ROBUST COMPRESSION AND CLASSIFICATION OF HYPERSPECTRAL IMAGES

by

Kyle Logan Webster

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

April 2022

ACKNOWLEDGMENTS

I would like to thank Dr. John Sheppard, my advisor and committee chair, for his support and guidance throughout this process, and my fellow members of the Numerical Intelligent Systems Laboratory at Montana State University for their valuable comments and suggestions.

I would also like to thank the members of my thesis committee, Dr. David Millman and Dr. Joseph Shaw, and the members of the Optical Technology Center at Montana State University, especially Riley Logan, for all of the information and help during the development of this work.

I cannot express my thanks to my significant other Dani Hatfield for her constant support and inspiration during this thesis writing process. Finally, I would like to thank my parents for their unconditional love and support of my academic career.

TABLE OF CONTENTS

TABLE OF CONTENTS – CONTINUED

TABLE OF CONTENTS – CONTINUED

LIST OF TABLES

LIST OF TABLES – CONTINUED

LIST OF FIGURES

LIST OF FIGURES – CONTINUED

LIST OF ALGORITHMS

ABSTRACT

Hyperspectral images are a powerful source of spectral data that have been utilized in a wide array of applications. The large size of hyperspectral images limits the applicable uses and necessitates effective compression methods. While many spectral-spatial compressors have been proposed in the past, there has been little work on the benefits of a spectral-only strategy. A spectral-only strategy not only has compressive capabilities but would also allow the classification of the compressed images, making the contributions of this thesis multi-fold. We present a Long Short Term Memory Autoencoder designed for the spectral compression of hyperspectral images. We show that this network can compress the images effectively with low reconstruction error, as well as require fewer training parameters to compress when compared to existing spectral-spatial compression methods. Existing learned compression models often require many of the pixels to be used to train an image. We demonstrate that our proposed network does not suffer a reduction in compression performance by reducing the number of training examples. Existing compression techniques are limited in capability by their inclusion of spatial information, requiring reprocessing for all images that have sufficiently different scenes. We demonstrate the proposed network's robustness by training a single model for use in multiple scenes without the requirement of retraining the model from scene to scene. Furthermore, using the feature extracting capabilities of an autoencoder, we analyzed the capabilities of the compressed image as a feature set for classification. Experimental results demonstrate that the unsupervised compressed features generated can be utilized for supervised machine learning classification tasks. We also demonstrate that the robustness of the compressor allowed for a single network to not require being retrained for compressing and then classifying new images without significant loss.

CHAPTER ONE

INTRODUCTION

Hyperspectral imaging systems have become a widely used source of information due to their ability to capture detailed spectral information based on hundreds of spectral bands [28]. With the increased level of spectral detail, users of hyperspectral imaging systems are capable of using a wide array of algorithmic techniques to better characterize and differentiate objects within an image based on the subtle differences in the spectral bands that could often not be captured with conventional cameras [39]. The images captured by these systems, called hyperspectral images (HSI) come at the cost of significantly higher storage requirements as well as the computational cost to analyze these images [82].

Hyperspectral imagers utilize a non-destructive spectroscopic technique that remotely captures spectral information [81]. One of the most common uses of this sensing technique found in the literature is the aerial and spatial sensors that capture land use and land cover information over large areas [50]. With recent work in the increasing accessibility of the sensors through reductions in economic and computational time savings, many diverse applications have found benefits of utilizing this sensing technology with examples such as food quality and safety [74], agricultural field analysis [78], water safety analysis [84], and medical applications [79].

While hyperspectral imaging technology has existed since the 1980s [101], it was not until a few years ago that the use of hyperspectral imaging systems was no longer limited to large platforms like aircraft and satellite platforms due to their large size, operational difficulties, and economic costs [115]. Recent advancements in sensing technology have produced small and low-cost hyperspectral sensors available for commercial use [61]. These

Figure 1.1: HYMSY hyperspectral sensor mounted on an Aerialtronics Altura AT8 v1 octocopter UAV [114]

imagers have the potential to be mounted on drone platforms with some images being mounted on tripods [42] which can capture a wider array of subjects to be analyzed with this technology.

As an example, the Resonon Pika L hyperspectral image platform is a small imager that can be mounted on a small drone [46]. The Kochia dataset, captured by Scherrer *et al.*, utilized this platform to analyze the differences between three different classes of herbicide-resistance of the Kochia weed (*Bassia scoparia)* [97]. The Hyperspectral Mapping System (HYMSY) sensor, as shown in Figure 1.1, can be mounted on a small commercially available drone for precision agriculture purposes. The sensor has both a GPS navigation system, a conventional camera, and a hyperspectral imager mounted on the small frame [114].

