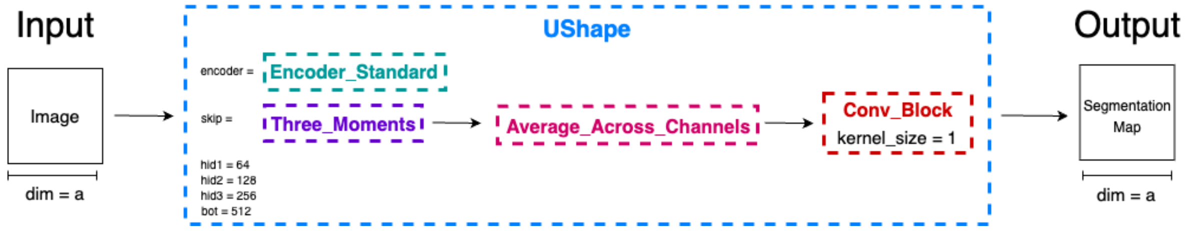


FIGURE 3.11: `Unet_3Moms_Skip_1` segmentation architecture.

This architecture employs the typical U-Net encoder block. The skip connections feed the input data through a `Three_Moments` block to compute representations of statistical moments, an `Average_Across_Channels` block to reduce the tensor to three dimensions, and then two learnable

1x1 convolutions to output a tensor with the same number of channels as the input channel dimension of the corresponding decoder block. Model `Unet_3Moms_Skip_2` (Figure A.3.11) is almost identical, with the only difference being that it utilizes the `Convolve_to_1Chan` block instead of the `Average_Across_Channels` block to reduce the tensor to three channels.

Model `Unet_3Moms_Skip_BigKern` (Architecture A.3.12) is also similar to model `Unet_3Moms_Skip_2`, except it employs a kernel size of 7 in the convolutional operation of the encoder, as opposed to the typical kernel size of 3. To provide a benchmark for this architecture, model `Unet_BigKern` (Architecture A.3.13) is a typical U-Net (with typical skip connections), and also employs a kernel size of 7 in the convolutional operation of its encoder.

Instead of replacing every skip connection with moment computations, `AE_Concat_3Moms` only uses the moment computation once as a parallel branch that is eventually concatenated with the output of an autoencoder (and synthesized with a 1x1 convolution), as shown in Figure 3.12. In previously-discussed architectures, the `Average_Across_Channels` or `Convolve_to_1Chan` block was necessary to maintain three channels corresponding to the first three moments, potentially abstracting the computations away from the statistical foundations that they were inspired by. Here, since the input into the `Three_Moments` block only has one channel to begin with, such blocks are not necessary and the more simplistic moment computation is more statistically-sound.

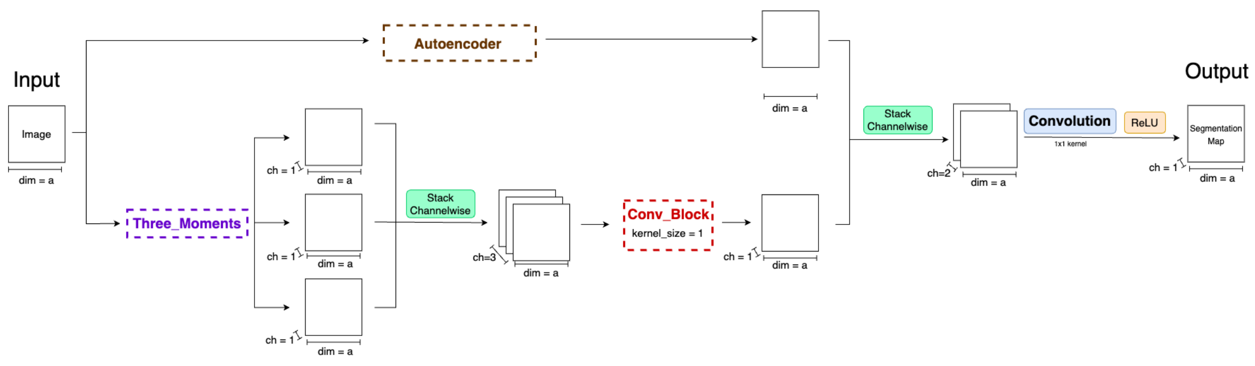


FIGURE 3.12: `AE_Concat_3Moms` segmentation architecture.

To provide a benchmark for model `AE_Concat_3Moms`, model `AE_Concat_Skip` (Architecture A.3.15) was developed, which mirrors the former but incorporates a conventional skip connection between the input image and output of the decoder.

Given that a crucial aspect of the U-Net's functionality relies on compressing the input into a lower-dimensional representation to facilitate global learning, model `3Moms_Concat_BigShrink` (Figure 3.13) seeks to incorporate this principle with a more computationally efficient architecture. This model significantly reduces the input's size with a max pooling layer, resulting in a tensor one-eighth the original height and width, which is equivalent to the bottleneck's spatial dimensions in this project's `Unet`. Subsequently, the tensor undergoes a convolutional layer followed by a transpose convolution to restore its original size. Additionally, to provide the network with statistical information regarding the image, this output is concatenated with the output of the image processed through the moment computation architecture, akin to the approach in architecture `AE_Concat_3Moms`.

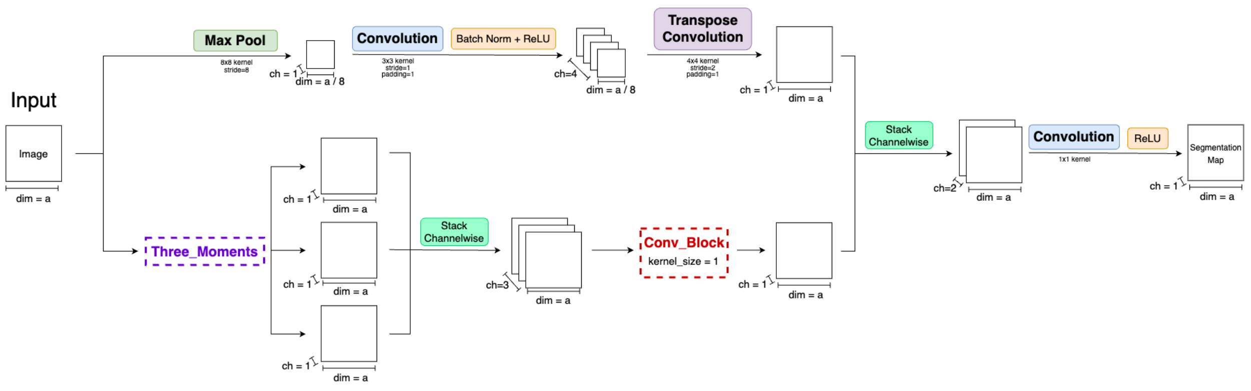


FIGURE 3.13: 3Moms_Concat_BigShrink segmentation architecture.

Model `3Moms_Concat_SmallShrink` employs the same architecture as `3Moms_Concat_BigShrink`, but only halves the input image in the max pooling layer.

The final architecture developed was model `3Moms_B4_Unet`, depicted in Figure 3.14. In this model, moment computation is employed for image pre-processing before inputting it into a standard U-Net. The idea behind this approach is that providing the U-Net with broader information about the input's distribution of pixels could lead to more homogeneous segmentation maps that focus on the high-level regions.

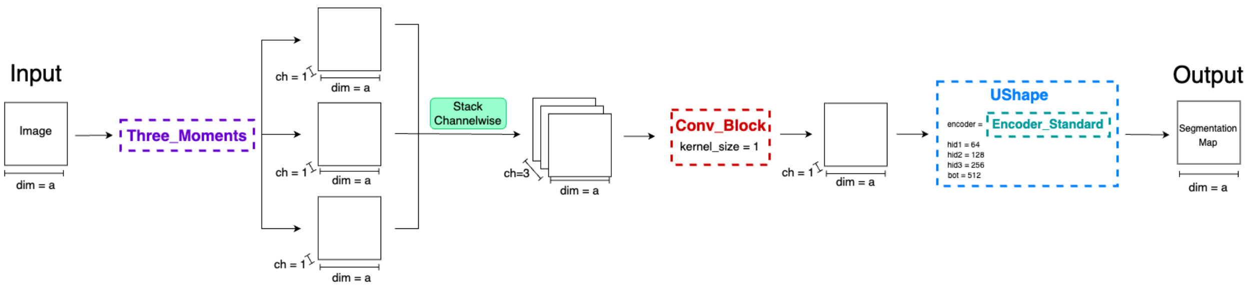


FIGURE 3.14: `3Moms_B4_Unet` segmentation architecture.

3.3 Disparity-Based Loss

As discussed in Section 2.2.3, stochastic distances provide a measure for the distance between probability distributions. They have found applications in various SAR contexts and if used for deep learning, are typically incorporated into the loss function. Although stochastic distances were not yet implemented, this project pursued a preliminary approach incorporating the Euclidean distance between the average pixel values of distinct regions into the loss function. This type of approach, similar to those used in stochastic distance applications, will be referred to as *disparity-based*, meaning that it relies on the assumption that an optimal segmentation will exhibit maximized deviation in the original pixels between the various perceived regions in the predicted segmentation. In the context of SAR, this means that distinct regions would be expected to exhibit maximized disparity between their corresponding roughness parameters. A major benefit of disparity-based losses is that no labels are required, which is extremely valuable for SAR applications where there are few high-quality, labeled datasets. This initial method serves as a starting point and has the potential for further enhancement into a more statistically-driven approach in the future.

The equation for computing the average pixel values of the input data for each perceived region in the predicted segmentation map is displayed below:

$$\hat{\mu}_0 = \frac{\sum_{i=1}^{n} w_{0_i} \cdot \ln(x_i)}{n} \quad \hat{\mu}_1 = \frac{\sum_{i=1}^{n} w_{1_i} \cdot \ln(x_i)}{n}, \tag{3.1}$$

where $\hat{\mu}_0$ is the average pixel value corresponding to predicted region 0, $\hat{\mu}_1$ is the average pixel value corresponding to predicted region 1, w_0 is the tensor corresponding to the $0th$ channel of the predicted segmentation, w_1 is the tensor corresponding to the $1st$ channel of the predicted segmentation, x_i is the input data at pixel i, and n is the number of pixels. Notably, by taking the natural log of x before computing the weighted sums, the equations in 3.1 resemble the first log moment of the perceived regions, which as noted in Section 2.2.2, are known to relate directly to the distribution's parameters.

To compute the loss employed in training, the Euclidean distance was taken between the two averages:

$$\text{ED} = \|\hat{\mu}_1 - \hat{\mu}_0\|^2, \tag{3.2}$$

where ED is the Euclidean distance.

This Euclidean distance was incorporated into the loss function in both a supervised and unsupervised approach. For the supervised approach, ED was combined with the typical binary cross-entropy (BCE) loss linearly as follows:

$$\text{Supervised Loss} = \text{BCE} - k * \text{ED}, \tag{3.3}$$

where k is the Euclidean distance loss coefficient. The ED is subtracted from the BCE, since exhibiting a large ED indicates a good segmentation performance, and should therefore correspond to a lower loss value. When $k = 1$, the two loss computations are factored into the sum equally, when $k < 1$, the overall loss prioritizes the BCE, and when $k > 1$, the overall loss prioritizes the Euclidean distance component. Various coefficients were experimented with to investigate how changing the relative importance of both losses impacts the model's accuracy and efficiency.

The unsupervised approach relies entirely on the ED metric to evaluate loss:

$$\text{Unsupervised Loss} = -\text{ED}. \tag{3.4}$$

Note that the negative sign is still necessary to encourage increased average pixel divergence between regions.

Throughout this section, the standard U-Net was trained on these loss functions to evaluate their relative performance.

Chapter 4

Results

4.1 Experimental Setup

4.1.1 Datasets

The synthetic dataset, whose generation is detailed in Section 3.1, comprises 825 256x256 images, with 80% allocated for training and the remaining 20% for testing.

Additionally, real SAR data samples from diverse sources were utilized. For example, many of the experiments make use of SAR data samples depicting San Francisco and an oil spill. Both images are commonly used in SAR analysis literature, including Fan and Neto (2023). The San Francisco data is $512x1024$ and the oil spill data is $512x512$. They are stored as TIFF files, which is a versatile, high-quality image format that supports lossless compression, multiple layers, and various color spaces (Frery, Wu, and Deniz, 2022). Therefore, it can preserve SAR's intensity data, which is crucial for effective analysis. Various real SAR data images were also retrieved from Capella Space's open-source datasets (Capella, 2021). Capella Space is a private aerospace company specializing in SAR satellite technology. Their SAR images are extremely high resolution, so for the purposes of this project, they were cropped to $1024x1024$ to accommodate Bowdoin HPC's base memory usage limits. They are also stored as TIFF files. Since the San Francisco, oil spill, and Capella data samples do not have labels, they can only be used for qualitative analysis.

Another dataset used in this project is the SARBuD Dataset (Wu et al., 2021; Wu et al., 2022), which contains $20,000$ optical SAR images with associated binary segmentation maps. Each image is a $256x256$ JPEG file. Unlike TIFF files, JPEG formats use lossy compression, which sacrifices some image quality to achieve smaller file sizes. These compression techniques can have the effect of corrupting SAR's statistical characteristics, which is often unideal for analyses, especially in this case where the statistical properties of SAR are of the utmost importance. Figure 4.1 illustrates samples from the aforementioned real SAR data sources:

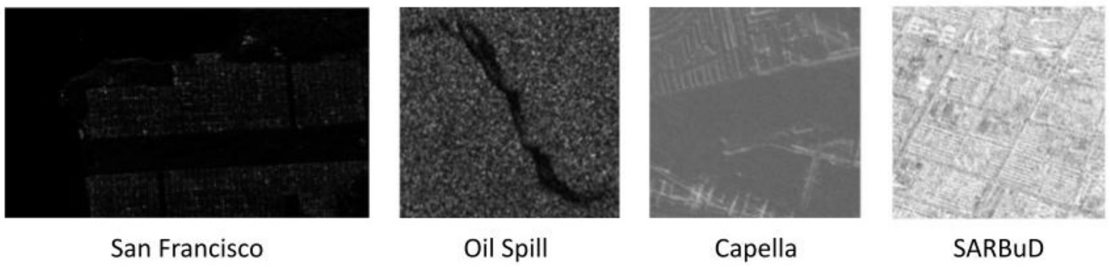


FIGURE 4.1: Real SAR data samples from various sources.

4.1.2 Data Preprocessing

As discussed in Section 2.2.2, there is a close relationship between log cumulants and the log of statistical moments of the same order. Consequently, taking the natural log of SAR data before processing has been shown to stabilize subsequent training. Therefore, data preprocessing involved taking the natural log and then applying standard normalization to the images before training.

The real SAR test images underwent resizing before processing, imposing a constraint that height and width dimensions were the nearest power of 2^8. Otherwise, the `U-Shape` architectures face problems when they attempt to reduce a tensor of odd height or width by half.

4.1.3 Hyperparameters for Model Training

Binary cross-entropy (BCE) was employed as the loss function for model training (besides, of course, when the loss function was the subject of experimentation). BCE is a commonly used loss function in binary classification tasks. It is defined by the formula:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right], \tag{4.1}$$

where y represents the ground truth labels (0 or 1), $\hat{y}$ represents the predicted probabilities, and N is the number of samples. By minimizing the value outputted from this loss function, algorithms penalize deviations between predicted probabilities and true labels, helping the model learn to accurately classify pixels as foreground or background. The machine learning library used to implement this project, PyTorch (Paszke et al., 2019), also automatically applies a *softmax* function to the raw y prediction before computing the cross-entropy. Softmax is a mathematical function that converts a vector of real numbers into a probability distribution. The softmax equation, defined in Equation 4.2 calculates the probability s_i of the i-th class given the input vector z and ensures that all probabilities sum up to 1:

$$s_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}. \tag{4.2}$$

All models in this project were trained using an initial learning rate of *0.001*, *70* epochs, a batch size of *8*, and the *ADAM Optimizer*. Some architectures also employ batch normalization and dropout (all `UShape` architectures employ dropout with probability *0.5* at the bottleneck). The models were trained using the NVidia Turing GeForce RTX 3080 10 GB available through Bowdoin's HPC.

The hyperparameters were chosen through random experimentation using the U-Net architecture. Extensive hyperparameter tuning was not necessary, since the focus was on ensuring consistency across hyperparameters for fair comparison across architectures and loss functions, rather than striving for the highest possible model performance.

4.1.4 Evaluation Metrics

Throughout the project, segmentation performance was evaluated using the *Intersection over Union* (IOU) metric:

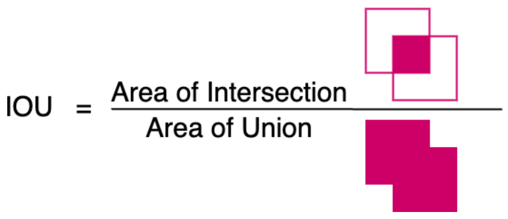


FIGURE 4.2: Intersection over Union metric.

