The Pennsylvania State University

The Graduate School

QUANTIZATION AND ADAPTIVITY OF WAVELET SCATTERING NETWORKS FOR CLASSIFICATION PURPOSES

A Thesis in

Electrical Engineering

by

Maxine R. Fox

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2020

The thesis of Maxine R. Fox was reviewed and approved by the following:

Ram M. Narayanan
Professor
Thesis Advisor

Timothy J. Kane
Professor

Raghu G. Raj
Head, Radar Imaging and Target ID Section and Senior Research Scientist

Kultegin Aydin
Professor of Electrical Engineering and Department Head

ABSTRACT

GPR has been proposed as a solution for mine clearance and the removal of unexploded ordnance. To detect targets, convolutional neural networks (CNNs) may be used. However, the training of CNNs for this purpose is hampered by the lack of extensive GPR data on buried targets; therefore, this thesis investigates the use and application of CNNs on the freely available MSTAR dataset for target classification.

Although CNNs improve upon the required memory of preceding neural networks, large CNNs still need large amounts of memory. Furthermore, despite the promising results of quantization methods, extensions to other datasets and architectures are not easily proven. To explore the effect of quantization, this thesis implements several quantization schemes on the wavelet scattering network (WSN), due to its shared properties with CNNs. Results indicate that the WSN is robust to few quantization levels; future work should focus on the application of these quantization schemes to CNNs, as well as the exploration of PDF-based quantization schemes.

In addition, an adaptive WSN (AWSN) is constructed using the backpropagation algorithm inherent in CNNs. Using particle swarm optimization (PSO), the parameters of the mother and father wavelets are adjusted. Multiple AWSN constructions are compared to static WSN networks and a fully-connected layer network. Though the validation accuracy never exceeded 0.75, the results were within 0.05 for all AWSN implementations using PSO. Future work should explore modifications to parameter updates, as well as the implementation of existing adaptive wavelet methods.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Ram M. Narayan, for providing this opportunity and his mentorship. I would also like to thank the remainder of my thesis committee, Professor Timothy J. Kane and Dr. Raghu Raj, for taking the time to evaluate and provide feedback on my work, as well as Zach Idris for his collaboration. In addition, I would like to thank my family and friends for their support and encouragement.

Finally, I would like to thank Dr. Raghu Raj of the Naval Research Laboratory (NRL) for his guidance and mentorship during the research. This project was supported by Dr. Joong Kim of the Office of Naval Research (ONR) under ONR Grant # N00014-16-1-2354.

The findings and conclusions do not necessarily reflect the views of the funding agency.

Chapter 1

Introduction

Buried explosive hazards, including anti-tank (AT), anti-personnel (AP) landmines, and unexploded ordnance (UXO), are dangerous and persistent threats whose presence has damaging impacts on individuals and communities. Ground penetrating radar (GPR), which has proven to be suitable for subsoil investigations, is a technique that has been exploited for mine and UXO detection. GPR has been proposed as a solution for mine clearance and the removal of unexploded ordnance because of its speed, safety and suitability as a non-invasive technique compared to commonly used but more dangerous invasive methods, such as excavations, or traditional detection methods, such as metal detectors.

Recently, deep convolutional neural networks (CNNs) have achieved state-of-the-art performance on several benchmark tasks in the computer vision literature. This suggests the possibility of applying CNNs to GPR data for detecting and classifying buried threats. CNNs are a supervised classification model, like those used previously for GPR data, but they have many more free model parameters. As a result, one well known limitation with CNNs, which was mentioned in existing CNN publications, is their need for large quantities of training data. This is a major challenge for GPR buried target detection, wherein the collection of training data is time consuming and expensive.

Hampered by the lack of extensive GPR data on buried targets, an investigation was carried out to explore the use and application of CNNs on the freely available MSTAR dataset for target classification. The results of this study can easily be extended to GPR data, when such data become available.

CNNs have found application to synthetic aperture radar (SAR) in recent works, with heavy focus on the MSTAR dataset [1]–[6]. These studies typically used predefined network architectures, that can contain many learnable parameters; other available architectures can contain upwards of a million parameters, which require significant amounts of memory. To reduce needed memory, the quantization of generalized or otherwise large CNNs has been studied for the classification of optical datasets [7], [8]. Despite promising results, extensions to other datasets and network architectures are not proven easily due to their variability. The success of a CNN design is dependent upon many factors, including the arrangement of layers, the number and initialization of learnable parameters in each layer, the learning rate and update methods implemented during training, as well as the size and complexity of the available training data. Given such variability, the development of a network for benchmarking these designs would prove useful.

Mallat and others [9]–[11] implemented a wavelet scattering network (WSN) to demonstrate the paramount properties of CNNs, particularly convolution, nonlinearity, and the layered architecture. Far less complex in its design parameters and requiring much less memory, the similar functionality of a WSN may be utilized as a benchmark for comparing quantization schemes among different CNN architectures in future work. Prior to this endeavor, an overview of the effect of quantization on the WSN is required and explored in this thesis.

The foundation of a WSN, the wavelet scattering transform is itself an effective instrument in feature extraction due to the combination and nature of the scattering wavelets that it employs. It is even used as a preprocessing measure in [12], in which a WSN performs preliminary feature extraction prior to the training of a deep neural network for localization.

Despite their similar functionality to CNNs, WSNs do not contain learnable parameters. However, adaptive wavelets have been explored for applications such as denoising, compression, and signal classification [11]–[15]. The implementation of adaptive wavelets in WSNs has not

been widely explored, providing a possible method for further improvement. Utilizing the foundations of backpropagation found in CNNs, an adaptive WSN (AWSN) was developed and tested using the MSTAR dataset.

The remainder of this thesis is organized as follows: the background and literature review of concepts crucial to CNNs and WSNs (Chapter 2); the exploration of the unique architecture of the WSN and the development of the quantization method and scales (Chapter 3) to be implemented on the WSN for classification of the MSTAR dataset (Chapter 4); and the development of an AWSN (Chapter 5), followed by a thorough look at its performance in comparison to similarly designed architectures (Chapter 6).

Chapter 2

Background and Literature Review

The WSN shares key properties with the CNN, making it a good study for quantization schemes. While the quantization study performed in this thesis focuses only on the WSN, the AWSN was developed using the foundations of backpropagation found in CNNs, as well as the concepts of the convolutional and fully connected layers. In this chapter, the general architecture of convolutional neural networks, as well as their classification and backpropagation methods are discussed prior to an overview of the wavelet scattering network. Note that due to the 2-D nature of the MSTAR dataset, discussion in this chapter, as well as the remainder of this thesis, will emphasize 2-D implementations.

Convolutional Neural Networks

CNNs apply a discriminative approach to feature abstraction and classification problems. Architectures implement a series of hidden layers for feature extraction, followed by some classification method. The hidden layers typically include combinations of convolutional layers in conjunction with a non-linear activation layer and pooling layers. Lastly, brief notes on design considerations are presented.

Convolutional Layer

Convolutional (CONV) layers contain the learnable network parameters. Convolution connects subregions of an image to a set of filters. An individual CONV-layer contains a set of F

filters, Ω_1, Ω_2, …, Ω_F, of the same size, each with an affiliated scalar bias, b_1, b_2, …b_F. Selection of the filter size is dependent upon the dataset as well as the location of the CONV-layer within the network. In two-dimensional analysis, commonly implemented sizes include 3×3, 5×5, and 7×7, and typically decreases for deeper layers.

If an image, $X \in \mathbb{R}^{D_1^X \times D_2^X \times C}$ passes through a CONV-layer with F filters of size $\Omega_f \in \mathbb{R}^{D_1^\Omega \times D_2^\Omega \times C}$, then the output to the subsequent layer contains F feature maps, S_1, S_2, …, S_F, which is notated as a 3-D matrix, $\boldsymbol{S} \in \mathbb{R}^{D_1^S \times D_2^S \times F}$, such that

$$S_f = \downarrow_d \left(\sum_{c=1}^{C} X(c) * \Omega_f(c) + b_f \right), \tag{2-1}$$

where c indexes along the third dimensions of X and the f-th filter Ω_f, such that $X(c) \in \mathbb{R}^{D_1^X \times D_2^X}$ and $\Omega_f(c) \in \mathbb{R}^{D_1^\Omega \times D_2^\Omega}$, the operator $*$ indicates 2-D convolution, and the function $\downarrow_d (\cdot)$ down-samples its input by a factor of d. Note that in CNN applications, the down-sampling factor is often referred to as stride, which is typically set to one in CONV layers. In addition to stride, the amount of padding (usually zero-padding) around the borders of the input image may also be set.

In this example, the number of learnable parameters in the CONV layer is

$$N_{\text{CONV}} = \left(D_1^\Omega D_2^\Omega + 1\right) F. \tag{2-2}$$

However, if X is a 3-D image with C channels, then there must be C channels per filter, increasing the number of learnable parameters to

$$N_{\text{CONV}} = \left(D_1^\Omega D_2^\Omega C + 1\right) F. \tag{2-3}$$

For application to 2-D image analysis, if $F > 1$, the input to any CONV layer excluding the first will have C channels, where C is equal to the number of filters in the previous CONV layer.

A special case of the CONV-layer, the fully connected (FC) layer reduces the dimensions of the input to a $K \times 1$ vector. For a vectorized input $\vec{X} \in \mathbb{R}^{Dx}$, the FC-layer performs the matrix operation

$$S = W\vec{X} + \vec{b}, \tag{2-4}$$

where the weight matrix $W \in \mathbb{R}^{K \times D_X}$ and the bias vector $\vec{b} \in \mathbb{R}^{K}$.

Non-Linear Activation Layers

The feature maps output from a CONV layer are usually passed through a non-linear activation layer. Non-linear activation functions allow the network to learn non-linear features shared by classes, which allows learning complex objects. The rectified linear unit (ReLU) layer, which instantiates the piecewise function

$$y = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}, \tag{2-5}$$

is a common choice. Other commonly used non-linear activation functions include the sigmoid and hyperbolic tangent functions.

Pooling Layers

Though it does not have a direct comparison within the WSN, pooling layers are briefly discussed here for completeness. Pooling layers serve to down-sample feature maps, thereby reducing the computational complexity of subsequent layers. Max-pooling is one of the most commonly implemented pooling layers.

The removal of pooling layers is argued in [16], wherein the authors state that parameter reduction can be achieved by increasing the stride in the CONV-layer.

Classification and Backpropagation

In a CNN, classification is commonly performed using the stable SoftMax function,

$$\mathcal{s}(\vec{z}) = \frac{e^{\vec{z}+A}}{\sum_k e^{z_k+A}} \tag{2-6}$$

where there are K potential classes, $\vec{z} \in \mathbb{R}^K$, and $A = \max\{z_k \in \vec{z}\}$. Note that the division in (2-6) is an element-wise operation. Prior to its implementation, an FC-layer is used to shape the feature maps into a $K \times 1$ matrix.

The SoftMax score is often used in conjunction with the negative log loss,

$$C(y, \vec{a}) = -\sum_{1<k<K} \mathbb{I}_{k=y} \ln(a_k), \tag{2-7}$$

where y is the actual class, $\mathbb{I}_{\{.\}}$ is the indicator function and a_k is the SoftMax score for the k-th class as calculated in (2-6). (2-7) provides the error to backpropagate through the network. Inserting (2-6) into (2-7), and taking its derivative with respect to each $z_k \in \vec{z}$, the error backpropagated through the layer is

$$\delta_k = \frac{\partial C}{\partial a_k} = a_k - \mathbb{I}_{k=y}. \tag{2-8}$$

Using the chain rule, the error calculated at the m-th layer, $\delta^{(m-1)}$, is the error input to the preceding layer. If $X^{(m)}$ is the input to the m-th layer, which performs the function $f^{(m)}$, then the backpropagation through that layer is

$$\delta^{(m-1)} = \frac{\partial f^{(m)}}{\partial X^{(m)}} \delta^{(m)}. \tag{2-9}$$

Backpropagation through an FC-layer is as follows:

$$\delta^{(m-1)} = W^T \delta^{(m)}. \tag{2-10}$$

In addition, layers with learnable parameters, such as the FC-layer, must perform updates to their weights and biases. This is most simply accomplished using stochastic gradient descent (SGD),

$$\Omega = \Omega - \alpha \cdot \Delta\Omega, \tag{2-11}$$

where α is the learning rate. In the FC-layer, the update values for the weights and biases are

$$\Delta W = \delta^{(m)} \left(X^{(m)}\right)^T \tag{2-12}$$

and

$$\Delta \vec{b} = \delta^{(m)}, \tag{2-13}$$

respectively.

Backpropagation through a CONV-layer is a more involved process. For a CONV-layer with F filters, the error, $\delta^{(m)} \in \mathbb{R}^{D_1^{\delta} \times D_2^{\delta} \times F}$, must be up-sampled along its first two dimensions to the size of the layer input, $X^{(m)} \in \mathbb{R}^{D_1^{X} \times D_2^{X} \times C}$. This is most easily accomplished using the Kronecker product,

$$\delta_{up}^{(m)}(f) = \delta^{(m)}(f) \otimes \bar{1}_{d_2 \times d_2}, \tag{2-14}$$

where $\delta^{(m)}(f) \in \mathbb{R}^{D_1^{\delta} \times D_2^{\delta}}$, $\bar{1}_{d \times d}$ is a matrix of ones in $\mathbb{R}^{d_1 \times d_2}$ and $d = \left[[D_1^X, D_2^X] / [D_1^{\delta}, D_2^{\delta}] \right]$ is the subsampling factor. Assuming a stride of one and no padding, consider the f-th filter with weights $\Omega_f \in \mathbb{R}^{D_1^{\Omega} \times D_2^{\Omega} \times C}$ and $b_f \in \mathbb{R}^{F \times 1}$, the error backpropagated along the c-th channel of the input is

$$\delta^{(m-1)}(c) = \sum_{1<f<F} \delta_{up}^{(m)}(f) * \mathrm{rot}_{180}\left(\Omega_f(c)\right). \tag{2-15}$$

where rot_{θ} indicates a physical rotation of a matrix. The updates to the weights and biases are computed as

$$\Delta\Omega_f(c) = \mathrm{rot}_{180}\left(X^{(m)}(c) * \mathrm{rot}_{180}\left(\delta^{(m)}(f)\right)\right) \tag{2-16}$$

and

$$\Delta b_f = \sum_{u,v} \left(\delta^{(m)}(f)\right)_{u,v} \tag{2-17}$$

respectively, where u and v index the rows and columns of the matrix.