1.1 Image Compression

Alongside the work to improve the capabilities and reduce the cost of hyperspectral imagers, work has to be done to reduce the cost of handling the captured images [21]. In

general, the increasing number of spectral bands from the fine spectral resolutions captured and the increased spatial resolution of the images quickly increase the size of the images [32]. The storing of the spectral information of these images is often represented as a three-dimensional data cube where the x and y axes represent the spatial coordinates of a pixel, and the z axis contains the spectral bands [5]. Often, a given band at a given pixel is represented either by a 12-bit value or a 16-bit value. With this data cube representation, these images often exceed hundreds of megabytes (MB) [37]. For example; the Hyperspectral Digital Imagery Collection Experiment (HYDICE) imager, developed in 1994, captured a single 1280×307 pixel image of the National Mall in Washington DC over 191 usable bands requiring 145 MB of storage [12].

Given the large size of the images, it is unrealistic to utilize the images without a compression process in both data transfer as well as when stored in a database for most applications [26]. Given that classifiers for these images significantly benefit from the fine levels of information contained within these images, we focused on the development of a compressor that has a low amount of reconstruction error without sacrificing the level of compression that many use cases require. In particular, we present the application of a Long Short Term Memory Autoencoder model that compresses spectral information. The bands of each pixel are systematically compressed into a small set of values defined by the size of the network encoding layer. We hypothesize that this architecture will be able to efficiently compress the images with a lower amount of reconstruction error when compared to contemporary compression techniques. We also hypothesize that the reduction in reconstruction error will still allow our proposed network to perform similarly to contemporary techniques with regard to the compression rate. Here, we talk about efficiency in network performance measured by the number of training parameters and the computational operations the model requires.

In a spatial-spectral compression model, the model considers not only the spectral

information of the pixel but also the bands of the surrounding pixels to influence the compression process. This dependence on spatial information influences the necessity to retrain models for sufficiently different scenes. If a model uses a spectral only compression model, removing the requirement to retrain our model for sufficiently different scenes, all that the model requires is that each sufficiently different subject is observed during the training process. It stands then that we can generalize the model to be robust against changing scenes by including the subjects of many models in the training process.

With this, we explore a variation in the training process that can compress multiple images with a single model. We also explore another variation in the training process where we take a trained model and compress an unseen image, demonstrating the generality of the model. We hypothesize that the removal of the spatial dependence will allow us to perform these training variations without a significant reduction in compressive capabilities.

Bidirectional LSTM Autoencoders create a duplication of the LSTM network at each layer and invert the order of the input data to better capture the latent information in the data. These networks have shown promise in improving the classification performance of LSTM Autoencoders [35, 71]. With this, we propose experimenting with implementing a Bidirectional LSTM model to perform each of the compression tasks that the proposed model. We hypothesize that the increased complexity and reduced compression rate will lead to a decrease in reconstruction error rates.

While there have been many successful compression approaches, a significant number of the approaches utilize a spatial-spectral compression strategy [24]. While this strategy has been effective for significant levels of compression, compressed images can only be decompressed. Alternatively, compressed images can be utilized as features. This multi-applicability of hyperspectral image compression could lead to increases in computational efficiency and further expand the applications of this imaging technique.

1.2 Classification Problem

In addition to the storage complexity of these images, the high number of spectral bands and the increasing spatial resolution leads to redundancy in the data [1]. This can make models for classifying these images difficult to train efficiently and leads to an increase in the risk of overfitting the data due to the rising spectral and spatial resolution [90]. The rapid increase in the number of parameters is often referred to as the "curse of dimensionality" phenomenon [122] or the "Hughes phenomenon" [112]. This phenomenon describes the observation that to obtain a reliable result, the amount of data required to train these models grows exponentially with the increased number of features. While this redundancy of information may be helpful for information recovery or super-resolution applications [67, 135], in most use cases the prioritization of computational efficiency requires the dimensionality to be reduced.

Feature extraction is the process of taking the features from a dataset to reduce dimensionality, thereby representing them as a smaller set of new features. Often, straightforward spectral methods are implemented with expert knowledge on a specific problem using hyperspectral images. For example, the normalized difference vegetation index (NDVI) uses the difference between near-infrared bands and red bands to quantify vegetation [38]. NDVI is calculated with the equation:

$$NDVI = \frac{(NIR - Red)}{(NIR + Red)}$$

where NIR and Red represent the spectral reflectance measurements acquired in the red (620-750 nm) and near-infrared (800-2,500 nm) regions respectively. While this form of feature extraction requires specific expertise to derive the features, the arithmetic calculation does not necessarily lead to dimensionality reduction. Often, these techniques are used to

obtain additional information to discriminate between land cover objects, and fine-tuning a classification algorithm would still make use of the full spectrum.

