8-6-2021

Exploring the use of neural network-based band selection on hyperspectral imagery to identify informative wavelengths for improving classifier task performance

Preston Chandler Darling
preston.c.darling@gmail.com

Exploring the use of neural network-based band selection on hyperspectral imagery to identify informative wavelengths for improving classifier task performance

By

Preston Chandler Darling

Approved by:

John E. Ball (Major Professor)
Ali C. Gurbuz
Stanton R. Price
Qian (Jenny) Du (Graduate Coordinator)
Jason M. Keith (Dean, Bagley College of Engineering)

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical Engineering
in the Electrical & Computer Engineering Department

Mississippi State, Mississippi

August 2021

Name: Preston Chandler Darling

Date of Degree: August 6, 2021

Institution: Mississippi State University

Major Field: Electrical Engineering

Major Professor: John E. Ball

Title of Study: Exploring the use of neural network-based band selection on hyperspectral imagery to identify informative wavelengths for improving classifier task performance

Pages of Study: 78

Candidate for Degree of Master of Science

Hyperspectral imagery is a highly dimensional type of data resulting in high computational costs during analysis. Band selection aims to reduce the original hyperspectral image to a smaller subset that reduces these costs while preserving the maximum amount of spectral information within the data. This thesis explores various types of band selection techniques used in hyperspectral image processing. Modifying Neural network-based techniques and observing the effects on the band selection process due to the change in network architecture or objective are of particular focus in this thesis. Herein, a generalized neural network-based band selection technique is developed and compared to state-of-the-art algorithms that are applied to a unique dataset and the Pavia City Center dataset where the subsequent selected bands are fed into a classifier to gather comparison results.

Key words: Remote Sensing, Machine Learning, Hyperspectral Imaging, Band Selection

DEDICATION

To my wife, Ali. Without you, I would not be where I am today. You inspire me to be the best I can be. Thank you for being there every step of the way.

Thanks to my major professor, Dr. John Ball, as well as my committee members Dr. Stanton Price and Dr. Ali Gurbuz for taking the time to help me grow as an engineer throughout this journey.

Thanks to my friends Jarrett, Brandon, and Simmers for spending too many nights rambling about this topic. It really helped shape this thesis, and I could not appreciate you all more for that help.

ACKNOWLEDGEMENTS

This work was supported in part by the Engineer Research and Development Center (ERDC) under the United States Army Corps of Engineers (USACE). The findings and opinions in this thesis belong solely to the author and are not necessarily representative of those who have supported the work.

Permission to use the unique hyperspectral data gathered for this thesis was given by the Survivability Branch of the Geotechnical and Structures Laboratory (GSL) of ERDC.

I thank my committee for their comments on this thesis, and I thank John E. Ball for directing this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Scope of the Problem

Hyperspectral images contain an abundance of data due to the nature of the modality. In general, an image has three dimensions—two spatial, one spectral. Typical multispectral images split an abritrary band of light into large spectral bins with the most familiar example being Red, Green, and Blue (RGB) images having three channels for wavelengths in the visible electromagnetic spectrum broadly corresponding to red, green, and blue light. A hyperspectral image in this same band of light would take many more slices resulting in a much finer spectral resolution and often times hundreds of "color" channels.[1] With the spectral dimesnion becoming roughly two orders of magnitude larger, the amount of computational resources required for analysis—especially in real time scenarios—can render certain applications infeasible.

Due to limitations of computational resources and consequences of situations such as the Hughes phenomenon,[2] it is desireable to reduce the amount of channels in a hyperspectral image such that a minimum amount of information is lost. The process by which the a subset of the spectral dimension of the hyperspectral image is downselected is called band selection. Band

[1] It is worth noting that it is often disagreed upon when a multispectral image can become a hyperspectral image. The simplest difference between the two is that multispectral images have a course sampling of the spectral dimension, and hyperspectral images have a fine sampling. This thesis does not make an attempt to elaborate further on this point of contention.

[2] Also known as the curse of dimensionality, this thesis uses this term to refer to the rule of thumb that a class can be discriminated by a machine learning classifier as long as 5 training examples per dimension are presented to that classifier.

selection is a heavily researched topic that will be discussed in more detail in Chapter II, but this thesis's primary concern is to explore neural network-based approaches to band selection.

There is a specific type of neural network architecture called an autoencoder whose task is to recreate the network's input at its output. In doing so, an autoencoder learns a transform that performs compression on the data. Various authors have tested using the encoding weight matrix of the autoencoder as a basis to determine which input bands were most important for compression—and through the decoding process, image reconstruction—yielding a type of band selection technique inherent to machine learning [1, 5, 9, 26]. This thesis explores if image reconstruction is the only basis by which sufficient band selection results can be computed—especially when the application of the data involves tasks such as classification. In other words, should the basis by which bands are selected be application agnostic?

1.2 Importance

As discussed in Section 1.1, band selection is important for performing dimensionality reduction (DR). For applications involving supervised learning, each dimension we keep in our hyperspectral image compounds 10 more data points onto the amount of training examples for a single class. Supposing a neural network is performing pixel classification and the data contains 10 classes, a simple RGB image would required at least $10 * 3$ or 30 training pixels per class in order to perform classification. Because there are so few dimensions, RGB images require much fewer examples in terms of raw data. In general, however, RGB images' limited spectral information makes pixel-wise classification tasks very difficult. With the spectral resolution inherent in hyperspectral imaging, the classification problem becomes both easier and harder. The network has access to

more dimensions making discrimination between different classes of pixels easier, but assuming the hyperspectral image has 200 channels, $10 * 200$ pixels are required per class in keeping with the curse of dimensionality. Band selection techniques are important not only to reduce the amount of computational resources to analyze the data but to also reduce the amount of examples a classifier would need to be sufficiently trained.

The selections produced through these techniques can give insight into some physical characteristics of the data. For example, consider an application where a hyperspectral pixel classifier needs to detect man-made materials such as concrete and steel in two scenes—one comprised mainly of vegatative materials and another comprised of mainly soil materials. Assuming the hyperspectral image is captured with a sensor tuned to the near infra-red (IR) and electro-optical (EO) bands, the vegatative-rich scene will show higher responses in wavelengths corresponding to the near IR and visually green regions[3], while the soil-rich scene will generally have duller responses across most bands except for those matching the visual color of the soil. One could imagine that the hyperspectral classifier would want the selected bands that provide the most discriminative information between the background—vegatation or soil in this example—and the target, concrete or steel. Depending on what the band selection technique considers "important", subsets that are mainly bands defining soil and vegatation can be just as valid as subsets defining maximum descripancy between vegatation and concrete. Hence, it is important to carefully consider the context under which the band selection technique is operating.

[3]About 495–570 nm

1.3 Contributions

This thesis examines different band selection techniques effects on hyperspectral pixel classification. Band selection aims to reduce data as much as possible while maintaining that amount of important information in the dataset. As alluded to in Section 1.2, different band selection techniques have different definitions of what is "important". With the data's intended purpose being hyperspectral classification, this thesis compares band selection techniques that select by maximizing statistical moments like variance and kurtosis, minimizing image reconstruction error, and maximizing class discrimination. A generic feedforward network for hyperspectral pixel classification is constructed to learn input weights for band selection similar to the techinique(s) described in [26]. To explore if the data's application should be taken into account when choosing a band selection technique, this neural network-based band selection (NNBS) technique is used as a specific comparison to observe effects when selecting bands based on minizing image reconstruction error versus maximizing classification accuracy. In addition to the Pavia City Center hyperspectral dataset, the ERDC collect a novel dataset for use in this thesis allowing for an examination of this technique's effect in heavily skewed datasets. In short, the author's contributions to this field of research include:

- Developing a new method of band selection based on deep feedforward architecture optimizing classification results.
- Demonstrating that any neural network architecture can be used to perform band selection (as opposed to being limited to convolutional neural networks and autoencoders).
- Publishing the results of the previous two contributions in an SPIE conference paper.
- Coding a custom machine learning library for more control over computations allowing for experimentation with layer-by-layer network analysis.

CHAPTER II

BACKGROUND

2.1 Hyperspectral Imaging

All imaging sensors—biological or not—operate under the same working principle: react proportionally to the amount of light reflected or transmitted towards the sensor. For man-made imagers, either a CCD-[1] or CMOS-based[2] sensor is used to convert the incident attenuation of light on each pixel of the sensor to an electric signal [15]. In color imagery, the light is separated using filters, prisms, or inherent transmission of light depending on wavelength through a material, and the separated light is captured by individual imagers [13]. For a standard RGB image, bandwidths of light corresponding to red, green, and blue colors are directed towards three different sensors that captures the three images for each color channel. These three images are then stacked on top of each other and displayed to form the visual spectrum of colors humans can see.

Multispectral imagery is loosely defined as multiple images covering large bandwidths of light stacked on top of each other. This can result in true color images like RGB, false color images like RGN[3] to more easily detect things like vegetation, and dual band thermal imaging fusing the MWIR and LWIR[4] information resulting in better detection of targets during nighttime conditions.

[1]Charge-Coupled Device

[2]Complementary Metal Oxide Semiconductor

[3]RGN stands for Red, Green, and Near IR. Near IR wavelengths take up the blue channel resulting in living vegetation appearing as a purplish color.

[4]Midwave Infrared and Longwave Infrared

Researchers tend to argue over when a multispectral image becomes a hyperspectral image. For this thesis's purposes, multispectral images are comprised of a few bands with large bandwidths (70–400 nm), whereas hyperspectral images are comprised of many more bands with a much narrower bandwidth (5–10 nm) [14].

Hyperspectral sensors operate similarly to multispectral sensors, but with the spectral dimension being one to two orders of magnitude larger, typical images are gathered using particular scanning methods. Two of most common scanning techniques used in hyperspectral imagers are whisk broom and push broom scanners [8]. Both sensor types utilize movement of the sensor on a drone, aircraft, or satelite to scan the downrange of an image, but differ how the crossrange is scanned. For whisk broom sensors, a rotating mirror oscillates back and forth scanning the crossrange pixel-by-pixel. Because of the moving parts associated with scanning these swaths, whisk broom sensors are expensive, break more easily, and take longer to collect data. As a consequence, push broom sensors are becoming the standard in collecting hyperspectral imagery as the entire crossrange is imaged at once, line-by-line.[5]

From a remote sensing point-of-view, hyperspectral images open up the possibility of classifying pixels based on their spectral responses. In other words, each pixel can be seen as an accumulation of materials' spectral responses. Multispectral images do not provide enough discriminatory information to classify pixels [12]. Hyperspectral sensors sample the spectral dimension at a fine enough resolution allowing for the spectral curve of a pixel to start resembling one that can be measured by spectroscopy. Material classification due to sensing material spectral responses allows for a wide variety of new remote sensing applications including improved image

[5]Push broom sensors are not without their faults as the varying sensitivity between each of the imagers requires precise calibration to obtain valid data.

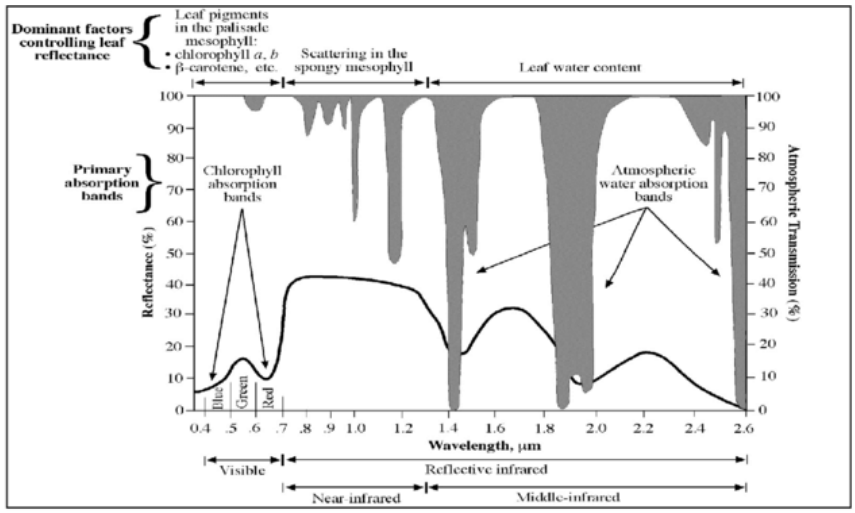

Figure 2.1

Water Absorption Curves in [16]

segmentation, improved performance of unwanted vegatation in agriculture applications over RGN images, and improved detection of man-made structures by looking for man-made materials such as steel or concrete [12].

As with any remote sensing modality, hyperspectral imaging faces challenges specific to its sensing domain. Because hyperspectral imagery is usually focused on sampling the visual and IR[6] bands, atmospheric attenuation is of great concern for sensing spectral curves [20]. Refer to figure 2.1 for an example of absorption due to water in the atmosphere versus wavelength [16]. To account for this, most preprocessing steps for hyperspectral imagery involve dropping bands most greatly affected by water absorption to avoid high variance in these bands.

Another issue hyperspectral sensing faces is due to the nature of downrange scanning. Because hyperspectral sensors are air-based and the scanning method is usually line by line, variations in

[6]There are many types of hyperspectral sensors. IR could be LWIR, MWIR, or SWIR.

downrange scanning due to air turbulence can cause each line to vary spatially. For example, if there is no turblence and the sensing platform moves at a constant speed, each pixel will be square as one would expect. However, a real flight path of a drone will experience some turbulence causing square pixels to become sheared into rectangles, or at worst, some form of quadrilateral. Mitigating this in real data can be very computationally intenseive but is generally performed by orthorectifying the data by using the sensing platform's IMU[7] data to account for turblence during data collects [2].

2.2 Neural Networks

This thesis investigates changes in band selection when varying neural network architecture. Machine learning is a vast research area, and covering all of it is outside the scope of this thesis. This section will focus only on the working principle behind neurons, the formulation of a fully connected neural network, autoencoders, and simplified explanations of the training process.

2.3 Components of Neural Networks

Machine learning algorithms in general take in data and perform operations resulting in outputs such as classification, decisions, or transformations. One of the more popular machine learning techniques is the artificial neural network (ANN). ANNs are loosely modeled after the biological systems of the brain where information is attenuated and transported through neurons and synapses. The purpose of machine learning neurons[8] and, ultimately, neural networks is to model any nonlinear function. Thus, neurons take in data and introduce nonlinearity to produce an output that propagates through the network eventually contributing to some task.

[7] Inertial Measurement Unit
[8] Also called perceptrons

Neurons take in a data vector $\mathbf{x}$ and output data y. In typical applications of this model, the connections from neuron to neuron has an associated weight, w, associated with it such that the data is attenuated as it moves along that connection. Mathematically, the data is multiplied against these weights and summed with the rest of the inputs in the neuron producing an unactivated output, z. At this point, z is a linear combination of weights in w and data in x. To introduce nonlinearity into the neuron, this unactivated output is passed through a nonlinear activation function, φ, producing the activated output, y. Assuming $\mathbf{x}$ is a column vector of length N and $\mathbf{w}$ is a row vector of length N, the entire operation is described as the inner product of $\mathbf{x}$ and $\mathbf{w}$ passed through the activation function, φ, shown in equation 2.1. Refer to figure 2.2 for a visual representation of this model.[9]

$$y = \varphi(z) = \varphi(\mathbf{wx}) \tag{2.1}$$

In order to learn more complicated nonlinear functions, a single neuron is not enough. The simplest form of an ANN is the multilayer perceptron (MLP). A MLP is a fully connected feed forward network comprised of an input layer, a hidden layer, and an output layer. Because there are multiple neurons in one layer, describing the forward propagation of data from one layer to the next is most easily described using linear algebra. Arranging the weights between two layers into a weight matrix $\mathbf{W}$ such that a column vector of weights for the first neuron $\mathbf{w_1}$ corresponds to the first column in $\mathbf{W}$, the second neuron's weights $\mathbf{w_2}$ corresponding to the second column, and so on results in a modification to equation 2.1 shown in equation 2.2.

$$\mathbf{y} = \varphi(\mathbf{z}) = \varphi(\mathbf{Wx}) \tag{2.2}$$

[9]The unactivated output models some linear hypersurface of the form $\mathbf{z} = \mathbf{wx}$ which is the slope-intercept equation where the hypersurface intersects through the origin. To allow this hypersurface move away from the origin, a bias b is generally added constructing the more familiar $\mathbf{z} = \mathbf{wx} + b$ equation. For this thesis, the bias is taken into account by allowing $\mathbf{x} = [1, x_1, x_2, ...]$ and $\mathbf{w} = [b, w_1, w_2, ...]$.