IOU provides a quantitative measure of the spatial overlap of the regions in the prediction and ground truth, as well as how accurately the pixel values line up. A perfect segmentation will produce an IOU of *1* and a predicted segmentation with no regional overlap with the ground truth will produce an IOU of *0*.

Qualitative analyses were also extremely valuable in this context. By aligning a segmentation map side-by-side with the original data, one can infer how accurately the segmentation model identified key regions and shapes in the SAR imagery.

4.2 Generated Synthetic Dataset

For each ground truth label out of the 200 generated segmentation maps, corresponding synthetic SAR images were produced by sampling from different combinations of roughness parameters. Specifically, for each label, data was generated by sampling from the G^0 distribution with all permutations of 11 α values ranging from -1.5 to -11. The dataset and segmentation maps were limited to two regions, constituting a binary segmentation task.

To evaluate whether a segmentation model could be trained on this data, a standard U-Net was first trained and texted on one combination of roughness parameters at a time. Table 4.1a presents the average test IOUs and Table 4.1b presents the average inference times of a U-Net model trained exclusively on the corresponding α combinations.

TABLE 4.1: IOU and inference times for U-Nets trained on one combination of roughness parameters (α) at a time (L=1).

α_2 \ α_1	-1.5	-3	-5	-9
-3	0.973			
-5	0.981	0.951		
-9	0.985	0.976	0.956	
-11	0.986	0.979	0.966	0.831

(A) Average IOU

α_2 \ α_1	-1.5	-3	-5	-9
-3	1.413			
-5	1.290	1.293		
-9	1.236	1.318	1.282	
-11	1.237	1.337	1.292	1.361

(B) Average Inference Time

Figure 4.3 provides a qualitative evaluation of these experiments, displaying a segmentation map produced from models trained on each roughness parameter combination. In the pairs of images corresponding to each roughness parameter combination, the leftmost image displays an example image generated by sampling from the G^0 distribution with those roughness parameter (α) values, and the rightmost image displays the predicted segmentation. The top right corner of the Figure contains the ground truth segmentation.

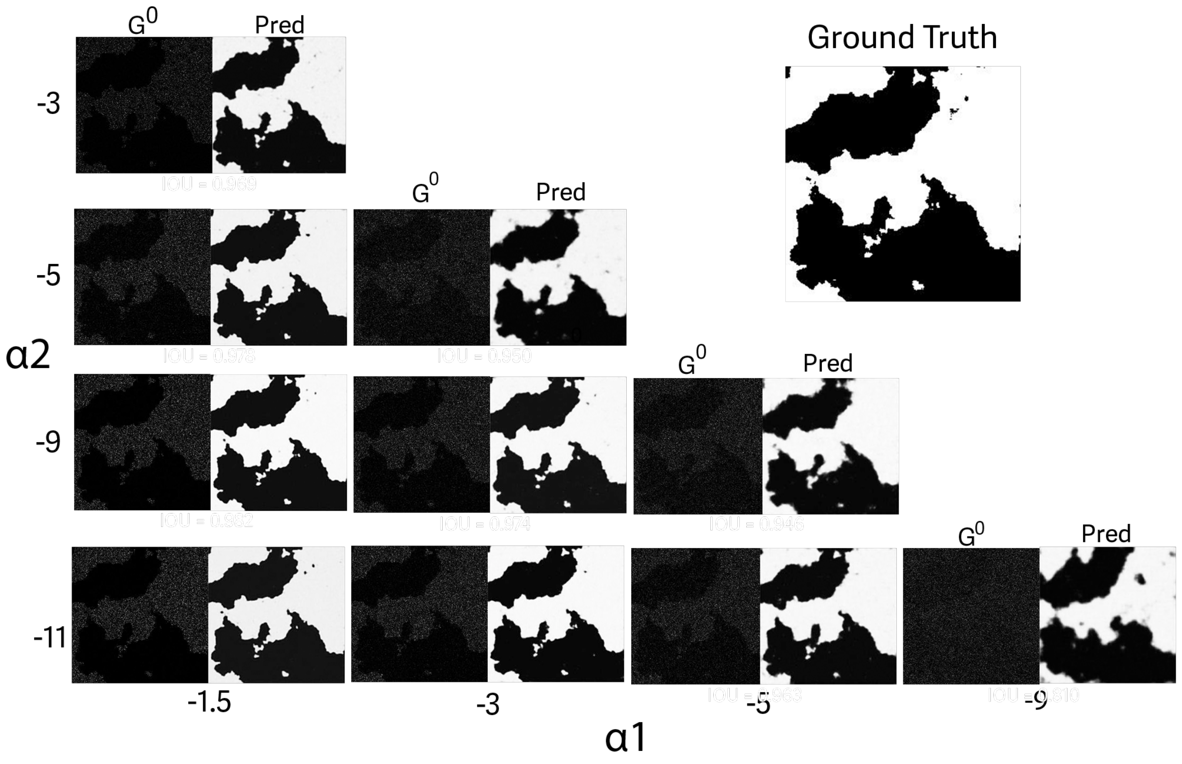


FIGURE 4.3: Synthetic data generation preliminary results trained on two αs at a time ($L = 1$).

Based on the metrics in Table 4.1 and the qualitative results Figure 4.3, U-Nets were able to train effectively on synthetic data samples generated from one combination of roughness parameters at a time. Moreover, models trained on more distinct α parameters performed better.

While the previous experiments provide a good indication of each roughness combination's capacity to train a U-Net, the exact roughness parameters in real SAR data varies and is unknown. Therefore, to emulate the diverse quality of SAR images that might be encountered in the real world, a U-Net was then trained on the entire synthetic dataset, which contains various combinations of roughness parameters. The quantitative results, including the average IOU and average inference time, of the U-Net's performance on the integrated roughness synthetic dataset is displayed below:

TABLE 4.2: Metrics from models trained on a large variety of α combinations ($L = 1$).

Average IOU:	0.822
Average Inference Time (ms):	1.342

A corresponding qualitative result is displayed in Figure 4.4, which shows example segmentation maps produced by a U-Net trained on integrated roughness combinations and tested on synthetic data generated by specified roughness parameters. Below each data-segmentation pair is the IOU of the particular corresponding segmentation displayed above it.

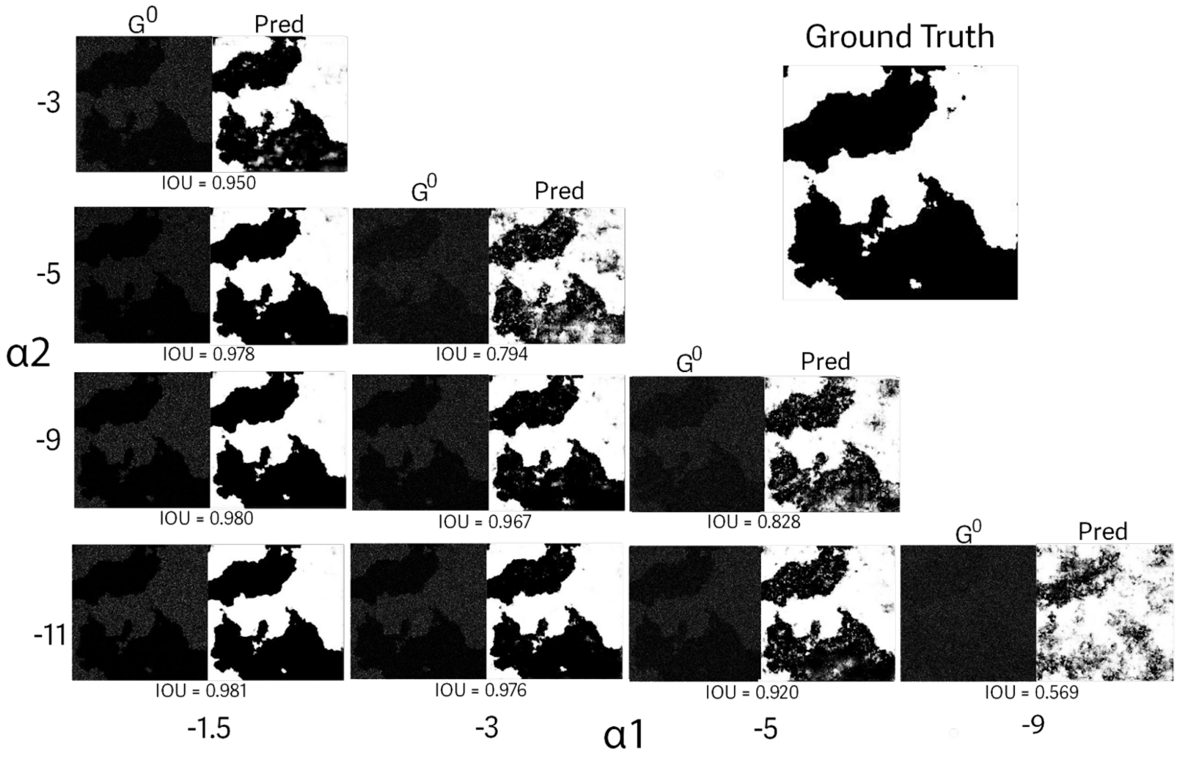


FIGURE 4.4: Synthetic data generation preliminary results trained on all α combinations at once ($L = 1$).

Although the U-Net performed worse when trained on a diverse combination of roughness parameters, it was still able to discern the various regions for most test examples, again having more difficulty when the α parameters were closer together.

Another dataset generation hyperparameter that was experimented with was the number of looks (L). Figure 4.5 shows the results of various `U-Nets` that were trained on individual datasets generated by sampling from the G^0 distribution using different Ls, while maintaining a constant $\alpha_1 = -1.5$ and $\alpha_2 = -3.0$. Below each data-segmentation pair is the average IOU over the entirety of each experiment.

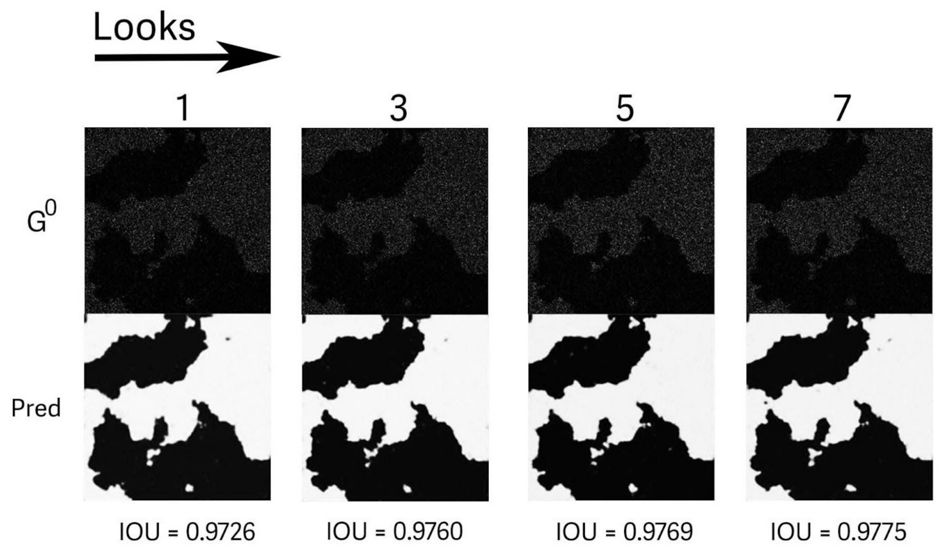
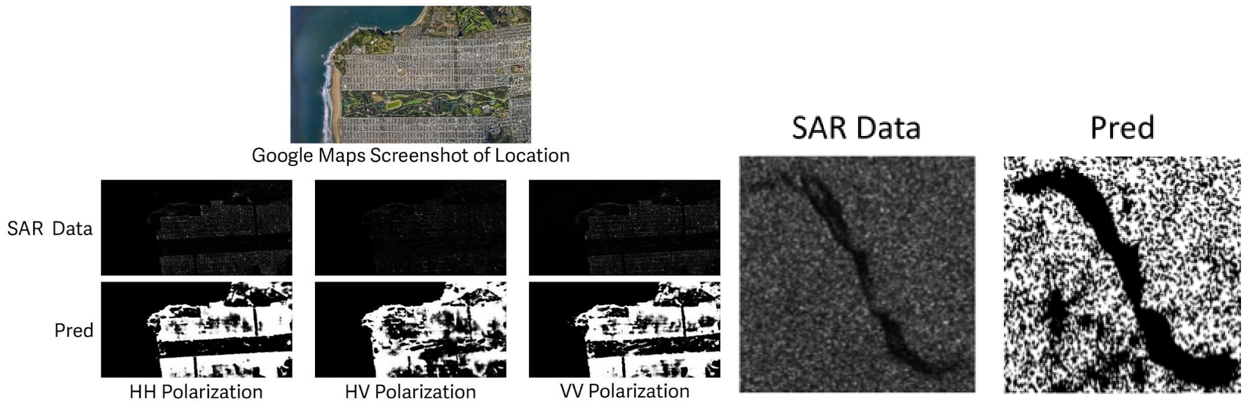


FIGURE 4.5: Synthetic data generation preliminary results trained on varying *look* parameter.

As shown in Figure 4.5, the performance of the U-Net is slightly better when trained on a larger number of looks, though not by a very large margin.

Finally, since the ultimate goal of synthetic data is to train a model that can be used on real SAR data, a U-Net trained on the integrated roughness synthetic dataset was tested on real SAR data. Figures 4.6a and 4.6b display the model's performance on real SAR data depicting San Francisco and an oil spill respectively.



(A) San Francisco (B) Oil Spill

FIGURE 4.6: Test reults of U-Net trained on synthetic data and tested on real SAR data.

Based on these segmentations, it appears that the U-Net trained on the integrated roughness dataset was able to discern regions in real SAR data. However, there is some granularity in the segmentation maps, which is especially evident in Figure 4.6b.

4.3 Statistically-Motivated Architecture Performance

4.3.1 Architectures Trained on Synthetic Data: Quantitative

The quantitative performance of all of the architectures trained and tested on the synthetic dataset are displayed in Table 4.3. The metrics include final test cross-entropy loss, final test IOU, and average training time per epoch. The best-performing architectures, with respect to final test cross-entropy loss and IOU, are denoted by stars, where the architectures signified by $***$, $**$, and $*$ symbols, achieved the best, second-best, and third-best scores respectively.

TABLE 4.3: Performance of architectures on synthetic dataset.

Architecture Name	Final Test Cross-Entropy Loss	Final Test IOU	Training Time Per Epoch (s)
Unet	0.292	0.861	726.98
Autoencoder*	0.133	0.943	1115.59
3Moms	0.586	0.561	25.07
AE_Avg	0.234	0.812	412.81
Unet_1Mom	0.248	0.828	638.39
Unet_3Moms_1	0.501	0.585	655.76
Unet_3Moms_2	0.344	0.776	664.64
Unet_3Moms_3Chans_1	0.52	0.641	75.52
Unet_3Moms_3Chans_2	0.504	0.64	77.64
Unet_3Moms_Skip_1	0.353	0.858	850.34
Unet_3Moms_Skip_2	0.361	0.851	863.62
Unet_3Moms_Skip_BigKern	0.16	0.92	1365.97
Unet_BigKern	0.134	0.924	1242.11
AE_Concat_3Moms ***	0.107	0.945	1127.49
AE_Concat_Skip**	0.118	0.944	1119.83
3Moms_Concat_BigShrink	0.578	0.528	30.79
3Moms_Concat_SmallShrink	0.536	0.584	31.42
3Moms_B4_Unet	3.988	0.622	737.32

As detailed in Table 4.3, the `Unet` trained on the synthetic dataset was able to achieve a final test cross-entropy loss of *0.29*, a final test IOU of *0.86* and took *726.98 s* to train on average per epoch. The `Autoencoder`, achieved a final test cross-entropy loss of *0.13* and a final test IOU of *0.94*, outperforming the `Unet` with respect to these metrics.

The only two models that outperformed the `Autoencoder` with respect to both final test cross-entropy loss and final test IOU were models `AE_Concat_3Moms` and `AE_Concat_Skip`, with model `AE_Concat_3Moms` exhibiting superior quantitative performance on the synthetic data among all of the architectures. They both also trained slightly faster than the `Autoencoder`.

While not quite matching the performance of the `Autoencoder`, albeit by a small margin, both the `Unet_3Moms_Skip_BigKern` and `Unet_BigKern` models vastly outperformed the standard `Unet` with respect to both loss and IOU. However, they exhibited the largest training time per epoch out of all of the other models.

Besides the previously mentioned front-runners, only the `AE_Avg`, `Unet_1Mom`, `Unet_3Moms_2`, `Unet_3Moms_Skip_1`, and Unet_3Moms_Skip_2 models obtained final test cross-entropy losses and final test IOUs comparable to the `Unet`, and all but the ladder two outpaced the `Unet` with respect to training time.