Dimensionality reduction, on the other hand, is looking for low-dimensional representations of HSIs [137]. Spectral analysis dimensionality reduction methods can be classified into either unsupervised or supervised. Unsupervised dimensionality reduction does not utilize labeled information to extract low-dimensional data patterns. One of the most common of these methods, principal component analysis (PCA), removes the redundancy in the spectral data and transforms the data into a smaller dimension by mapping to the k principal components [120]. This is accomplished by performing an orthogonal linear transformation to transform the data into a new coordinate system where each coordinate (referred to as principal components) is ranked by variance. PCA often has the limitation of not being able to model nonlinear relationships efficiently [111].

One approach to resolve this limitation has been Stacked Autoencoders (SAE). These networks allow for the modeling of nonlinear relationships to perform dimensionality reduction. Additionally, SAEs hierarchically extract deep features, which allows them to be connected to other models directly for classification [17].

Supervised dimensionality reduction, on the other hand, uses the labeled information to separate the data points. Deep learning has been very popular recently with convolutional neural networks (CNN) being one of the most common networks [60]. These networks can generate both high-level spatial and spectral features. CNNs have been successful at discovering spatial features in addition to spectral features [16].

Recent works have integrated a spatial-spectral dimension reduction process that passes the data to a Convolutional Neural Network (CNN) [137]. One of the weaknesses of this approach is the single application of the features in classification. The feature extraction significantly reduces the complexity of the classification models [70], but the end-user still has the original large image to deal with as well as having to train multiple models for a

single purpose.

Given that the autoencoder has a history of being an effective feature extraction tool, we propose an analysis of the quality of the features generated by the compressed images to be reused for classification purposes [87]. Based on our review of the HSI compression literature, this appears to be the first attempt to consider the effectiveness of classifying compressed hyperspectral images. We hypothesize that the feature extraction process used to compress the images can be reused effectively to perform classification on the compressed images. We also hypothesize that the multi-image and generalized modifications do not have a significant impact on classification performance.

1.3 Proposed Contributions

This work makes several contributions regarding effective hyperspectral image compression models and analysis of compression techniques being applied to hyperspectral image classification. Our specific contributions are summarized as follows:

- We present a low-cost Long Short Term Memory Autoencoder model for hyperspectral image compression. We show that this model yields lower reconstruction error without reducing the compressive capabilities. We also demonstrate that our proposed model is more computationally efficient than contemporary models by requiring fewer trainable parameters and fewer training samples without detriment to its performance.

- We present a multi-image training variation and a generalized training variation that expand the generality of the compression model and eliminate the requirement to retrain the compressor for new scenes. We demonstrate that this increase in generality can reduce the reconstruction error compared to the standard training process.

- We analyzed the capabilities of the compressed image being utilized for classification tasks. We demonstrate that effective classifiers can be learned to operate on compressed

images.

- We demonstrate that increasing the generality of the compressor does not have a significant reduction in the classification performance on compressed images.

These contributions allow for an effective compression strategy that, through the model's robustness, does not require the model to be retrained for new scenes. The proposed model also allows further computational efficiency by the reusability of the compressed image being able to be used as a reduced set of features for image classification. This improvement in the computational efficiency of the model allows for a trained model to be able to be efficiently utilized across many hyperspectral image processing tasks.

For hyperspectral image classification, feature extraction often is performed to reduce the complexity and reduce the risk of overfitting for classification models. Feature extraction techniques are focused on this single task. utilizing the spectral compression of hyperspectral images as a feature extraction process would be able to both address the problem storage of these images as well as the classification dimensionality reduction problem.

Based on our literature review of the HSI compression literature, we believe we performed the first analysis of the capabilities of classifying compressed hyperspectral images. The capability of classifying compressed images would remove this additional processing and further reduce the cost of classifying hyperspectral images. Our classification analysis demonstrates that the images compressed by our proposed model can be reused as a reduced set of extracted features for classifying the compressed image as well. The analysis also further demonstrates the robustness of the model to be utilized across many scenes without retraining the model, further reducing the computational cost of processing hyperspectral images.