When training in batches, the updates to the learnable parameters are averaged with respect to the number of samples.

Design Considerations

As previously mentioned, architectures can be highly variable among applications, their design more witchcraft than science. For example, though ResNet [17] and VGG [18] were trained using the ImageNet database [19] to classify images into 1000 categories. Architectures may also vary within applications: In [6], the MSTAR dataset was used to demonstrate the

comparable performance of a long network (ten layers) and a short network (five layers), each of which used the same first three layers (CONV-ReLU-POOL).

Depending upon the number of layers in a network (some architectures exceed hundreds of layers), the number of settable attributes can quickly become unwieldy. Ignoring the weight and bias initializations of filters, the filter size and stride for a CONV-layer must be set. These design choices must also be made for the pooling layers.

In addition, though the CNN reduces the memory requirements of preceding neural networks by sharing parameters within its CONV-layers, designs remain highly memory expensive. To reduce required memory, quantization of CONV-layer parameters and the output feature maps has been proposed in [7] and [8] for the classification of the CIFAR-10 [20] dataset with GoogleNet [21] and of the ImageNet [19] dataset with AlexNet [22], respectively. In future work, the application of these quantization methods to the WSN should be performed.

Wavelet Scattering Networks

The WSN shares the key properties of a CNN, primarily convolution, nonlinearity, and layer-wise architecture. A WSN is a windowed scattering transform that extracts features at multiple resolutions. This section provides a background on scattering wavelets and the scattering transform.

Scattering Wavelets

Wavelets provide time and frequency localization of a signal. A 2-D wavelet, ψ, must satisfy the admissibility condition

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{\left|\hat{\psi}(\omega_1,\omega_2)\right|^2}{|\omega_1,\omega_2|^2}d\omega_1\,d\omega_2<\infty, \tag{2-18}$$

where $\hat{\psi}$ is the Fourier transform of the wavelet and ω_1 and ω_2 are the frequencies. Daughter wavelets are constructed from a mother wavelet, ψ, at various scales and orientations. For a scaling factor of 2^j and rotation angle of θ, a 2-D daughter wavelet is

$$\psi_{j,\theta}=2^{2j}\psi\left(2^j r_\theta^{-1}\vec{u}\right), \tag{2-19}$$

where r_θ is the rotation matrix and the position vector, $\vec{u}=[x,y]^T$. For convenience, this thesis uses the notation ψ_λ, where λ indicates a combination of scale and orientation.

A subset of wavelets, scattering wavelets aid in achieving translation invariance, stability, and the linearization of small diffeomorphisms. A scattering wavelet that fulfills (2-18) can be written as

$$\psi(\vec{u})=e^{i\eta x}\vartheta(\vec{u}), \tag{2-20}$$

where $\hat{\vartheta}$ is a real-valued function with primary support $0\leq\omega<\pi$, and the Fourier transform of (2-20) is centered at the frequency η.

Windowed Scattering Transform

A WSN is a windowed scattering transform, consisting of multiple windowed scattering propagators, typically of multiple scattering orders, like that shown in Figure **2-1** for a maximal scattering order of $M=3$. The scattering propagators are formed using a set of daughter wavelets and the windowing is performed by a son wavelet.

The collection of daughter wavelets, ψ_λ ($\lambda \in \Lambda$), defined for J scales and L orientations at Q wavelets per octave, act as high-pass filters. The scale of each is $2^{j/Q}$ for unique $j = 0,1,\dots,J-1$, where Q acts as a quality factor for the employed filter bank; however, for image data, a large Q is unnecessary, and is set to one in this thesis, as in the example in the ScatNet documentation [23]. The orientations, $\theta \in \Theta = \{\ell\pi/L\}_{\ell=0}^{L-1}$.

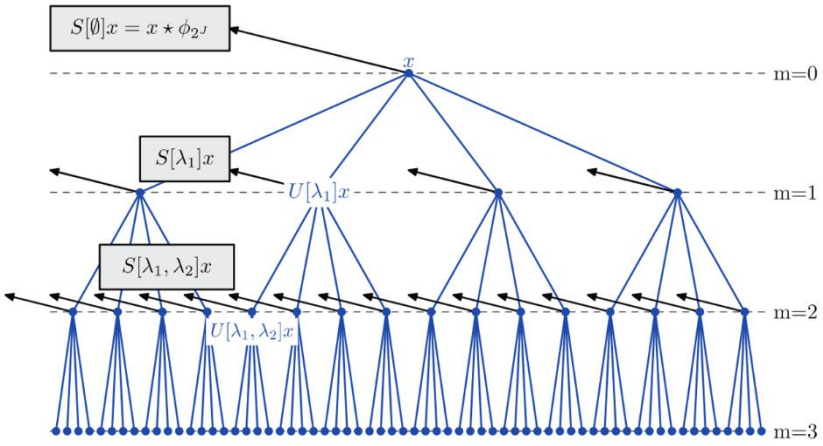


Figure **2-1**: The wavelet scattering network computes the windowed scattering transform [9].

The Littlewood-Paley wavelet transform of an input image, X, with a daughter wavelet is defined as

$$\psi_\lambda * X, \tag{2-21}$$

where each element of X is convolved. As with the wavelet transform, the output of (2-21) is subsampled according to its frequency bandwidth:

$$W[\lambda]X = \downarrow_{2^d} (\psi_\lambda * X). \tag{2-22}$$

The $\log_2$ subsampling rate d is determined as a function of the $\log_2$ filter resolution, r_{ψ_λ}, and the resolution of the input at the m-th scattering order, $r_X^{(m)}$, such that

$$d = \max\left\{0, r_{\psi_\lambda} - r_X^{(m)} - \zeta\right\}, \tag{2-23}$$

where $r_{\psi_\lambda} = \lfloor j/Q \rfloor$ and ζ is the oversampling factor. The resolution of the output is

$$r_X^{(m+1)} = r_X^{(m)} + d. \tag{2-24}$$

Equation (2-22) is not translation invariant; therefore, like the non-linear activation functions in the CNN, it is passed through a non-linear function for demodulation. The modulus of (2-22) may be used for this purpose. The complete process of the wavelet transform and its demodulation is described by the scattering operator $U[\lambda]$.

$$U[\lambda]X = |W[\lambda]X|. \tag{2-25}$$

A scattering propagator applies (2-25) at each λ along a path $p = (\lambda_1, \lambda_2, \dots)$, where each λ_f in p is unique. For an m-th order path—that is, a path of length m—the scattering propagator is defined as

$$U[p]X = U[\lambda_m] \dots U[\lambda_2]U[\lambda_1]X, \tag{2-26}$$

where $U[\emptyset] = I$ and I is the identity matrix. In a WSN, the set of paths, P, that define a scattering propagator are unique, and vary in length between 0 and the maximal scattering order, M. The number of paths in P is restricted so that only filters of increasing scale can be used ($j_m \geq j_{m-1} + Q$).

To form a windowed scattering propagator, a son wavelet, ϕ_J, is constructed from a father wavelet, ϕ, such that

$$\phi_J = 2^{-2(J-1)}\phi\left(2^{-(J-1)}\vec{u}\right), \tag{2-27}$$

which acts as a low-pass filter. (Note that if $Q \neq 1$, then the scaling of ϕ_J is $2^{(J-1)/Q}$.) The windowed scattering propagator subsamples the convolution of (2-27) with the scattering propagator in (2-26):

$$S_J[p]X = \downarrow_{2^d} \left(U[p]X * \phi_J\right). \tag{2-28}$$

Note that the path length of the windowed scattering propagator is the same as the scattering propagator.

Figure **2-2** provides an example of the paths found in P for $J = 4$, $L = 2$, $Q = 1$, and $M = 2$. Note that these paths overlap: For example, the path $p = \left((0, \theta_1)\right)$ is part of the path $p = \left((0, \theta_1), (1, \theta_2)\right)$. The output from each $U[p]$ is passed to ϕ_4, as well as to all orientations of the high-pass filters with scales 2^j, such that $j_{m-1} + 1 \leq j < J$, where $2^{j_{m-1}}$ is the scale of the most recent filter along the path.

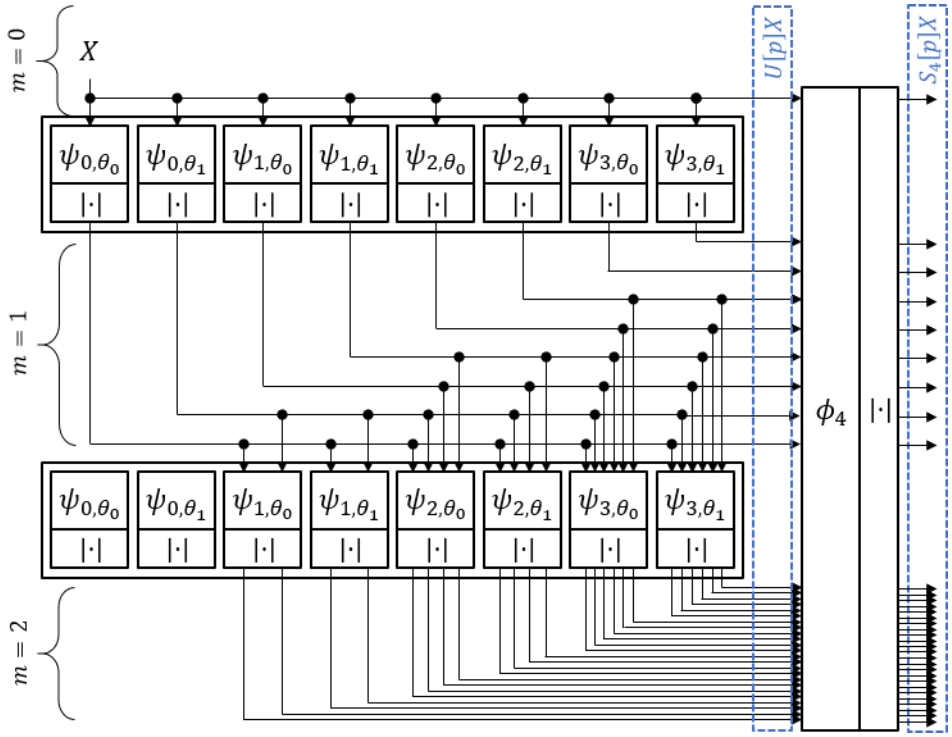


Figure **2-2**: WSN paths with $J = 4$, $L = 2$, and maximal scattering order $M = 2$.

Each $S_J[p]X$ is a feature map akin to those output from the CONV-layers of a CNN, and may be input to a classifier, such as a support vector machine (SVM) or the stable SoftMax function, following additional processing.

Code Implementation

ScatNet, the code implementation of the windowed scattering transform used in this thesis is publicly available [23]. For a 2-D WSN, in addition to the values of Q, J, L, and M, the user has the selection of the wavelet filter bank, as well as the parameters of the wavelets. There are two 2-D wavelet filter banks from which to choose, the Morlet and the Shannon. According to ScatNet documentation, the irregular Fourier transform of the Shannon wavelets causes poor localization; thus, this thesis considers the Morlet filter bank is considered. It consists of the Morlet wavelet and a Gaussian window. The Morlet wavelet is defined as

$$\psi(\vec{u}) = \left(e^{i\xi x} - K\right)e^{-\frac{x^2+s^2y^2}{2\sigma_\psi^2}}, \tag{2-29}$$

where s is the slant, or eccentricity, of the elliptical Gaussian envelope, σ_ψ is the standard deviation of the elliptical Gaussian envelope, and K is a constant set so that the average of (2-29) is zero. For almost unitary operators, the daughter wavelets are renormalized by $\mu_p = \sqrt{0.5 \max\left\{\sum_{\lambda\in\Lambda}\left|\hat{\psi}_\lambda\right|^2\right\}}$.

For the Morlet wavelet defined in (2-29), the Gaussian envelope

$$\phi(\vec{u}) = e^{-\frac{\left(x^2+y^2\right)^2}{2\sigma_\phi^2}} \tag{2-30}$$

is a common choice for the windowing function.

The defaults for the parameters σ_ψ and σ_ϕ are 0.8. The remainder of ψ parameters are calculated as functions of J, L, and/or Q. In addition, by default, the filter sizes are dynamically set to the smallest multiple of 2 with a minimum margin of $2^{J/Q}$.

In the MATLAB code of [23], a 2-D WSN can be implemented for a set number of scales and orientations. Because the size of the image varies along the path of the windowed scattering propagator, the code implements the same filter at each required resolution. Figure **2-3** provides an updated view of the scattering path found in Figure **2-2**. Filters at different resolutions are denoted with the superscript r, where $r = 0$ is the resolution of the original filter. The filter $\Omega^{(r)}$ is constructed via the periodization of the Fourier transform.