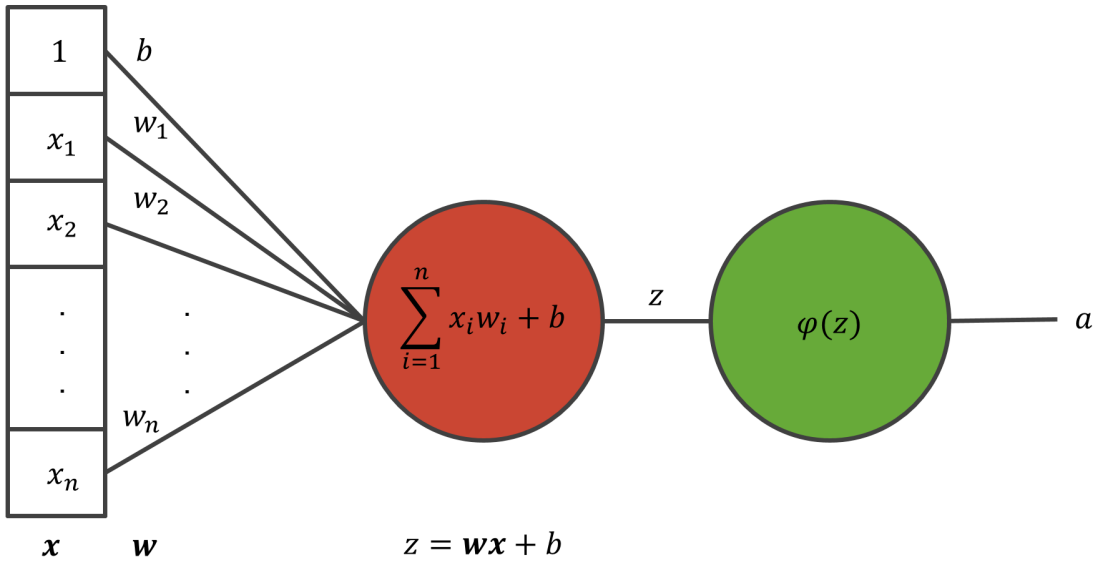

Figure 2.2

Standard Representation of Single Artificial Neuron

A MLP is the simplest form of a fully connected deep feed forward network where there could be multiple hidden layers. The above computations computations describe forward propagation throughout any feed forward network. Starting at the input layer, the input data, **x**, is propagated through the first layer producing the first layer of activated outputs, $\mathbf{a_1} = \mathbf{W_1 x}$. The second layer uses the previous layer's outputs as inputs producing the second layer of activated outputs, $\mathbf{a_2} = \mathbf{W_2 a_1}$. This can be generalized as shown in equation 2.3.

$$\mathbf{a_l} = \mathbf{W_{l-1,l} a_{l-1}} \tag{2.3}$$

Assuming there are L layers, equation 2.3 is implemented such that $\mathbf{a}_0 = \mathbf{x}$ and $l \in [1, L]$. Refer to figure 2.3 for a visual example of a simple deep feed forward network.

2.4 Backpropagation

Machine learning boils down to an optimization algorithm. For example, suppose there is an ANN designed to classify images on whether or not a cat or dog was present in the image. To do this, the ANN has two output neurons forming an output column vector, $\mathbf{o}$. Given some test data, each image has an associated label, $\mathbf{y}$, such that $\mathbf{y} = [1, 0]$ indicates the prescence of a cat and $\mathbf{y} = [0, 1]$ indicates the prescene of a dog in an image. When performing forward propagation on input data x, the output can be compared to the label and determined how incorrect the network was compared to the expected output, i.e. label, using a cost function. For instance, given an output vector $\mathbf{o} = [0.75, 0.2]$, a label $\mathbf{y} = [1, 0]$, and a loss function of $C = (\mathbf{y} - \mathbf{o})^T (\mathbf{y} - \mathbf{o})$, the associated cost for this example would be $C = 0.1025$. For ANNs, the machine learning task is to learn every layer's weight matrix, W, such that the cost is minimized.[10] This leads to the actual training technique of the neural network—backpropagation.

At its core, backpropagation takes advantage of the optimization problem. As weights vary, the cost function varies creating local optima. The goal of training is to find the global minimum of the cost function. Gradient descenet is an efficent method by which derivative of the cost function with respect to the network's weights are used to find changes in the weights leading to a minimum of the cost function. In practice, the error signal calculated from the cost function propagates backwards through the network determining how each weight should change—hence the name, backpropagation.

[10]The cost function used in this example is the L2 cost function for simplicity's sake. Because this is a classification problem with the output vectors being one-hot encoded, the cross entropy cost function would be more ideal in this scenario.

Throughout the explanation of backpropagation in this section, the cost function used will be the L2-norm cost function, i.e. $C = \frac{1}{2}(\mathbf{o} - \mathbf{y})^T(\mathbf{o} - \mathbf{y})$. The derivative of the cost function with respect to weight of the ith neuron in layer l connected to the jth neuron in layer $l + 1$ is broken down using the chain rule in equation 2.4.

$$\frac{\delta C}{\delta w_{ij}} = \frac{\delta C}{\delta a_j}\frac{\delta a_j}{\delta z_j}\frac{\delta z_j}{w_{ij}} \tag{2.4}$$

The motivation behind this particular application of the chain rule is that z_j is directly related to the weights as described in equation 2.1, so connecting its derivative to the derivative of C is desireable. Since this computation is performed starting at the output layer, equation 2.4 evaluated at the output layer with n neurons is shown in equation 2.5.

$$\begin{aligned}
\frac{\delta C}{\delta a_j} &= \frac{\delta C}{\delta o_j} = \frac{\delta}{\delta o_j}\left[\frac{1}{2}\left(o_j - y_j\right)^2\right] = o_j - y_j \\
\frac{\delta a_j}{\delta z_j} &= \frac{\delta \varphi\left(z_j\right)}{\delta z_j} = \varphi'\left(z_j\right) \\
\frac{\delta z_j}{\delta w_{ij}} &= \frac{\delta}{\delta w_{ij}}\left(\sum_{k=1}^{n} w_{kj}x_k\right) = \frac{\delta}{\delta w_{ij}} w_{ij}x_i = a_i \\
\frac{\delta C}{\delta w_{ij}} &= \frac{\delta C}{\delta a_j}\frac{\delta a_j}{\delta z_j}\frac{\delta z_j}{w_{ij}} = (o_j - y_j)\varphi'(z_j)a_i
\end{aligned} \tag{2.5}$$

In order to make future computation easier, the introduction of an "error" signal of the jth neuron, δ_j, is defined in equation 2.6. With this definition, equation 2.4 is reformulated to a more general form of the chain rule shown in equation 2.7.

$$\delta_j = \frac{\delta C}{\delta a_j}\frac{\delta a_j}{\delta z_j} \tag{2.6}$$

$$\frac{\delta C}{\delta w_{ij}} = \delta_j a_i \tag{2.7}$$

When attempting graident calculations in hidden layers, computing $\frac{\delta C}{a_j}$ is not as simple as computing it in the output layer. To compute this, the chain rule is utilized in succession to

compute the total derivative starting at the output layer and ending at the hidden layer. As long as the computation is performed backwards, repeated calculations of gradients is unnecessary. This leads to the recursive definition described in equation 2.8. In other words, $\frac{\delta C}{a_j}$ of layer l can be found as long as that term is known for every neuron in the $l+1$ layer. For equation 2.8, let this $l+1$ layer be a layer with n neurons.

$$\frac{\delta C}{\delta a_j} = \sum_{k=1}^{n} \frac{\delta C}{\delta a_k} \frac{\delta a_k}{z_k} \frac{\delta z_k}{\delta a_j} = \sum_{k=1}^{n} \delta_k w_{jk} \tag{2.8}$$

Equation 2.9 fully defines δ_j no matter where in the network the computation is performed.

$$\begin{aligned} \delta_j &= C'(o_j, y_j) = \left(o_j - y_j\right) \varphi'(z_j), \text{ if } j \text{ is an output neuron} \\ \delta_j &= \sum_{k=1}^{n} \delta_k w_{jk}, \text{ if } j \text{ is a hidden neuron} \end{aligned} \tag{2.9}$$

Using equations 2.7 and 2.9, each training iteration updates weights using equation 2.10.

$$\begin{aligned} \Delta w_{ij} &= -\eta \frac{\delta C}{\delta w_{ij}} = -\eta \delta_j a_i \\ w_{ij}^{\text{new}} &= w_{ij}^{\text{old}} + \Delta w_{ij} \end{aligned} \tag{2.10}$$

Where η is the learning rate and the introduced negative sign ensures that changes proceed towards a minimum, i.e., gradient descent. While this section fully describes the derivation of backpropagation, it is more easily formulated using linear algebra in practice. This formulation can be found in appendix A.

2.5 Hyperparameters

With forward and back propagation fully defined, an ANN can be trained to perform tasks. Before a network can be trained, the model must be constructed. When constructing a model, there

are many different considerations to take into account. Before an ANN can begin training, the weights between each layer must be initialized in order for any of the calculations to be performed. Decisions on various hyperparameters such as learning rate, minibatch size, and activation function type can make a dramatic impact on network performance. Finally, various learning strategies can alter convergence rate of networks as well as change hyperparameters during training based on various factors that come up during training.

During intialization of an ANN, the weight matrices between each layer are randomly initialized. There are some methods of choosing better intial weights depending on the choice of activation functions such as Xavier and Kaiming intialization, but the standard method is to select random weights from either a uniform or normal distribution of mean 0 and variance 1 [4].[11] Small, random initial weights allow each neuron to gradually learn which connections are important and change those weights proportionally to the backpropagated error gradients.

Hyperparameters are incredibly important for any network to function. A neuron's activation function determines how gradients are calculated and how the neuron actually fires in the network. Activation functions for neurons in a network are carefully chosen to meet certain criteria. In general, activation functions need to avoid shifting the gradient of the cost function toward zero,[12] be computationally inexpensive to perform, and be differentiable.[13] Popular activation functions in wide use include the sigmoid, softmax, and the hyperbolic tangent. While not a requirement, the reason these types of functions are used is that their derivatives are easy to compute and are

[11] In applications of transfer learning, "initial" weights between early layers are pre-trained and are not updating during training with only the weights towards the end of the network trained for "fine-tuning".

[12] Also called the vanishing gradient problem.

[13] Another desireable characteristic of some activation functions is to be zero-centered. However, some activation functions like the Rectified Linear Unit (ReLU) are not zero-centered, so it was not included in the general list of requirements.

usually put in terms of the original function. For instance, if the activation function is the sigmoid where $\varphi(x) = (1 + e^{-x})^{-1}$, then its derivative is $\varphi'(x) = \varphi(x)\,(1 - \varphi(x))$.

Learning rates are of particular importance as they define how much the ANN corrects itself with each training iteration. With a large learning rate, the model can converge towards a set of weights faster, but it may miss finding the global minimum due to exploding gradients. Small learning rates provide finer model corrections, slower convergence, and possible vanishing gradients, but it is easier to find minimum if the model can converge. Number of layers and number of neurons in a layer determine how "deep" the network is and greatly affect the ability of the network to model nonlinear functions. Finally, networks can be trained using an online or offline method. Online methods look at one training example at a time, while offline methods look at multiple examples called a batch in one weight update also called a training iteration. The minibatch size hyperparameter determines how many training examples the network sees during one training iteration, and the number of epochs determines how many times the network trains on the whole dataset. In short, all of these hyperparameters affect some key aspect during training affecting overall network performance and must be carefully tailored to each networks' applications.

2.6 Learning Strategies

Setting up a network for training requires a learning strategy. The learning strategy dictates how exactly the weights are updated and how the network actually handles the training data. The learning strategy described in 2.10 describes standard gradient descent (GD). A simple extension of this straetgy is called stochastic gradient descent (SGD) where the data is taken in small batches and the weights are updated based on the average change throughout each minibatch. Updating the

network based on multiple data points rather than one at a time not only speeds up computation, but also, the randomness associated with the batches tends to improve convergence to a global minimum.

Other popular learning strategies incorporate the use of a momentum term. The momentum term simply adds a small proportion of the previous iteration's weight change into the current iterations weight change. This is shown in equation 2.11 during iteration t where $\alpha \in [0, 1]$ controls how much the momentum term is present during updates.

$$\Delta w_{ij}^{t} = -\left(\eta \frac{\delta C}{\delta w_{ij}} + \alpha \Delta w_{ij}^{t-1}\right) \tag{2.11}$$

Intuitively, momentum encourages the weights to change in such a way that it has changed in the past. This can help the network converge to a solution despite local minima being present on the path towards the global minimum.

Stochastic Gradient Descent with momentum is described above, but variations on this can be seen in applications of learning strategies that aim to deemphasize the choice of the learning rate. A popular alternative to SGD with momentum is Adaptive Moment Estimation (ADAM). Adam is similar to the simple momentum term described in 2.11, but it keeps a running average of the first and second statistical moments of the cost gradient. Because of the way the weight update is computed, the learning rate is automatically adjusted each training iteration based on the calculated first and second moments. Equation 2.12 defines the update step for ADAM as described in [10]. Note that θ refers to all the layers' weights condense into a single vector at iteration t, β_1 and β_2

are decay parameters for the moments, $\mathbf{m}_0$ and $\mathbf{v}_0$ are intially set to zero vectors, and ϵ is set to a very small number to avoid divide by zero errors.

$$\begin{aligned}
\mathbf{m}_t &= \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\,\Delta_{\boldsymbol{\theta}} C\left(\boldsymbol{\theta}_{t-1}\right) \\
\mathbf{v}_t &= \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\left(\Delta_{\boldsymbol{\theta}} C\left(\boldsymbol{\theta}_{t-1}\right)\right)^2 \\
\hat{\mathbf{m}}_t &= \frac{\mathbf{m}_t}{1-\beta_1^t} \\
\hat{\mathbf{v}}_t &= \frac{\mathbf{v}_t}{1-\beta_2^t} \\
\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}
\end{aligned} \tag{2.12}$$

Another method by which the learning rate can be automatically updated is through the scaled conjugate gradients (SCG) method. This method is used in MATLAB's deep learning toolbox in order to update fully connected networks. The general idea of this method is to estimate the second order gradient of the cost function in order to automatically choose step sizes that follow the steepest descent path for the weight updates. The algorithm itself can be found in algortihm 1 as shown in [17].