4.3.2 Architectures Trained on Synthetic Data: Qualitative

This section showcases visual representations of three validation images, each with varying levels of anticipated segmentation difficulty, that have undergone segmentation by the top eight performing architectures. When analyzing the performance of the varying architectures, it is useful to compare the quantitative measures (final test cross-entropy loss, final test IOU, training time), as well as observe examples of segmentation maps produced by the models, since the the visual information can reveal aspects of the performance that are not obvious through the quantitative metrics. At the top of each figure, alongside the synthetic data example, is the associated ground truth segmentation map.

The synthetic image in Figure 4.7 was expected to be 'easy' to segment, as the two regions have the most distinct α values of the three examples: $\alpha_1 = -2$ and $\alpha_2 = -11$. This is also evident visually, as the eye can clearly discern the dark and light regions, despite the noise.

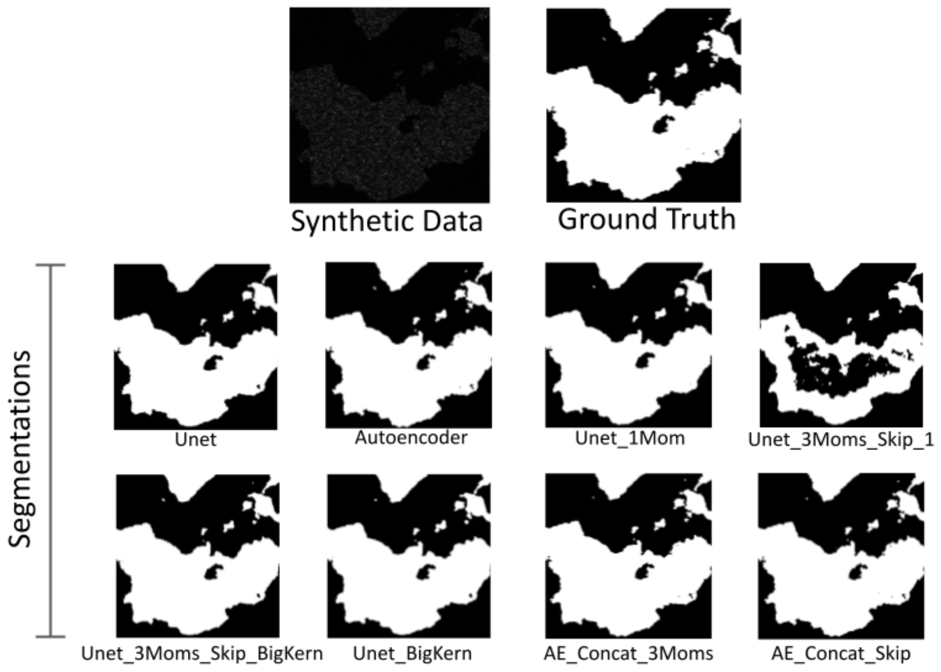


FIGURE 4.7: Segmentations for synthetic data generated from $\alpha_1 = -2$ and $\alpha_2 = -11$ across architectures.

The synthetic data example in Figure 4.8 was expected to be of 'medium' difficulty for the models to segment, with α values: $\alpha_1 = -2$ and $\alpha_2 = -4$. Visually, the various regions are not hard to discern, but not as clear as the previous data example.

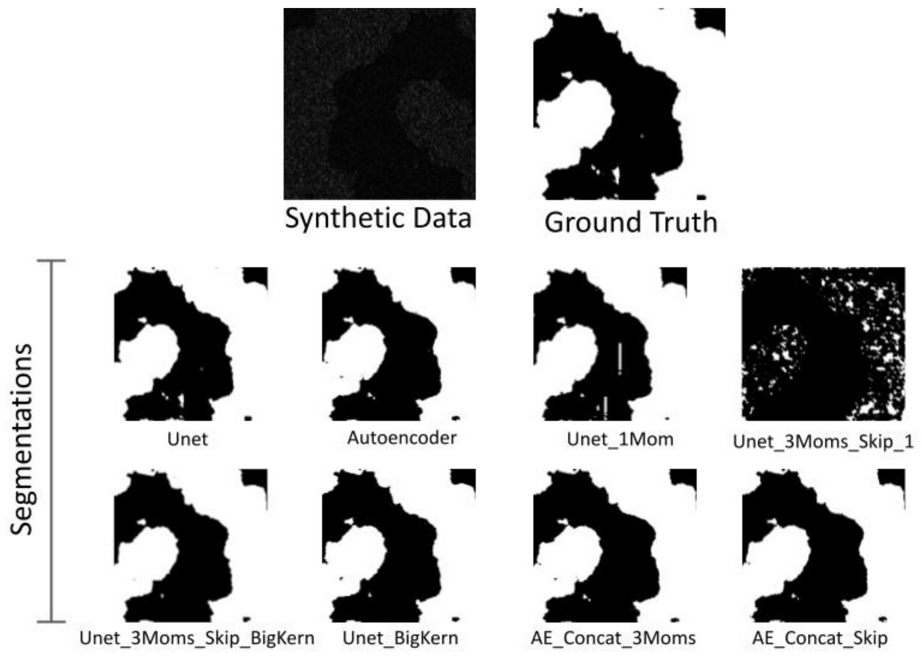


FIGURE 4.8: Segmentations for synthetic data generated from α_1=-2 and α_2=-4 across architectures.

The synthetic data example in Figure 4.9 was expected to be 'hard' for the models to segment, as the two regions have the least distinct α values of the three examples: $\alpha_1 = -6$ and $\alpha_2 = -7$. Evidently, it is very difficult to discern the various regions with the human eye.

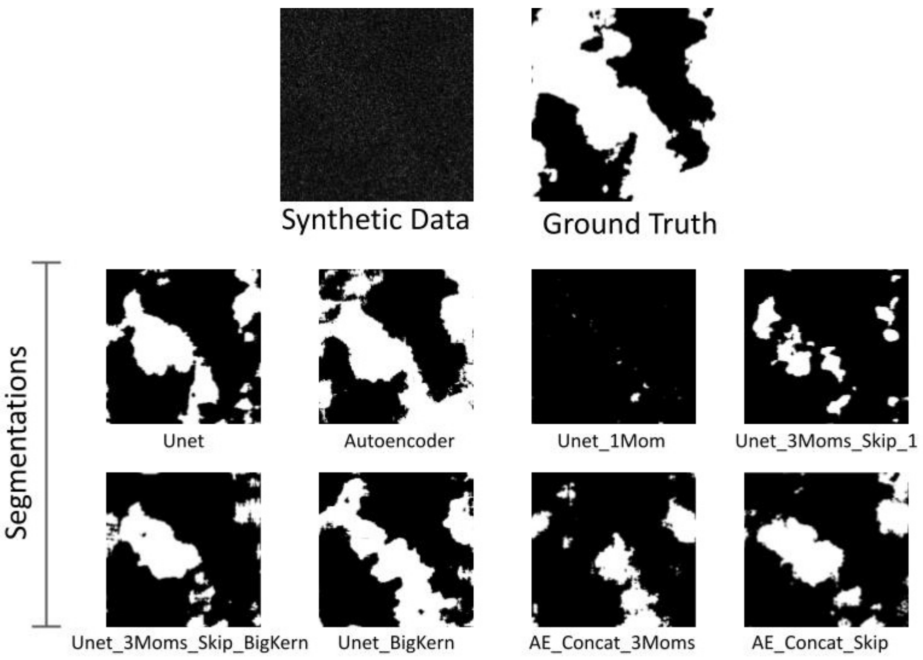


FIGURE 4.9: Segmentations for synthetic data generated from $\alpha_1 = -6$ and $\alpha_2 = -7$ across architectures

The relatively successful validation examples in Figures 4.7, 4.8, and 4.9 support the quantitative results for these eight highest-performing architectures. The `Unet_3Moms_Skip1` and `Unet_1Mom` models' segmentations of the 'easy' and 'medium' examples appear slightly less accurate than the other architectures, while the `Autoencoder`, `Unet`, and `Unet_BigKern` models appear to produce the most successful segmentations of the 'difficult' example.

The segmentation maps displayed in Figures 4.10, 4.11, and 4.12 present the results of testing the top eight performing models on real SAR data depicting San Francisco, an oil spill, and Capella data, respectively.

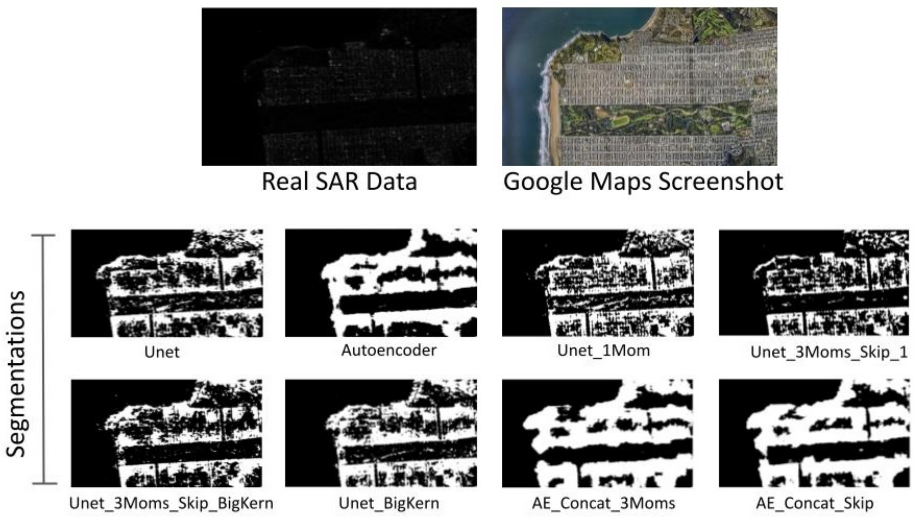


FIGURE 4.10: Segmentations for real SAR data of San Francisco across architectures.

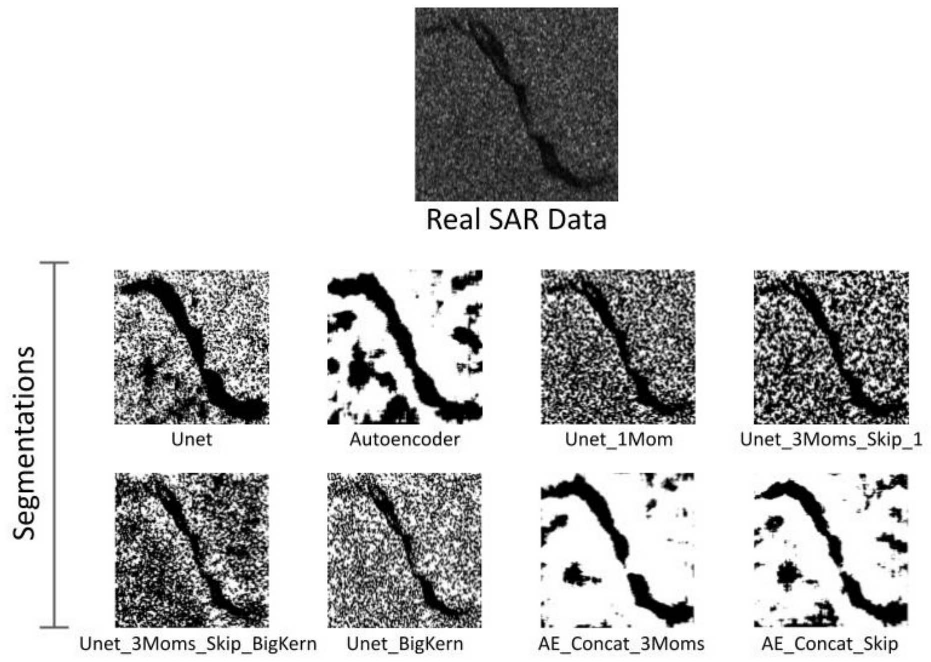


FIGURE 4.11: Segmentations for real SAR data depicting oil spill across architectures.

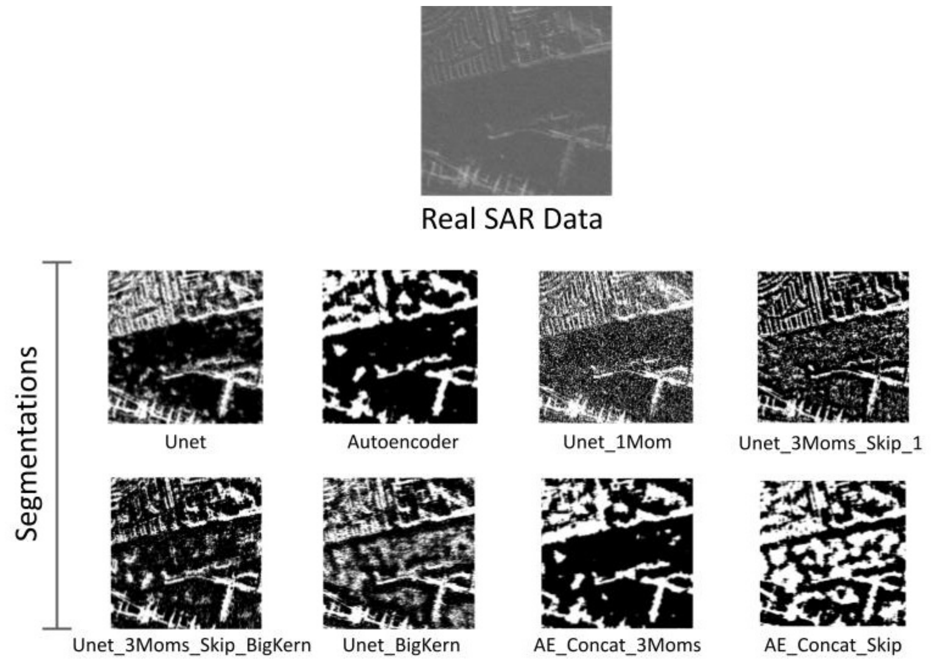


FIGURE 4.12: Segmentations for real SAR data from Capella dataset across architectures.

The architectures appear to all produce segmentations with varying degrees of success and fine-grained detail. Specifically, the autoencoder-based models produced much more homogeneous segmentation maps, particularly the `Unet_Concat_3Moms`, while models incorporating skip connections, such as model `Unet_1Mom`, produced segmentation maps with much more detail.

4.3.3 Architectures Trained and Tested on Optical SAR Imagery

Table 4.4 displays metrics of eight models that were trained and tested on optical SAR imagery from the SARBuD dataset. These architectures are the eight models that performed the best on the synthetic dataset. The best-performing architectures, with respect to final test cross-entropy loss and IOU, are denoted by stars, where the architectures signified by $***$, $**$, and $*$ symbols, achieved the best, second-best, and third-best scores respectively.

TABLE 4.4: Performance of best architectures trained and tested on real SAR dataset (SARBuD).

Architecture Name	**Final Test Cross-Entropy Loss**	**Final Test IOU**	**Training Time Per Epoch (s)**
Unet*	0.197	0.777	22200.32
Autoencoder	0.25	0.736	33754.87
AE_Avg	0.429	0.514	12564.2
Unet_1Mom	0.966	0.429	19221.04
Unet_3Moms_Skip_1**	0.199	0.778	25904.16
Unet_3Moms_Skip_2***	0.175	0.785	26181.81
Unet_3Moms_Skip_BigKern**	0.182	0.777	41332.65
Unet_BigKern	0.281	0.73	37719.52
AE_Concat_3Moms	0.296	0.639	34168.75
AE_Concat_Skip	0.296	0.668	33863.97

Based on Table 4.4, the `Unet_3Moms_Skip_2`, `Unet_3Moms_Skip_1`, `Unet_3Moms_Skip_BigKern`, and `Unet` models performed the best, with respect to final test cross-entropy loss and IOU, on the SARBuD dataset. However, these metrics do not match the performance of the architectures trained and validated on the synthetic dataset.

Figure 4.13 shows a qualitative example of the models' performance on a validation image from the SARBuD dataset that the models did not see during training.

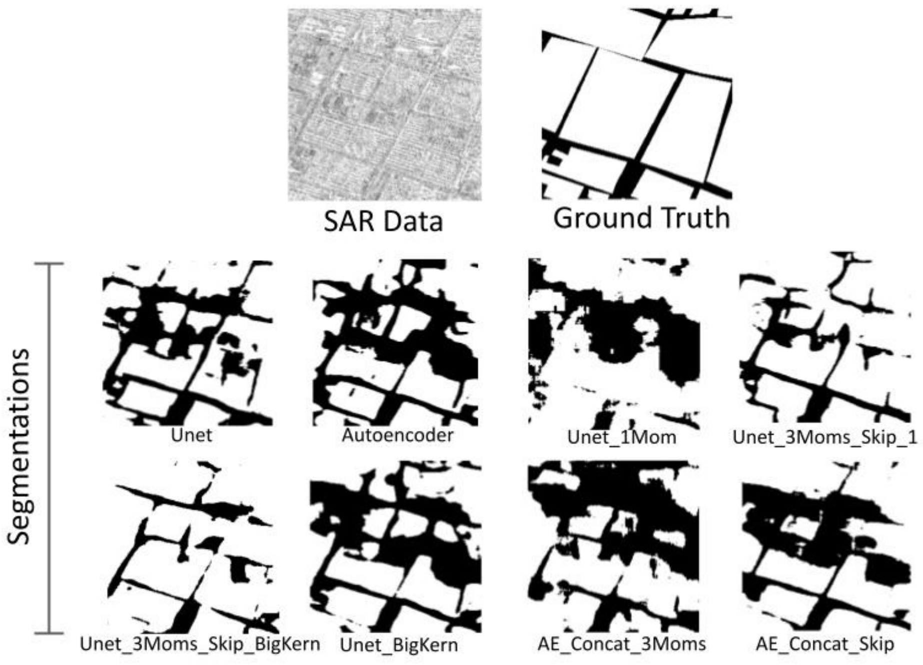


FIGURE 4.13: Segmentations for validation SARBuD image across architectures trained on SARBuD.