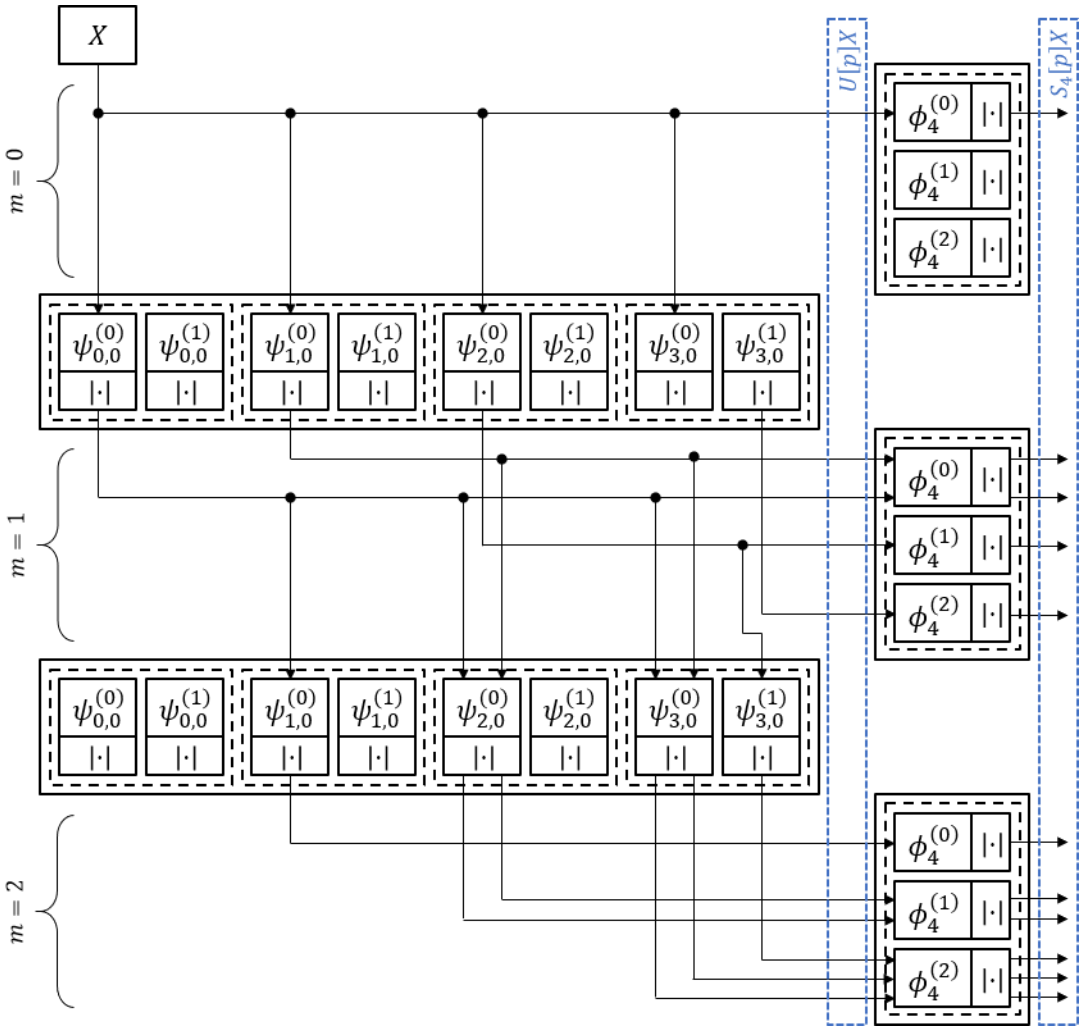


Figure **2-3**: WSN paths accounting for different resolutions with $J = 4$, $L = 1$, and maximal scattering order $M = 2$.

Note that another implementation of the scattering network exists that follows a tree-search algorithm, which is more memory efficient that may be considered for future work [24].

Chapter 3

Quantization of a Wavelet Scattering Network

The WSN requires less memory than the large CNNs used for classification applications. Due to its functional similarity to CNNs, a WSN can provide a benchmark for the comparison of quantization schemes. This section explores the quantization of a wavelet scattering network using a set encoding method. Several quantization schemes are discussed prior to their implementation Chapter 4.

For an input to a network, X, each windowed scattering propagator, $S_J[p]X$, for all paths $p \in P$, as well as the intermediate operations of the propagator, $U[\lambda_1]X, U[\lambda_1, \lambda_2]X, \dots, U[p]X$, are quantized, such that the output along a path of length T undergoes $T + 1$ quantizations. The calculation and application of quantization levels is based on the ScatNet [23] implementation, wherein the m-th order scattering operations, $U\left[p^{(m)}\right] = U[\lambda]U\left[p^{(m-1)}\right]X$, and the windowing operations for the previous scattering order, $S_J\left[p^{(m-1)}\right]X$, are applied in one iteration of the scattering code.

The outputs, $\mathcal{Y} = \left(U\left[p^{(m)}\right]X \forall p^{(m)} \in P, S_J\left[p^{(m-1)}\right]X \forall p^{(m-1)} \in P\right)$, from this scattering layer, or s-layer, are fed to a quantization layer, or q-layer, wherein each input, $Y \in \mathcal{Y}$, is quantized to K unique levels. These quantization levels are generated using the values in Y. Figure **3-1** provides a visualization of the s-layers and their corresponding q-layers for an $M = 2$ network with $L = 1$. Note that the quantization processes within all q-layers in a network are isolated events.

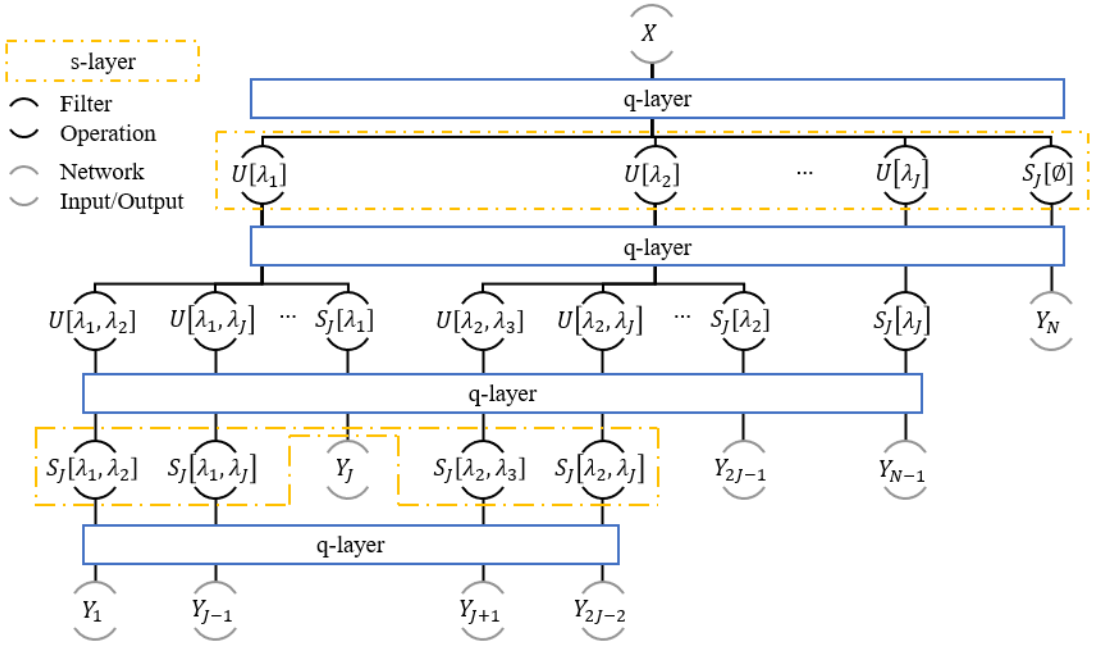
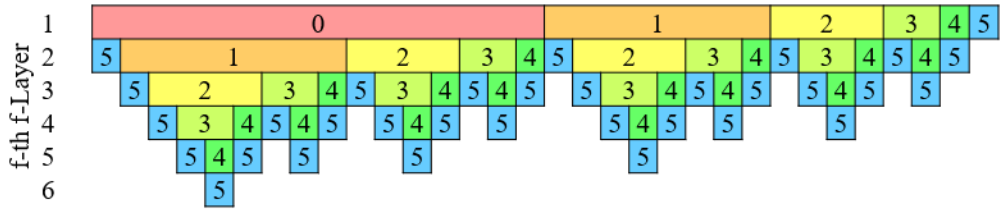


Figure **3-1**: Location of the q-layers in a WSN where $L = 1$.

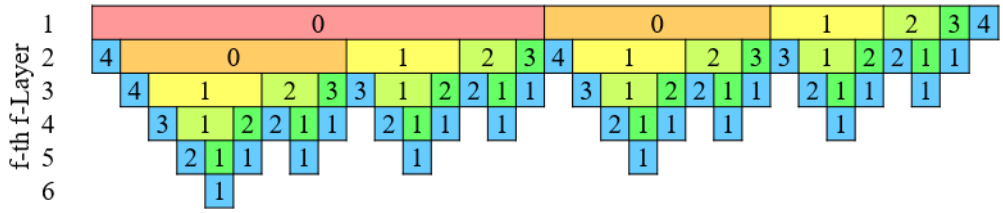
The purpose of this quantization scheme is to provide an initial comparison for quantization by limiting the number of unique values permitted. The effectiveness of such a quantization scheme is dependent upon the size of each $Y \in \mathcal{Y}$. Suppose that $|\mathcal{Y}| = N$ for an s-layer. If each value in Y requires b bits for representation, then the total number of bits required to represent all the values in Y is Vb. However, the sizes of the N outputs contained in $\mathcal{Y}$ vary with the scales of the filters used in their computation, with the maximum number of values in a Y, $\hat{V} \leq D_X$, where D_X is the number of values in the input to the network, X. In addition, the number of outputs in $\mathcal{Y}$ vary with the depth of the s-layer in the network.

Figure **3-2** provides an example for an WSN for $J = 5$, $L = 1$, $Q = 1$, $\zeta = 1$, and $M = 5$. In Figure **3-2(a)**, the number of calls to each scattering wavelet filter, ψ_λ, with $j > 1$, increases then decreases with each s-layer; ψ_λ with $j = 1$ are called once only in the first and second s-layers; and ψ_λ with $j = 0$ is called only once in the first s-layer. The windowing filter, ϕ_J, is called during each s-layer, once for each call to a ψ_λ in the previous s-layer and once in the first

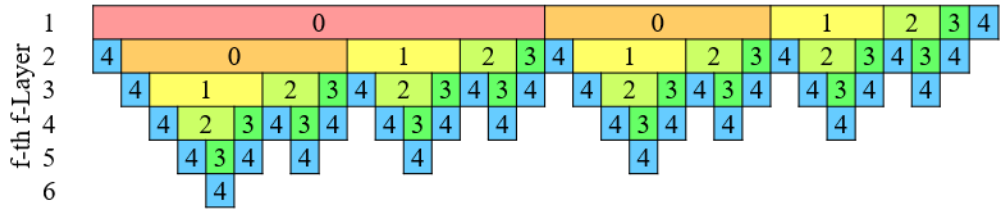
s-layer. The $\log_2$-down-sampling rate of each output, provided in Figure **3-2(b)**, is used to determine the size of the output from each filter, $D_X/2^d$, where D_X is the number of elements in the input. The size of the output along any path is quantified as the sum of the $\log_2$-down-sampling rates, or the $\log_2$-resolution of the output, which is found in Figure **3-2(c)**.



(a) ψ_λ and ϕ_J Filter Scales used for $J = 5$ (j).



(b) $\log_2$-Down-Sampling Rate (d_f).



(c) Cumulative $\log_2$-Down-Sampling Rate ($r_j = \sum d_f$).

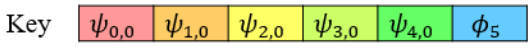


Figure **3-2**: Overview of the scales (a) and resolutions (b) of each filter used and the down-sampling factor (c) and cumulative down-sampling factor (d) of the input at each filter for all possible s-layers in a network where $J = 5$ and $L = 1$.

For a general network with $Q = 1$, a mathematical derivation for the size of the output images and the number of outputs from an s-layer is provided in the following two sections.

Sizes of Filter Outputs

The input along a path of length T, $p^{(T)}$, undergoes $T+1$ operations; that is, the input passes through T bandpass filters associated with the operators $U[\lambda_t]$, $\psi_{\lambda_1}, \psi_{\lambda_2}, \dots, \psi_{\lambda_t}, \dots, \psi_{\lambda_T}$, with scales 2^{j_t} ($j_t = 0,1, \dots, J-1$), and one windowing filter, ϕ_J, with scale 2^J. For simplicity, the f-th intermediate output along a scattering propagator $U\left[p^{(T)}\right]X$ is denoted as $U\left[p_f^{(T)}\right]X$, such that $U\left[p_f^{(T)}\right]X = U\left[p^{(f)}\right]X$, where $f \leq T$.

Setting $Q = 1$, the resolution of a filter used in the f-th s-layer, and $r_{\psi_\lambda}^{(f)}$, reduces to the scale of the filter, j_f, thereby simplifying (2-23) to

$$d_f = \begin{cases} 0 & j_f < r_X^{(f)} + \zeta \\ j_f - r_X^{(f)} - \zeta & j_f \geq r_X^{(f)} + \zeta \end{cases}, \tag{3-1}$$

where $r_X^{(f)}$ is the resolution of the input to the f-th filter.

Equation (2-24) can be reduced by examining the two cases of (3-1). In the first case, where $j_f < r_X^{(f)} + \zeta$, d_f and $r_X^{(f)}$ are always 0; this allows the simplification of the case statement to $j_f < \zeta$. For the second case, where $j_f \geq r_X^{(f)} + \zeta$, substituting (3-1) into (2-24) yields

$$r_X^{(f+1)} = j_f - \zeta. \tag{3-2}$$

The result is independent of $r_X^{(f)}$, therefore, independent of the scales of the previous filters along the path. This simplifies the case statement to $j_f \geq \zeta$. Similarly, the resolution of the output from the f-th filter in a path is

$$r_X^{(f+1)} = \begin{cases} 0 & j_f < \zeta \\ j_f - \zeta & j_f \geq \zeta \end{cases}. \tag{3-3}$$

According to (3-3), $r_X^{(f+1)}$ depends only upon the most recent filter scale. For this reason, the notation $r_j = r_X^{(f+1)}$ will denote the resolution of the output from a filter with scale 2^j for any f.