Algorithm 1: Scaled Conjugate Gradients Strategy

input : weight vector $\mathbf{w}$, $0 < \sigma \leq 10^{-4}$, $0 < \lambda_0 \leq 10^{-6}$, $\bar{\lambda_0} = 0$, $\mathbf{p}_0 = \mathbf{r}_0 = \nabla C(\mathbf{w})$, $k = 1$,

`success` = true

output Weight Vector $\mathbf{w}_{k+1}$

:

if *`success` is true* **then**

$\sigma_k = \sigma/|\mathbf{p}_k|$; $s_k = \nabla \left[C\left(\mathbf{w}_k + \sigma_k \mathbf{p}_k\right) - \nabla C\left(\mathbf{w}_k\right)\right]/\sigma_k$; $\delta_k = \mathbf{p}_k^T \mathbf{s}_k$;

$\delta_k = \delta_k + \left(\lambda_k - \bar{\lambda}_k\right)|p_k|^2$;

if $\delta_k \leq 0$ **then**

$\bar{\lambda}_k = 2\left(\lambda_k - \delta_k |p_k|^2\right)$; $\delta_k = -\delta_k + \lambda_k |p_k|^2$; $\lambda_k = \bar{\lambda}_k$;

$\mu_k = \mathbf{p}_k^T \mathbf{r}_k$; $\alpha_k = \mu_k/\delta_k$; $\Delta_k = 2\delta_k \left[E\left(\mathbf{w}_k\right) - E\left(\mathbf{w}_k + \alpha_k \mathbf{p}_k\right)\right]/\mu_k^2$;

if $\Delta_k \geq 0$ **then**

$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{p}_k$; $\mathbf{r}_{k+1} = -\nabla C\left(\mathbf{w}_{k+1}\right)$; $\bar{\lambda}_k = 0$; `success` = true;

if k *mod* $N = 0$ **then**

$\mathbf{p}_{k+1} = \mathbf{r}_{k+1}$;

else

$\beta_k = \left(|\mathbf{r}_{k+1}| - \mathbf{r}_{k+1}^T \mathbf{r}_k\right)/(\mu_k)$; $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$;

if $\Delta_k \geq 0.75$ **then**

$\lambda_k = 0.25 * \lambda_k$;

else

$\lambda_k = \lambda_k + \left[\delta_k \left(1 - \Delta_k\right)\right]/\left(|\mathbf{p}_k|^2\right)$;

if $\mathbf{r}_k \neq \mathbf{0}$ **then**

$k = k + 1$; continue;

else

return $\mathbf{w}_{k+1}$;

2.7 Types of Neural Networks

This section has focused mainly on the simplest form of neural networks, the multilayer perceptron (MLP) network, where there is one input layer, one hidden layer, and one output layer. More generally, this form of network can be called a deep feed forward (DFF) network. A network is considered to be a deep network when there is more than one hidden layer. Many other types of networks can be built off the concepts described in this section. A few of these networks are dicussed below.

One of the simplest extensions of the multilayer perceptron is the radial basis function neural network (RBFNN). Like the MLP, the RBFNN strictly has one input layer, one hidden layer, and one output layer. One key difference from the MLP is that the hidden layer activation functions of RBFNNs are radial basis functions like the Gaussian function. RBFNNs operate very similar to k-means clustering in that their goal is to cluster data points that are closer together. The basic behaviour of training an RBFNN is to update initial cluster centroids[14], radius of the radial basis function, and weights leading to the output. During training, the centroids converge to their positions describing a feature vector that is then projected onto the transformation space described at the output neurons. In other words, each centroid contributes some amount to a sample data point's prediction based on how close it is to other centroids in the RBFNN.

Another simple extension from MLP networks is the autoencoder. Standard autoencoders consist of one input layer, one hidden layer, and one output layer. Autoencoders seek to reconstruct the input matrix at the output resulting in a compressed version of the input at the hidden layer. Because of this, the weight matrix between the input and hidden layer is considered an encoder,

[14]These can be determined via k-means clustering or randomly sampling the training data.

and the weight matrix between the hidden layer and the output is the decoder. Furthermore, the compressed output due to the limited number of neurons in the hidden layer is generally called the autoencoder's latent representation of the input. Because the output is compared to the input, the cost function of these networks resemble a type of regression, but training autoencoders is similar to that of deep feed forward networks. Other varieties of autoencoders exists such as the variational autoencoder where the hidden neurons contribute to two latent representations for the mean and variance of a normal distribution, the denoising autoencoder where portions of training vectors are set to zero in order to learn ways to remove noise from noisy inputs, and the sparse autoencoder that includes more hidden neurons in addition to sparsity regularization in the cost function to increase performance over standard autoencoders.

Recurrent Neural Networks (RNN) extend DFF networks by allowing hidden neurons to incorporate a feedback loop enabling the capability to deal with sequence data, i.e., time dependent data where data at time t is dependent on data at time $t-1$ and before. RNNs are generally used in applications of speech recognition and computational chemistry as they are useful for taking into account context within a data sequence [23]. RNNs setup and training is unique in that the hidden neuron contains an aggregate sum of past inputs to the system. Because this can lead to vanishing or exploding gradient problems, alternative solutions involving truncating the history of the memory in the hidden neurons have been suggested. One of the more widespread types of RNNs is the Long Short-Term Memory LSTM network where a memory unit contains a forget gate, input gate, and output gate that controls how much of the network should remember as well as what the hidden neuron's state should be based on this information.

Perhaps one of the most popular types of neural networks are Convolutional Neural Networks (CNN) due to their applications in computer vision. A standard DFF network may flatten a 2D image into a vector and perform classification tasks based on that vector. However, depending on the spatial and spectral resolution of the image, the input vector could have hundreds of thousands of dimensions resulting in a network that takes too long to converge. More sophisticated algorithms would extract features from the raw data into a feature vector which is then fed into a DFF network, whereas CNNs aim to perform both feature extraction and classification into one network. CNNs utilize convolution neurons to learn filters that extract features in various layers of the CNN. Once the CNN has extracted the features, the last portion of the CNN is essentially a DFF that performs whatever task the CNN needs to do, e.g., classification. Because of this process, CNNs are able to perform classification on images with increased performance when compared to an equivalent DFF network [18].

2.8 Dimensionality Reduction

As discussed in section 2.1, one potential issue with hyperspectral imagery from an analysis perspective is that the increase in spectral data results in longer computational resources needed for analysis—especially for machine learning algorithms. Since hyperspectral imagery is highly dimensional, it is important to perform dimensionality reduction (DR) on the data. There are many different methods of DR towards any type of data, but these methods generally fall into two categories—methods based on original dataset subsambling and methods based on trasnforming the original data into a new space. Regardless of the method, the end goal of DR is to select dimensions that provide the most information of the original set of data while discarding dimensions that are

redundant or provide uninformative information. Below are descriptions of various DR methods seen in [24] and the references found in chapter III.

One of the simplest examples of dimensionality reduction is based on principle component analysis (PCA). The goal of PCA is to find k orthogonal basis vectors from the data X such that variance is maximized along each of the new basis vectors, or principle components (PC), i.e., the maximum amount of variance of the dataset is described with the first PC axis, the second greatest amount of variance is described with the second PC axis and so on. In doing this, a form of compression is found if N PC are selected such that $N < k$. Performing in analysis of the data in the transformed PC space is one of the simpler versions of DR. To perform subsambling in the original space, methods such as the ones used in [11] utilize the eigenvectors and eigenvalues for transforming to the PC space as a basis for selecting the most informative subset in the original space with the argument being that the magnitude of the transformation vector for a dimension should be greater than the magnitudes of transformation vectors for less important dimensions [7].

A natural extension of the PCA DR approach is to use independent component analysis (ICA). Simply put, ICA works similarly to PCA with two changes. For one, the independent component (IC) basis vectors are not necessarily orthogonal resulting in both linear and non-linear transformations being possible. Additionally, instead of constructing these IC axes based on maximizing variance in the data, ICA seeks to maximize kurtosis of the data for each IC. Maximizing kurtosis leads to the main applications of ICA involving viewing the data as a mixture of source signals.[15] This is especially useful in DR for hyperspectral imagery in that hyperspectral pixels can be viewed as a mixture of source materials. With that in mind, ICA can be used to determine the most

[15]See the blind source separation problem is an example of ICA applications.

informative bands that provide the most information on the source materials similar to the method described for PCA [6].

Another popular DR technique involves implementing Linear Discriminant Analysis (LDA). LDA aims to find a projection such that class separability is maximized while minimizing scatter within each class. This is done through closed form solutions using class means, interclass variance, and intraclass variance. Similar to the projection methods described above, the transformed data is lower in dimensionality than the original, but the transformation matrix holds information describing which input dimensions are important for maximizing class separability and minimizing the spread within a single class allowing for selecting a subset of the original data. An example of DR using LDA can be found in [3]. In general, there are a variety of techniques that that can perform DR on arbitrary datasets. In [28], the DR strategy combines a searching algorithm that minimizes the correlation between selected dimensions based on the volume of the hyperellipsoid that the subset of bands creates, i.e., search for a subset that minimizes hyperellipsoid volume. Refer to [24] and [7] for a survey of various techniques for DR.

An emerging pattern in many DR techniques is that some transformation from the input data space to a new data space is computed. After that computation, either the transformed data space is already reduced in dimensionality or the transformation itself is used as a basis for selecting a subset of the original data. Machine learning has an obvious place in this format. For instance, the latent representation that autoencoders compute can be synonymous with PCA compression and fed into other algorithms as the reduced dataset. Other networks such as those in [26] use the input weight matrix, synonymous with the transformation matrices in PCA, ICA, and LDA, as a basis for selecting subsets of data for DR.

The remainder of this thesis focuses on methods that select subsets of the original data in its original space. In the context of hyperspectral imagery, this is equivalent to a process known as band selection where a subset of the spectral dimension is found in order to reduce computational costs of hyperspectral analysis. A comprehensive review of state-of-the-art band selection techniques are listed and compared in [25]. These methods are loosely broken into six categories: ranking based, searching based, clustering based, sparsity based, embedding-learning based, and hybrid-scheme based band selection. In [25], a state-of-the-art method in each category is selected and compared to every other category. This thesis selects specialized versions of these methods from the ranking based, searching based, and embedding-learning based categories. In chapter III, band selection is generalized from those found in papers such as [26] and [9] to utilize deep feed forward networks as the architecture for band selection.

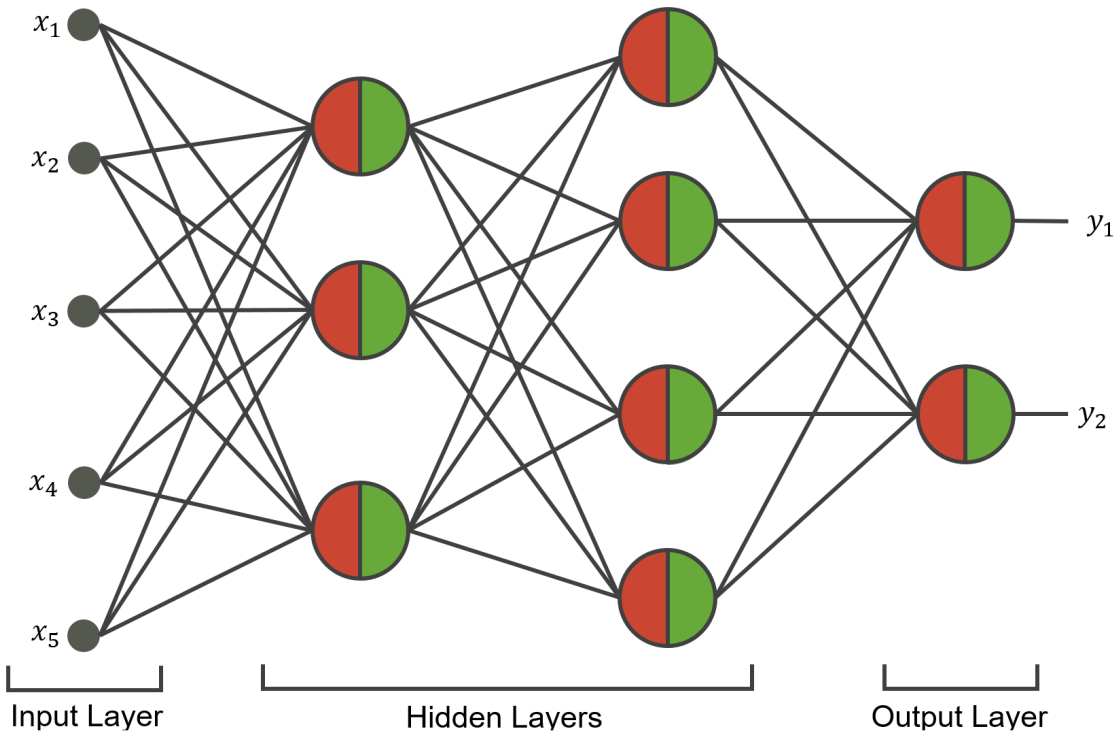

Figure 2.3

Simple Representation of a Deep Feed Forward Network

CHAPTER III

CONSTRUCTING AND ANALYZING NEURAL NETWORK-BASED BAND SELECTION

The following is a conference paper submitted and published to the 2021 SPIE Defense and Commercial Sensing conference. The formatting has been slightly altered to fit this thesis's. Minor grammatical changes have been made with some additional information added to Section 3.4.

3.1 INTRODUCTION

Data analysis is incredibly important in order to accomplish goals such as optimizing product designs, automating increasingly complex tasks, and predicting trends in data to make more intelligent decisions. As sensors and various fusion algorithms continue to improve, the amount of data generated continues to increase in size and can become unmanageable. It is becoming apparent that the sheer volume of data in some tasks can overwhelm analysis algorithms and negatively impact performance. With this in mind, algorithms such as band selection are designed to select the most informative subsets of hyperspectral imagery's (HSI) spectral dimension. In other words, the purpose of band selection generally aims to reduce 200 channels of spectral information down to a more manageable number for computation. These algorithms can be used to reduce the amount of data negatively impacting computation time yet still maintain algorithm performance parameters such as classification accuracy.

In this paper, the type of data that will be the focus of our analysis is HSI. HSI is very similar to RGB multispectral imagery in that they both have a third dimension in which material properties affect specific bands of light differently. However, while color images typically have red, green, and blue channels, hyperspectral imagery has many channels splitting a broad bandwidth of light into fine slices. HSI is beneficial from a remote sensing point-of-view in that a high spectral resolution results in each pixel capturing the material characteristic response of objects within a scene. With the size of this third dimension typically increasing by a factor of 100 of a common RGB image, the computation time needed to perform various analyses can suffer.

Herein, we propose a machine learning-based band selection algorithm we are calling Neural Network Band Selection (NNBS). We apply our algorithm to select informative bands of a novel HSI dataset to reduce computational complexity while maintaining classifier performance using the reduced dataset. Using this dataset allows us to test our technique against other state-of-the-art band selection algorithms on spectral curves of unknown materials.

3.2 RELATED WORK

Most methods of band selection involve using statistical analysis on the data itself in order to reduce the amount of bands present. This usually involves leveraging some information metric in order to rank the bands and cut the least informative bands present. Simple methods utilizing principle component analysis (PCA) are widely used to transform the data in such a way as to maximize the variance between channels. This can be extended to independent component analysis (ICA) to leverage the fact that hyperspectral imagery has a limited amount of classes in any datacube. Taken to the extreme, the data projections can be generalized to a point to where

the shape of the data itself can be used as a measure to select bands composing an optimal subset of the data. As an alternative to transforming the data to maximize some statistical measure, some methods opt to maximize measures based in information theory such as information gain (IG). However, the most widely used method of performing dimensionality reduction (DR) on data utilizes some form of machine learning.

Many DR methods prefer to reorient the band selection problem utilizing standard transformation methods such that the data is transformed and analysis perfomed in this lower-dimensionality space. Rodarmel and Shan use PCA to reduce hyperspectral images' dimensionality and found that using around 10 PCA bands is comparable in classification results to using the original 60 bands of their hyperspectral dataset resulting in a 83% reduction in dimensionality [22]. Koonsanit et al extend PCA band selection by integrating the information gain metric in their selection process to allow selection of bands to be more intelligent based on class separability information [11]. While these methods produce competitive results, band selection is not performed on the original dataset which does not allow for any additional analysis in the original information space.