Mirroring the quantitative metrics, the `Unet_3Moms_Skip_1` and `Unet_3Moms_Skip_BigKern` models appeared to have produced the most accurate segmentation maps, while the `Unet_1Mom` model had the most trouble.

Figures 4.14 and 4.15 provide test examples displaying how well the architectures trained on the SARBuD dataset performed on real SAR data samples from alternative sources.

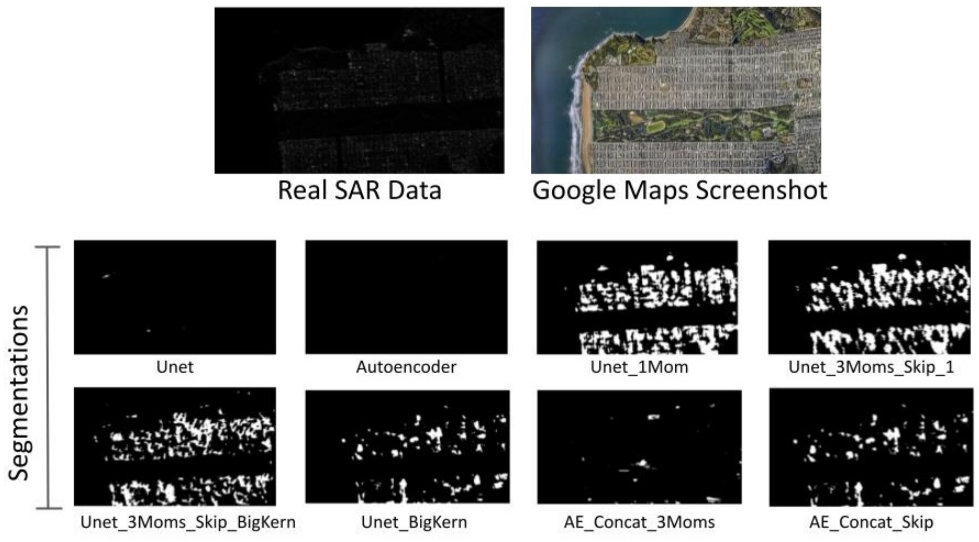


FIGURE 4.14: Segmentations for real SAR data of San Francisco across architectures trained on SARBuD.

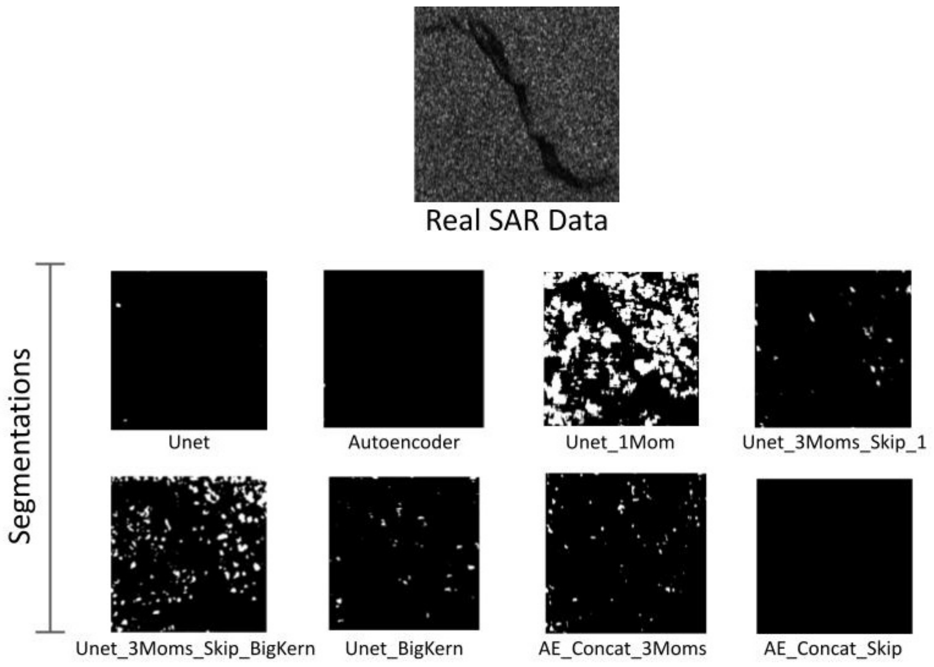


FIGURE 4.15: Segmentations for real SAR data of oil spill across architectures trained on SARBuD.

Especially compared to the synthetically-trained models, the architectures trained on the SARBuD dataset were less successful in discerning regions of real SAR data. Particularly in Figure 4.14, the `Unet_1Mom`, `Unet_3Moms_Skip_1`, and `Unet_3Moms_Skip_BigKern` architectures appeared to produce slightly more successful segmentations than the other architectures.

4.3.4 Architectures Trained on Synthetic Data and Tested on Optical SAR Imagery

Table 4.5 showcases the final test cross-entropy losses and final test IOU of eight models that were trained on the synthetic dataset, and tested on the SARBuD dataset. These architectures are the eight models that performed the best on the synthetic dataset. The best-performing architectures, with respect to final test cross-entropy loss and IOU, are denoted by stars, where the architectures signified by $* * *$, $**$, and $*$ symbols, achieved the best, second-best, and third-best scores respectively.

TABLE 4.5: Performance of architectures trained on synthetic dataset and tested on SARBuD dataset.

Architecture Name	Final Test Cross-Entropy Loss	Final Test IOU
Unet	3.536	0.3
Autoencoder	2.695	0.367
AE_Avg	3.264	0.271
Unet_1Mom	3.099	0.291
Unet_3Moms_Skip_1 ***	0.846	0.444
Unet_3Moms_Skip_2	2.407	0.341
Unet_3Moms_Skip_BigKern**	2.052	0.378
Unet_BigKern	2.426	0.36
AE_Concat_3Moms*	2.502	0.379
AE_Concat_Skip	2.775	0.352

All of the synthetically-trained models performed badly when tested on the SARBuD dataset.

4.4 Disparity-Based Loss Performance

4.4.1 Supervised Approach: Combining Cross-Entropy and Disparity-Based Losses

Table 4.6 showcases the standard U-Net's quantitative performance across various Euclidean distance loss coefficients. When the coefficient is 0, the loss function is entirely cross-entropy loss. When the coefficient is < 1, the loss function emphasizes cross-entropy loss more than the Euclidean distance loss. When the coefficient is > 1, the loss function emphasizes the Euclidean distance loss more than the cross-entropy loss. The Euclidean loss coefficients that led the U-Net towards the best results, with respect to final test IOU, are denoted by stars, where the coefficients signified by $* * *$, $**$, and $*$ symbols, led to the best, second-best, and third-best scores respectively.

TABLE 4.6: U-Net trained incorporating Euclidean distance between region average pixel values into loss function.

Euclidean Distance Loss Coefficient	Final Test IOU	Training Time Per Epoch (s)
0***	0.865	746.81
0.001**	0.861	747.15
0.01*	0.852	747.8
0.1*	0.852	748.82
1	0.454	747.25
10	0.851	747.37
100	0.212	747.24

As shown in Table 4.6, the U-Net generally performed worse as the disparity-based portion of the combined loss was given a greater weight.

Figure 4.16 showcases the U-Net's validation performance on a synthetic data sample that it had not yet seen before. The data was sampled from $\alpha_1 = -2$ and $\alpha_1 = -4$, which has previously been regarded as a 'medium' difficulty image to segment.

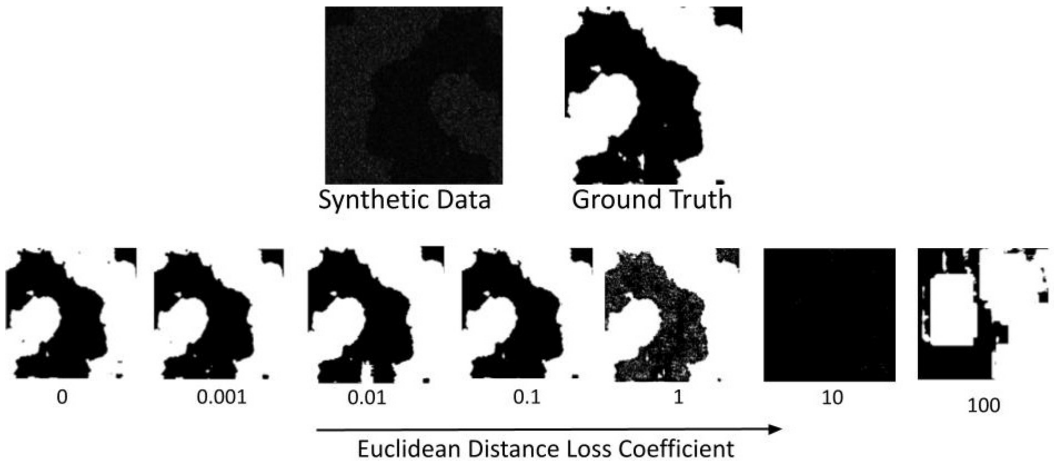


FIGURE 4.16: U-Net segmentation on validation synthetic data across Euclidean distance loss coefficients (data sampled from $\alpha_1 = -2$ and $\alpha_2 = -4$).

Similar to the quantitative results, the best segmentation maps were produced by U-Nets that gave the disparity-based loss the least weight.

The models that were trained with varying Euclidean distance loss coefficients were then tested on real SAR data samples. Figures 4.17 and 4.18 display the segmentation results of the U-Nets tested on SAR data of San Francisco and from the Capella dataset respectively.

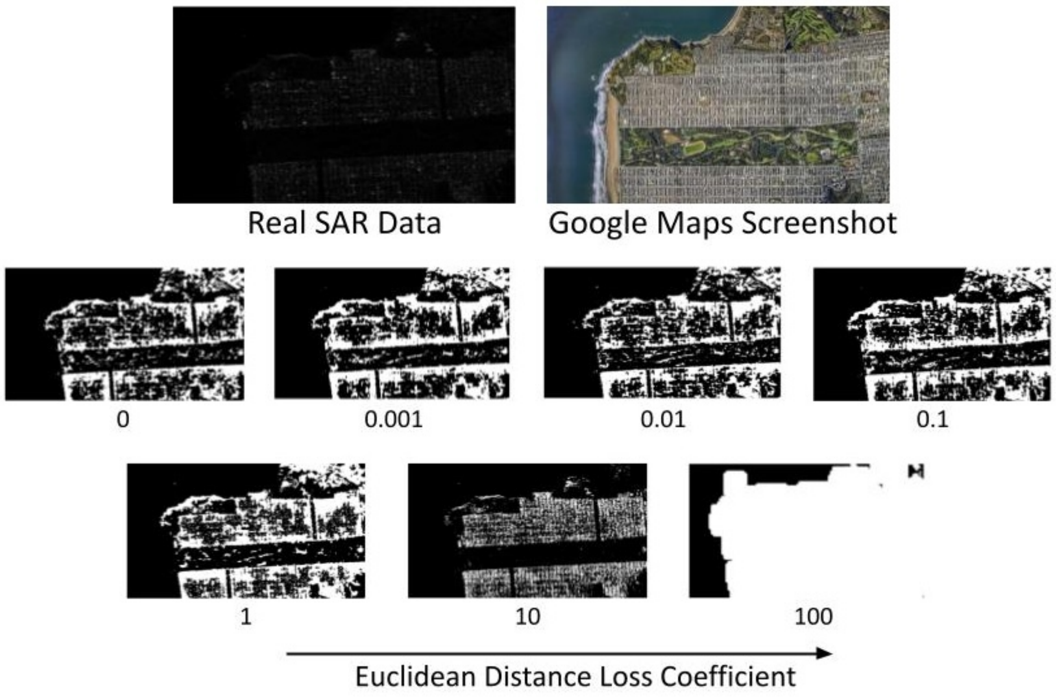


FIGURE 4.17: Segmentations for real SAR data of San Francisco across Euclidean distance loss coefficients.

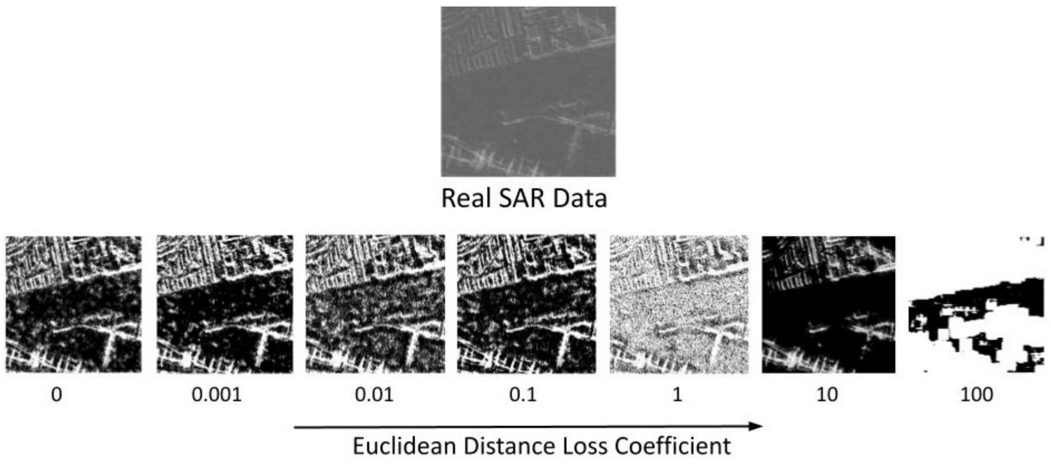


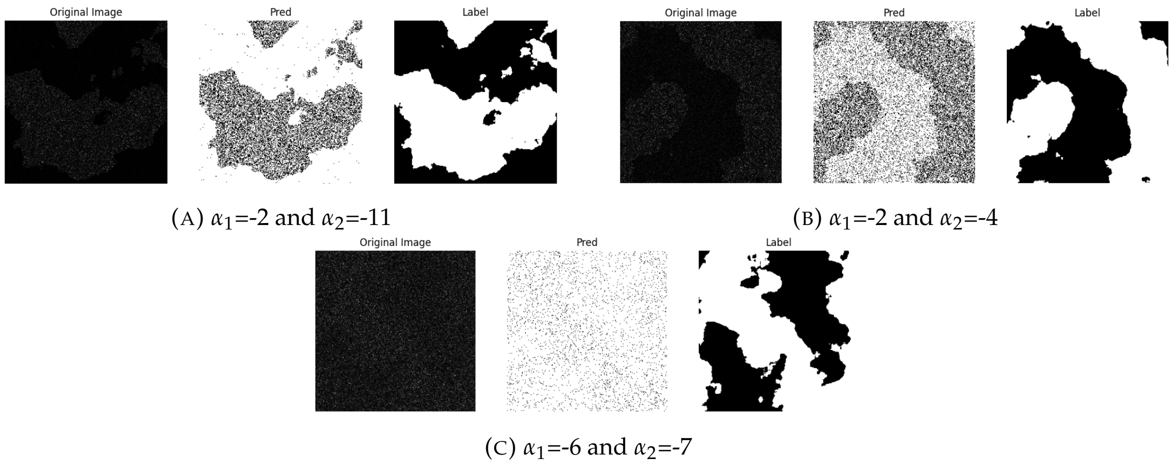
FIGURE 4.18: Segmentations for real SAR data from Capella dataset across Euclidean distance loss coefficients.

Once again, the models' performance worsened when the Euclidean loss coefficient was greater. However, when the loss coefficient was given the greatest weight of 100, it appeared to have been able to discern the regions in an extremely general sense.

4.4.2 Unsupervised Approach: Disparity-Based Loss

When the U-Net was trained and tested on synthetic data and only considered the Euclidean distance between the average pixel values of the predicted regions as the loss function, it achieved a final test IOU of *0.71*. Its training also ended with a final test loss of *-0.29*, though the interpretation of this loss cannot be compared to the previously documented cross-entropy loss values. The training took about *746.79 s* per epoch on average, which is functionally equivalent to the training time results for the typical cross-entropy loss.

Figure 4.19 displays the performance of the U-Net using the unsupervised Euclidean distance loss on various synthetic data samples that come from the training dataset, but the model did not train on.