As described in (2-23), the resolution of the output is the summation of the $\log_2$-down-sampling rate, d_n, for $1 \leq n \leq f$. (The final resolution of the output, $r_X^{(T+2)}$, i.e. a feature map, is the summation of d_f, for $1 \leq f \leq T+1$). Therefore, for a network input, $X \in \mathbb{R}^{D_1 \times D_2}$, the number of values in the output, Y, from the f-th s-layer is

$$V_j^{(f)} = \frac{D_1 D_2}{2^{2(r_j+1)}}, \tag{3-4}$$

where the +1 is attributed to the down-sampling by the ϕ_J. The size of an output is independent of the s-layer in which it is produced, again depending upon only the resolution of the output. To reduce the number of unique values in Y, the number of levels, $K < V_j^{(f)}$.

The total number of values submitted from the f-th s-layer to the following q-layer is

$$V^{(f)} = U^{(f)} R^{(f)} + \sum_j V_j^{(f)} N_j^{(f)}, \tag{3-5}$$

where $N_j^{(f)}$ is the number of outputs from each bandpass filter with a scale 2^j, and $U^{(f)}$ is the number of values in each of the $R^{(f)}$ lowpass filter outputs. In the next section, calculations for $N_j^{(f)}$ and $R^{(f)}$ are discussed.

Number of Filter Outputs per s-layer

The total number of s-layers in the network is equal to $M+1$. The number of outputs from a filter in each s-layer varies with the maximum scale, J, the number of orientations per scattering wavelet, L, and the scattering order, m, associated with the operations in an s-layer.

Recall that in the f-th s-layer, layer inputs undergo the operations $U\left[p^{(m)}\right]$ for all $p^{(m)} \in P$ and $S_J\left[p^{(m-1)}\right]$ for all $p^{(m-1)} \in P$.

If $M = 0$, then the total number of operations in the first s-layer, and the entire network, is one ($S_J[\emptyset]X$). however, if $M > 0$, the first s-layer contains the operations $U\left[p^{(1)}\right]X$ for each path $p^{(1)} \in P$ and $S_J[\emptyset]X$, such that each filter has one output $N_j^{(1)} = 1$, making the total number of outputs equal the number of filters.

$$N^{(1)} = \begin{cases} J \cdot L + 1 & M > 0 \\ 1 & M = 0 \end{cases}. \tag{3-6}$$

The second s-layer accepts $N^{(1)} - 1$ outputs from the first layer $U\left[p^{(1)}\right]X$ as inputs to the filters ($S_J[p]X$ constitutes a terminated path for any p). The windowing filter operates on all $J \cdot L$ inputs to the s-layer, while the bandpass filters only operate on those inputs with paths ending with a smaller scale: An input with path $p^{(1)} = \left((j_2, \theta \in \Theta)\right)$ is operated upon by each ψ_λ, where $\lambda = (j, \theta \in \Theta)$ with $j_2 + Q \leq j < J$. That is for a given j_2, $j = j_1 + Q, j_1 + 2, \dots, J - 1$. Using $a_j = \psi_\lambda$ and $L = 1$, the total number of outputs from the ψ_λ filters can be stated as a function of j:

$$\begin{aligned} \sum_{k=0}^{J-2} \sum_{j=k+1}^{J-1} a_k &= \left(a_1 + a_2 + a_3 + \cdots + a_{J-2} + a_{J-1}\right) \\ &\quad + \left(a_2 + a_3 + \cdots + a_{J-2} + a_{J-1}\right) \\ &\quad + \left(a_3 + a_4 + \cdots + a_{J-2} + a_{J-1}\right) + \cdots + \left(a_{J-2} + a_{J-1}\right) \\ &\quad + \left(a_{J-1}\right) \\ &= a_1 + 2a_2 + 3a_3 + \cdots + (J-1)a_{J-1} \\ &= \sum_{k=0}^{J-1} k a_k \end{aligned}$$

$$= \sum_{k=0}^{J-1} N_k^{(2)}\, a_k.$$

For $L > 1$, the number of outputs from each ψ_λ is $N_j^{(2)} = jL$. Therefore, the total number of outputs from the second s-layer is

$$N^{(2)} = \left(N^{(1)} - 1\right) + L \cdot \sum_{k=0}^{J-1} k. \tag{3-7}$$

If $M > 2$, then this pattern continues, with each of the $N^{(f-1)} - N^{(f-2)}$ inputs to an s-layer. The number of outputs from any bandpass filter in the s-layer is

$$N_j^{(f)} = \begin{cases} L \cdot \sum_{k=0}^{j-1} N_k^{(f-1)} & f > 1 \\ 1 & f = 1 \end{cases} \tag{3-8}$$

The summation may also be expressed in terms of $N_j^{(f)}\Big|_{L=1}$, the number outputs from a filter in the f-th layer for $L = 1$:

$$L \cdot \sum_{k=0}^{j-1} N_k^{(f-1)} = L \cdot \sum_{k_f=0}^{j-1} \left(L \cdot \sum_{k_{f-1}=0}^{k_f-1} \ldots \left(L \cdot \sum_{k_1=0}^{k_2-1} N_{k_1}^{(1)} \right) \right)$$

$$= L^{(f-1)} \cdot \sum_{k=0}^{j-1} N_k^{(f-1)}\Big|_{L=1}.$$

Furthermore, because $j_f > j_{f-1}$, $N_j^{(f)} = 0$ for $j < f - 1$, which modifies (3-8) to

$$N_j^{(f)} = \begin{cases} L^{(f-1)} \cdot \sum_{k=f}^{j-1} N_k^{(f-1)}\Big|_{L=1} & j \geq f - 1 > 0 \\ 0 & j < f - 1 \\ 1 & f = 1 \end{cases}. \tag{3-9}$$

The total number of outputs for an s-layer is

$$N^{(f)} = \begin{cases} N^{(f-1)} - N^{(f-2)} + \sum_{k=f}^{j-1} N_k^{(f)} & 1 < f \leq M \\ N^{(f-1)} - N^{(f-2)} & f = M+1 \\ 1 & f = 1 \end{cases}. \tag{3-10}$$

Quantization Scales

In this section, several quantization scales are explored to create K quantization levels. In this section, the quantization levels of a $Y \in \mathcal{Y}$, are denoted as v_Q.

Uniform Scale

The uniform, or linear, quantization scale provides a good performance benchmark. The uniform scale was constructed by uniformly spacing values, such that the K levels are

$$v_Q \in \left(\min(Y), \min(Y) + dv_Q, \min(Y) + 2dv_Q + \cdots + \max(Y)\right)$$

where $dv_Q = (\max(Y) - \min(Y) + 1)/2^K$.

Log-Scale

Log-scale quantization provides another simple benchmark; however, to prevent values $v_Q \in Y$ that exist outside the domain of the log function, $\log(z) \in (0, \infty)$, some value, dz, must be added to all values in Y. The log-scale is constructed as follows:

$$Y \to Z: z = \log(v + dz)\ \forall z \in Z$$

$$Z_Q \in \left(\min(Z), \min(Z) + dv_Q, \min(Z) + 2dv_Q + \cdots + \max(Z)\right)$$

$$v_Q \in \left(e^{\min(Z_Q)-dz}, e^{\min(Z_Q)-2dz}, \ldots, e^{\max(Z_Q)}\right)$$

where $dv_Q = (\max(\mathrm{Y}) - \min(\mathrm{Y}) + 1)/2^K$. Though the SAR magnitude data preclude the possibility of negative values, as x approaches 0, $\log(z)$ approaches $-\infty$. To prevent dealing with unreasonably large numbers (which would require more memory for representation) the shifting value $dz = 1$.

K-Means Scale

Quantization via k-means clustering is a common method. In this paper, the k-means scaling was implement using Lloyd's algorithm with random initializations for K centroids, or quantization levels. Following the convergence of the clustering algorithm, the nearest neighbor method maps the values in Y to the K quantization levels. Note that this method assumes convergence for the success of this quantization scale, and convergence to a global minimum depends on the centroid initialization; therefore, the k-means++ algorithm proposed in [25] is used to heuristically select centroid initializations for improved convergence time and successful clustering.

Note that this method requires a random number generator (RNG). In the next section, two additional methods implementing an RNG are discussed.

Probability Distribution Scales

To explore the impact of the data on the quantization levels, the probability distribution function (pdf) and the output from each s-layer, is used to generate quantization levels. Best-fit

pdfs for the data were selected using maximum likelihood estimation (MLE). The global best-fit pdf was determined by assessing the individual fits to the s-layer outputs for each class. Figure **3-3** provides an example of such a fitting using the inverse Gaussian distribution on the MSTAR dataset.

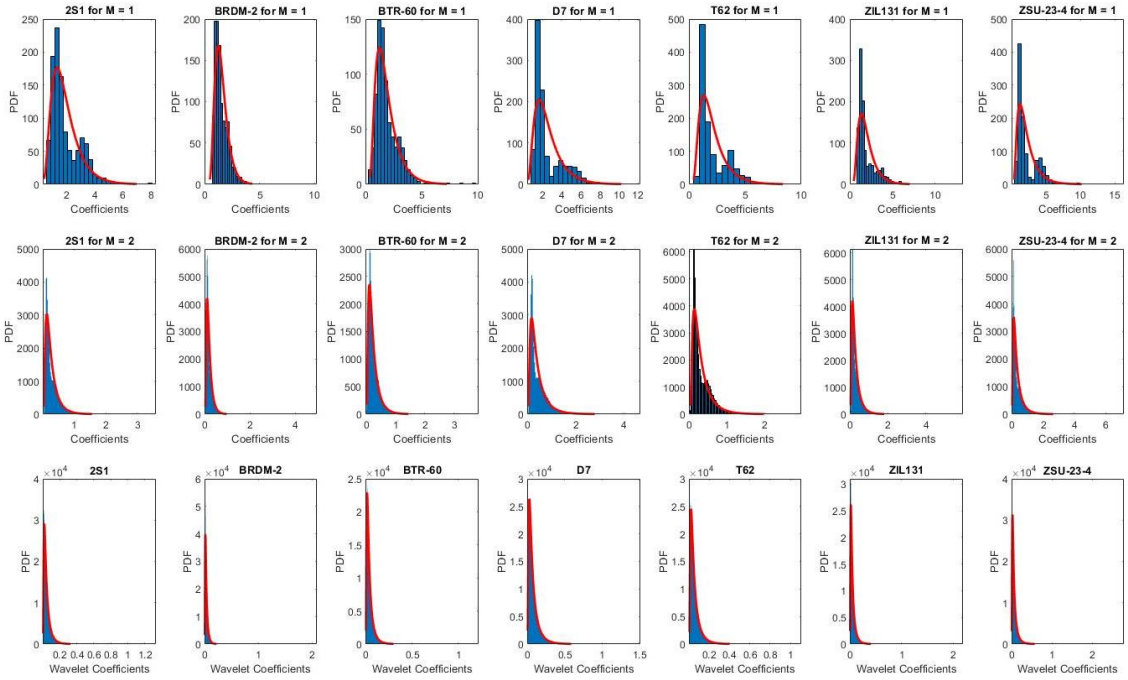


Figure **3-3**: Histogram of MSTAR dataset separated by class for each s-layer in a WSN of $M = 2$, $Q = 1$, $J = 5$, and $L = 8$, as well as a fitted inverse Gaussian pdf.

The inverse Gaussian and the gamma distributions were selected to generate quantization scales. At each s-layer, the data in Y are fitted to one of these distributions. Quantization levels are then determined using random number generation, until K unique levels are found. For the inverse Gaussian scale, the values are generated using the algorithm in [26]; for the gamma scale, the values were generated with Marsaglia and Tsang's method [27]. Lastly, each $y \in Y$ is mapped to a level using nearest-neighbor. Note that the generation of the random variate v_Q uses an RNG.

As with the log-scale, domain restrictions must be handled. The inverse Gaussian distribution has support on $[0, \infty)$, therefore shifting the data such that $\min(y \in Y) \geq 0$ aids in obtaining the fitted pdf. The support of the gamma distribution, $(0, \infty)$ requires shifting the values such that $\min(y \in Y) > 0$; the value of dz should be as small as possible. For practical purposes, the rounding error of the implementing system, multiplied by some factor, might be considered.

Following the calculation and application of the quantization levels, the data is shifted by $-dz$.

Quantile Scale

Another pdf-based quantization scale is considered that requires no RNG. The distribution of the values in Y are divided into K quantiles. The midpoint of each quantile is then used as a quantization level, v_Q. This provides more static fit of a pdf to the data; moreover, there are no support considerations required for its implementation.

Chapter 4

Experiments: Quantized Wavelet Scattering Network

The performances of the quantization method and scales were tested using a second-order WSN with Morlet wavelets at five scales ($J = 5$) and eight orientations ($L = 8$), and the Gaussian windowing function. The parameter values of the mother Morlet wavelet and the Gaussian function, as defined in (2-29) and (2-30), respectively, are provided in Table **4-1**. The size of the filters at $r = 0$ was 2112×2112 pixels.

Table **4-1**: Morlet Wavelet and Gaussian Envelope Settings

Parameter	σ_ϕ	σ_ψ	s	ξ
Value	0.8	0.8	0.5	2.356

The output of the WSN is modified to form a feature vector $\vec{a} \in \mathbb{R}^{\sum N^{(f)}}$, such that each element is equal to $\sum_{\forall y \in Y} y$ for a unique windowed scattering propagator.

Using the MSTAR dataset, each quantizer was tested with 2, 4, 16, and 256 quantization levels. At each location, the same quantization scheme was implemented. The effect of random number generation on the results for the RNG-based quantization scales, k-means, inverse Gaussian, and gamma, was evaluated by seeding the RNG with 10 different seeds. Lastly, the effect of noise addition and removal were assessed for SNR = 2-dB and 10-dB.