To perform various analyses on the hyperspectral sensors and its captured data, it is important to perform DR in its original space of N variables. Du et al use projection-based methods to analyze and rank the original hyperspectral bands by using ICA to determine which bands are important for separating original source signals [6]. Ball et al use standard methods that take advantage of classifier performance measured in the area under receiver operating characteristic (AUROC) curves to select bands that maintain classifier performance to improve level set image segmentation [3]. Zhang et al take advantage of geometrical intrepretations of data covariance in order to select bands based on maximizing statistical measures [28]. As noted in several papers,

geometrical and statistical measures such as spectral angle mapper (SAM) and spectral information divergence (SID) can be used to denote band relevance for selection purposes [3].

As more big data problems are being solved with machine learning techinques, there are also many band selection techinques based in machine learning to allow more seamless integration with existing architectures [27, 21, 19]. A majority of techinques based in machine learning make use of the weights connecting various layers in the network. For instance, some networks take advantage of an assumption that the weights connecting to the final feature extraction layer will activate sparsely when an interesting object needs to be classified, combine that with an anomaly detection calculation based on statistics, and select bands based on which features cause the anomalies [21]. Other methods use dropout methods involving randomly removing bands to take note of the degradation in performance [5, 21]. The bigger hit to performance correlates to more information contained in those bands. Finally, other methods use autoencoder architectures to compress the hyperspectral imagery and minimize reconstruction error at its output [9, 26, 1]. Once this is trained, the input weight layer gives insight to which bands are needed for accurate reconstruction thus automatically ranking the bands in the process.

Band selection techniques unique to machine learning implement some form of dropout in the input space and detect how this affects classifier performance. Chandra and Sharma directly implement input variable masking to accomplish band selection by asserting that drops in autoencoder reconstruction error directly correlate with the relevance of the masked band [5]. Zhan et al implement a similar approach as the previous authors, but they segment the bands using a distance density metric and test random band combinations before selecting a set demonstrating the highest performance for their convolutional neural network (CNN) [27]. Some authors acknowledge that

an exhaustive search of band combinations can realize optimal results but is generally impractical in HSI processing [3].

Other methods based in machine learning tend to integrate previously mentioned methods into their own architecure or take advantage of machine learning's nature as a black box. As an example of integrating band selection into machine learning architecture, Ribalta et al construct an attention-based CNN by extending the Elliptical Envelope algorithm from the standard, geometry-based band selection method to an anomaly detector under the assumption that highly informative bands should be viewed as anomalies [21]. By far, the most popular band selection approaches unique to machine learning leverage network activations or layer weights in the band selection process. Prasvita uses a method similar to Ball and Bruce by training many weak classifiers in a one-against-all scenario for each class, using the activations to calculate band contributions for each class, and select the best bands to improve overall classifer performance [19]. Mateus et al utilize autoencoder architecture to reconstruct HSI so that input weights can be used to rank each band in terms of their magnitudes [9]. This method is improved by integrating previously discussed segmenting tactics to improve computational performance [26]. Other authors discourage segmentation in the spectral dimension to allow for comparison between all bands. Instead, Ahmad et al segment spatially and implement denoising autoencoders to provide competitive results [1].

3.3 IMPLEMENTATION

Our method aims to extend the autoencoder band selection techniques—referred to in this paper as autoencoder band selection (AEBS)—to a generalized network architecture we are calling neural network band selection (NNBS) for performing task-specific band selection. While the bands

selected via AEBS produce acceptable results, the bands themselves are selected on the same basis that the network's input weights are tuned—minimizing image reconstruction error. Important processing applications of hyperspectral imagery include segmenting images into similar materials or classifying each hyperspectral pixel as a specific material. It is unclear if image reconstruction is the best metric for selecting bands for classification tasks. Therefore, it is desireable to select bands on the basis of a specific task to be optimized. Herein, we propose a generalization of the autoencoder methods listed in section 3.2 by shifting the task of the network to classification. Our method aims to show that performance can be maintained—possibly even improved—regardless if the network task has shifted from image reconstruction to classifying hyperspectral endmembers.

In NNBS, we use the learned input weights as a basis for selecting which bands contribute the most to our network's classification task. The input weight matrix between the input neurons and the first hidden layer are organized such that the n^{th} row contains a vector of weights connecting the n^{th} input neuron to the hidden layer. This creates a weight matrix $\mathbf{W}$ that can be used to calculate band contributions to task performance. In viewing the weight matrix as a column of row vector entries, we can utilize the Frobenius norm to get the magnitude of each row vector producing a contributions vector, $\mathbf{c}$, that describes the magnitude that each band in the hyperspectral dataset contributes to our network's task. Refer to equation 3.1 for a quick calculation of $\mathbf{c}$.

$$\mathbf{c} = \text{diag}\left(\mathbf{W}^T \times \mathbf{W}\right) \tag{3.1}$$

This method assumes that a sufficiently trained network will produce an input weight matrix, $\mathbf{W}$, where the magnitude of each row will be treated as that neuron's contribution to the overall task. The n^{th} element of the calculated contributions vector $\mathbf{c}$ describes the contributions of the n^{th} input

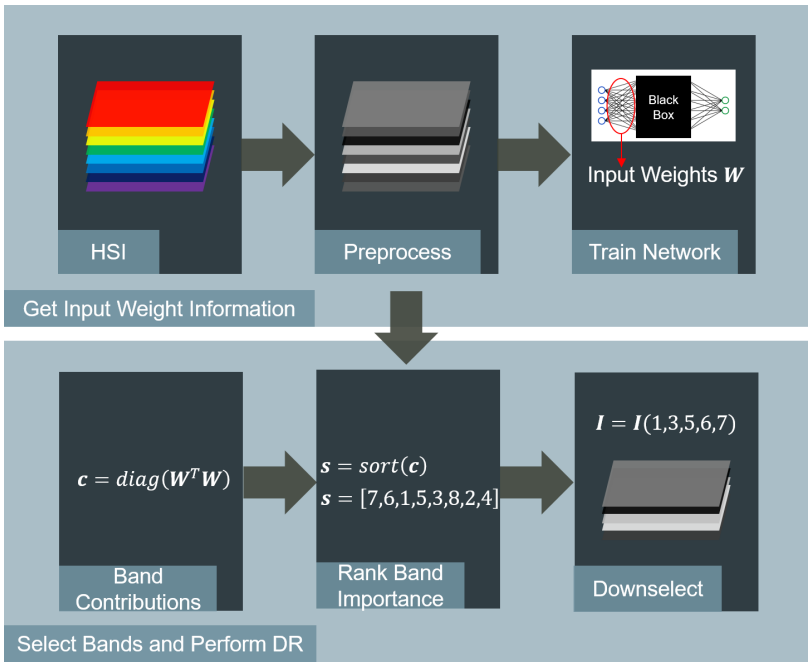

Figure 3.1

Neural Network Band Selection Diagram

neuron. In simpler terms, each input neuron corresponds to a band in the hyperspectral dataset. With the contributions vector, we can simply rank each band according to their contributions by sorting **c** and selecting an amount of bands to continue analyzing. Refer to figure 3.1 for a complete diagram of how NNBS functions as a band selection technique.

A major component of this paper is to show that using input weights as a ranking mechanism for band selection is possible for any network. To shift focus away from any specific neural network architecture, we utilize a simple multilayer perceptron architecture to classify pixels in a unique dataset and apply the input weight matrix ranking method to this architecture. Architecture parameters such as number of input layers and number of neurons in a specific layer are parameterized in terms of the hyperspectral data presented to the network to further abstract specific network architecture. Given some hyperspectral data of size XxYxD and number of material classes, N, our network's architecture has D input neurons to classify individual hyperspectral pixels and each subsequent hidden layer decreases in number of neurons by half of the previous layer. This continues as necessary until the output layer having N output neurons to classify each pixel as a material out of N classes. For simplicity purposes, the number of output neurons is assumed to be much less than the previous layer's number of hidden neurons. In our case, this behavior is represented in the following: $N \ll D/4$. Refer to figure 3.2 for an abstraction of our simplified architecture.

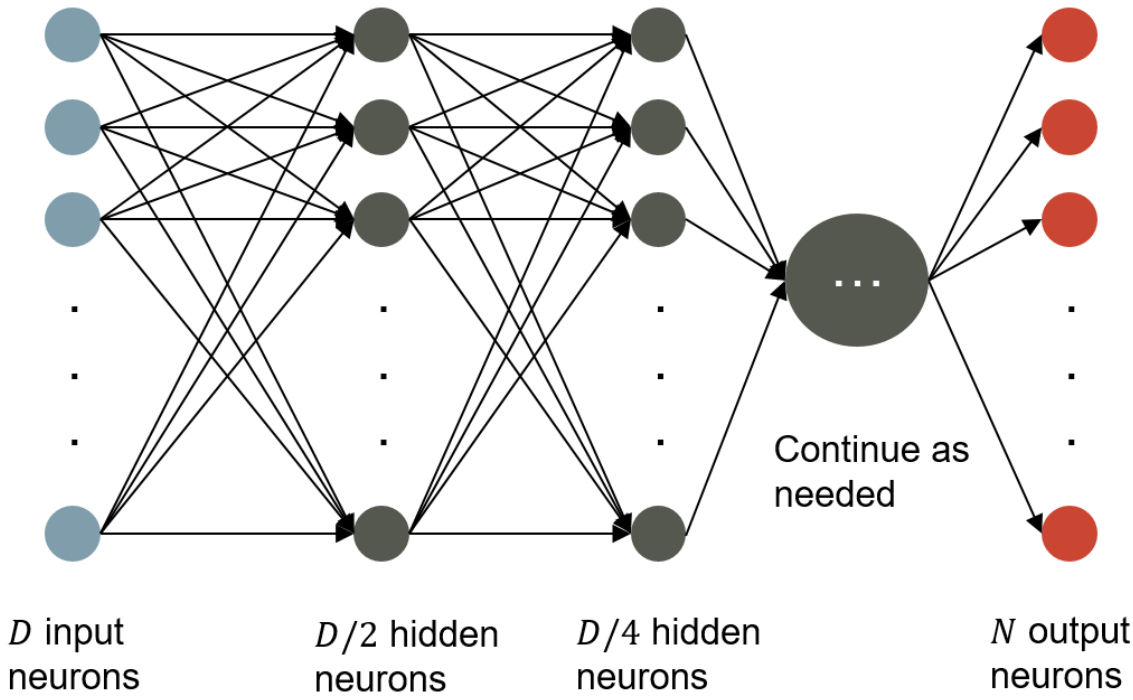

Figure 3.2

Network Architecture

An important question to consider is the following: why should we focus only on the input weight matrix? For NNBS to function, an arbitrary network must be trained on the original hyperspectral data such that we can obtain the learned transformation matrices between each layer that accomplish the task of the network—in our case, endmember classification. Once these weight matrices are learned, we can use the magnitudes of each neuron's weight vector as an indication of how important a layer's feature contributes to the next layer's set of features. In doing so, this enables us to compare contributions of the input features to the learned first layer features, contributions from first layer features to second layer features, and so on until we get to the output layer. However, because of the difference in activation values between each layer, there is no clear method to compare input feature contributions to third layer features for instance. Instead, we take advantage of the standard black box approach to neural network analysis allowing us to simplify this process by only focusing on the input weight matrix for direct contributions to the output of the black box. Algorithm 2 describes the band selection technique described in figure 3.1 using only input weight contributions.

Algorithm 2: Neural Network Band Selection Algorithm

input : HSI I, Labels Y, Number of Selected Bands n

output Selections s

:

X, Y = preprocessData(I,Y)

options = setupNetworkParameters()

net = trainNetwork(X,Y,options)

W = net.input.W

c = diag(W'*W)

val, ind = flip(sort(c))

s = ind(1:n)

3.4 RESULTS

The hyperspectral data was taken with a sensor dividing the visual and near infrared spectrums into 272 channels. The objects in the image range from callibration equipment, natural materials, and standardized CUBI objects with various surface materials applied to them. The data was hand labeled to identify 15 classes. Before being analyzed for band selection, the data is preprocessed to including data whitening to ensure cross channel correlation is minimized. The image is then reorganized into a matrix to be fed pixel-by-pixel into our network. Refer to figure 3.3 for the image used in this study.

Figure 3.3

Hyperspectral Data Taken by the ERDC

The networks used for NNBS and AEBS were constructed using built-in functions via MATLAB and MathWorks' Deep Learning Toolbox. Both network parameters were primarily kept to default settings with the only notable changes being that the NNBS network was trained for 500 epochs via the Adam algorithm. The pixels were randomly separated into 70% training data and 30% testing data, and NNBS was run multiple times to ensure that band selections were not sensitive to network initialization. On average, the band selections for NNBS varied only by one or two ranks per run, so the mode was taken for each selection to account for slight variations in specific band rankings. It is worth noting that band selections from the AEBS method resulted in much more varied rankings per run, and the same processing method was applied to the AEBS rankings.

The method of evaluation is to perform DR based on the results of various band selection algorithms and feed that new data through two classifiers. A few varying band selection algorithms from section 3.2 were chosen and implemented to compare results with NNBS. The selected methods for comparison were independent component analysis band selection (ICA-BS) [6], optimal projection-based band selection (OPBS) [28], and AEBS. To characterize DR's impact on classifier performance, the classifiers are also run on the unaltered dataset. Finally, two of the most common HSI classifiers—neural networks and k-nearest neighbors—are used to produce baseline performance results.

Because the amount of data points labeled as *soil* is so large in this dataset, we perform a type of undersampling on the *soil* class to avoid skewed class distribution issues. To inform our undersampling ratio, we use an anomaly detector to classify pixels between *soil* and *other*, effectively creating a one-against-all classifier. This gives us an accuracy of around 98.5% which

Table 3.1

Neural Network Classification Accuracies for Various Band Selection Methods

Method	**Accuracy**		
No Band Selection	94.3%		
	50% DR	**75% DR**	**90% DR**
ICA-BS	93.4%	90.4%	**83.1%**
OPBS	92.5%	90.2%	81.9%
AEBS	93.0%	90.5%	81.5%
NNBS	**93.7%**	**91.4%**	81.3%

we use as an indicator for how much we should undersample the *soil* pixels. While training on this undersampled data gives us an optimistic classifier, the entire classifer including the anomaly detector results in a well-balanced system in which pixels containing soil are screened before making it to a more balanced classifier for the objects in our scene. Refer to tables 3.1 and 3.2 for comparisons of classifier accuracies when varying band selection method and amount of DR applied. Note that 50%, 75%, and 90% DR applied refers to using 136, 68, and 27 bands respectively for classification.

Table 3.2

k-Nearest Neighbors Classification Accuracies for Various Band Selection Methods

Method	**Accuracy**		
No Band Selection	91.7%		
	50% DR	**75% DR**	**90% DR**
ICA-BS	**91.8%**	90.2%	**90.0%**
OPBS	90.6%	89.8%	88.9%
AEBS	91.3%	89.9%	88.2%
NNBS	**91.8%**	**90.4%**	89.0%

As is clear in both tables, NNBS tends to perform best in general when less DR is used. Even at extreme levels of DR, NNBS continues to remain competitive with the other band selection methods. It is important to note that once the data is reduced to 10% the original data's dimensionality, our assumption of network architecture, $N \ll D/4$, is no longer true. This can cause issues during training for NNBS. With tweaks to hyperparameters controlling training behavior and network architecture, NNBS can be improved. These tweaks are not implemented to maintain the architecture agnostic methodology.

3.5 FUTURE WORK

This paper focused on simple architectures to further explore the idea of generalizing band selection using magnitudes of input weight vectors. Through our results, we observe that NNBS is competitive for standard applications of dimensionality reduction. In doing so, we surmise that the learned transformation matrices in neural networks can be utilized to perform various tasks from selecting important input features—as is this paper's goal—to analyzing higher level features found in deep layers of complicated machine learning architectures. The next step in this work is to analyze the importance of features selected in convolutional neural networks and of optimal solutions of genetic algorithms.