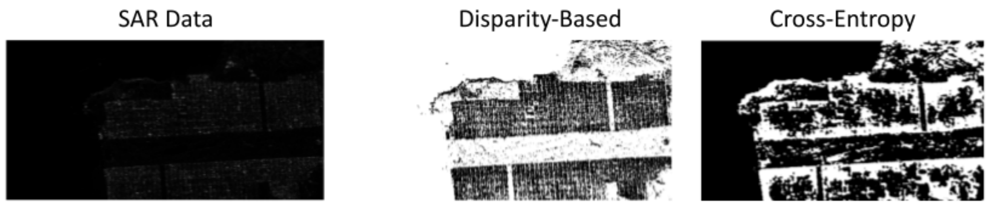
(A) α_1=-2 and α_2=-11

(B) α_1=-2 and α_2=-4

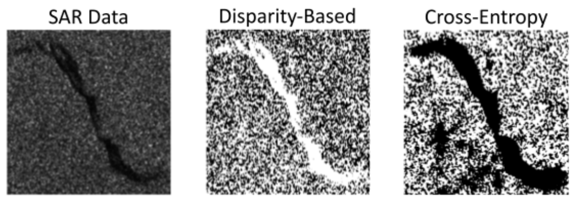
(C) α_1=-6 and α_2=-7

FIGURE 4.19: U-Net segmentation performance on validation synthetic data samples using unsupervised Euclidean distance loss.

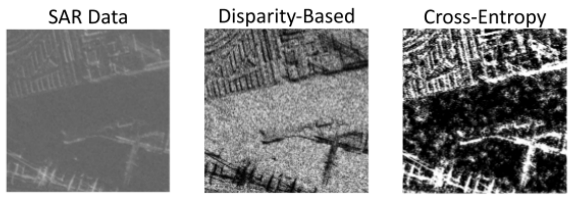
Figure 4.20 displays the performance of the U-Net using the unsupervised Euclidean distance loss on various test data samples containing real SAR data from various sources. To provide a benchmark for comparison, the predicted segmentation produced by a U-Net guided by the unsupervised disparity-based loss was placed alognside the predicted segmentation produced by a U-Net guided by typical cross-entropy loss.



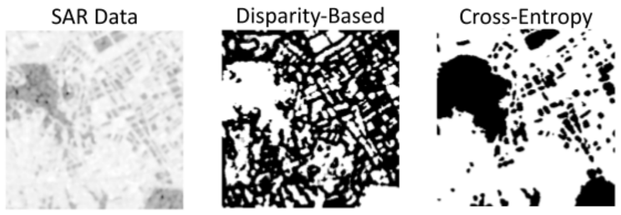
(A) San Francisco



(B) Oil Spill



(C) Capella



(D) SARBuD

FIGURE 4.20: U-Net segmentation on real SAR test data samples using unsupervised Euclidean distance loss.

Figures 4.19 and 4.20 indicate that the disparity-based loss is effective when used as the sole loss metric. The resulting segmentation maps, however, are extremely fine-grained.

Chapter 5

Discussion

5.1 Synthetic Dataset

The test results displayed in Table 4.1 and Figure 4.3 indicate that the synthetic dataset exhibits a diverse range of visual interpretability, with increased difficulty when the roughness (α) parameters in one image are close together, particularly for α values that are more negative. This trend is also visible in the qualitative results, as well as in the average IOU metrics. However, even in the worst case, $\alpha_1 = 9$ and $\alpha_2 = -11$, the model still achieves an IOU of around *0.83*, and the segmentation map still retains the general trends of the ground truth, if not the small details.

As anticipated, Table 4.2 and Figure 4.4 indicate that the U-Net trained on a dataset consisting of various combinations of roughness parameters had more difficulty discerning various regions, particularly for data samples generated from similar roughness values. However, the average IOU of *0.82* is still relatively high and the increased diversity of the training dataset likely resulted in a model that was more versatile and effective for discerning regions in real SAR data. Moreover, it is worth noting that the average inference time for the U-Net trained on the synthetic dataset was quite fast, especially compared to statistical models which take at least a few seconds to generate a segmentation map.

Additionally, as depicted in Figure 4.5, increasing the number of looks did result in more successful U-Net performance, but not so much so to make a very significant impact. Since the U-Net performed relatively well on data generated using one look in all of these tests, the rest of the project will assume a look parameter (L) of 1.

The qualitative test results in Figure 4.6a and Figure 4.6b are especially promising, indicating that a U-Net trained on synthetic data was able to discern the regions of real SAR data in a general sense. Although this attests to the dataset's efficacy in training a model for applied segmentation tasks, the granularity of the segmentation map in Figure 4.6b suggests the potential for further improvements.

5.2 Statistically-Motivated Segmentation Architectures

5.2.1 Benchmark Architectures

As noted in the Results Section, it is important to consider examples of predicted segmentation maps (using both synthetic and real test SAR data) when comparing the performance of segmentation architectures, in addition to the quantitative measures. The significance in doing so is apparent when considering the relative performance of the two benchmark architectures: the `Unet` and the `Autoencoder`. As displayed in Table 4.3, the `Autoencoder` appears to have outperformed the `Unet` with respect to both final test cross-entropy loss and final test IOU. This conclusion is supported by the qualitative validation results in Figure 4.9, where it appears that the `Autoencoder` was able to better discern the general shape of the synthetic input data's ground truth segmentation map. However, it is more clear in the test images in Figures 4.10, 4.11, and 4.12 that the primary distinction between the architectures was in the level of detail in their segmentation maps.

The Unet appears to capture more detailed features, such as the water regions of the park in Figure 4.10, while the Autoencoder produced more homogeneous regions. This is expected, since the skip connections in U-Nets enable the preservation of low-level features, and autoencoders have no such connections. Instead, they rely only on the condensed, latent representation of the image at the bottleneck to reconstruct a segmentation. It is likely that if hyperparameters were chosen that resulted in synthetic data with more fine-grained details, the Unet would have performed better quantitatively than the Autoencoder. With no labels for the real SAR data, there is not necessarily a correct answer to which segmentation map is 'better'. Typically, computer scientists strive to balance the benefits of specificity and homogeneity, with decisions contingent upon the particular domain and task at hand. This trade-off will be important to bear in mind throughout the entirety of this analysis.

5.2.2 Architectures Employing Branches in Parallel with an Autoencoder

As indicated in Table 4.3, the best-performing architecture, with respect to quantitative metrics, was model AE_Concat_3Moms (Architecture A.3.14). It achieved the lowest final cross-entropy test loss of *0.107* and the highest final test IOU of *0.945*. It evidently also performed well on the 'easy' (Figure 4.7) and 'medium' (Figure 4.8) validation data samples, though it did not seem to perform as well as other models on the 'hard' validation sample (Figure 4.9). This one data sample could have been an outlier, with model AE_Concat_3Moms more significantly outperforming the other models on other data samples. Based on the qualitative tests on real SAR data (Figures 4.10, 4.11, and 4.12), model AE_Concat_3Moms was able to discern the various regions across all of the real SAR data sources, producing more homogenous regions than even the Autoencoder. This result coincides with the statistical background of this model. As discussed, the Autoencoder is more likely to produce regions with less fine-grained detail, due to its lack of skip connections. The additional moment computation performed on the input image in architecture AE_Concat_3Moms would ideally provide even more information about the probability distributions of the regions, leading to more flexible models that would be able to take into account the speckle quality of SAR. This reasoning is further supported by the qualitative segmentation maps produced by AE_Concat_3Moms's benchmark model, model AE_Concat_Skip (Architecture A.3.15). AE_Concat_Skip is identical to AE_Concat_3Moms except its alternative branch in parallel with the autoencoder is a regular skip connection, containing no computations. It is evident in Figure 4.11 especially that the AE_Concat_3Moms model's outputted segmentation map is a bit less homogeneous than the AE_Concat_3Moms model's segmentation, indicating that the moment computation route in AE_Concat_3Moms was successful in computing useful information about the regions' distributions. Overall, AE_Concat_3Moms has grounded statistical backings and seems to perform very well. The downside of this model, however, is that it took *1127.49s* per epoch to train, which is one of the slowest architectures. This is logical, as the additional computational aspects of the moment computation route would make training such a model take take longer than training a traditional autoencoder or an autoencoder with a regular skip connection parallel branch.

5.2.3 Architectures Employing Statistically-Motivated Computations in the Encoder of a U-Net

Additional architectures that performed well with respect to the metrics documented in Table 4.3 were the AE_Avg (Architecture A.3.4) and the Unet_1Mom (Architecture A.3.5) models. Because these models are very similar in implementation and performance, and as explained in Section 3.2, the Unet_1Mom model is a bit more statistically-sound, this analysis will focus on the Unet_1Mom model. Qualitatively, the Unet_1Mom performed well on the 'easy' validation example (Figure 4.7), but there are notable misclassifications in the center of the 'medium' example (Figure 4.8) and the 'hard' segmentation (Figure 4.8) was very inaccurate. For the real SAR test images, the model appeared to discern the various regions accurately and with more detail than any of the other models. This quality is especially evident in the Capella data example (Figure 4.12), where the segmentation map appears much more granular than the other maps. Because the U-Shape path of the architecture now only retains moment information, it likely no longer learns a global latent representation of the input itself at the bottleneck, instead now only learning a global representation of the data's statistics. Consequently, the model may rely more on the skip connections for the segmentation, providing granular segmentation maps that more closely resemble the input retrieved from the skip connections. One benefit of this model is that it trains extremely fast, at about *638.39s* per epoch, which is faster than the `Unet` and all of the other higher-performing models. Since this model has some sort of statistical foundation and trains even faster than the standard `Unet` (due to the fact that it only employs 1x1 convolutions), it could be the more favorable architecture to use in practice if more fine-grained segmentation maps are desired.

5.2.4 Architectures Employing Statistically-Motivated Computations in the Skip Connections of a U-Net

To avoid the problem in model `Unet_1Mom`, where the encoders might lose too much information about the input by only keeping track of the data's statistics, the next best-performing statistically-inspired models, `Unet_3Moms_Skip_1` (Architecture A.3.8) and `Unet_3Moms_Skip_2` (Architecture A.3.9), address this effectively by incorporating the moment computation into the skip connections instead of in the encoders. Since `Unet_3Moms_Skip_1` and `Unet_3Moms_Skip_2` performed about the same and their only difference is their mechanisms for reducing the moments representations to three dimensions, this analysis will focus on `Unet_3Moms_Skip_1`, which achieved slightly superior performance metrics by using averaging to reduce the moment tensors to three dimensions. Compared to the other models, model `Unet_3Moms_Skip_1` clearly had the most trouble discerning the regions in the synthetic validation tests, even on the 'easy' data (Figure 4.7). It was able to identify regions in the real SAR test examples, but its segmentation maps appeared to be even more fine-grained than the `Unet`'s. These results indicate that the moment computation skip connections did not provide the decoders with valuable statistical information to help produce segmentations that take into account the regions' probability distributions. This could be due to the fact that since the skip connections take in inputs that have already undergone transformations from the encoder, the architecture is attempting to compute moments on tensors that are too abstract to glean meaningful statistical information from. Since the `Unet` also outperforms the `Unet_3Moms_Skip_1` model with respect to final test cross-entropy, final test IOU, and training time, it seems that incorporating the three moment computation into all of the skip connections has no clear benefit.

5.2.5 Architectures Employing Larger Kernels in the Encoder

Besides the `AE_Concat_Skip model` and the standard `Autoencoder`, the next best performing architectures with respect to quantitative metrics were the `UNet_BigKern` and `Unet_3Moms_Skip_BigKern` models. Since both of these architectures employ encoder blocks with increased kernel sizes, they are able to learn more feature representations and therefore output a more accurate segmentation map. It is evident in both the validation synthetic data and real SAR test qualitative examples that these models were able to produce segmentations with significant detail, more like the standard `Unet` than the `Autoencoder`. The major downside of employing a larger kernel in these architectures is that they took the longest to train out of any of the architectures.

5.2.6 Poorly Performing Models

None of the other architectures performed well:

- Unet_3Moms_1 and Unet_3Moms_1: These models likely underperformed because, similar to architectures AE_Avg and Unet_1Mom, computing three moments in a series of encoder blocks becomes too abstract to actually represent the statistics of the input, and instead only loses information about the original image.
- Unet_3Moms_3Chans_1 and Unet_3Moms_3Chans_2: These models likely face the same problem as Unet_3Moms_1 and Unet_3Moms_1, which is compounded by the fact that limiting the channel dimensions at each encoder and decoder block to 3 leads to even more information loss.
- 3Moms_Concat_SmallShrink and 3Moms_Concat_BigShrink: Though logical in their attempt to mimic the U-Net's ability to glean global information about the image by compressing it, these models likely underperformed because simply shrinking and resizing the image in one convolutional operation does not adequately simulate all of the learning that encoder and decoder blocks do throughout the many layers of the U-Net, even when supplemented by a statistical moment computation of the original image that could theoretically provide valuable statistical information.
- 3Moms: This architecture never condenses the image so there is no way for the model to relate pixels that are far away from each other.
- 3Moms_B4_Unet: Since so much spatial information is lost during the moment computation process, it is no surprise that this model was not able to reconstruct the outputs of the moments into accurate segmentation maps with a U-Net downstream.

5.2.7 Architecture Performance When Trained and Tested on Optical SAR Imagery

The architectures were then trained and tested on the SARBuD dataset. Based on the metrics shown in Table 4.4, the models were able to learn the synthetic data with more ease than the SARBuD data. This is unsurprising, as the synthetic data generated from the Perlin Noise while effective, is still quite not as diverse as all of the available type of terrains found on Earth.

Additionally, while the autoencoder-based architectures performed the best when trained on the synthetic data, the U-Net-based architectures performed better when trained on SARBuD. Specifically, the Unet_3Moms_Skip_2, Unet_3Moms_Skip_BigKern, Unet, and Unet_3Moms_Skip_1 models performed the best (and very similar to each other) with respect to final test IOU and test cross-entropy loss. This is mirrored in the validation tests depicted in Figure 4.13, where the Unet_3Moms_Skip_1 and Unet_3Moms_Skip_BigKern models in particular were able to discern the general regions quite accurately.

Interestingly, the architectures trained on the SARBuD dataset performed much worse on the real SAR test data than the models trained on synthetic data. For instance, both the Unet and the Autoencoder were unable to discern the regions in both test examples depicting San Francisco and an oil spill (Figures 4.14 and 4.15). The fact that the Unet and Autoencoder were able to discern the regions of this SAR data when trained on the synthetic dataset serves as a testament to the synthetic data's quality. As explained in Section 4.1.1, the SARBuD dataset is stored as JPEG files and consequently likely lost the original statistical qualities of the source SAR data. Therefore, even though the models were relatively successful in learning to segment the SAR JPEG images, they clearly regarded them as standard visual data, not learning any statistical patterns that they could apply to the SAR data stored as TIFF files. Conversely, the effectiveness of the models trained on the synthetic data in segmenting the real SAR data stored in TIFF files demonstrates that the statistical nature of the synthetic dataset's generation was of crucial importance.

Therefore, since the previous results confirm that the SARBuD dataset does not follow the traditional G^0 distribution that characterizes SAR, the relatively successful validation results in Table

4.4 and Figure 4.13 demonstrate that these statistical architectures can be effective in scenarios where the input data does not necessarily exhibit defined statistical characteristics.

Moreover, even though the standard `Unet` was unable to segment the San Francisco and oil spill SAR test samples, the `Unet_1Mom`, `Unet_3Moms_Skip_1`, and `Unet_3Moms_Skip_BigKern` models were a bit more successful in segmenting the real SAR data. As displayed in Figure 4.14, all three models clearly performed better than the other architectures on the San Francisco data, though still not as well as the models trained on synthetic data. It is particularly notable that the `Unet_1Mom` model, which exhibited the worst quantitative performance out of all of the architectures on the validation SARBuD data, was still able to perform segmentation on real SAR data relatively successfully. These results indicates that incorporating the statistical moment computations within the `Unet` architecture may have indeed improved its ability to recognize and process the statistical properties of the San Francisco and oil spill SAR data, even if it was not trained to do so.

5.2.8 Synthetically-Trained Model Performance on Optical SAR Imagery

Finally, all of the models trained on the synthetic dataset were tested on the SARBuD dataset. As shown in Table 4.5, all of the models performed very badly. There could be many reasons for this result, but it is likely because of the compression undergone by all of the SARBuD data. Since the models learn to identify regions based on the statistically-generated synthetic SAR data, the optical SARBuD data is different than anything it has ever seen before. Again, this poor performance attests to the synthetic dataset's quality and the ability of the models to learn statistical patterns, rather than treating the SAR data as regular images. It also emphasizes why there is a great need for more datasets with high-quality SAR data stored as TIFFs, not JPEGs.