Description of the MSTAR Dataset and Augmentations

Eight of the available classes from the mixed target subset of the MSTAR database were used for classification (see Figure **4-1**). In existing works that apply CNNs to SAR data, the 15° and 17° depression angle data are used, typically separated into training and test data. While the

difference between 15° and 17° may be negligible, to remove any possible confusion during analysis, only the 15° data were used. In future work, division of SAR datasets by viewing angle could afford the assessment of the generalizability of a learning algorithm. If an algorithm can successfully train on some minimum number of view angles, the amount of data collection required for application would be greatly reduced. Future work might also implement larger datasets with more variable viewing angles.

(a) 2S1 (b) BRDM-2 (c) BTR-60 (d) D7

(e) SLICY (f) T62 (g) ZIL-131 (h) ZSU-23-4

Figure **4-1**: Classes in the MSTAR dataset.

The number of samples from each class in the 15° subset of the MSTAR database are provided in Table **4-2**. Note that the number of samples for BTR-60 was approximately 70% of the other classes, making this an unbalanced dataset. Each of the samples were cropped to 64×64 images, centered on the target, to reduce the effect of the target's surroundings on classification results.

Table **4-2**: Number of MSTAR images in used for training and validation

2S1	BRDM-2	BTR-60	D7	SLICY	T62	ZIL-131	ZSU-23-4
274	274	195	274	274	273	274	274

To assess the effectiveness of classification in the presence of noise, Gaussian noise was added to the training and test data. Though this method of noise addition does not reflect the actual presence of noise in SAR imagery, it can provide an idea of the robustness of the network

in classifying more complex data. Prior to any data augmentation or feature extraction, the samples were shuffled to avoid any impact of RNG artifacts during quantization.

In addition to training and testing the noisy dataset, the network was trained on the same noisy data after noise removal via Biorthogonal spline wavelets. The threshold values were determined using the Birge-Massart strategy.

Figure **4-2** provides an example of the noise addition and removal. Note that the noise removal process is imperfect, and the resulting image contains a blur. The imperfection of the noise removal method allows another assessment of the robustness of the WSN-SVM network, as well as the underlying separability of the MSTAR dataset.

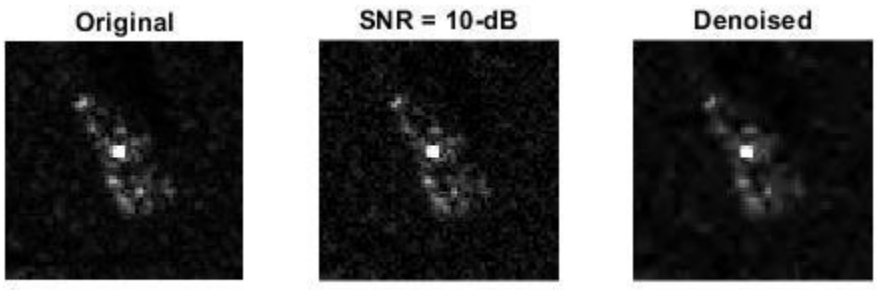


Figure **4-2**: SAR image of a 2S1 image from the MSTAR dataset with 10-dB of added Gaussian noise before and after wavelet noise removal.

Evaluation Metrics

Classification was performed using a one-vs.-all SVM. No consensus for assessing the success of a multi-class classification algorithm exists. For unbalanced datasets, such as the MSTAR dataset as described in Table **4-2**, the accuracy cannot be calculated via the traditional binary methods. [28] suggests the use of a balanced accuracy,

$$\mathrm{ACC} = \frac{1}{2}(\overline{\mathrm{TPR}} + \overline{\mathrm{TNR}}), \tag{4-1}$$

where $\overline{\mathrm{TPR}}$ and $\overline{\mathrm{TNR}}$ are the true positive and negative rates, respectively. This was expanded to a multi-class, one-vs.-all SVM by calculating the balanced accuracy for each class, with the true positive and negative rates calculated as macro-averages. To accommodate the small size of the dataset, 3-fold validation was used.

Results

The results of the five experiments are presented in this section. For reference, the performance of the non-quantized WSN for all five datasets is provided in Table **4-3**.

The method of noise removal did little to improve performance, the differences negligible for most dataset folds. Lack of improvement was likely due to the method of noise addition. Another possibility is that the wavelet noise removal did successfully remove the added noise as well as removed important data points within the image as well. In the future, a more tailored method of noise removal should be considered.

Table **4-3**: SVM Accuracies for Non-Quantized WSNs

SNR = ∞	**SNR = 10-dB**	**SNR = 10-dB with WNR***	**SNR = 2-dB**	**SNR = 2-dB with WNR**
0.9620 ± 0.0093	0.6718 ± 0.0097	0.6851 ± 0.0065	0.5828 ± 0.0013	0.5664 ± 0.0064

Effects of RNG Seeding

Figure **4-3** shows the results of the RNG-based quantization scales for 10 different seeds. Regardless of the number of quantization levels, the variability in network performance was negligible. Of the three quantization scales, k-means outperformed the pdf-based methods, with

accuracies greater than 0.6 for all folds at only $K = 2$. This, as well as the under-performance of the gamma and inverse Gaussian distributions is discussed further in the following section.

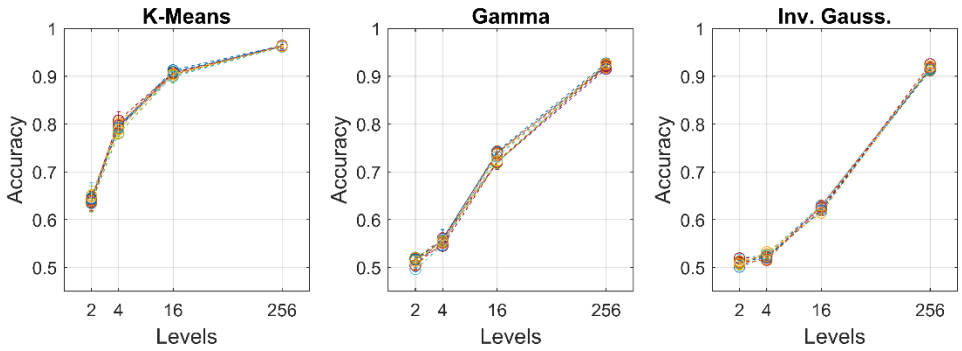


Figure **4-3**: Accuracy of the networks implementing RNG-based quantization scales for 10 different initial seeds.

Noiseless Dataset

The effect of quantization for the noiseless dataset for all values of K assessed is provided in Figure **2-1**. As expected, a smaller K yielded poorer accuracy. The steepest drop-off in accuracy occurred in the gamma and inverse Gaussian scale between $K = 256$ and $K = 16$. This was likely caused by poor choice of the quantization values at the input and the first s-layer. The input to the network contained 4096 values (64^2), which were quantized to K unique values. For $K = 256$, the ratio of potentially representative quantization levels to unique non-quantized values was, at worst, $256/4096 = 0.0625$; the remaining fraction of the input, 0.9375, or 3840 values, would be quantized to the 256 levels. For $K = 16$, the ratio of potentially representative values to true values was 0.0039, resulting in the quantization of at least 4080 values, which left even more potential for error. Poor selection of initial quantization levels would therefore have a significant impact throughout the network. In addition, by quantizing to fewer levels, the output

from the following layer likely loses the distribution shape found in the non-quantized data, making the distributions poor fits within the WSN.

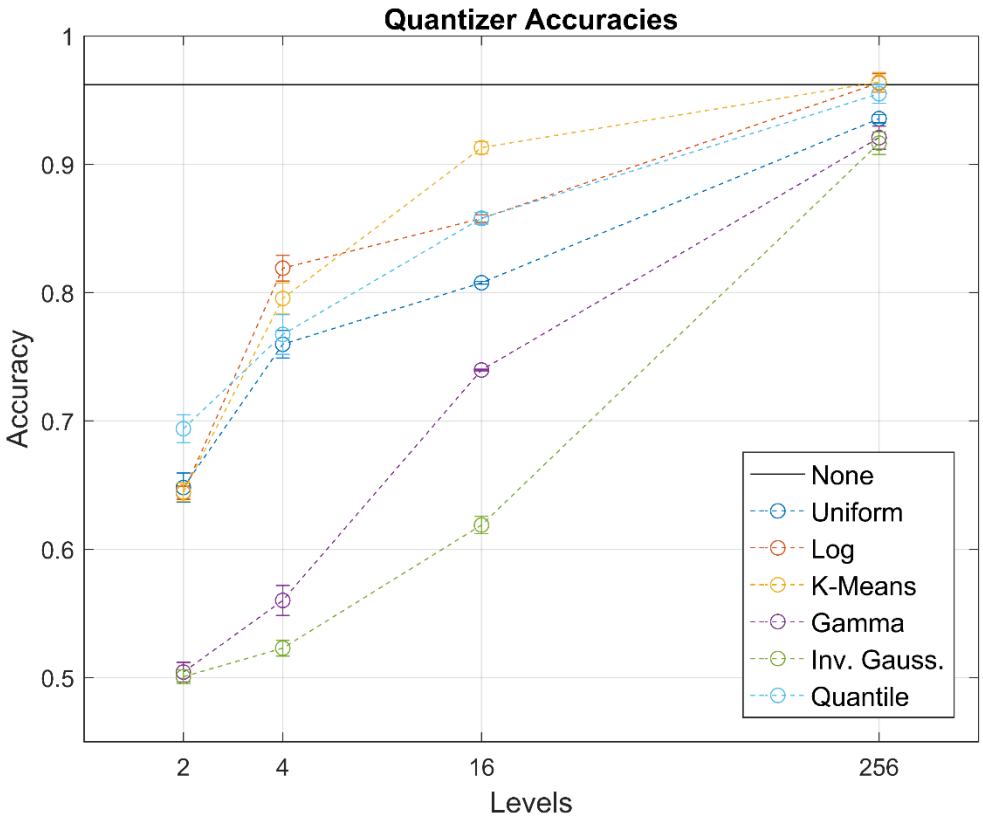


Figure **4-4**: Accuracy of the quantizer scales for SNR = ∞.

The underlying cause of the poor performance of the gamma and inverse Gaussian quantization scales was likely the RNG that they employ. While the k-means scale also used an RNG, the initial centroids were heuristically chosen to improve performance. In addition, the potential for a pdf-based scale is strengthened by the high accuracy of the quantile scale, which nears 0.7 even for $K = 2$.

Dataset with Added Gaussian Noise

As expected from the results of the non-quantized WSN, the addition of noise to produce 10-dB and 2-dB SNR resulted in significantly lower accuracies (see Figure **4-5** and Figure **4-6**). Additionally, the performance was not improved with noise removal. The reasons for this are the same for the noiseless images. The method of adding noise was unnaturally done to SAR image, and the noise removal filter may have removed too many identifying features or too little noise.

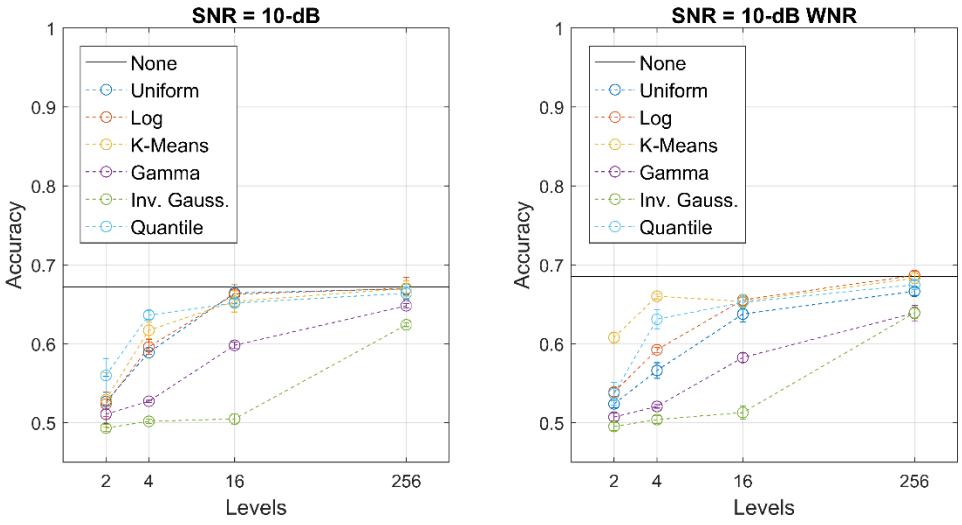


Figure **4-5**: Accuracy of the quantizer scales for SNR = 10-dB (left) and that after wavelet noise removal.

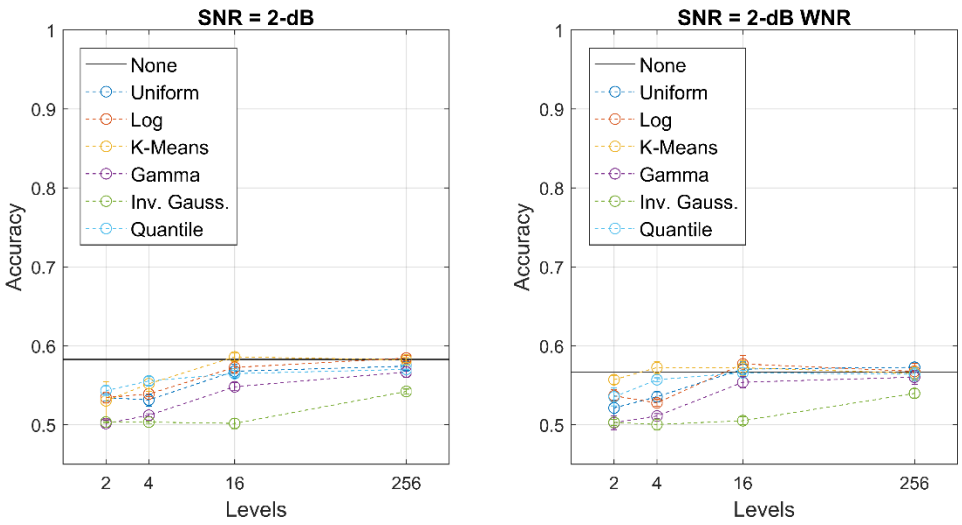


Figure **4-6**: Accuracy of the quantizer scales for SNR = 2-dB and that after wavelet noise removal (WNR).