1.4 Summary of Hypotheses

In this section, we summarize and label our hypotheses to be referenced throughout this thesis. This will provide a reference point for each of the experiments throughout this thesis.

(H1) We hypothesize that our proposed single image compression model will be able to reduce the reconstruction error in comparison to state-of-the-art models without a significant reduction in the compression rate.

(H2) We hypothesize that the reduced complexity from a spectral only compression approach will allow us to train our proposed model on fewer pixels than state-of-the-art spatial-spectral compression models.

(H3) We hypothesize that our model will be able to perform multi-image compression without a significant increase in reconstruction error when compared to the single image compression model.

(H4) We hypothesize that our model will be able to be generalized to compress images that were not observed in the training set without a significant increase in reconstruction error when compared to the single image compression model.

(H5) We hypothesize that the proposed Bidirectional LSTM Autoencoder (BLSTM-AE) modification of the proposed LSTM-AE model will be able to further reduce the reconstruction error but it will have a lower reconstruction rate than the LSTM-AE model.

(H6) We hypothesize that our model's focus on spectral compression will reduce the computational complexity of compression when compared to state-of-the-art learned compression models.

(H7) We hypothesize that the feature extraction process used to compress the hyperspectral images can be reused effectively to perform classification on the compressed images directly.

(H8) We hypothesize that expanding the model to perform multi-image compression does not significantly impact the classification performance on the compressed hyperspectral images.

(H9) We hypothesize that generalizing the model to compress the unseen images does not significantly impact the classification performance of the compressed image.

1.5 Publications

The results of our proposed compression algorithm are currently under review with the potential publication being:

- K. Webster and J. Sheppard. Robust Spectral Based Compression of Hyperspectral Images using LSTM Autoencoders. *International Joint Conference on Neural Networks*, Padua, Italy 2022

Part of the results concerning our classification analysis is under review with the potential publication being:

- K. Webster and J. Sheppard. Classifying Compressed Hyperspectral Images. *International Joint Conference on Neural Networks*, Padua, Italy 2022

1.6 Organization

The remainder of this thesis is organized as follows. In Chapter 2, we discuss the required information to make the reader familiar with the various topics discussed in this

thesis including hyperspectral imagery, Stacked Autoencoders, Long Short-Term Memory networks, Convolutional Neural Networks, and hyperspectral image compression.

In Chapter 3, we introduce the datasets that will be utilized throughout the thesis. In Chapter 4, we present the Long Short-Term Memory network Autoencoder architecture for hyperspectral image compression. We also present a multiple image variant that allows one model to compress many hyperspectral images. In addition, we demonstrate the use of a generalized model to compress unseen images. We present experimental results testing our compression techniques on five public remote sensing datasets, demonstrating a reduction in the reconstruction error rates without a loss in the compression ratio. In Chapter 5, we present the results of using a highly effective convolutional neural network to perform the classification of compressed hyperspectral images. The presented experimental results test the convolutional neural network's classification performance using the same five public remote sensing datasets, demonstrating effective classification performance. Finally, we conclude and discuss future work in Chapter 6.

CHAPTER TWO

BACKGROUND

In this chapter, we describe the main concepts and methods that form the foundation for the work presented in the later chapters of the thesis. In particular, we present hyperspectral imaging systems, Stacked Autoencoders, sequential data processing using Long Short-Term Memory networks, the integration of the Autoencoder and Long Short-Term Memory networks, and Convolutional Neural Networks.

2.1 Hyperspectral Imaging Systems

The ability to make predictions about the contents of a material based on its absorption of light has been understood since before camera systems were developed with Schulze's Scotophors discovery around 1717 [27]. Since the development of the non-visible remote sensing field starting during World War II [91], great strides have been made towards capturing more spectral information for a wide array of analyses. Digital sensors capture the reflected light of a subject and represent the continuous wavelength with a discrete representation. For example, digital cameras capture the reflected light at the spectrum wavelengths ranging from 400 to 700nm and save the information into three bands: red (625 to 740nm), blue (440 to 485nm), and green (500 to 565nm) [51].