5.3 Disparity-Based Loss

5.3.1 Supervised Approach

The supervised loss function experiments involved combining the loss derived from the Euclidean distance between the average pixel value of the regions and the typical binary cross-entropy loss. As depicted in Table 4.6, the U-Net achieved the highest IOU of *0.865* when it solely used cross-entropy loss and the lowest IOU of *0.212* when the Euclidean distance loss coefficient (k) was the highest.

The same trend is apparent in the validation example depicted in Figure 4.16, which has previously been employed as an example of data that is of 'medium' difficulty to segment. The predicted segmentation map seems to remain the same quality as k increases from 0 to 1. At $k = 1$, the model produces a slightly worse segmentation map. When k is 10, the model predicts the same region for all pixels. When k is 100, the U-Net is able to produce a segmentation map with two regions represented, but it is an incredibly abstract and inaccurate depiction of the regions. The qualitative test results using real SAR data displayed the same trend, such as the example segmentation of the San Francisco image pictured in Figure 4.17. Between $k = 0$ to $k = 1$, the quality varies a bit, but the U-Net is still able to discern the regions in the image with a lot of detail. The model's performance appears considerably worse at $k = 10$. Then, at $k = 100$, the U-Net seems to discern the land as one giant blob, which is likely too general to be useful. Moreover, the test segmentation results of the Capella data, as displayed in 4.18, depict the same pattern that was found in the validation and San Francisco experiments.

These results indicate that adding the two types of losses in this fashion is not an effective approach. The fact that the model performed the best when it solely employed cross-entropy loss leads to the conclusion that either the disparity-based loss is ineffective, or that it must be combined with cross-entropy in a more sophisticated way to be successful. The result in Figure

4.17 showing an approximate, if blurry, segmentation for San Francisco when the disparity-based loss is considered in much higher proportion than the cross-entropy suggests that perhaps the former might work as a loss function if used in isolation.

5.3.2 Unsupervised Approach

In fact, the disparity-based loss did appear to be more successful when used on its own, rather than in conjunction with the cross-entropy. Quantitatively, the U-Net using the opposite of the Euclidean distance as the sole loss function achieved a final test IOU of *0.71*. This quantitative result is only so-so, but it does indicate that the unsupervised disparity-based loss may guide the model towards learning some important region information. This IOU metric was calculated slightly differently than in the supervised experiments. As evident in all of the examples in Figure 4.19 and Figure 4.20, the segmentation shapes are often fairly accurate, but display regions with swapped classifications. This is because the disparity-based loss is only trying to maximize the distance between the average values of the regions, without regard to which region is which. Therefore, $max(IOU, 1 - IOU)$ was used to calculate the IOU in order to ensure that the metric solely reflected the model's capacity to discern different regions, without considering the specific class assignments. Since the disparity-based loss is independent of the region assignments, the predicted region assignments were expected to be completely random. However, after running the code multiple time, the models always produced segmentations with the regions swapped in comparison to the associated ground truths. Since the disparity-based loss is not exposed to the ground truth, it is not obvious why this is. Future work could involve investigating this phenomena.

The qualitative results in Figures 4.19 and 4.20 provide further evidence that the unsupervised disparity-based loss was somewhat successful in guiding the model towards discerning the various regions. It was even able to somewhat accurately segment optical SAR data from the SARBuD dataset, as depicted in Figure 4.20d, where the data does not exhibit SAR's typical statistical characteristics. However, all of the test segmentation maps displayed in Figure 4.20 are very granular. Ideally, incorporating actual stochastic distances into the loss, instead of just the Euclidean distance between the average pixel values, would guide the models toward predicting more homogeneous regions from the SAR data.

Chapter 6

Conclusion

6.1 Overview

This project sought to advance SAR image segmentation techniques by exploring statistically-grounded synthetic data generation, deep learning architectures, and loss functions.

1. **Synthetic Data**: The synthetic data experiments exhibited promising outcomes. Models trained on the synthetic dataset demonstrated the capability to effectively capture statistical patterns present in real SAR data. Effective synthetic SAR datasets are extremely valuable, since real, high-quality datasets are scarce.

2. **Architectures**: Several statistically-principled architectures emerged as viable alternatives to the benchmark architectures, offering comparable or superior performance while often achieving faster efficiency. Notably, architectures like the `AE_Concat_3Moms` model demonstrated promise in scenarios where homogenous regions are desired in the segmentation maps, such as for identifying oil accumulations in ocean imagery. Models like `Unet_1Mom` and `Unet_3Moms_Skip_1` proved effective and quicker alternatives to the `Unet` in applications necessitating finer-grained segmentation maps. Such architectures could be useful, for instance, in urban domains, where segmentation maps that delineate individual houses could be beneficial. In general, the statistical moment computations were particularly effective when they were not abstracted out but rather applied directly to the input image. Even beyond the motivation of enhanced model performance, the value of having end-to-end segmentation algorithms capable of incorporating statistically-inspired computations all within a deep learning architecture is apparent. This departure from previous techniques, which often relied solely on deep learning without statistical considerations, or were entirely statistical but lacked flexibility and speed, indicates a significant potential for advancement.

3. **Loss Function**: While the results for the loss function experiments require further exploration, the preliminary findings suggest that incorporating disparity-based losses in the SAR context is promising, particularly in unsupervised scenarios where labeled data is limited.

Overall, this project presents encouraging evidence supporting the development of statistically-informed deep learning algorithms for SAR image segmentation. While the efficacy of these techniques is a promising start, there remains room for further exploration and refinement. It will be exciting to witness whether statistical domain knowledge can be effectively integrated into deep learning systems in the future. SAR analysis, with its statistical foundations and capacity for deep learning processing, presents a compelling opportunity to explore the critical question of explainability within deep learning algorithms. Innovation in this field is crucial not only for advancing knowledge in computer science and computer vision, but also for addressing the numerous impactful applications for which SAR is used.

6.2 Future Recommendations

Looking ahead, several avenues of experimentation warrant exploration:

- Synthetic Dataset
  - Diversify dataset by employing varying persistance parameters in the Perlin Noise.
  - Explore which parameters lead to datasets that are better suited for specific applications.
- Architectures
  - Develop explainability/interpretability techniques and experiments to evaluate whether the architectures are really learning statistical information.
  - Seek out and test architectures on additional large, labeled real dataset of TIFF SAR images for evaluation, acknowledging the challenges associated with this endeavor.
- Loss Functions
  - Implement stochastic distance losses that compare distances between probability distributions, rather than just simple distances between average pixel values of regions.
  - Explore one-shot unsupervised learning to determine if model weights can be learned from a single image.
  - Investigate why the unsupervised Euclidean distance loss always predicted segmentation maps with the regions swapped in comparison to the labels.
  - Develop a more sophisticated approach to combining cross-entropy and disparity-based losses.