Conclusion

Though the WSN is not itself adaptive, its similar functionality to CNNs may provide a good benchmark for future work in the quantization of neural networks. In this chapter, the classification accuracy of a quantized WSN with a multi-class SVM was explored. The compounding of error resulting from poorly selected quantization levels can have a drastic effect on the classification performance of the entire network, as seen in the inverse Gaussian and gamma scales. Though these RNG-based pdf quantization scales were not fruitful, the RNG-based k-means scale performed well, as did the more statically generated levels of the pdf-based quantile scale, providing an interesting avenue for future work. However, the performance of any method may be hampered by unexpected noise or error added to the training samples, as was seen with addition and removal of Gaussian noise.

Chapter 5

Designing an Adaptive Wavelet Scattering Network

The purpose of this AWSN is to update the parameters of the scattering wavelet and windowing function to produce more accurate classifications. The AWSN outlined here is designed as a four-layer network, as shown in Figure **5-1**, akin to the serial layers of a CNN. The WSN-layer contains the WSN; the feature map manipulation layer, or FEAT-layer, rearranges the output from the WSN-layer into a format readable by the FC-layer that follows; and the classification, or CLASS-layer performs the classification and calculates the error for backpropagation. In this design, both the WSN-layer and the FC-layer contain learnable parameters.

Figure **5-1**: Layers of the designed AWSN.

The forward and backward operations of the FC-layer and the CLASS-layer are described in Chapter 2. The remainder of this chapter provides a detailed description of the WSN-layer and FEAT-layer and their forward and backward processes in the context of a 2-D application.

The WSN-Layer

The WSN-layer contains the entirety of the WSN architecture. For this layer, design parameters can be categorized into two sets, that of the scattering wavelet and the windowing function, and that of the network architecture. The scattering wavelet and windowing-function

parameters provide the design for the wavelet and windowing function. For example, the Morlet wavelet as described in (2-29) contains adjustable parameters σ_ψ, ξ, and s, and the Gaussian windowing-function of (2-30) contains the parameter σ_ϕ. The network design parameters include the number of wavelet scales, J, the number of wavelet orientations, L, and the maximal scattering order, M. Additional parameters, such as the oversampling factor for filter output, may also be adjusted, but are not considered in this thesis.

During forward propagation, the input, $X^{(1)}$, is passed through a WSN, outputting a set of F feature maps, $X^{(2)} = \mathcal{S}_J X^{(1)} = \left(S_J[p_1]X^{(1)}, S_J[p_2]X^{(1)}, \ldots S_J\left[p_f\right]X^{(1)}, S_J[p_F]X^{(1)}\right)$, where $S_J\left[p_f\right]X^{(1)} \in \mathbb{R}^{A\times B}$ and $p_f \in P$.

During backpropagation, the error backpropagated from the FEAT-layer $\delta^{(1)} = \left(\delta_{p_1}^{(1)}, \delta_{p_2}^{(1)}, \ldots, \delta_{p_F}^{(1)}\right)$, where $\delta_{p_f}^{(1)} \in \mathbb{R}^{A\times B}$ and $\delta_{p_f}^{(1)}$ corresponds to $S_J\left[p_f\right]X^{(1)}$ for all $p_f \in P$. For each p_f, the backpropagation process described for the CONV-layer in Chapter 2 is applied for each operator in the windowed scattering propagator, $S_J\left[p_f\right]$. This requires the retention of the intermediate outputs from the operator $U[\lambda_n]$ along the path p_f, $Y_{p_f,n}$. For notational convenience, $Y_{p_f,n}$, where $n \in \mathbb{N}$, is defined as

$$Y_{p_f,n} = \begin{cases} X^{(1)} & n = 0 \\ U[\lambda]Y_{p_f,n-1} & 1 \le n \le \left|p_f\right| \end{cases} \tag{5-1}$$

and the intermediate error, $\delta_{p_f,n}^*$, is defined as

$$\delta_{p_f,n}^* = \begin{cases} \delta_{p_f}^{(1)} & n = \left|p_f\right| \\ \left(\delta_{p_f,n}^* \otimes \bar{1}_d\right) * \mathrm{rot}_{180}(\Omega_n) & 1 \le n < \left|p_f\right|, \\ \delta_{p_f}^{(0)} & n = 0 \end{cases} \tag{5-2}$$

where $\left(\delta^*_{p_f,n} \otimes \bar{1}_d\right) * \mathrm{rot}_{180}(\Omega_n)$ is the backpropagation described by (2-14) and (2-15), and Ω_n defines the windowing function, ϕ_J, for $n = |p_f|$ and the scattering wavelet, ψ_{λ_n}, for $1 \leq n < |p_f|$. Using this notation, the filter updates in (2-16) can be rewritten in terms of the WSN for $n \neq 0$:

$$\Delta\Omega_n = \mathrm{rot180}\left(Y_{p_f,n} * \mathrm{rot180}\left(\delta^*_{p_f,n} \otimes \bar{1}_d\right)\right). \tag{5-3}$$

While (5-3) provides the general method of determining filter updates, the actual process is complicated by the representation of filters at multiple resolutions and the reuse of filters among different scattering paths, as well as the desire to retain the wavelet structure of the filters (see Chapter 3). For each operation along the scattering propagator, $U[p]X$, backpropagation through a ψ_λ is preceded by the backpropagation through the modulus function:

$$f(z) = \sqrt{\Re\{z\}^2 + \Im\{z\}^2}, \tag{5-4}$$

where $\Re\{\cdot\}$ and $\Im\{\cdot\}$ denote the real and imaginary components, and $z \in \mathbb{C}$. Because (5-4) is complex differentiable, the error backpropagated to the filter is

$$\delta^{(f-1)} = \frac{\Re\{z\}\Re'^{\{z\}} + \Im\{z\}\Im'\{z\}}{\sqrt{\Re\{z\}^2 + \Im\{z\}^2}}\delta^{(f)} \tag{5-5}$$

The remainder of this section discusses methods for balancing these concerns.

Weight Updates

Due to the implementation of the filters at different resolutions, the method of updates in (5-3) must be modified slightly to accommodate the subsampled filter representations. Updates to each filter along the propagator $S_J[p_f]$ occur at potentially different resolutions:

$$\Delta\Omega_n^{(r)} = \mathrm{rot}_{180}\left(Y_{p_f,n} * \mathrm{rot}_{180}\left(\delta_{p_f,n}^{*} \otimes \bar{1}_d\right)\right). \tag{5-6}$$

Updates to the individual resolutions of each filter are referred to as weight updates, and AWSNs implementing this method will be denoted as AWSN-W. As previously mentioned in Chapter 2, multiple references to each filter and its resolutions may occur for $M > 1$. Applying updates after each calculation of $\Delta\Omega_n^{(r)}$ would change the coefficients of $\Omega_n^{(r)}$ for each scattering order, thereby affecting the backpropagation of error through earlier uses. To prevent modifications to the backpropagation algorithm of a CNN, the updates are averaged and applied after $\delta_{p_f}^{(0)}$ has been computed for all $p_f \in P$.

The coefficient updates are only applied to the coefficients of the filters at various resolutions, potentially resulting in a non-wavelet function. Even if the resulting filters were wavelet functions, the updates to the multiple resolutions of Ω_n may correspond to different wavelets, e.g. different wavelet scales or orientations. In the next section, a method of mapping wavelets is described to counter these contingencies.

Wavelet Updates

To retain the properties of the scattering wavelets sought by the WSN, the coefficient updates must be mapped to the original resolution of the filter ($r = 0$). For a scattering wavelet,

such as the Morlet wavelet in (2-29), the updates to the daughter wavelets must be up-sampled and rotated such that they correspond to a single scale of the wavelet representation at $r = 0$. The up-sampling must account for both the change in resolution as well as the scaling of the daughter wavelet. Rather than scaling to the mother wavelet ($j = 0$), it is more convenient to scale to the maximum scale ($j = J - 1$). Therefore, the entire up-sampling process may be summarized as

$$\Delta\psi_{J-1,\theta}^{(0)}(Da, Db) = \Delta\psi_{j,\theta}^{(r)}(a, b), \tag{5-7}$$

where a and b are indices, and $D = 2^{(r+J-j-1)}$. For $\theta \in \{\ell\pi/L\}_{\ell=0}^{L-1}$, rotation to a single orientation is most easily accomplished when the orientation is $\theta = 0$. The calculation for one update then becomes

$$\Delta\psi_{J-1,0}^{(0)}(Da, Db) = \mathrm{rot}_{-\theta}\left(\Delta\psi_{j,\theta}^{(r)}(Da, Db)\right). \tag{5-8}$$

For non-directional filters, such as the Gaussian windowing function of (2-30), only up-sampling must be performed:

$$\Delta\phi_{J}^{(0)}(2^r a, 2^r b) = \Delta\phi_{J}^{(r)}(a, b). \tag{5-9}$$

Subsequent to the scale and orientation mapping process, the application of the updates must be applied to $\psi_{J-1,0}^{(0)}$ and $\phi_J^{(0)}$, such that the resulting functions retain the same underlying shape of the mother wavelet and windowing function, respectively. Future work should explore a wavelet mapping function . However, for the purposes of this thesis, particle swarm optimization (PSO) is applied to the updated coefficients of the filter to determine a set of learnable parameters in the wavelet and windowing functions. AWSNs that utilize PSO for their parameter adaptation will be denoted as AWSN-PSO.

The FEAT-Layer

There are no learnable parameters in the FEAT-layer. Instead, it reformats the output from the WSN-layer to fit the input to the FC-layer. Recall that the input to the FEAT-layer, $X^{(2)} = \left(S_J[p_f]X^{(1)}\right)_{f=1}^{F}$, where $S_J[p_f]X^{(1)} \in \mathbb{R}^{A\times B}$. In Chapter 3, the feature maps were formatted into a feature vector, $\vec{a} \in \mathbb{R}^{C}$, such that each element of $\vec{a}$, $a_n = \sum_{y\in S[p_n]X} y$. This method does not translate to the backpropagation processing implemented in this AWSN. Instead, the feature maps are reshaped and concatenated, such that the output from the layer,

$$\begin{aligned} X^{(3)} &= \text{cat}\left(a_1, a_2, \dots, a_f, \dots a_F\right) \in \mathbb{R}^{F\cdot A\cdot B} \\ a_f &= \text{vec}\left(X^{(2)}(f)\right), \end{aligned} \tag{5-10}$$

where $\text{vec}(\cdot)$ is the vectorization of a matrix. For backpropagation through the FEAT-layer, the error, $\delta^{(2)} \in \mathbb{R}^{F\cdot A\cdot B}$, must be formatted into a set of errors, $\delta^{(1)} = \left(\delta_{p_1}^{(1)}, \delta_{p_2}^{(1)}, \dots, \delta_{p_F}^{(1)}\right)$, corresponding to each feature map:

$$\delta_{p_f}^{(1)} = \text{vec}^{-1}(\delta^{(2)}(1 + (f-1)AB, fAB). \tag{5-11}$$

Chapter 6

Experiments: Adaptive Wavelet Scattering Network

Description

Using a 2-D Morlet filter bank, the performance of an AWSN with weight updates (AWSN-W) and an AWSN with PSO updates (AWSN-PSO) were evaluated for different combinations of M, J, and L. To provide a more direct comparison to CNNs, the input was padded with zeros to the filter size, rather than potentially reducing any boundary effects via a symmetric boundary condition, as was done in Chapter 4. In addition, in order to accommodate the potentially non-periodic updates to the AWSN-W, the periodized Fourier method of convolution default in ScatNet was modified to a Fourier domain operation.

The combinations of M, J, and L implemented in this chapter are provided in Table **6-1**. Three and five scales were used to understand the effect of both the scale, as well as the network complexity. Both scales were used for $M = 0$ networks to better understand the effect of the scale in ϕ_J on the output, as they should behave as CONV-layers with one filter. For $M > 0$, differing J and L were implemented to explore the effect of network complexity, particularly with the application of the updates during backpropagation. Note that in the ScatNet framework, at $M = 1$, all daughter wavelets, ψ_λ, are utilized once at $r = 0$, and the windowing function, ϕ_J, at multiple resolutions depending upon the value of J; at $M = 2$, all ψ_λ with $j > 2$ are utilized at more than one resolution.

Table **6-1**: Number of Paths in a WSN

M	J	L	Number of Paths
0	-	-	1
1	3	1	4
1	3	2	7
1	3	8	25
1	5	1	6
1	5	2	11
1	5	8	41
2	3	1	7
2	3	2	19
2	5	1	16
2	5	2	51
2	5	8	681

The AWSN-PSO architectures were updated using the mean-squared-error (MSE) of a daughter wavelet at a particular scale and orientation (see Chapter 5) and the father wavelet, ϕ. Their initial parameter values and bounds are provided in Table **6-2**. Note that σ_ϕ and σ_ψ were trained separately; their maximum values were determined as a function of the filter and input sizes,

$$\sigma = 2^{-J-1}(D_\Omega - D_X). \tag{6-1}$$

The filter size varied with J (for $J = 3$, the filter size was 120; for $J = 5$, the filter size was 160), to help prevent border effects.