An interesting path forward for this work involves testing more complex architecture, but also applying band selection principles to a new application. For instance, certain NN architectures learn base features towards the beginning of the network and learn higher order features towards the end of the network. Band selection aims to down select input features by determining which of these input features contribute the most information. With all this in mind, "band" selection

could be applied to higher level features to guide machine learning algorithms to more informative solutions.

We have shown that the band selection technique underlying AEBS—ranking based on input weight matrix magnitudes—can be generalized to a simple multilayer perceptron architecture. The evaluation of classifier performance on those evaluated bands was comparable, if not preferable, to other standard band selection techniques. While NNBS proved to be superior when more bands are retained in the reduced subset, other methods continue to be competitive when the dimensionality is further reduced. It appears that NNBS prioritizes class separability as this is the underlying optimization task that generates the learned input weight matrices as opposed to data reconstruction or compression. As such, we propose that selecting optimal bands for dimensionality reduction is not an objective process but a subjective one tailored to the type of tasks being performed on the data itself.

CHAPTER IV

ADDITIONAL RESULTS AND DISCUSSION

4.1 Additional Results

This section is dedicated to results obtained after publishing the paper described in chapter III. The popular Pavia City Center dataset[1] is used alongside additional results obtained using the data gathered by the ERDC. The Pavia City Center dataset is captured with a hyperspectral sensor diving its targeted bandwidth into 102 bands spanning 430 nm to 860 nm. The dataset has nine classes in total with a tenth class covering the pixels that are unlabeled. Because this dataset is much less skewed than the ERDC dataset, no subsample balancing is performed on the data. Refer to tables 4.1 and 4.2 for the number of examples per class in each dataset.

[1]This dataset was gathered by Prof. Paolo Gamba from the Telecommunications and Remote Sensing Laboratory at Pavia University.

Table 4.1

Class Distribution in the ERDC Dataset

Class Name	**Number of Examples**
clay	15875
paint1	646
paint2	616
paint3	650
paint4	624
paint5	635
paint6	572
paint7	551
paint8	615
cubi1	179
cubi2	181
cubi3	187
cubi4	164
other	443

Table 4.2

Class Distribution in the Pavia City Center Dataset

Class Name	**Number of Examples**
Water	824
Trees	820
Asphalt	816
Self-Blocking Bricks	808
Bitumen	808
Tiles	1260
Shadows	476
Meadows	824
Bare Soil	820

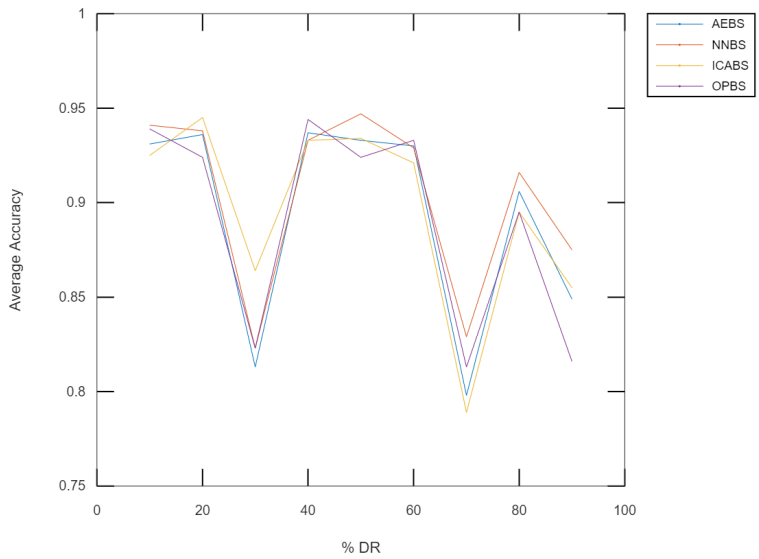

Figure 4.1

Average Accuracy vs % DR (DFF Classifer, ERDC Dataset)

In chapter III, the four methods are compared using average overall accuracy of a DFF network and a K-means clustering algorithm at 50%, 75%, and 90% applied DR. Refer to figures 4.1–4.4 for this same information ranging from 10% reduction to 90% reduction in number of bands when varying selection method, classifer, and dataset.

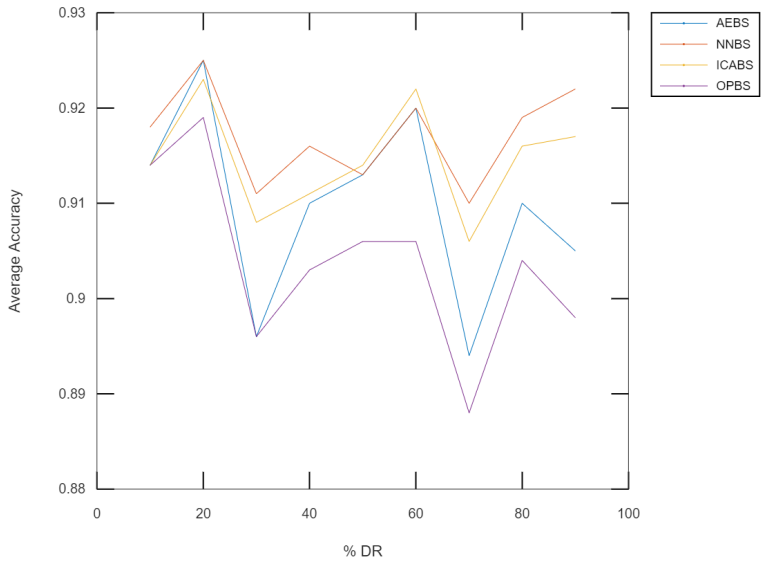

Figure 4.2

Average Accuracy vs % DR (KNN Classifer, ERDC Dataset)

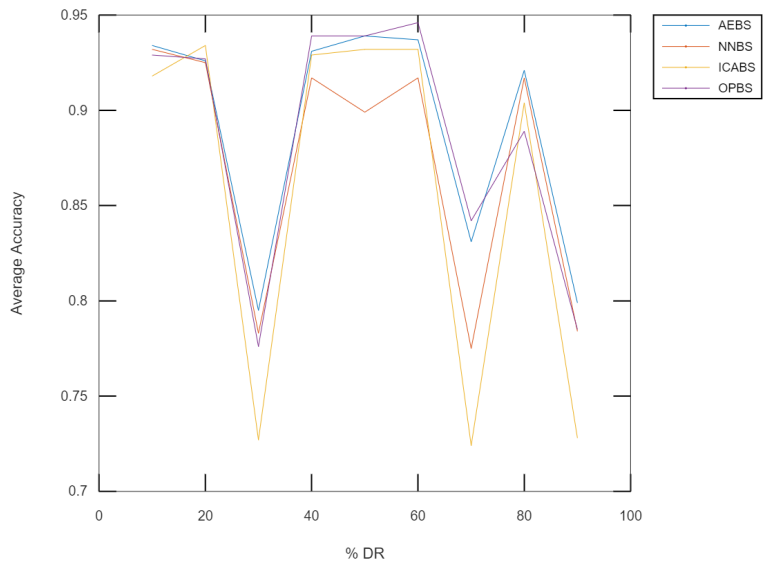

Figure 4.3

Average Accuracy vs % DR (DFF Classifer, Pavia City Center Dataset)

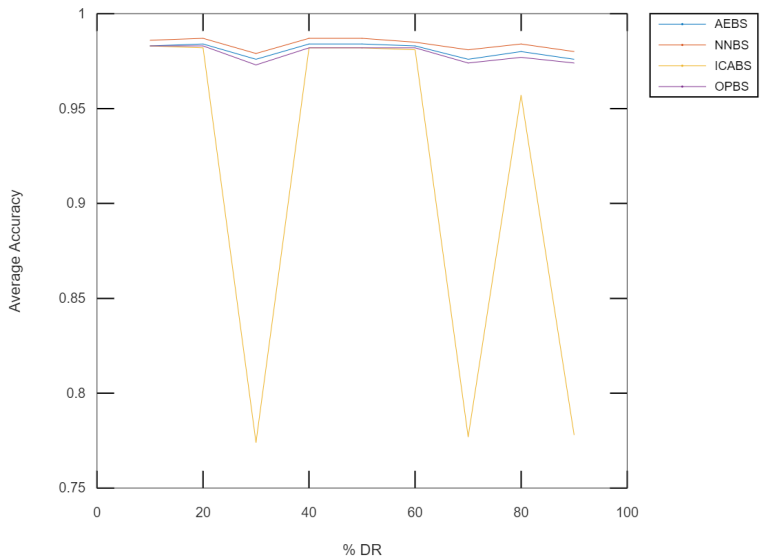

Figure 4.4

Average Accuracy vs % DR (KNN Classifer, Pavia City Center Dataset)

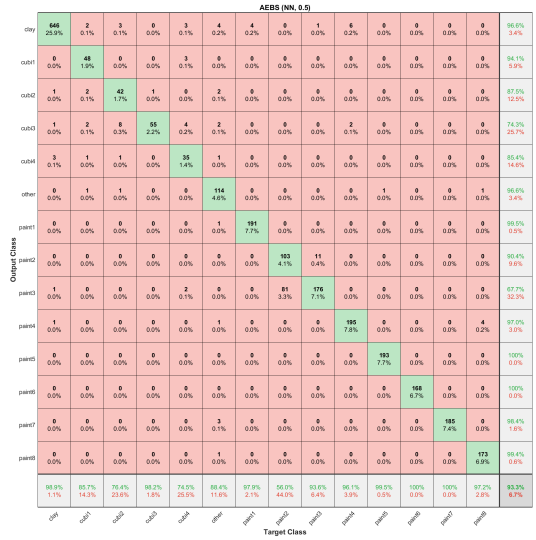

Figure 4.5

AEBS Confusion Matrix (DFF Classifer, ERDC Dataset, 50% Applied DR)

Average overall accuracy is useful for a quick look into classifier performance, but it can hide problems with classifier generality—especially if the dataset is skewed. Because of this, confusion matrices are used to look deeper into classifier performance on a class-by-class basis. Not only is overall average accuracy given, but the confusion matrix can show how often class A is mislabeled as class B and vice versa for every class in the dataset. Because classifier type, amount of DR applied, and dataset are all varied, 72 confusion matrices were constructed in total for this thesis. Each confusion matrix generally displays the same type of information, so amount of DR applied is fixed at 50% and only the AEBS method is directly compared to the NNBS method for the sake of brevity. Refer to figures 4.6–4.11 for confusion matrices varying dataset and classifier type.

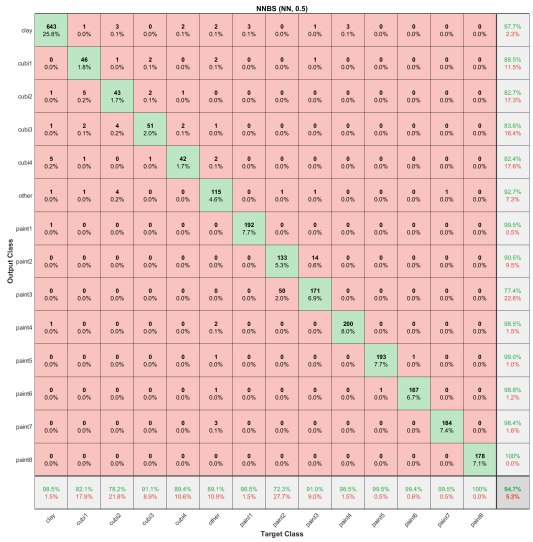

Figure 4.6

NNBS Confusion Matrix (DFF Classifer, ERDC Dataset, 50% Applied DR)

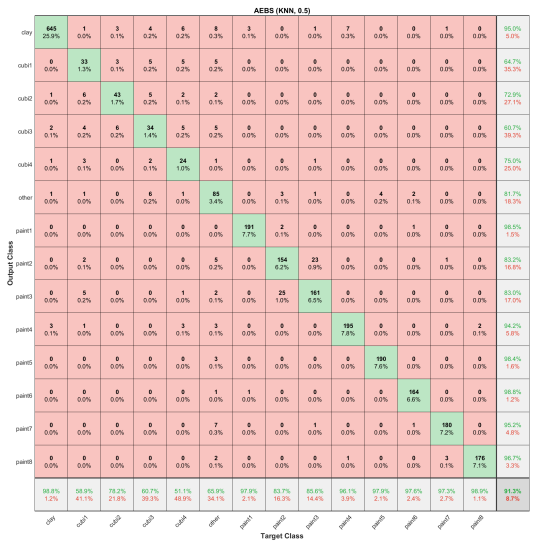

Figure 4.7

AEBS Confusion Matrix (KNN Classifer, ERDC Dataset, 50% Applied DR)

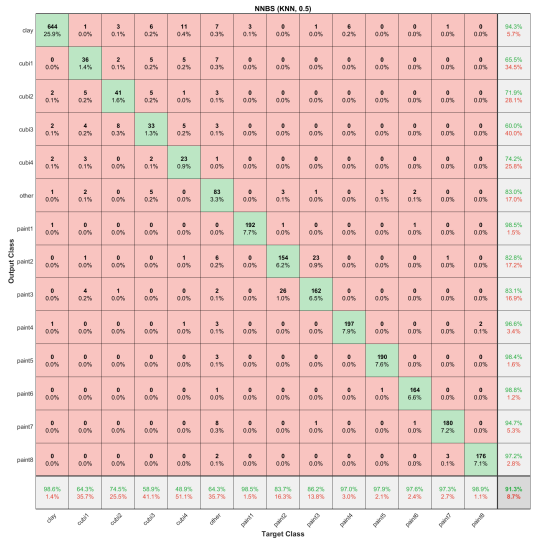

Figure 4.8

NNBS Confusion Matrix (KNN Classifer, ERDC Dataset, 50% Applied DR)

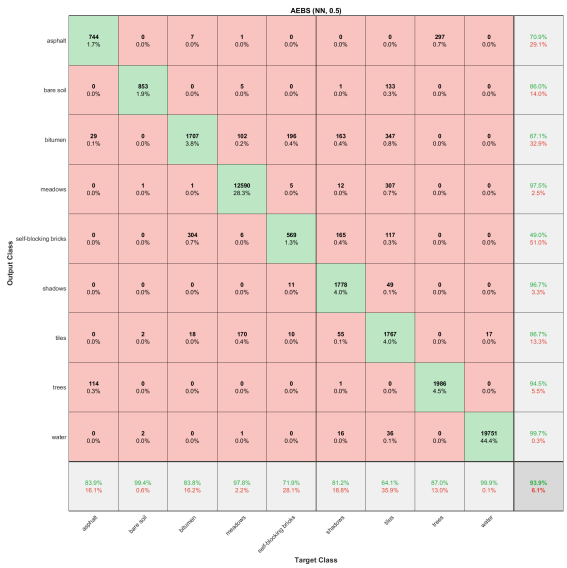

Figure 4.9

AEBS Confusion Matrix (DFF Classifer, Pavia City Center Dataset, 50% Applied DR)