Appendix A

Architectures Tested

A.1 Helper Blocks

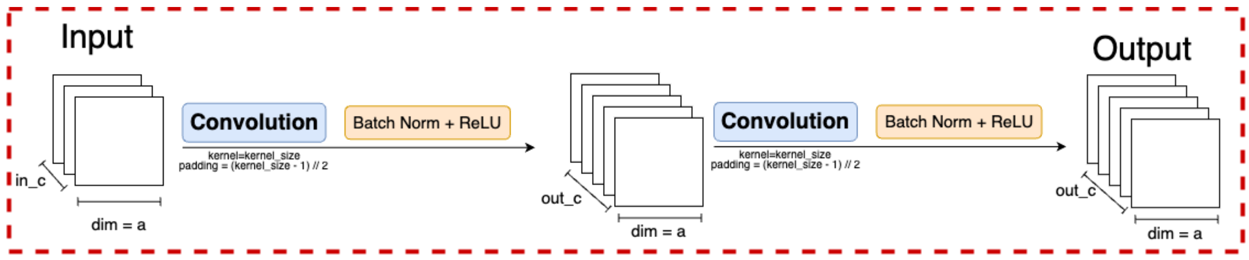


FIGURE A.1.1: Conv_Block

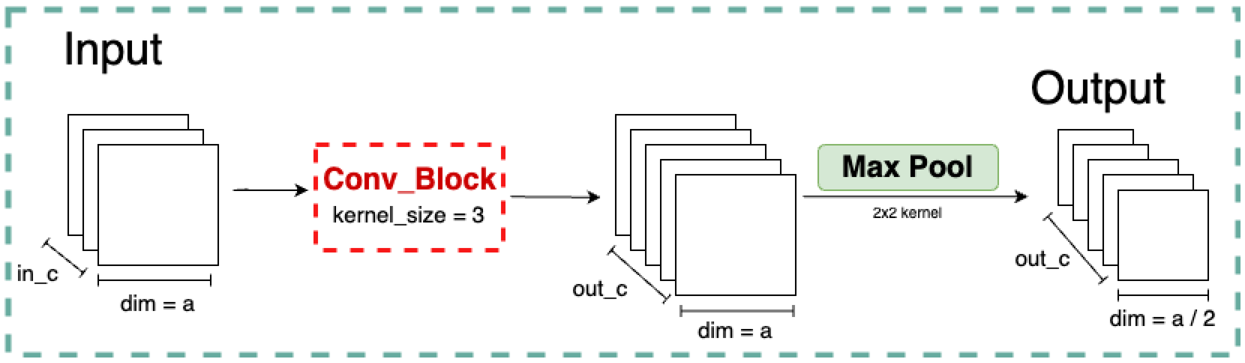


FIGURE A.1.2: Encoder_Standard

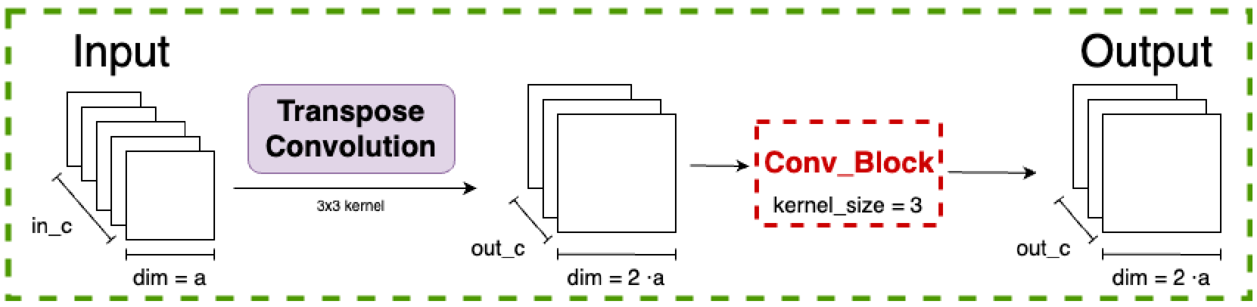


FIGURE A.1.3: Decoder_Block

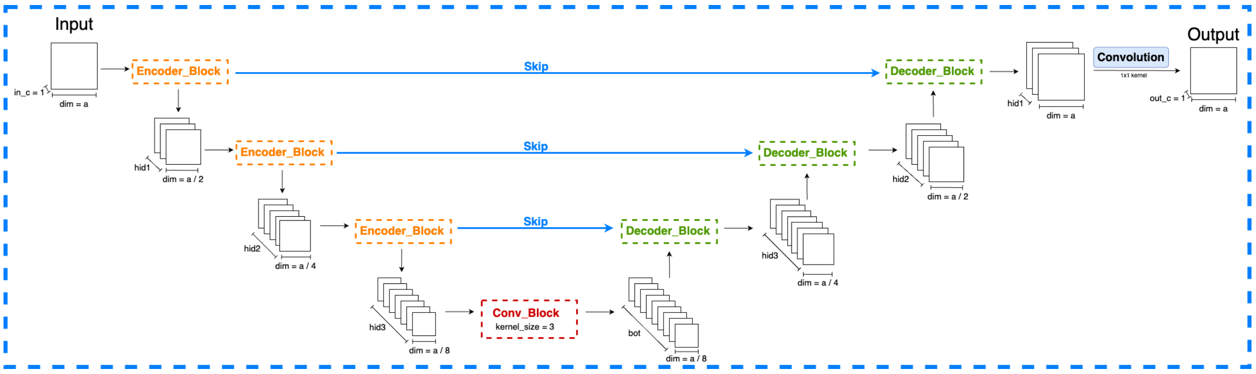


FIGURE A.1.4: UShape

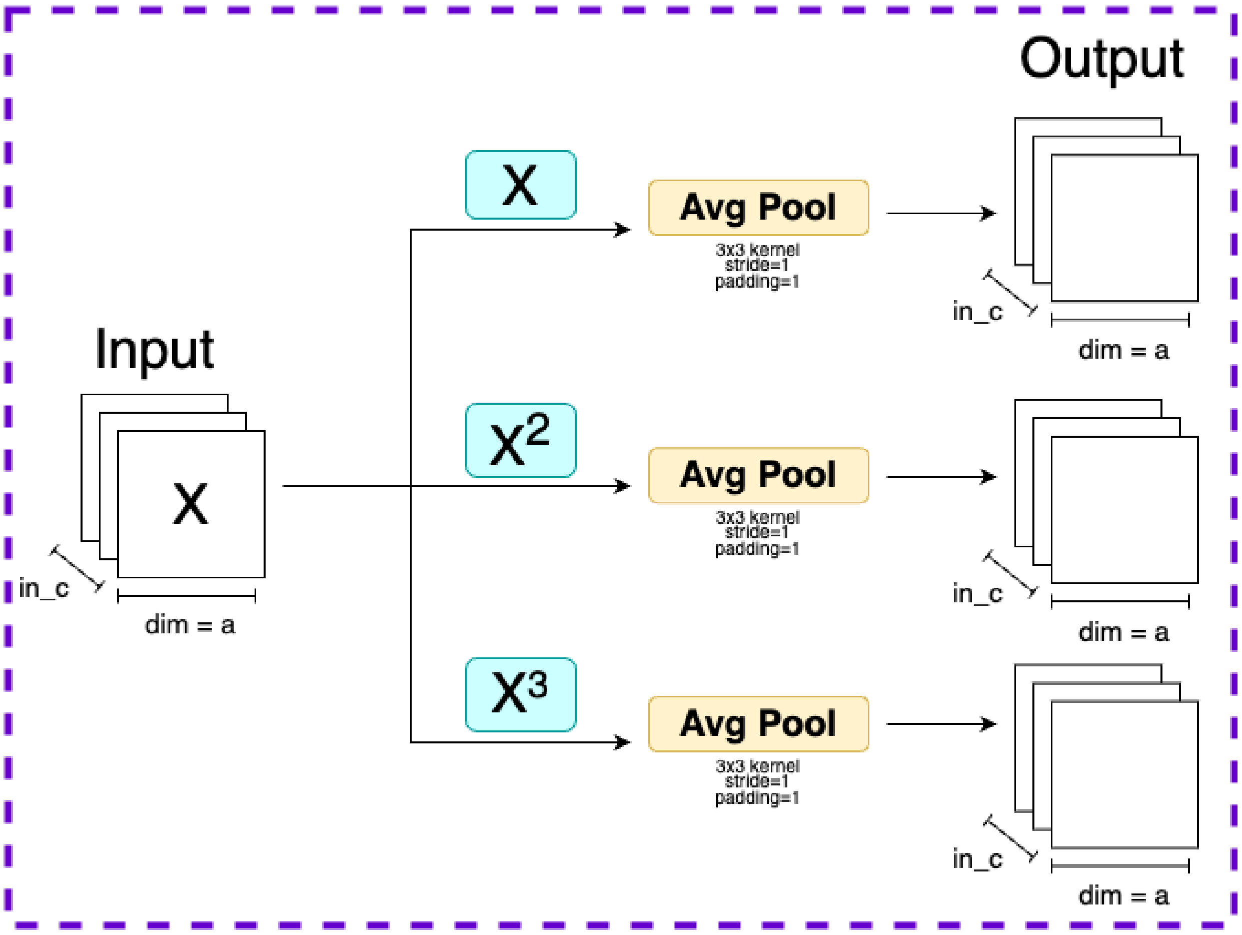


FIGURE A.1.5: Three_Moments

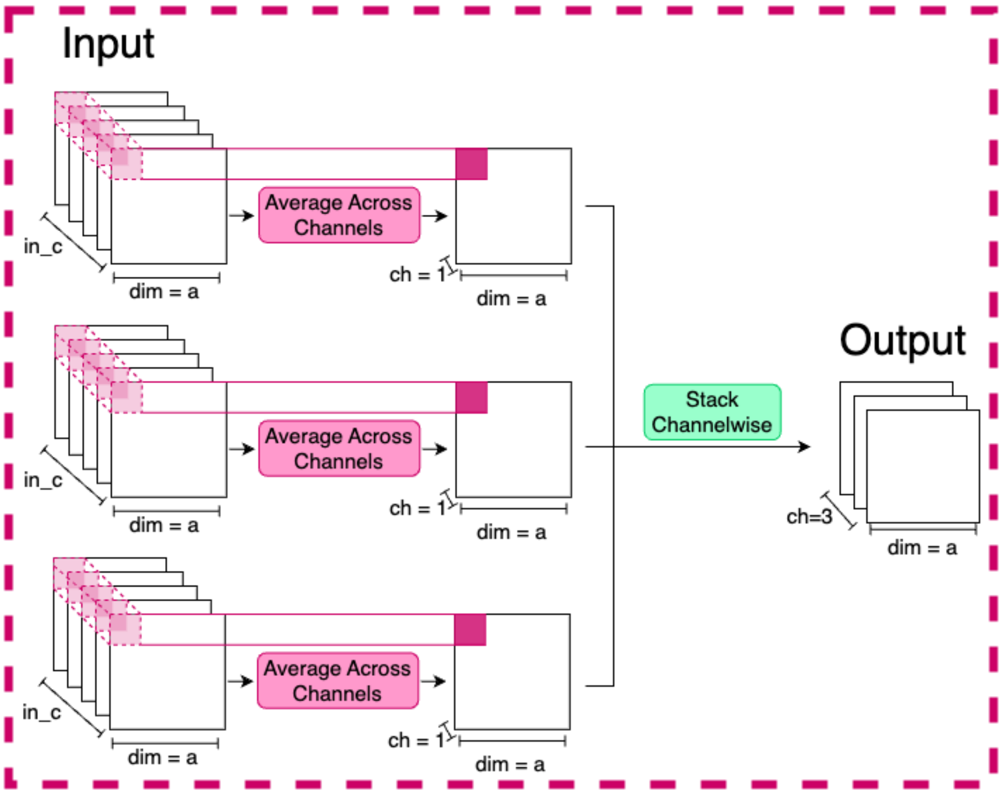


FIGURE A.1.6: Average_Across_Channels

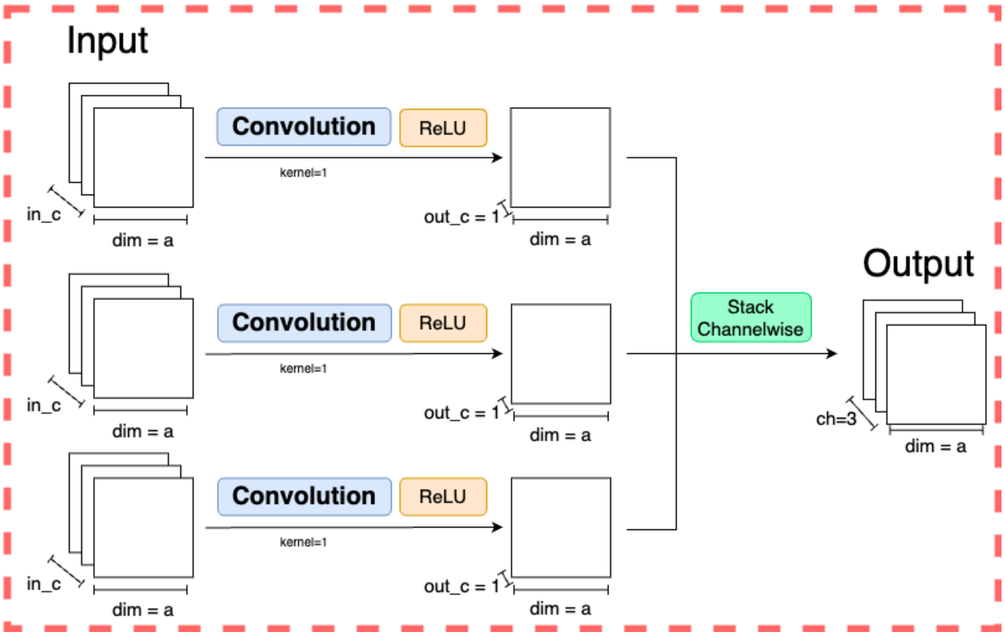


FIGURE A.1.7: Convolve_to_1Chan

A.2 Experimental Encoder Blocks

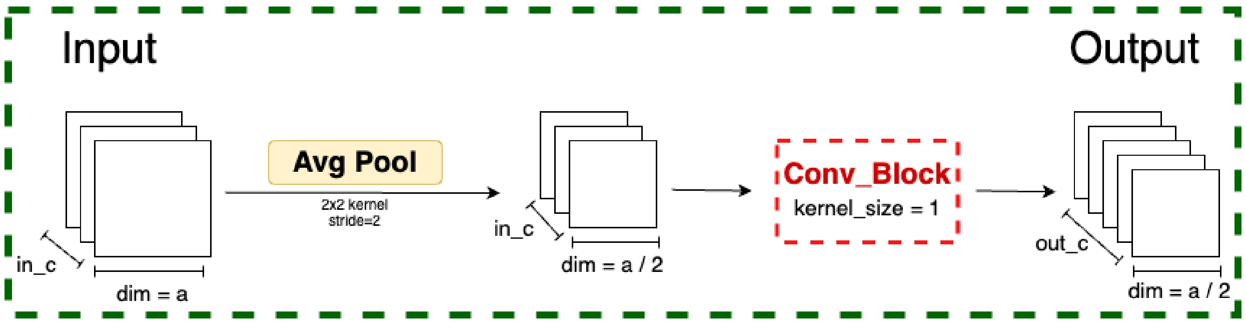


FIGURE A.2.1: Encoder_Avg

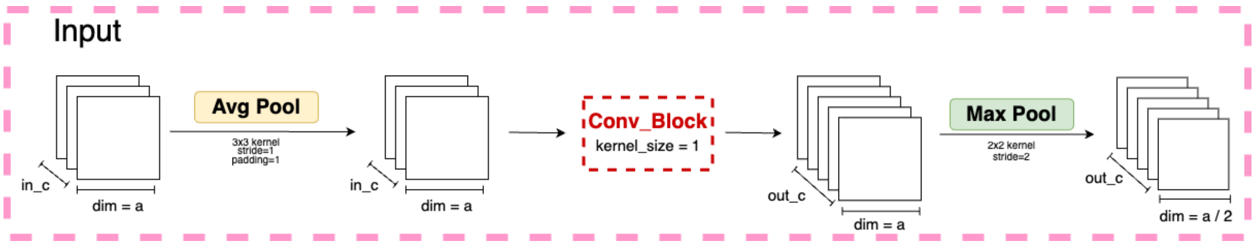


FIGURE A.2.2: Encoder_1Mom

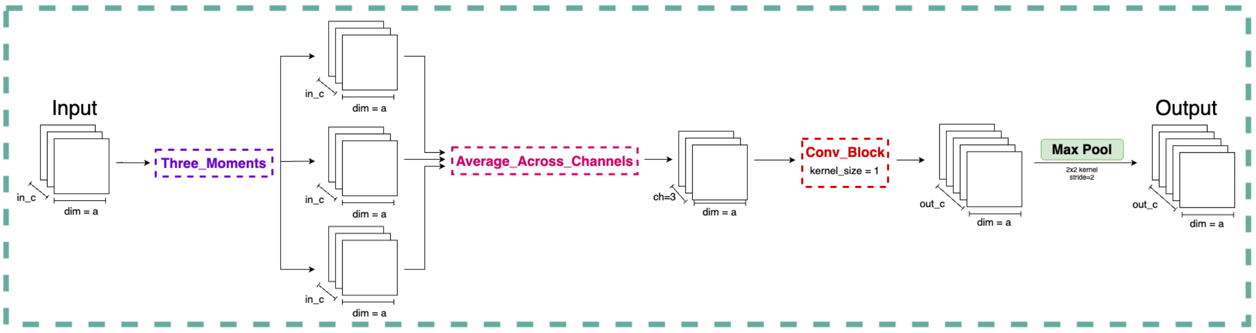


FIGURE A.2.3: Encoder_3Moms_1

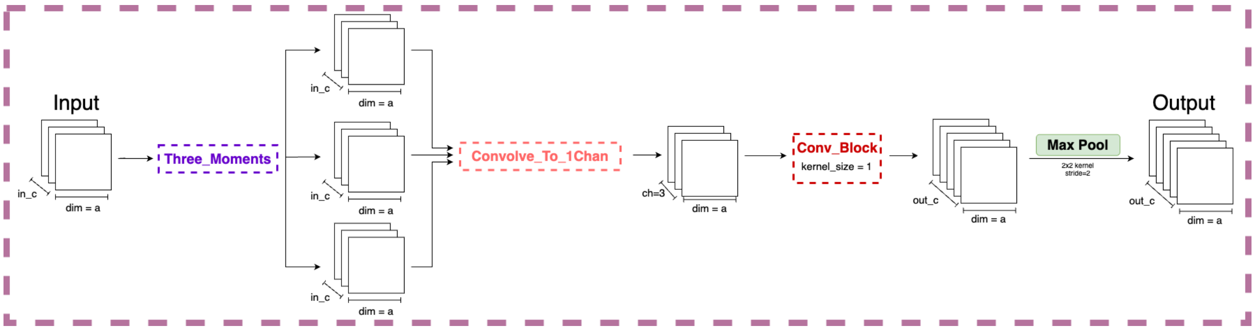


FIGURE A.2.4: Encoder_3Moms_2

A.3 High-Level Architectures

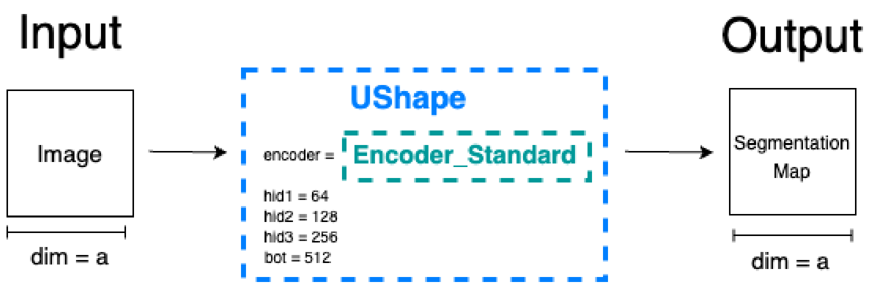


FIGURE A.3.1: Unet

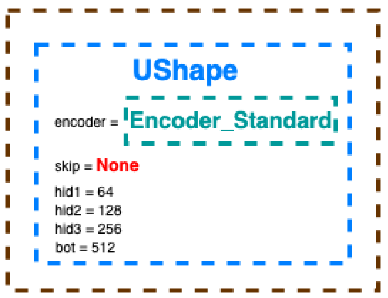


FIGURE A.3.2: Autoencoder

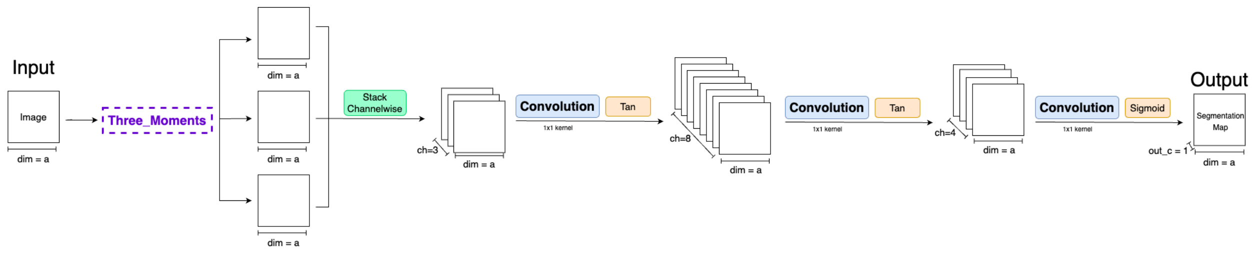


FIGURE A.3.3: 3Moms

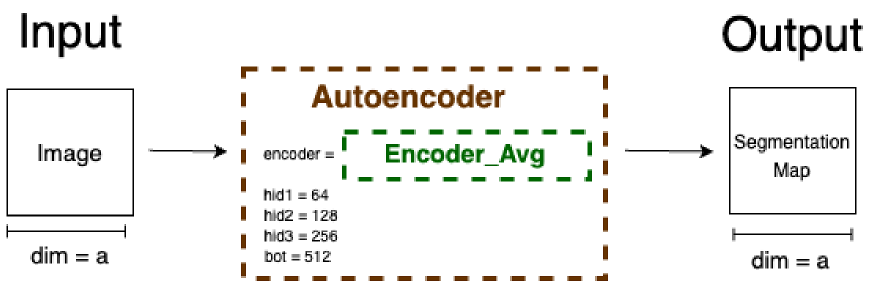


FIGURE A.3.4: AE_avg

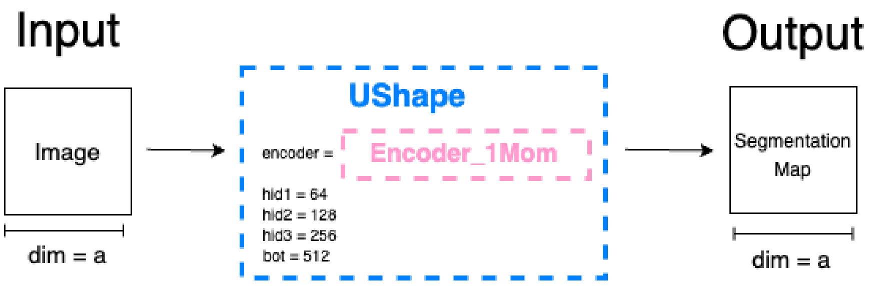


FIGURE A.3.5: Unet_1Mom

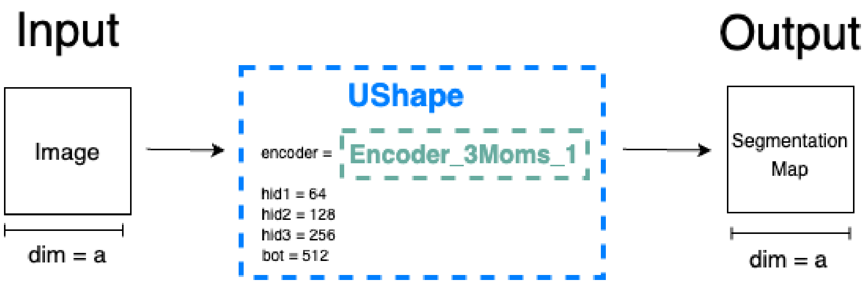


FIGURE A.3.6: Unet_3Moms_1

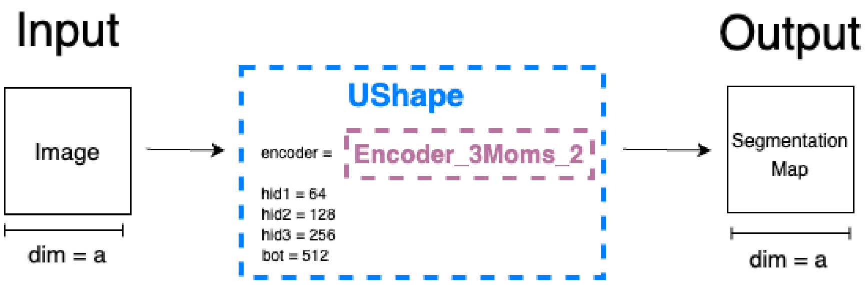


FIGURE A.3.7: Unet_3Moms_2

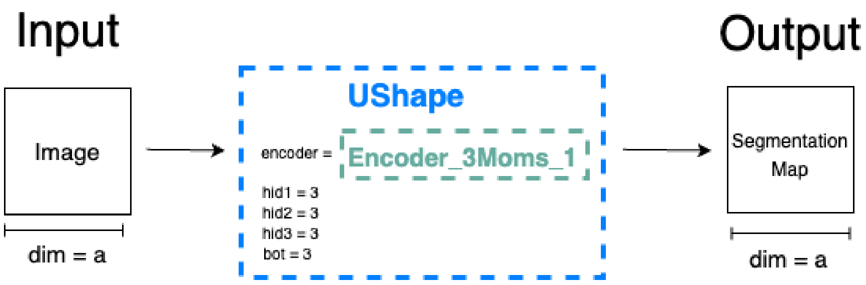


FIGURE A.3.8: Unet_3Moms_3Chans_1

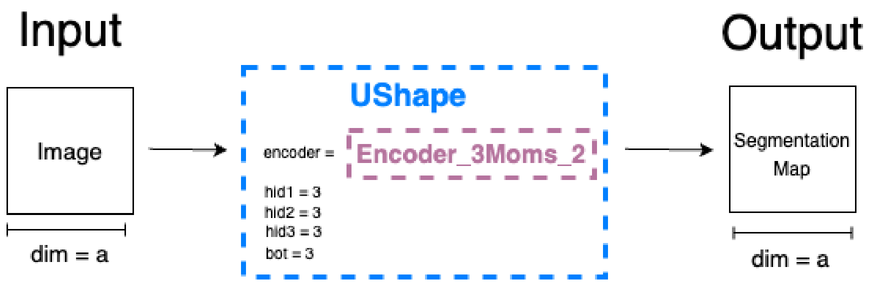


FIGURE A.3.9: Unet_3Moms_3Chans_2

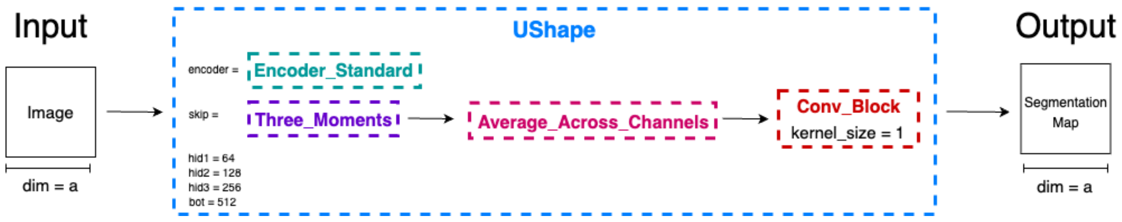


FIGURE A.3.10: Unet_3Moms_Skip_1

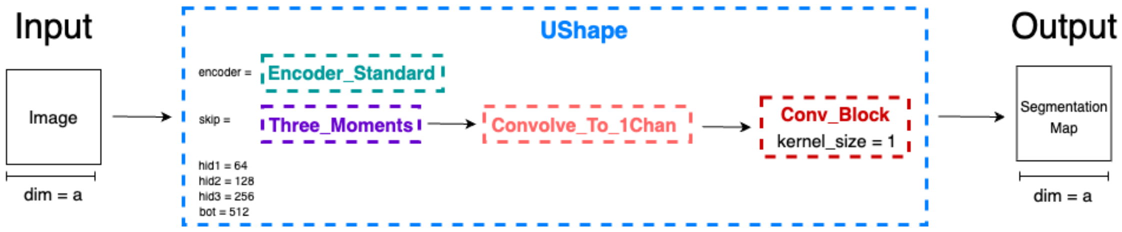


FIGURE A.3.11: Unet_3Moms_Skip_2

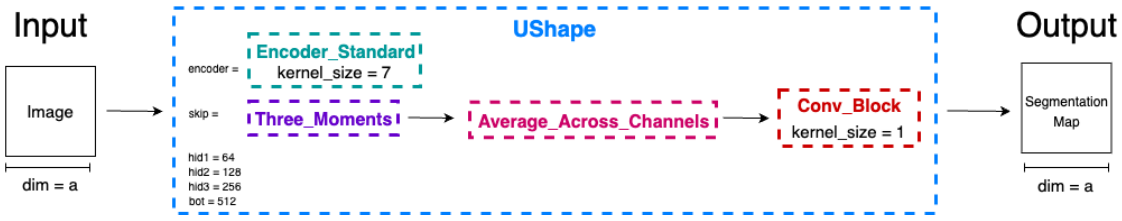


FIGURE A.3.12: Unet_3Moms_Skip_BigKern

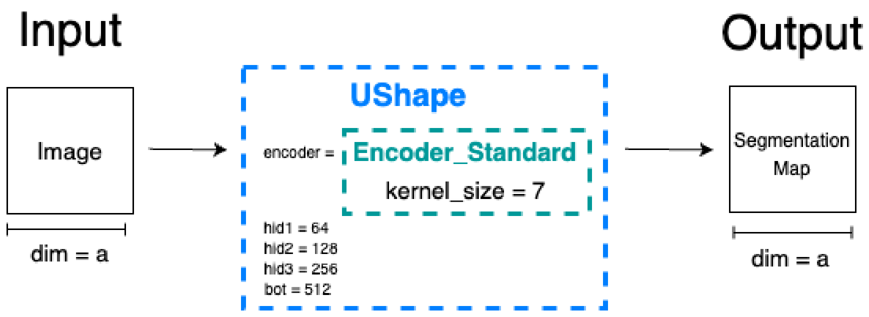


FIGURE A.3.13: Unet_BigKern

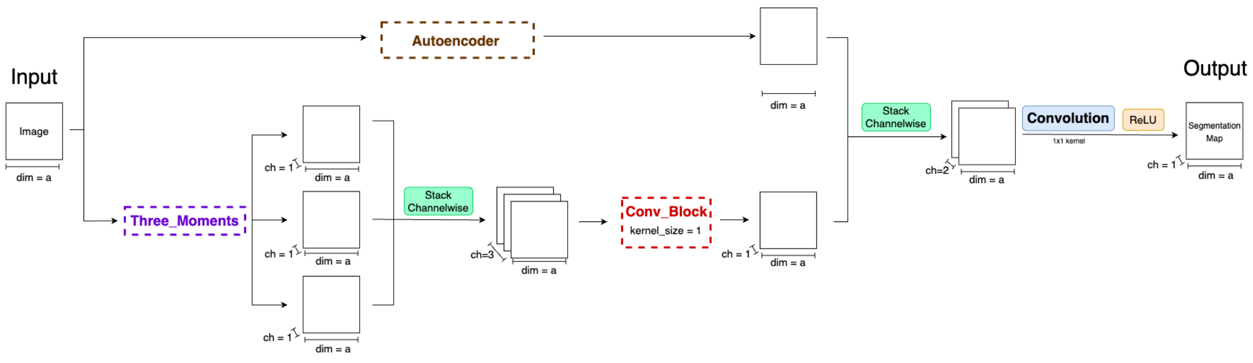


FIGURE A.3.14: AE_Concat_3Moms

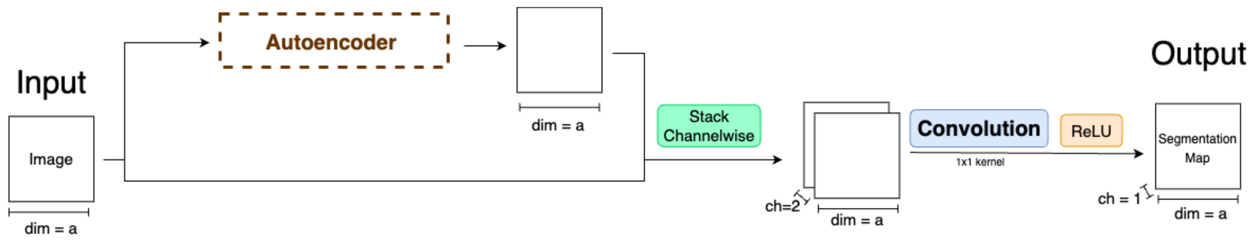


FIGURE A.3.15: AE_Concat_Skip

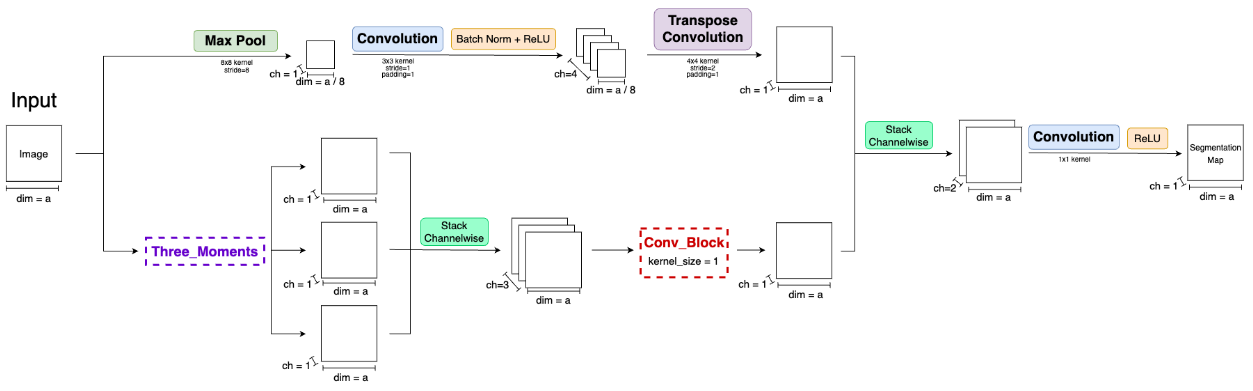


FIGURE A.3.16: 3Moms_Concat_BigShrink

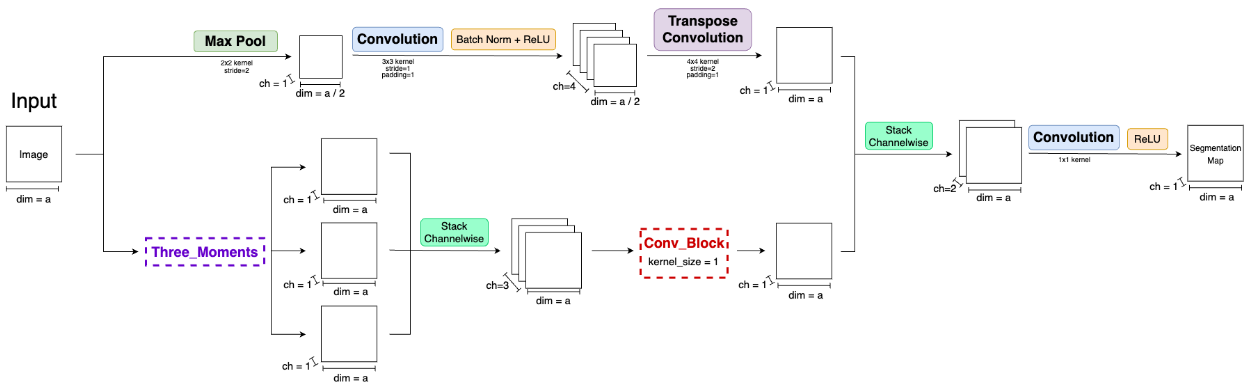


FIGURE A.3.17: 3Moms_Concat_SmallShrink

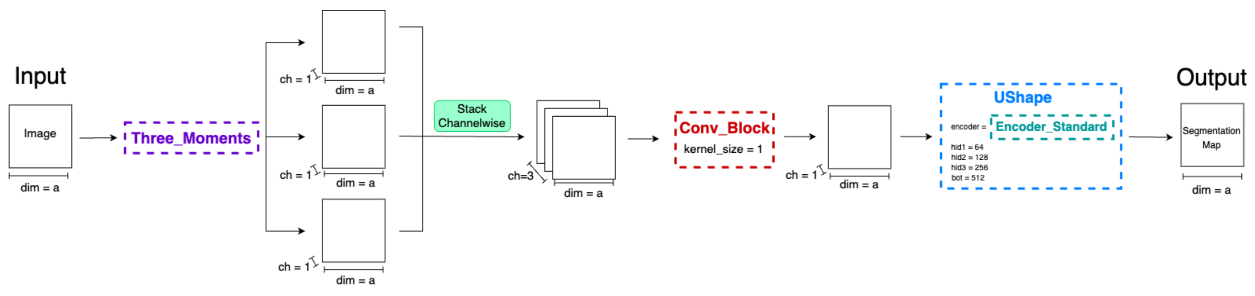


FIGURE A.3.18: 3Moms_B4_Unet

Bibliography

Capella (2021). *Capella Space Synthetic Aperture Radar (SAR) Open Dataset*. Accessed on May 2, 2024. URL: https://registry.opendata.aws/capella_opendat.

Cheng, Jianghua et al. (Jan. 2013). "An Improved Scheme for Parameter Estimation of G° Distribution Model in High-Resolution SAR Images". In: *Progress In Electromagnetics Research* 134, pp. 23–46. DOI: 10.2528/PIER12082308.

Dumoulin, Vincent and Francesco Visin (2018). *A guide to convolution arithmetic for deep learning*. arXiv: 1603.07285 [stat.ML].

Fan, Li and Jeova Farias Sales Rocha Neto (2023). *Using Neural Networks for Fast SAR Roughness Estimation of High Resolution Images*. arXiv: 2309.03351 [cs.CV].

Frery, Alejandro, Jie Wu, and Luis Deniz (Aug. 2022). *SAR Image Analysis – A Computational Statistics Approach: With R Code, Data, and Applications*. ISBN: 9781119795292. DOI: 10.1002/9781119795520.

Frery, Alejandro et al. (June 1997). "A model for extremely heterogeneous clutter". In: *Geoscience and Remote Sensing, IEEE Transactions on* 35, pp. 648 –659. DOI: 10.1109/36.581981.

Gambini, Juliana et al. (Jan. 2015). "Parameter Estimation in SAR Imagery Using Stochastic Distances and Asymmetric Kernels". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 8.1, 365–375. ISSN: 2151-1535. DOI: 10.1109/jstars.2014.2346017. URL: http://dx.doi.org/10.1109/JSTARS.2014.2346017.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Hartmann, Andreas et al. (2021). "Bayesian U-net for segmenting glaciers in SAR imagery". In: *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*. IEEE, pp. 3479–3482.