Table **6-2**: Morlet Wavelet and Gaussian Envelope Parameter Initializations and Bounds

Parameter	σ $J = 3$	σ $J = 5$	s	ξ
Initial	0.8	0.8	$4/L$	2.356
Minimum	0.1	0.1	0.017	0.1
Maximum	3.5	1.5	∞	∞

For benchmarking their performance, the AWSN-W and AWSN-PSO were compared to other feature extraction methods implementing a combination of static WSNs and FC-layers (WSN-FC), static WSNs and an SVM classifier (WSN-SVM), as well as an FC-layer only CNN (FC). The WSN-based networks employed the same 2-D Morlet wavelets and Gaussian windowing functions. All networks ending with an FC-layer used the SoftMax classifier and

negative log loss for backpropagation; updates during backpropagation were made using SGD without regularization.

The networks trained on the MSTAR dataset described in Chapter 4 without the synthetic target, SLICY, to provide a more uniform level of confusion. Three-fold validation and balanced accuracy were used, again. Each network was trained for up to 100 epochs; for those networks training via backpropagation, the learning rates were determined experimentally.

Results

An overview of the validation accuracy for the WSN-based networks is provided in Table **6-3**; the average accuracy for the FC network was 0.909. A complete table of the training and validation information is provided in the appendix.

Table **6-3**: Average Validation Accuracies for All Networks

M	***J***	***L***	**AWSN-W**	**AWSN-PSO**	**WSN-SVM**	**WSN-FC**
0	3	-	0.914	0.714	0.606	0.906
0	5	-	0.612	0.672	0.594	0.709
1	3	1	0.716	0.743	0.642	0.923
1	3	2	0.726	0.752	0.706	0.949
1	3	8	0.811	0.731	0.735	0.957
1	5	1	0.508	0.699	0.722	0.762
1	5	2	0.721	0.712	0.806	0.824
1	5	8	0.698	0.706	0.869	0.818
2	3	1	0.561	0.733	0.648	0.924
2	3	2	0.534	0.747	0.719	0.950
2	3	8	TBD	TBD	TBD	TBD
2	5	1	0.522	0.710	0.765	0.773
2	5	2	0.549	0.710	0.901	0.855
2	5	8	TBD	TBD	TBD	TBD

AWSN-W

For $M = 0$, the best performers were the AWSN-W with $J = 3$ (0.914) and the FC-layer (0.909); however, for $J = 5$, the AWSN-W yielded a 0.612 accuracy, outperforming only the

WSN-SVM. The lesser performance of $J = 5$ in comparison to $J = 3$ likely indicates that the blurring occurring in the first path of the network is masking identifying features of the target, resulting in poor initial updates. Figure **6-1** provides an example from the third fold of training. After one epoch, ϕ_3 retained its general shape, though not perfectly; for ϕ_5, the Gaussian was no longer discernable. After training, the general shapes of both were gone, though the updates to ϕ_3 appear more uniform.

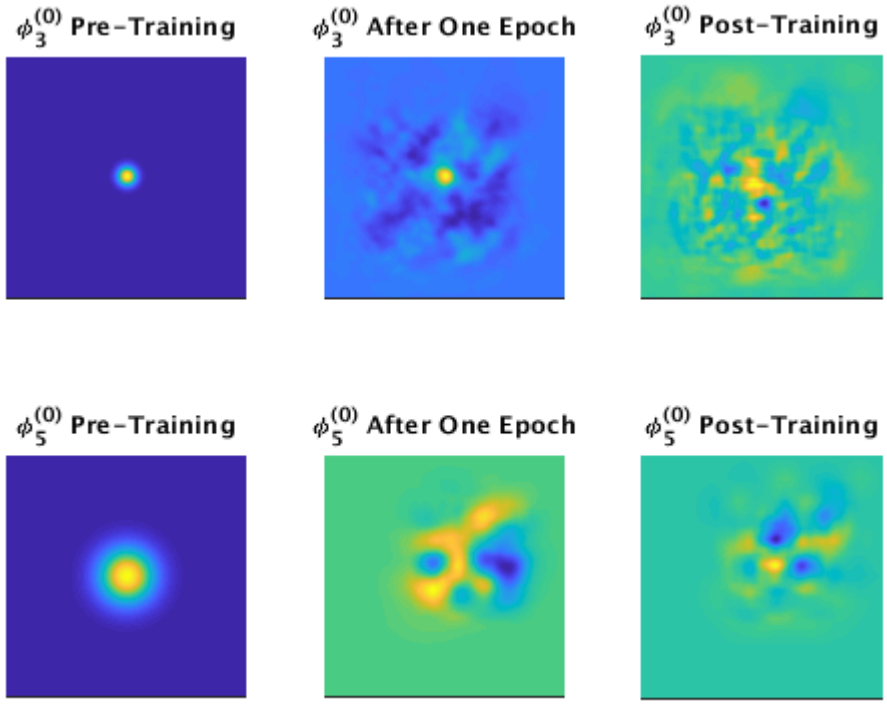


Figure **6-1**: Filters $\phi_3^{(0)}$ and $\phi_5^{(0)}$ before training, after one epoch, and after training (third fold).

For $M = 1$ and $L = 1$ or $L = 8$, the $J = 3$ networks again outperformed the $J = 5$ networks; however, their performance was comparable at $L = 2$. At $M = 2$, though performance degraded for the tested networks, the performance was again comparable. These results indicate that the network complexity can both improve and degrade the results of backpropagation, as applied in this thesis.

AWSN-PSO

The AWSN-PSO underperformed the AWSN-W for all $M = 0$ and $J = 3$, though this appears a function of worsened performance of AWSN-W with $J = 5$; however, the AWSN-PSO performed comparably among all combinations of M, L, and J, yielding a mean accuracy of 0.737.

The trained parameter values for each fold of the AWSN-PSO networks are provided in Table **6-4**. Individually, the parameters have no discernable pattern to their behaviors, varying greatly even among folds. Given the relatively consistent accuracy, there are likely tradeoffs among individual class accuracies. In future work, more folds should be used for validation. Additionally, the implementation of symmetric padding instead of zero padding may improve results.

Table **6-4**: Trained AWSN-PSO Parameters

			σ_ϕ		σ_ψ		s		ξ	
M	L	Fold No.	$J=3$	$J=5$	$J=3$	$J=5$	$J=3$	$J=5$	$J=3$	$J=5$
0	-	1	3.50	0.10	-	-	-	-	-	-
0	-	2	3.50	0.10	-	-	-	-	-	-
0	-	3	0.72	0.10	-	-	-	-	-	-
1	1	1	0.37	0.10	3.35	1.50	676.45	0.02	539.53	0.10
1	1	2	1.26	0.10	3.50	1.50	5093.59	9339.77	457.71	4271.33
1	1	3	0.52	0.14	2.16	1.12	0.02	4177.66	2896.22	0.10
1	2	1	0.56	0.12	2.66	0.34	2651.73	8214.51	1048.90	0.10
1	2	2	1.34	0.11	3.50	1.50	241.92	2.38	1344.40	0.02
1	2	3	0.52	0.10	3.50	1.08	4096.99	0.02	0.10	904.48
1	8	1	0.72	0.15	0.61	1.10	977.26	1286.03	1672.85	212.76
1	8	2	3.50	0.25	3.50	0.46	4246.95	1362.46	0.02	1671.89
1	8	3	0.95	0.17	0.10	0.19	0.02	1443.13	0.10	747.55
2	1	1	0.41	0.10	0.26	1.13	311.53	0.02	1655.09	0.10
2	1	2	1.19	0.10	2.58	1.50	3416.82	2094.15	1146.07	3065.50
2	1	3	0.51	0.10	3.50	0.48	690.98	0.02	527.48	0.10
2	2	1	0.62	0.14	3.50	1.50	0.02	0.02	50.47	0.10
2	2	2	0.87	0.11	1.29	1.50	4570.17	116.56	3348.56	0.02
2	2	3	0.73	0.17	3.50	1.50	0.02	0.02	0.10	654.51
2	8	1	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD
2	8	2	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD
2	8	3	TBD	TBD	TBD	TBD	TBD	TBD	TBD	TBD

The finitely bounded parameters, σ_ϕ and σ_ψ, however, showed a tendency toward their extrema. This was particularly true for $J = 5$, where all σ_ϕ tend toward the lower bound. A

tendency toward a smaller σ_{ϕ} may indicate correction from over-blurring experienced in the initial stages of training seen in the AWSN-W.

The value of σ_{ψ} tended more toward the upper bound and middle range of values, though their changes did not appear highly correlated to the accuracy. Given the drastic effect of ϕ_J for $J = 5$, future work might combine the two spread parameters and calculate an update with the MSE for the Morlet wavelet function only. This could potentially reduce the drastic changes observed in the other parameters, s and ξ. Note that some values of ξ exceed the lower boundary; an earlier implementation of the code may have been used; these cases should be retested.

The apparent cap on accuracy in the AWSN-PSO networks may derive from the complexity with which the parameter updates were implemented. By receiving updates at multiple resolutions across multiple scales and orientations, an averaging affect may occur, preventing progress past a certain value when individual updates have highly conflicting values. Revisiting the method by which the parameters update would be prudent in future work.

Still, another cause may be the lack of regularization in the SGD update equation, which is a common practice in neural networks, as it can reduce drastic changes to coefficients.

Chapter 7

Conclusion

The similar functionality of a WSN to a CNN was explored in this thesis. First, the effect of quantization on a WSN with multi-class SVM was discussed and evaluated. The WSN showed robustness to decreasing numbers of quantization levels. The k-means and the quantile quantization schemes were leading performers. Though the RNG-PDF-based quantization schemes yielded disappointing results, the inclusion of data-driven information in the selection of quantization levels provides opportunity for future work.

Following was the development of an AWSN using the backpropagation techniques of a CNN. Though the results were not particularly impressive, several avenues for their development were noted, including the methods by which the parameters update. At present, updates from each scaled and oriented filter are mapped onto a single daughter wavelet via up-sampling and matrix rotation for the implementation of the PSO algorithm, which yielded a static accuracy of approximately 0.7 regardless of the network complexity. In the future, updates to the mother wavelet should be explored, via existing methods, as well as methods that project the updates into a wavelet space. Additionally, the spread parameters of the scattering wavelet and the gaussian window might be updated as a shared value.

References

[1] A. Profeta, A. Rodriguez, and H. S. Clouse, “Convolutional neural networks for synthetic aperture radar classification,” in *Proc. of SPIE Algorithms for Synthetic Aperture Radar Imagery XXIII*, 2016, vol. 9843, p. 98430M.

[2] R. J. Soldin, D. N. MacDonald, M. D. Reisman, L. R. Konz, R. Rouse, and T. L. Overman, “HySARNet: a Hybrid machine learning approach to Synthetic Aperture Radar automatic target recognition,” in *Automatic Target Recognition XXIX*, 2019, vol. 10988, no. May.

[3] J. Shao, C. Qu, and J. Li, “A performance analysis of convolutional neural network models in SAR target recognition,” in *2017 SAR in Big Data Era: Models, Methods and Applications (BIGSARDATA)*, 2017, no. 188, pp. 1–6.

[4] D. A. E. Morgan, “Deep convolutional neural networks for ATR from SAR imagery,” in *Proc. SPIE Algorithms for Synthetic Aperture Radar Imagery XXII*, 2015, vol. 9475, p. 94750F.

[5] M. Cha, A. Majumdar, H. T. Kung, and J. Barber, “Improving Sar Automatic Target Recognition Using Simulated Images Under Deep Residual Refinements,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, vol. 2018-April, pp. 2606–2610.

[6] M. R. Fox and R. M. Narayanan, “Application and performance of convolutional neural networks to SAR,” in *Proc. of SPIE Radar Sensor Technology XXII*, 2018, vol. 1063304.

[7] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, “Fixed Point Quantization of Deep Convolutional Networks,” in *Proceedings of the 33rd International Conference on Machine Learning*, 2015.

[8] B. Jacob *et al.*, “Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 2704–2713, 2018.

[9] J. Bruna and S. Mallat, “Invariant Scattering Convolution Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1872–1886, Aug. 2013.

[10] S. Mallat, “Group Invariant Scattering,” *Commun. Pure Appl. Math.*, vol. 65, no. 10, pp. 1331–1398, Oct. 2012.

[11] E. Oyallon, S. Mallat, and L. Sifre, “Generic Deep Networks with Wavelet Scattering,” 2013.

[12] B. Soro and C. Lee, “A wavelet scattering feature extraction approach for deep neural network based indoor fingerprinting localization,” *Sensors (Switzerland)*, vol. 19, no. 8, 2019.

[13] Q. Xiao, G. Ge, and J. Wang, “The neural network adaptive filter model based on wavelet transform,” *Proc. - 2009 9th Int. Conf. Hybrid Intell. Syst. HIS 2009*, vol. 1, no. 1, pp. 529–534, 2009.

[14] H. Xiong, T. Zhang, and Y. S. Moon, “A translation- and scale-invariant adaptive wavelet transform,” *IEEE Trans. Image Process.*, vol. 9, no. 12, pp. 2100–2108, 2000.

[15] H. H. Szu, “Why adaptive wavelet transform?,” in *Visual Information Processing II*, 1993, vol. 1961, no. August 1993, pp. 280–292.

[16] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Work. Track Proc.*, pp. 1–14, 2015.

[17] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, vol. 2016-Decem, pp. 770–778.

[18] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[19] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[20] A. Krizhevsky, V. Nair, and G. Hinton, “CIFAR-10 (Canadian Institute for Advanced Research).” [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html.

[21] C. Szegedy *et al.*, “Going deeper with convolutions,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 1–9, 2015.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[23] J. Andén, L. Sifre, M. Kapoko, E. Oyallon, and V. Lostanlen, “ScatNet.” 2013.

[24] M. Andreux *et al.*, “Kymatio: Scattering Transforms in Python,” no. 2012, pp. 2012–2017, 2018.

[25] D. Arthur and S. Vassilvitskii, “K-means++: The Advantages of Careful Seeding,” in *SODA ’07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete Algorithms*, 2007, pp. 1027–1035.

[26] J. R. Michael, W. R. Schucany, and R. W. Haas, “Generating random variates using transformations with multiple roots,” *Am. Stat.*, vol. 30, no. 2, pp. 88–90, 1976.