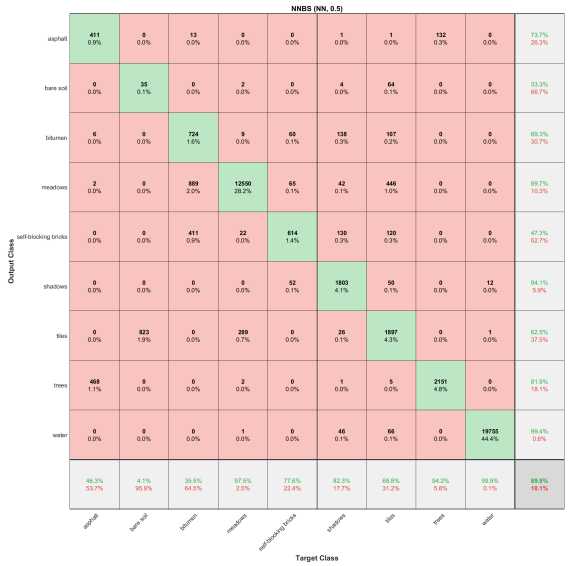

Figure 4.10

NNBS Confusion Matrix (DFF Classifer, Pavia City Center Dataset, 50% Applied DR)

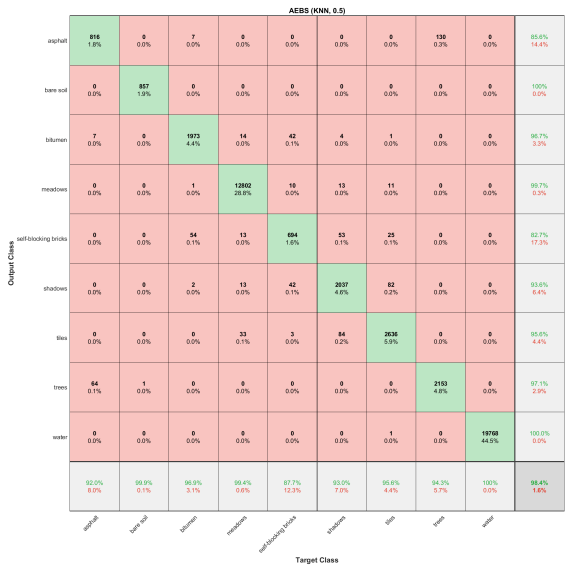

Figure 4.11

AEBS Confusion Matrix (KNN Classifer, Pavia City Center Dataset, 50% Applied DR)

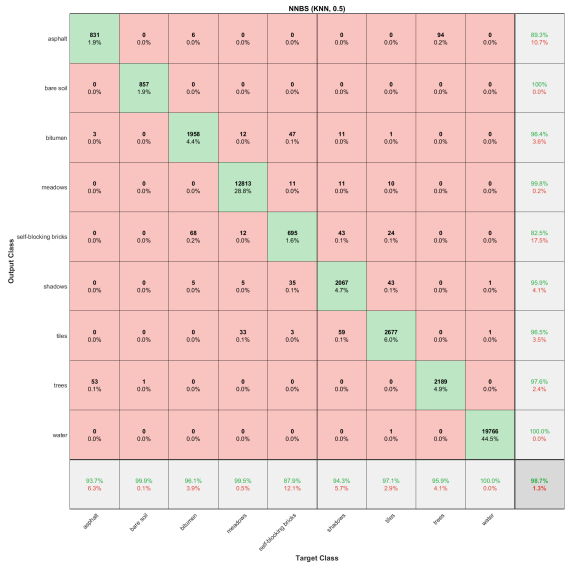

Figure 4.12

NNBS Confusion Matrix (KNN Classifer, Pavia City Center Dataset, 50% Applied DR)

Finally, the selection results of each method are plotted against the average radiance of each hyperspectral endmember. With this, it is easier to compare what each method focuses on in terms of the data that is presented to it. As NNBS and its direct comparison to AEBS is the focus of this thesis, the plots directly comparing these two methods are listed in figures 4.13 and 4.14. For NNBS comparisons to ICABS and OPBS, refer to appendix B.

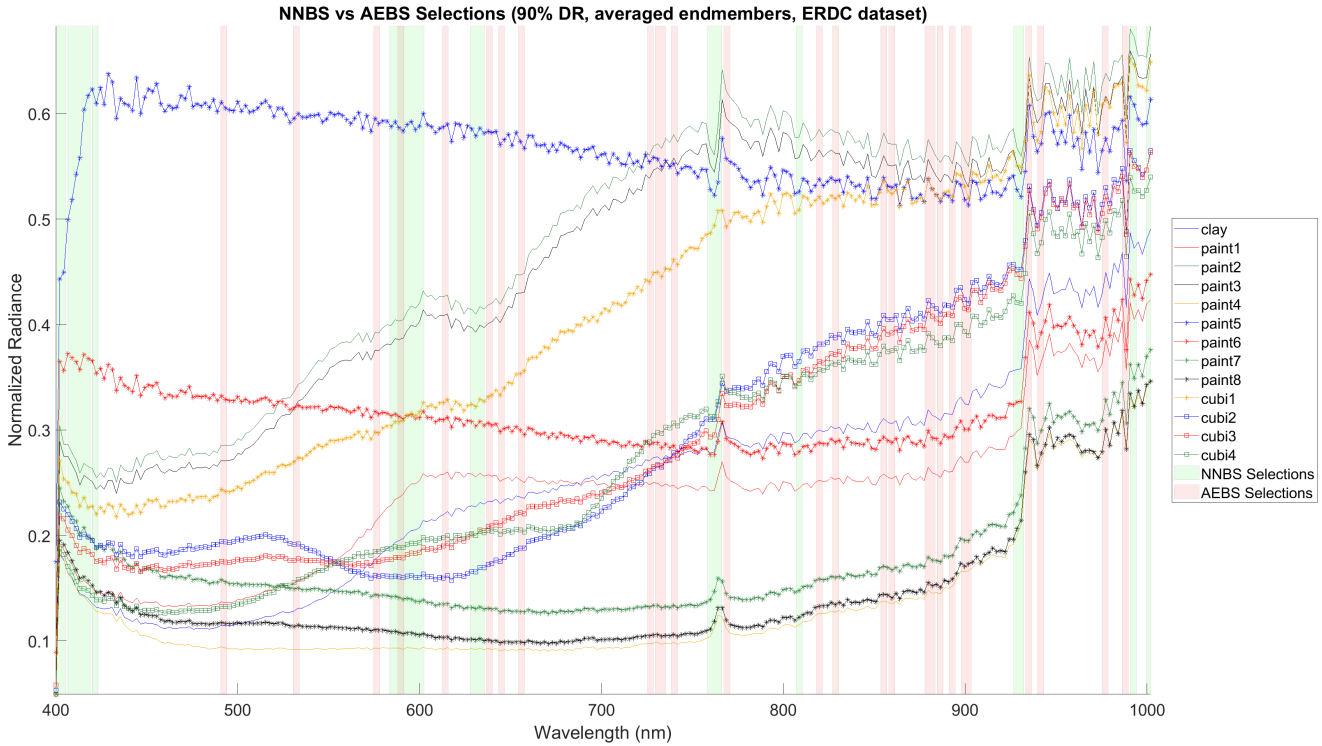

Figure 4.13

NNBS vs AEBS Selections (ERDC Dataset, 90% Applied DR)

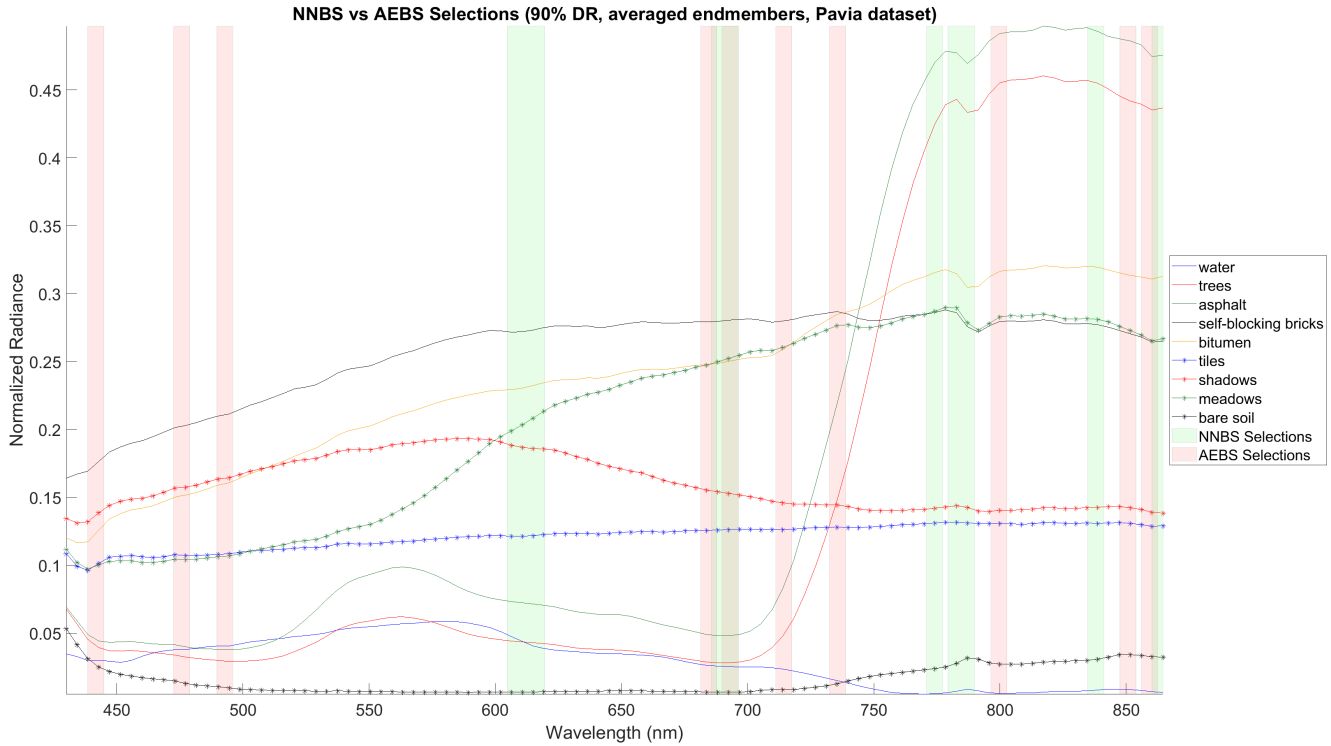

Figure 4.14

NNBS vs AEBS Selections (Pavia Dataset, 90% Applied DR)

4.2 Discussion

For the ERDC dataset, the majority class—clay—was subsampled to allow for the neural network classifier as well as the overall accuracy results to not be dominated by the overwhelming presence of that class. Even though the clay class if subsampled, it remains the majority class during testing. The rest of the objects in the scene are low emissivity coatings that generally vary in amount of light reflected, coatings that somewhat match the signature of clay, and living materials. With this in mind, discriminators intuitively focus on the SWIR bands for living materials and bands that differ provide the most difference between a clay color and the targeted material to be detected.

The Pavia City Center dataset is much more well-balanced when compared to the ERDC dataset. The sensor used in the collectio of this dataset only covers up to 860 nm which is just a bit into the near IR range. The classes in this dataset are of a much higher variety covering two types of living materials—trees and meadows—with the rest being either water or a type of man-made material such as asphalt or bitumen. With this in mind, the selected bands in general focus on the near IR for infromation on living-materials, some bands in the blue wavelengths in order to classify the majority class—water—and various methods select different bands in order to gather some information to discriminate between non-organic materials like tiles (metal sheets), self-blocking bricks, and bricks.

ICA-based Band Selection (ICABS) focuses on finding a components comprising a transform that maximizes the kurtosis of the data. Essentially, the data is assumed to be a mixture of non-Gaussian signals and kurtosis is used as a measure of non-normality to find those signals. In hyperspectral data, each pixel is some mixture of the materials present in the scene. ICABS finds

bands that best separate the signals in each pixel throughout the entire dataset. Although not entirely the same, this method is used as stand-in method alongside NNBS for an LDA-based approach in which bands are selected that best maximize class separability. In general, this performs very well. In both datasets, ICABS selects areas that allow for the best separability between endmembers. This typically occurs in the wavelengths corresponding to blue in each dataset. It should be noted that for the Pavia City Center dataset, ICABS hyperfocuses on the areas corresponding to heavy local variance for the `water` class. It is possible that this is the reason that ICABS is heavily dependent on variations of applied DR as seen in 4.4. Refer to figures B.1 and B.2 for plots showing the selection ICABS provides when limited to 10% of the original dimensionality.

Orthogonal Projection-based Band Selection (OPBS) is a fast version of Maximal Ellipsoidal Volume (MEV) combined with sequential forward search (MEV-SFS). MEV-SFS adds to a subset band by band such that each new band maximizes the amount of information in the subset while minimizing the correlation between bands. Because OPBS defines maximizing information as maximizing local variance, OPBS selections are used as a stand-ing for PCA-based approaches. As such, OPBS selections prioritize regions where the endmember curves have the most local variance. In the ERDC dataset, OPBS focuses much more strongly on the near IR wavelengths as this is where the endmembers vary the most as seen in figure B.3. In the Pavia City Center dataset, the band rankings oscillate between the longest and shortest wavelengths consistently despite the amount of DR applied changing. This is mainly due to the fact that water absorption bands are already taken out of this dataset as well as the fact that the poorer spectral resolution does not capture the variance as well as the sensor used in the ERDC dataset.

Autoencoder-based Band Selection (AEBS) trains an autoencoder to reconstruct the hyperspectral dataset. The weights connecting the input to the network are used as a basis for selection, i.e., the stronger the connections between an input neuron and the network, the more important that band is to image reconstruction. The selections made by AEBS are very similar to those made by OPBS although with a stronger emphasis on more equally sampling the original hyperspectral endmembers allowing for better compression. In particular, AEBS seems to target regions that offer the most entropy as these areas would contain the most amount of information for compression and therefore, maximize the image reconstruction objective function.

Neural Network-based Band Selection (NNBS) implements the same process as AEBS but with a general DFF architecture focused on hyperspectral pixel classification. Due to the nature of minimizing the loss function, NNBS prioritizes weights that transform the input to make classification easier. In other words, NNBS learns weight transforms that increase class separability much like LDA. In the ERDC dataset, the selected bands are biased more towards the shorter wavelengths as these regions offer the most class separability of endmembers. Because living materials are anomalies within this dataset, NNBS also prioritizes selecting green and near IR wavelengths to account for classifying these endmembers. In the Pavia City Center dataset, NNBS mostly ignores the shorter wavelengths in favor of the green, red, and near IR wavelengths. This is evident due to the endmembers being very close together in the shorter wavelengths and exhibiting much more separability in the longer wavelengths.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

At the outset, this thesis sought to explore three basic questions. One, can the AEBS method be extended to any architecture without producing selections that did not help classifier performance? Two, does the new method, NNBS, generate novel selections in comparison to AEBS, or do both methods produce the same selections? And finally, do these different selection produce better results? In other words, the evaluating task required pixel classification to be improved. NNBS uses pixel classification to perform band selection. With this in mind, does NNBS produce better results than AEBS?

First, to answer the question of machine learning band selection that is independent of autoencoder architecture, the selections made by NNBS need to be consistent from run to run. As there is a stochastic element to NNBS, there is the concern that the selections in NNBS are giberish and that they could vary due to random initialization. During testing, NNBS was performed multiple times in order to accomplish two things: ensure that the selections do not vary run to run and in the case that they do vary, is the variation significant enough to render NNBS selections useless. After observing the results, the selections do not significantly vary from run to run as long as the network is trained to a sufficiently small error which varies depending on architecture and the dataset. Any variations present are within the actual rankings of the selections. Experimentally, the rankings

vary on average one rank up or down. While it is possible to overfit the network to allow for even more consistency in rankings, it is unclear if this is preferable for the selections themselves. All of these observations of NNBS apply to AEBS as both methods are based on a neural network architecture optimizing a cost function. In short, NNBS produces consistent selections implying that the selections can be repeated and are selected intelligently.