Kingma, Diederik P. and Jimmy Ba (2017). *Adam: A Method for Stochastic Optimization*. arXiv: 1412.6980 [cs.LG].

Kumar, Vedant (2020). *Towards Data Science*. Accessed on May 2, 2024. URL: https://towardsdatascience.com/convolutional-neural-networks-f62dd896a856.

Larsen, Richard J and Morris L Marx (2005). *An introduction to mathematical statistics*. Prentice Hall Hoboken, NJ.

Liu, Yan et al. (2018). "Efficient Patch-Wise Semantic Segmentation for Large-Scale Remote Sensing Images". In: *Sensors (Basel, Switzerland)* 18. URL: https://api.semanticscholar.org/CorpusID:52878014.

Maity, Alenrex et al. (2015). "A Comparative Study on Approaches to Speckle Noise Reduction in Images". In: *2015 International Conference on Computational Intelligence and Networks*, pp. 148–155. URL: https://api.semanticscholar.org/CorpusID:2603031.

Marques, R.C.P., Fatima Medeiros, and Juvêncio Nobre (Oct. 2012). "SAR Image Segmentation Based on Level Set Approach and G_A^0 *Model*". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, pp. 046–2057. DOI: 10.1109/TPAMI.2011.274.

Mazza, Antonio et al. (2019). "TanDEM-X forest mapping using convolutional neural networks". In: *Remote Sensing* 11.24, p. 2980.

Nascimento, Abraao D. C., Alejandro C. Frery, and Renato J. Cintra (Mar. 2019). "Detecting Changes in Fully Polarimetric SAR Imagery With Statistical Information Theory". In: *IEEE Transactions on Geoscience and Remote Sensing* 57.3, 1380–1392. ISSN: 1558-0644. DOI: 10.1109/tgrs.2018.2866367. URL: http://dx.doi.org/10.1109/TGRS.2018.2866367.

Nascimento, A.D.C., R.J. Cintra, and A.C. Frery (Jan. 2010). “Hypothesis Testing in Speckled Data With Stochastic Distances”. In: *IEEE Transactions on Geoscience and Remote Sensing* 48.1, 373–385. ISSN: 1558-0644. DOI: 10.1109/tgrs.2009.2025498. URL: http://dx.doi.org/10.1109/TGRS.2009.2025498.

Nava, Lorenzo et al. (2022). “Rapid mapping of landslides on SAR data by attention U-Net”. In: *Remote Sensing* 14.6, p. 1449.

Neto, Jeová Farias Sales et al. (Sept. 2019). “Level-Set Formulation Based on an Infinite Series of Sample Moments for SAR Image Segmentation”. In: *IEEE Geoscience and Remote Sensing Letters* PP, pp. 1–4. DOI: 10.1109/LGRS.2019.2933149.

Niemietz, Ricardo Cancho (2008). *Wikipedia*. Accessed on May 2, 2024. URL: https://en.wikipedia.org/wiki/File:RGB_channels_separation.png.

Nobre, Ricardo H. et al. (2016). “SAR Image Segmentation With Rényi’s Entropy”. In: *IEEE Signal Processing Letters* 23.11, pp. 1551–1555. DOI: 10.1109/LSP.2016.2606760.

Paszke, Adam et al. (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *CoRR* abs/1912.01703. arXiv: 1912.01703. URL: http://arxiv.org/abs/1912.01703.

Perlin, Ken (1985). “An image synthesizer”. In: *ACM Siggraph Computer Graphics* 19.3, pp. 287–296.

Ren, Yibin et al. (2021). “Development of a dual-attention U-Net model for sea ice and open water classification on SAR images”. In: *IEEE Geoscience and Remote Sensing Letters* 19, pp. 1–5.

Rodrigues, Francisco et al. (Jan. 2016). “SAR Image Segmentation Using the Roughness Information”. In: *IEEE Geoscience and Remote Sensing Letters* 13, pp. 1–5. DOI: 10.1109/LGRS.2015.2496340.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: arXiv: 1505.04597 [cs.CV].

Wu, F. et al. (2021). “Built-up area mapping in China from GF-3 SAR imagery based on the framework of deep learning”. In: *Remote Sensing of Environment* 262, p. 112515.

Wu, F. et al. (2022). “SARBuD1.0: A SAR Building Dataset Based on GF-3 FSII Imageries for Built-up Area Extraction with Deep Learning Method”. In: *National Remote Sensing Bulletin* 26.4, pp. 620–631.

Zhu, Xiao Xiang et al. (2021). “Deep learning meets SAR: Concepts, models, pitfalls, and perspectives”. In: *IEEE Geoscience and Remote Sensing Magazine* 9.4, pp. 143–172.