[27] G. Marsaglia and W. W. Tsang, “A simple method for generating gamma variables,” *ACM Trans. Math. Softw.*, vol. 26, no. 3, pp. 363–372, Sep. 2000.

[28] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, “The balanced accuracy and its posterior distribution,” *Proc. - Int. Conf. Pattern Recognit.*, pp. 3121–3124, 2010.

Appendix

Network Training Results for AWSN Experiments

The training information for all networks used in Chapter 6 is provided in Table **A-1**.

Table **A-1**: Training Options for Backpropagation-Based Networks

Network	M	J	L	Learn Rate	Fold No.	Epoch	Training Accuracy	Validation Accuracy
AWSN-PSO	0	3	-	4.00E-01	1	88	0.649	0.712
AWSN-PSO	0	3	-	4.00E-01	2	73	0.645	0.690
AWSN-PSO	0	3	-	4.00E-01	3	89	0.659	0.739
AWSN-PSO	1	3	1	4.00E-01	1	99	0.664	0.746
AWSN-PSO	1	3	1	4.00E-01	2	93	0.649	0.735
AWSN-PSO	1	3	1	4.00E-01	3	79	0.654	0.747
AWSN-PSO	1	3	2	4.00E-01	1	95	0.658	0.797
AWSN-PSO	1	3	2	4.00E-01	2	83	0.628	0.737
AWSN-PSO	1	3	2	4.00E-01	3	62	0.687	0.721
AWSN-PSO	1	3	8	4.00E-01	1	92	0.669	0.750
AWSN-PSO	1	3	8	4.00E-01	2	97	0.687	0.714
AWSN-PSO	1	3	8	4.00E-01	3	84	0.627	0.731
AWSN-PSO	2	3	1	4.00E-01	1	75	0.648	0.731
AWSN-PSO	2	3	1	4.00E-01	2	83	0.641	0.729
AWSN-PSO	2	3	1	4.00E-01	3	57	0.649	0.739
AWSN-PSO	2	3	2	4.00E-01	1	80	0.674	0.732
AWSN-PSO	2	3	2	4.00E-01	2	95	0.673	0.783
AWSN-PSO	2	3	2	4.00E-01	3	76	0.658	0.727
AWSN-PSO	2	3	8	TBD	1	TBD	TBD	TBD
AWSN-PSO	2	3	8	TBD	2	TBD	TBD	TBD
AWSN-PSO	2	3	8	TBD	3	TBD	TBD	TBD
AWSN-PSO	0	5	-	4.00E-01	1	73	0.564	0.662
AWSN-PSO	0	5	-	4.00E-01	2	81	0.608	0.688
AWSN-PSO	0	5	-	4.00E-01	3	35	0.500	0.667
AWSN-PSO	1	5	1	4.00E-01	1	99	0.514	0.681
AWSN-PSO	1	5	1	4.00E-01	2	87	0.565	0.706
AWSN-PSO	1	5	1	4.00E-01	3	96	0.594	0.711
AWSN-PSO	1	5	2	4.00E-01	1	66	0.519	0.693
AWSN-PSO	1	5	2	4.00E-01	2	90	0.580	0.736
AWSN-PSO	1	5	2	4.00E-01	3	98	0.536	0.708
AWSN-PSO	1	5	8	4.00E-01	1	54	0.586	0.698
AWSN-PSO	1	5	8	4.00E-01	2	79	0.554	0.722
AWSN-PSO	1	5	8	4.00E-01	3	75	0.610	0.699
AWSN-PSO	2	5	1	4.00E-01	1	63	0.537	0.686

AWSN-PSO	2	5	1	4.00E-01	2	95	0.604	0.749
AWSN-PSO	2	5	1	4.00E-01	3	77	0.547	0.696
AWSN-PSO	2	5	2	4.00E-01	1	91	0.604	0.703
AWSN-PSO	2	5	2	4.00E-01	2	71	0.559	0.728
AWSN-PSO	2	5	2	4.00E-01	3	76	0.571	0.699
AWSN-PSO	2	5	8	TBD	1	TBD	TBD	TBD
AWSN-PSO	2	5	8	TBD	2	TBD	TBD	TBD
AWSN-PSO	2	5	8	TBD	3	TBD	TBD	TBD
AWSN-W	0	3	-	8.00E-03	1	100	0.811	0.919
AWSN-W	0	3	-	8.00E-03	2	97	0.779	0.914
AWSN-W	0	3	-	8.00E-03	3	99	0.773	0.910
AWSN-W	1	3	1	4.00E-04	1	97	0.509	0.715
AWSN-W	1	3	1	4.00E-04	2	100	0.505	0.711
AWSN-W	1	3	1	4.00E-04	3	95	0.491	0.721
AWSN-W	1	3	2	4.00E-04	1	97	0.500	0.707
AWSN-W	1	3	2	4.00E-04	2	99	0.500	0.723
AWSN-W	1	3	2	4.00E-04	3	99	0.500	0.748
AWSN-W	1	3	8	1.00E-03	1	98	0.509	0.806
AWSN-W	1	3	8	1.00E-03	2	92	0.587	0.800
AWSN-W	1	3	8	1.00E-03	3	77	0.519	0.826
AWSN-W	2	3	1	1.00E-04	1	15	0.514	0.562
AWSN-W	2	3	1	1.00E-04	2	11	0.519	0.565
AWSN-W	2	3	1	1.00E-04	3	17	0.522	0.556
AWSN-W	2	3	2	1.00E-04	1	7	0.519	0.537
AWSN-W	2	3	2	1.00E-04	2	4	0.519	0.531
AWSN-W	2	3	2	1.00E-04	3	2	0.540	0.533
AWSN-W	2	3	8	TBD	1	TBD	TBD	TBD
AWSN-W	2	3	8	TBD	2	TBD	TBD	TBD
AWSN-W	2	3	8	TBD	3	TBD	TBD	TBD
AWSN-W	0	5	-	4.00E-04	1	99	0.507	0.615
AWSN-W	0	5	-	4.00E-04	2	98	0.520	0.624
AWSN-W	0	5	-	4.00E-04	3	92	0.512	0.598
AWSN-W	1	5	1	1.00E-04	1	53	0.500	0.507
AWSN-W	1	5	1	1.00E-04	2	1	0.500	0.501
AWSN-W	1	5	1	1.00E-04	3	54	0.500	0.517
AWSN-W	1	5	2	4.00E-04	1	89	0.500	0.727
AWSN-W	1	5	2	4.00E-04	2	75	0.500	0.711
AWSN-W	1	5	2	4.00E-04	3	69	0.534	0.726
AWSN-W	1	5	8	4.00E-04	1	100	0.500	0.662
AWSN-W	1	5	8	4.00E-04	2	100	0.500	0.693
AWSN-W	1	5	8	4.00E-04	3	99	0.500	0.740
AWSN-W	2	5	1	1.00E-04	1	99	0.500	0.521
AWSN-W	2	5	1	1.00E-04	2	94	0.500	0.525
AWSN-W	2	5	1	1.00E-04	3	98	0.500	0.521
AWSN-W	2	5	2	1.00E-04	1	26	0.500	0.546
AWSN-W	2	5	2	1.00E-04	2	28	0.500	0.556
AWSN-W	2	5	2	1.00E-04	3	28	0.500	0.544
AWSN-W	2	5	8	TBD	1	TBD	TBD	TBD
AWSN-W	2	5	8	TBD	2	TBD	TBD	TBD

AWSN-W	2	5	8	TBD	3	TBD	TBD	TBD
FC	-	-	-	4.00E-01	1	68	0.952	0.906
FC	-	-	-	4.00E-01	2	94	0.954	0.921
FC	-	-	-	4.00E-01	3	59	0.916	0.901
WSN-FC	0	3	-	4.00E-01	1	100	0.782	0.903
WSN-FC	0	3	-	4.00E-01	2	97	0.816	0.909
WSN-FC	0	3	-	4.00E-01	3	98	0.800	0.907
WSN-FC	1	3	1	4.00E-01	1	92	0.859	0.918
WSN-FC	1	3	1	4.00E-01	2	95	0.838	0.925
WSN-FC	1	3	1	4.00E-01	3	96	0.819	0.928
WSN-FC	1	3	2	4.00E-01	1	99	0.775	0.938
WSN-FC	1	3	2	4.00E-01	2	97	0.796	0.952
WSN-FC	1	3	2	4.00E-01	3	100	0.783	0.955
WSN-FC	1	3	8	4.00E-01	1	100	0.795	0.949
WSN-FC	1	3	8	4.00E-01	2	88	0.764	0.958
WSN-FC	1	3	8	4.00E-01	3	100	0.789	0.964
WSN-FC	2	3	1	4.00E-01	1	91	0.832	0.918
WSN-FC	2	3	1	4.00E-01	2	98	0.782	0.925
WSN-FC	2	3	1	4.00E-01	3	96	0.815	0.928
WSN-FC	2	3	2	4.00E-01	1	100	0.781	0.939
WSN-FC	2	3	2	4.00E-01	2	96	0.749	0.955
WSN-FC	2	3	2	4.00E-01	3	100	0.802	0.956
WSN-FC	2	3	8	TBD	1	TBD	TBD	TBD
WSN-FC	2	3	8	TBD	2	TBD	TBD	TBD
WSN-FC	2	3	8	TBD	3	TBD	TBD	TBD
WSN-FC	0	5	-	4.00E-01	1	99	0.673	0.715
WSN-FC	0	5	-	4.00E-01	2	100	0.642	0.694
WSN-FC	0	5	-	4.00E-01	3	99	0.664	0.718
WSN-FC	1	5	1	4.00E-01	1	100	0.699	0.759
WSN-FC	1	5	1	4.00E-01	2	99	0.687	0.756
WSN-FC	1	5	1	4.00E-01	3	98	0.724	0.771
WSN-FC	1	5	2	4.00E-01	1	99	0.612	0.810
WSN-FC	1	5	2	4.00E-01	2	96	0.630	0.823
WSN-FC	1	5	2	4.00E-01	3	98	0.660	0.839
WSN-FC	1	5	8	4.00E-01	1	58	0.637	0.787
WSN-FC	1	5	8	4.00E-01	2	96	0.656	0.831
WSN-FC	1	5	8	4.00E-01	3	99	0.638	0.837
WSN-FC	2	5	1	4.00E-01	1	97	0.708	0.775
WSN-FC	2	5	1	4.00E-01	2	100	0.702	0.766
WSN-FC	2	5	1	4.00E-01	3	97	0.691	0.777
WSN-FC	2	5	2	4.00E-01	1	100	0.723	0.823
WSN-FC	2	5	2	4.00E-01	2	100	0.678	0.860
WSN-FC	2	5	2	4.00E-01	3	100	0.682	0.883
WSN-FC	2	5	8	TBD	1	TBD	TBD	TBD
WSN-FC	2	5	8	TBD	2	TBD	TBD	TBD
WSN-FC	2	5	8	TBD	3	TBD	TBD	TBD
WSN-SVM	0	3	-	-	1	-	0.603	0.604
WSN-SVM	0	3	-	-	2	-	0.609	0.597
WSN-SVM	0	3	-	-	3	-	0.597	0.617

WSN-SVM	1	3	1	-	1	-	0.649	0.651
WSN-SVM	1	3	1	-	2	-	0.650	0.635
WSN-SVM	1	3	1	-	3	-	0.650	0.641
WSN-SVM	1	3	2	-	1	-	0.733	0.708
WSN-SVM	1	3	2	-	2	-	0.725	0.704
WSN-SVM	1	3	2	-	3	-	0.720	0.708
WSN-SVM	1	3	8	-	1	-	0.774	0.729
WSN-SVM	1	3	8	-	2	-	0.769	0.728
WSN-SVM	1	3	8	-	3	-	0.764	0.748
WSN-SVM	2	3	1	-	1	-	0.655	0.645
WSN-SVM	2	3	1	-	2	-	0.668	0.644
WSN-SVM	2	3	1	-	3	-	0.659	0.655
WSN-SVM	2	3	2	-	1	-	0.746	0.722
WSN-SVM	2	3	2	-	2	-	0.744	0.716
WSN-SVM	2	3	2	-	3	-	0.741	0.718
WSN-SVM	2	3	8	-	1	-	TBD	TBD
WSN-SVM	2	3	8	-	2	-	TBD	TBD
WSN-SVM	2	3	8	-	3	-	TBD	TBD
WSN-SVM	0	5	-	-	1	-	0.598	0.590
WSN-SVM	0	5	-	-	2	-	0.612	0.594
WSN-SVM	0	5	-	-	3	-	0.602	0.598
WSN-SVM	1	5	1	-	1	-	0.737	0.721
WSN-SVM	1	5	1	-	2	-	0.728	0.727
WSN-SVM	1	5	1	-	3	-	0.729	0.718
WSN-SVM	1	5	2	-	1	-	0.831	0.795
WSN-SVM	1	5	2	-	2	-	0.815	0.815
WSN-SVM	1	5	2	-	3	-	0.823	0.807
WSN-SVM	1	5	8	-	1	-	0.918	0.860
WSN-SVM	1	5	8	-	2	-	0.899	0.886
WSN-SVM	1	5	8	-	3	-	0.907	0.863
WSN-SVM	2	5	1	-	1	-	0.799	0.763
WSN-SVM	2	5	1	-	2	-	0.791	0.761
WSN-SVM	2	5	1	-	3	-	0.786	0.771
WSN-SVM	2	5	2	-	1	-	0.943	0.896
WSN-SVM	2	5	2	-	2	-	0.929	0.902
WSN-SVM	2	5	2	-	3	-	0.926	0.903
WSN-SVM	2	5	8	-	1	-	TBD	TBD
WSN-SVM	2	5	8	-	2	-	TBD	TBD
WSN-SVM	2	5	8	-	3	-	TBD	TBD