After demonstrating that NNBS produces consistent selections, it is desireable to have these selections be unique from AEBS. If NNBS results match AEBS, then architecture does not matter, and the "best" deep learning-based band selection method would use the architecture that converges the faster or converges more consistently. However, selections produced by NNBS do not match those produced by AEBS as is evident by figures 4.13 and 4.14. This can be intuited by simply acknowledging that NNBS utilizes a DFF that is optimizing a cost function depending on pixel classification accuracy, while AEBS utilizes an autoencoder optimizing a cost function that minimizes image reconstruction error. With this, it is clear that NNBS and AEBS prioritize different aspects of hyperspectral endmembers and have different definitions of what is "important" information for band selection purposes.

Finally, does NNBS produce better results than AEBS or the other methods discussed in this thesis? NNBS is shown to slightly outperform other methods, but this is not consistent when varying applied DR and classifier type. All of these selection methods provide differing results for the "best" subsets of the original bands. For instance, in the Pavia City Center dataset, AEBS considers band 102 as the worst band, while NNBS considers band 102 to be the best. When comparing classifier performance on this dataset, AEBS selections perform better when the classifier is a DFF network, and NNBS selections perform better when the classifier is a k-means

clustering algorithm. With this in mind, it is important to note that each selection is shown to be at a minimum competitive with other methods as shown by the figures in sections 4.1 and 3.4. Instead, it is more important to consider the application for which the selections are the most important. If the application of band selection is to design a cheaper sensor for detecting certain materials in an image, then a band selection method that utilizes band grouping as well as one that takes into account class separability. If the application of band selection is to retain as much information as possible, approaches described in OPBS and AEBS would be more valuable.

5.2 Future Work

Before concluding this thesis, it is important to discuss where this research can move forward into the future. For one, the most obvious take away is that weights learned via statistical transforms and machine learning methods are important for analyzing data. As such, pre-trained networks' weights can be used as a basis for analyzing all sorts of data. In the context of this thesis, neural network weights hyperspectral endmembers can be analyzed to determine regions of the spectral curves providing the most discriminative information for classification. Sensors can be selected and tuned such that the spectral regions that most greatly discriminate weeds from important crops are prioritized over bands that only introduce confusion. To step outside the hyperspectral modality, CNN feature maps can be used to analyze important geometric features of objects such that important features can be prioritized in the design of future algorithms.

A natural step in the future for the topics discussed in this thesis is that neural networks can analyze their own parameters during runtime to make intelligent decisions during training. For example, this thesis used a DFF network trained on the entire dataset to perform band selection.

A separate DFF network is trained on the reduced subset in order to compare selections between different methods. An implementation of NNBS can be experimented with such that the input weights are analyzed during runtime in order to preemptively self-prune input nodes that are appearing to be unimportant. This sort of self-pruning can be applied to other networks in the context of analyzing important features as well.

Finally, this thesis focused on analyzing input weights to make decisions about the input data. However, a neural network uses more than the input weights in order to transform the input data to its appropriate output. Therefore, the weights between each hidden layer holds important information to make the network function. As alluded to in section 3.3 and experimented with in [21], hidden layer weight information can be aggregated with the information present in input weights in order to track input data information throughout the entire network ending at the output. This allows the input data importance to be compared directly in terms of the output. With this, steps into explainable neural networks can be taken which is important for applying machine learning to delicate tasks such as cancer identification and autonomous vehicles.

In this thesis, various band selection methods are tested to the Pavia City Center dataset as well as a novel dataset captured courtesy of the Engineer Research and Development Center. Furthermore, an extension of the AEBS method called NNBS is applied to these datasets. When spectral subsets are used in a DFF and k-means clustering classifer, each band selection method optimizes different types of information with NNBS is shown to be competitive with other state-of-the-art methods. It is clear from the results that each method prioritizes different regions showcasing various characteristics of each hyperspectral endmember. With NNBS, future work in

explainable machine learning and of utilizing machine learning to analyze input data rather than simply using the input to produce a desired output can be considered.

REFERENCES

[1] M. Ahmad, M. Alqarni, A. Khan, R. Hussain, M. Mazzara, and S. Distefano, "Segmented and Non-Segmented Stacked Denoising Autoencoder for Hyperspectral Band Reduction," *Optik*, vol. 180, 11 2018.

[2] H. Ayman, W. Xiong, F. He, L. Yang, and M. Crawford, "Improving Orthorectification of UAV-Based Push-Broom Scanner Imagery Using Derived Orthophotos From Frame Cameras," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, 02 2016, pp. 1–15.

[3] J. Ball, T. West, S. Prasad, and L. Bruce, "Level set hyperspectral image segmentation using spectral information divergence-based best band selection," 08 2007, pp. 4053 – 4056.

[4] W. Boulila, M. Driss, M. Al-Sarem, F. Saeed, and M. Krichen, "Weight Initialization Techniques for Deep Learning Algorithms in Remote Sensing: Recent Trends and Future Perspectives," 02 2021.

[5] B. Chandra and R. Sharma, "Exploring autoencoders for unsupervised feature selection," 07 2015, pp. 1–6.

[6] H. Du, H. Qi, X. Wang, R. Ramanath, and W. Snyder, "Band selection using independent component analysis for hyperspectral image processing," 11 2003, pp. 93– 98.

[7] I. Fodor, "A Survey of Dimension Reduction Techniques," 2002.

[8] J. E. Fowler, "Compressive pushbroom and whiskbroom sensing for hyperspectral remote-sensing imaging," *2014 IEEE International Conference on Image Processing (ICIP)*, 2014, pp. 684–688.

[9] M. Habermann, V. Fremont, and E. Shiguemori, "Unsupervised Band Selection in Hyperspectral Images using Autoencoder," 01 2018, pp. 6 (6 pp.)–6 (6 pp.).

[10] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations*, 12 2014.

[11] K. Koonsanit, C. Jaruskulchai, and A. Eiumnoh, "Band Selection for Dimension Reduction in Hyper Spectral Image Using Integrated InformationGain and Principal Components Analysis Technique," *International Journal of Machine Learning and Computing*, vol. 3, 01 2012, pp. 248–251.

[12] B. Lu, Y. He, and P. D. Dao, "Comparing the Performance of Multispectral and Hyperspectral Images for Estimating Vegetation Properties," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 6, 2019, pp. 1784–1797.

[13] R. Lukac and K. Plataniotis, "Color Filter Arrays for Single-Sensor Imaging," 01 2006, vol. 2006, pp. 352 – 355.

[14] S. Madry and J. N. Pelton, "Electro-optical and Hyper-spectral Remote Sensing," 2013, pp. 729–738.

[15] S. Mehta, A. Patel, and J. Mehta, "CCD or CMOS Image sensor for photography," *2015 International Conference on Communications and Signal Processing (ICCSP)*, 2015, pp. 0291–0294.

[16] E. Mohamed, A. Saleh, A. Belal, and A.-A. Gad, "Application of near-infrared reflectance for quantitative assessment of soil properties," *The Egyptian Journal of Remote Sensing and Space Science*, vol. 21, 02 2017.

[17] M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, no. 4, 1993, pp. 525–533.

[18] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks,", 2015.

[19] D. Prasvita, "Post-processing and band selection for hyperspectral image data classification with AdaBoost.MH," 11 2017, pp. 6–13.

[20] N. Rani, V. R. Mandla, and T. Singh, "Evaluation of atmospheric corrections on hyperspectral data with special reference to mineral mapping," *Geoscience Frontiers*, vol. 8, no. 4, 2017, pp. 797–808, Special Issue: Deep Seated Magmas and Their Mantle Roots.

[21] P. Ribalta Lorenzo, L. Tulczyjew, M. Marcinkiewicz, and J. Nalepa, "Hyperspectral Band Selection Using Attention-Based Convolutional Neural Networks," *IEEE Access*, vol. PP, 03 2020, pp. 1–1.

[22] C. Rodarmel and J. Shan, "Principal Component Analysis for Hyperspectral Image Classification," *Surv Land inf Syst*, vol. 62, 01 2002.

[23] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network," *Physica D: Nonlinear Phenomena*, vol. 404, 2020, p. 132306.

[24] C. Sorzano, J. Vargas, and A. Montano, "A survey of dimensionality reduction techniques," 03 2014.

[25] W. Sun and Q. Du, "Hyperspectral Band Selection: A Review," *IEEE Geoscience and Remote Sensing Magazine*, vol. 7, no. 2, 2019, pp. 118–139.

[26] J. Tschannerl, J. Ren, J. Zabalza, and S. Marshall, "Segmented Autoencoders for Unsupervised Embedded Hyperspectral Band Selection," 11 2018, pp. 1–6.

[27] Y. Zhan, D. Hu, H. Xing, and X. Yu, "Hyperspectral Band Selection Based on Deep Convolutional Neural Network and Distance Density," *IEEE Geoscience and Remote Sensing Letters*, vol. PP, 11 2017, pp. 1–5.

[28] W. Zhang, X. Li, Y. Dou, and L. Zhao, "A Geometry-Based Band Selection Approach for Hyperspectral Image Analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. PP, 03 2018, pp. 1–16.

APPENDIX A

FORMULATING BACKPROPAGATION USING LINEAR ALGEBRA

A.1 Forward Propagation using Linear Algebra

This appendix aims to formulate the entirety of neural networks in terms of linear algebra using the derivation provided in sections 2.3 and 2.4. Implementations of neural networks generally use the linear algebra formulation as it is more compact, and there exist more efficient methods of computing operations such as inverse matrices and matrix multiplication leading to faster computation time.

For this formulation, $\mathbf{X}$ will be an input data matrix to layer l whose columns are individual data vectors of length N such that $[\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M]$. Therefore, $\mathbf{X}$ is a NxM data matrix with M entries and N variables. It is important to view the data this way to account for offline training. If the minibatch size is equal to one, then the data matrix collapses to a single data vector, effectively implementing online training. This allows for the formulation in this appendix to be used regardless of minibatch size.

Similarly, a weight matrix $\mathbf{W}_{l-1,l}$ between layers $l-1$ and l can be described in a similar way such that the weights of a neuron in layer $l-1$ are organized into a column vector $\mathbf{w}$. With each column containing P weights connecting an arbitrary neuron in layer $l-1$ to all the neurons in layer l, the weight matrix of size PxN is formulated as $\mathbf{W}_{l-1,l} = [\mathbf{w}_1, \mathbf{w_2}, ..., \mathbf{w_N}]$. Furthermore, to incorporate the idea of a bias term, $\mathbf{X}$ and $\mathbf{W}_{l,l+1}$ are modified such that $\mathbf{X} = \left[\overrightarrow{\mathbf{1}}, \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M\right]$ and $\mathbf{W}_{l-1,l} = [\mathbf{b}, \mathbf{w}_1, \mathbf{w_2}, ..., \mathbf{w_N}]$ such that $\mathbf{b}$ is a vector of biases for the P neurons in layer $l+1$. Forward propagation from layer $l-1$ and to layer l creates the unactivated output matrix $\mathbf{Z}$ of size PxM as defined in equation A.1.

$$\mathbf{Z} = \mathbf{W}_{l-1,l}\mathbf{X} \tag{A.1}$$

The unactivated output matrix $\mathbf{Z}$ is then passed through an activation function, $\varphi(\cdot)$, that acts on every element of the matrix. This produces an activated output matrix that can be seen in equation A.2.

$$\mathbf{Y} = \varphi(\mathbf{Z}) = \varphi\left(\mathbf{W}_{l-1,l}\mathbf{X}\right) \tag{A.2}$$

An alternative definition of forward propagation can be found in equation A.3 that more clearly defines the iterative role of the matrix computations throughout a network. If the layer is an input layer, i.e. $l = 1$, then $\mathbf{A}_{l-1} = \mathbf{A}_0 = \mathbf{X}$ is the input to the network. Similarly, if the layer is the output layer, i.e. $l = L$, then $\mathbf{A}_l = \mathbf{A}_L = \mathbf{Y}$ or the output of the network. Refer to figure A.1 for a labeled reference of an arbitrary layer in a fully connected feedforward network.

$$\mathbf{A}_l = \varphi(\mathbf{Z}_l) = \varphi\left(\mathbf{W}_{l-1,l}\mathbf{A}_{l-1}\right) \tag{A.3}$$

A.2 Backward Propagation using Linear Algebra

Similar to section 2.4, backpropagation basically sums up to determining what the error signal is for each layer. Each data entry in the input matrix $\mathbf{X}$ will create an entry in the output matrix $\mathbf{Y}$. A consequence of this is that the error signal for each neuron will create an error vector, $\boldsymbol{\delta}$. For each entry in $\mathbf{Y}$, there will be an associated error vector entry organized in $\boldsymbol{\Delta} = [\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, ..., \boldsymbol{\delta}_M]$.

Reformulating the equations for δ_j as described in equation 2.9 results in equation A.4 describing the error matrix $\boldsymbol{\Delta}_{l,l-1}$ associated between layers $l-1$ and l. Note that the matrix $\mathbf{O}$ is a matrix containing the training labels to be compared to the output of the network, matrix $\mathbf{Y}$.

$$\begin{aligned} \boldsymbol{\Delta}_{L-1,L} &= (\mathbf{O} - \mathbf{Y}) \odot \varphi'(\mathbf{Z}_L) \text{ if } l = L \text{ is an output layer} \\ \boldsymbol{\Delta}_{l-1,l} &= \left(\boldsymbol{\Delta}_{l,l+1}^T \mathbf{W}_{l,l+1}\right) \odot \varphi'(\mathbf{Z}_l) \text{ if } l \text{ is a hidden layer} \end{aligned} \tag{A.4}$$

The $\odot$ operator is the Hadamard product, or put simply, it is element-wise multiplication between matrices of the same size. It is important to note that in practical applications of back-propagation in linear algebra, the $\varphi'(\mathbf{Z_l})$ term is calculated during forward propagation as the network already has easy access to the unactivated outputs matrix $\mathbf{Z}_l$. After the $\mathbf{\Delta}$ matrices have been backpropagated, the update equation for the weight matrix $W_{l-1,l}$ between two layers then becomes the following equation which is a reformulation of equation 2.10. Again, refer to figure A.1 for a representation of where these matrices are located in an arbitrary layer of a network.

$$\mathbf{W}_{l-1,l}^{t+1} = \mathbf{W}_{l-1,l}^{t} - \eta \mathbf{\Delta}_{l-1,l} \mathbf{A}_{l-1}^{T} \tag{A.5}$$

To implement various learning strategies such as ADAM, terms that implement momentum, L2-regularization, or sparsity regularizaition using Kullback-Leibler (KL) divergence must be reformulated to take into account matrix computations using the output matrix $\mathbf{Y}$. Once these terms are incorporated into the cost function, the above equations proceed as defined.

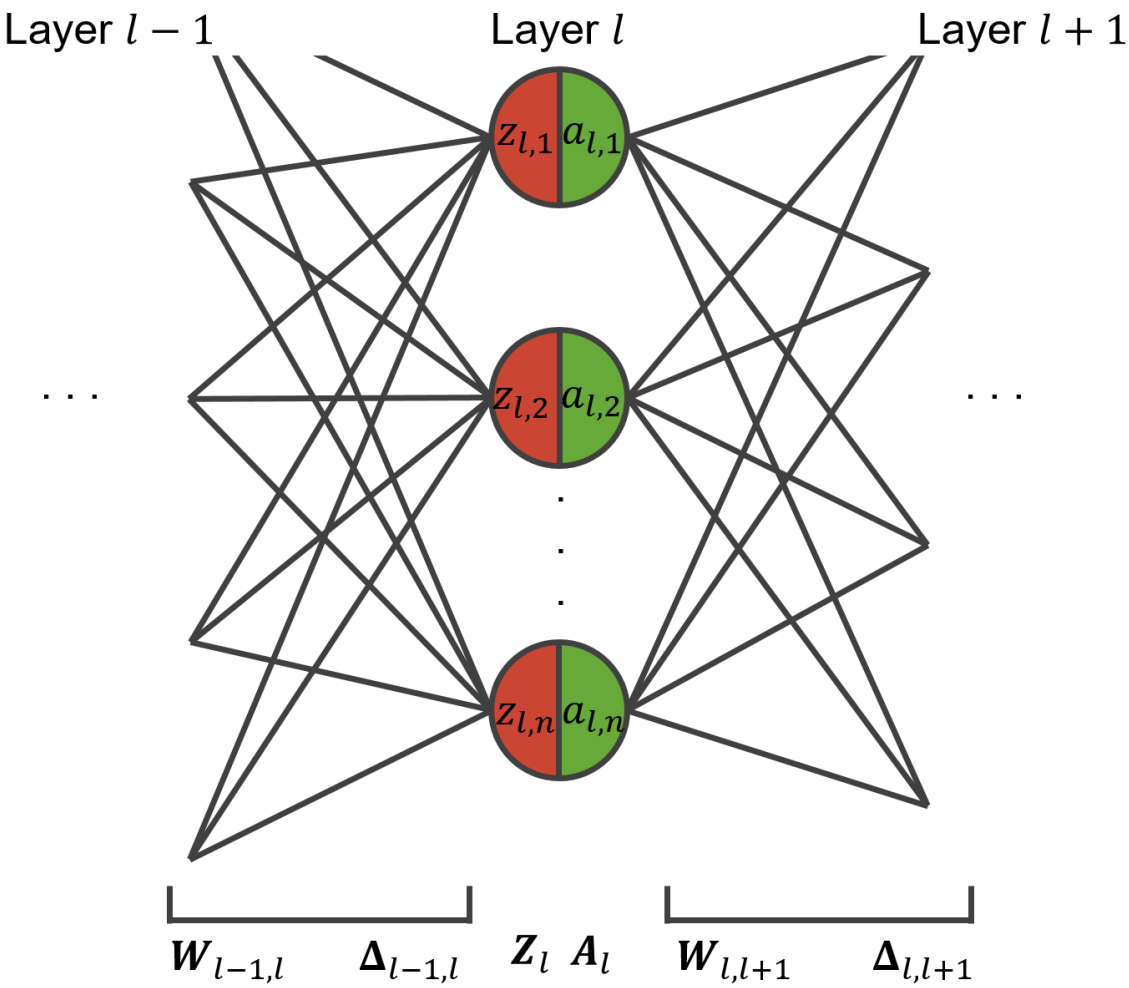

Figure A.1

Generic Labeled Layer of a Fully Connected Feedforward Network

APPENDIX B

ADDITIONAL RESULTS

B.1 Selection Comparison Plots

Below are additional plots for comparing various selections to the selections from the NNBS method.

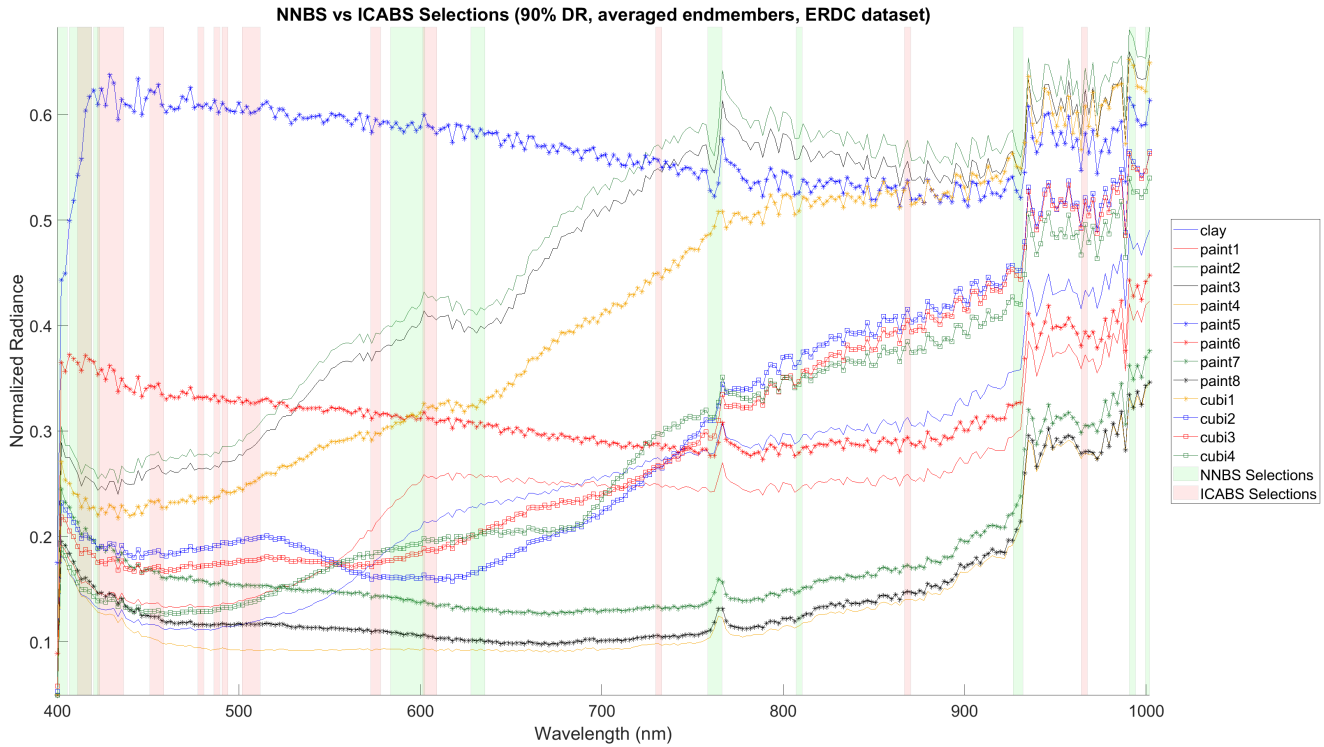

Figure B.1

NNBS vs ICABS Selections (ERDC Dataset, 90% Applied DR)

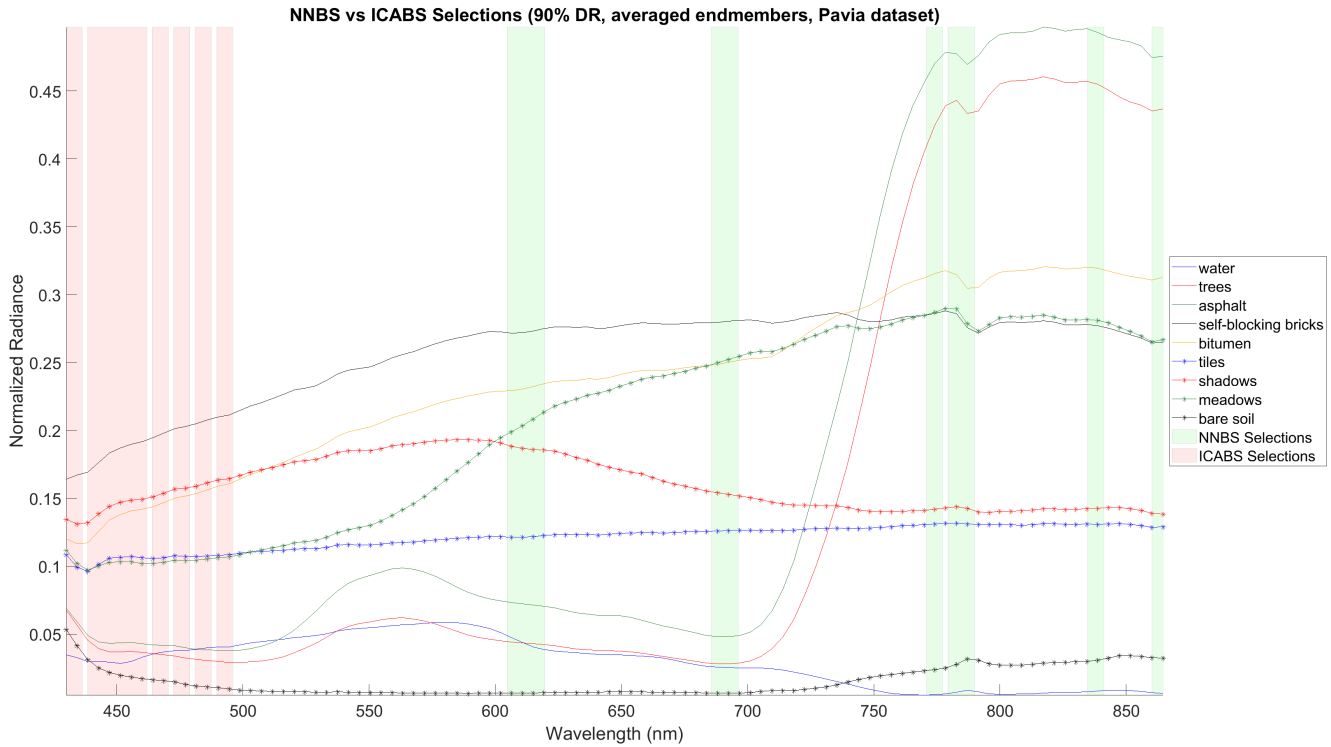

Figure B.2

NNBS vs ICABS Selections (Pavia Dataset, 90% Applied DR)

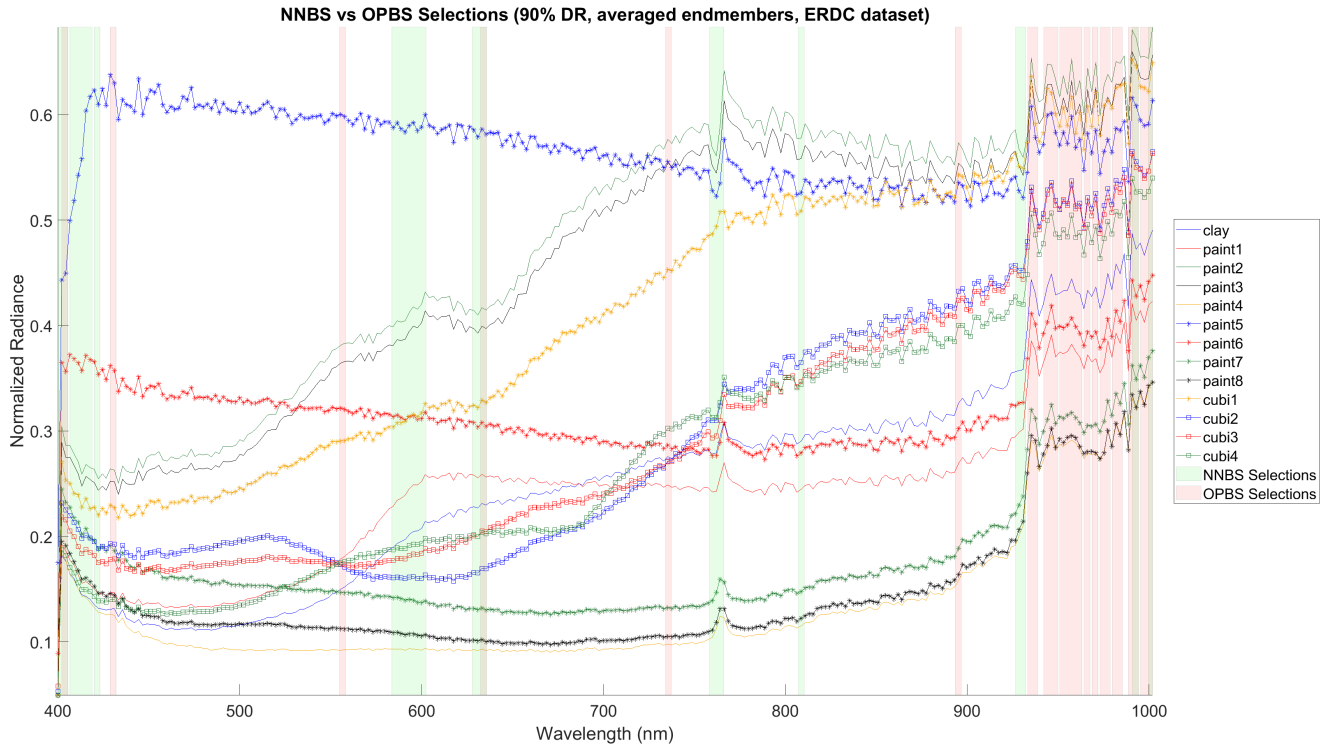

Figure B.3

NNBS vs OPBS Selections (ERDC Dataset, 90% Applied DR)

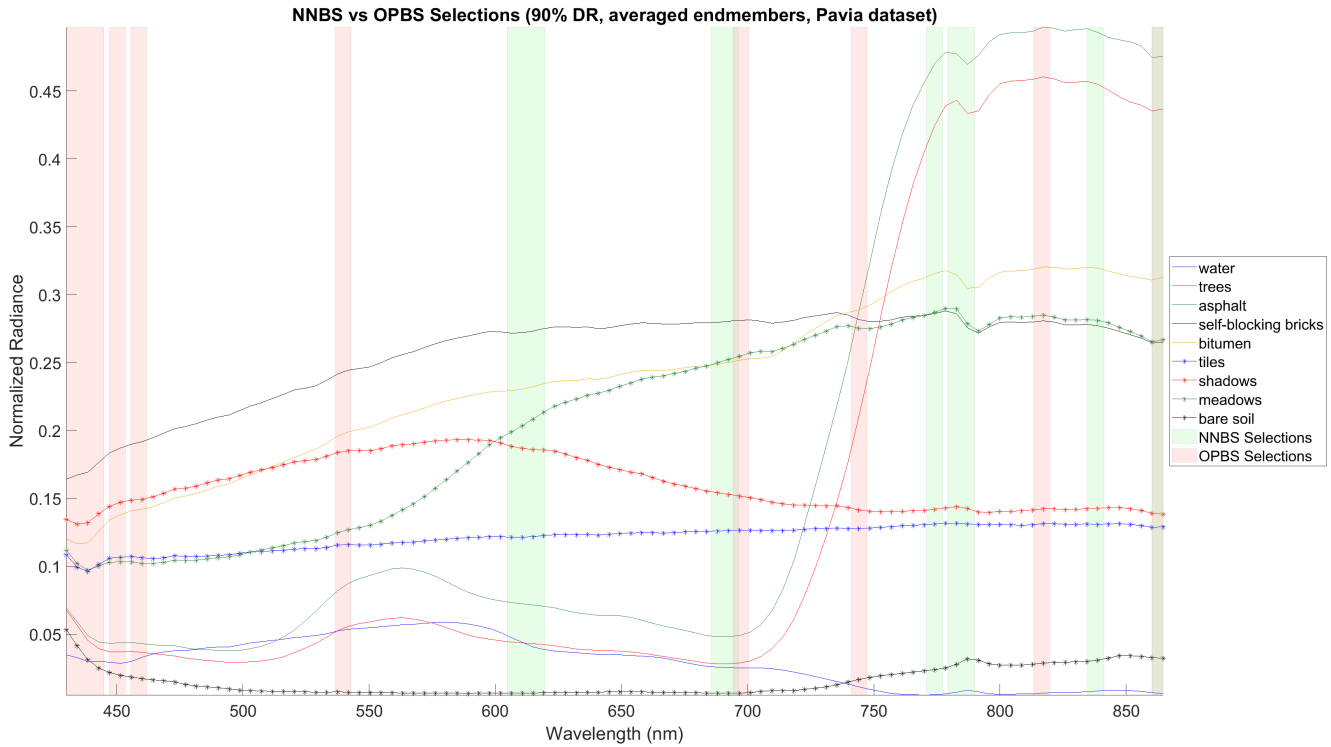

Figure B.4

NNBS vs OPBS Selections (Pavia Dataset, 90% Applied DR)