Hyperspectral images (HSI) are a specific categorization of electromagnetic sensing images that contain many continuous bands where each band represents a narrow range of captured wavelengths. Hyperspectral imaging systems collect information from a wide range of the electromagnetic spectrum. The collection of all of the bands represents hundreds of wavelengths often ranging from 400 to 2500nm [89]. These sensors capture an array of reflected radiation wavelengths as a three-dimensional data cube. Through the exploitation

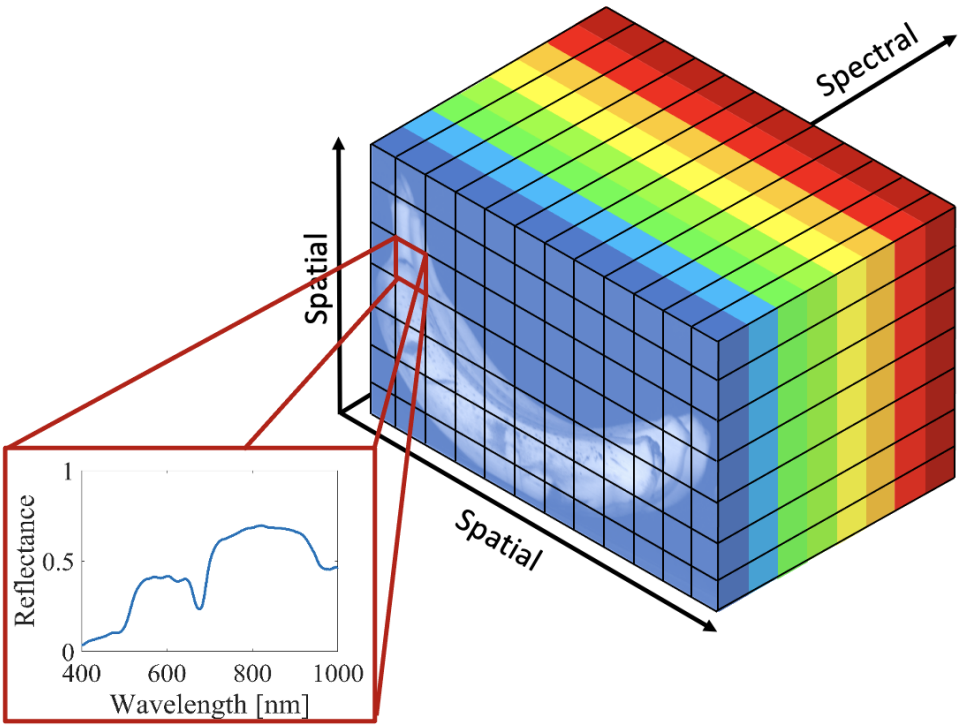


Figure 2.1: Example of a hyperspectral image data cube. Each pixels consists of a continuous reflection spectrum at its position. [77]

of spatial and spectral information in these images, powerful classifiers have been built to analyze the images [65].

HSI contains two spatial dimensions and one spectral dimension. These data cubes are often thought of as stacks of two-dimensional images to represent the same spatial object over the continuous spectral bands. Using the example data cube in Figure 2.1; if we selected one pixel from the image, we can extract the values across the spectral dimension and obtain a spectral reflectance curve. The spectral reflectance curve represents the percentage of light that the captured object reflects at each wavelength. With this representation, an image can be used to identify the properties of a subject by the absorption characteristics of the bands in complement to the high spatial resolution of the image [33, 85].

We can also collect spatial information by comparing a pixel to the surrounding pixels.

With the example data cube in Figure 2.1, a selected set of pixels can be compared to determine the boundaries of each of the river rocks. With this analysis, an image can be utilized to perform object detection and further segment the image [71].

The highly complex data present in hyperspectral imagery allows us to extract information that alternative sensors can not [56]. The continuous values of the images also allow us to extract information from the highly correlated spatially and spectral data points. With this information, HSIs have been used in a variety of applications such as remote sensing [11, 118, 142], medicine [45, 79], and food quality [75].

2.2 Stacked Autoencoder

An Autoencoder is a specialized type of artificial neural network designed as a nonlinear feature extraction method [72]. In contrast to latent space approaches that map the data to a higher dimension, the goal of an Autoencoder is to learn a lower-dimensional representation of data by mapping the original data into a lower-dimensional space [87].

An Autoencoder maps an input vector $\mathbf{x} \in \mathbb{R}^n$ to a lower-dimensional vector $\mathbf{y}$. To translate the dimension of the input vector, a new function maps the vector $\mathbf{y} \in \mathbb{R}^m$ to a new vector $\mathbf{x}' \in \mathbb{R}^n$. Specifically, the process of dimension reduction in the network can be formulated as

$$\mathbf{y} = f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}_\mathbf{x}$$

where $\mathbf{W}$ is a learned weight matrix and $\mathbf{b}_\mathbf{x}$ is the bias vector $\mathbf{b}_\mathbf{x} \in \mathbb{R}^n$. The decoding reconstruction process in the network can be formulated as

$$\mathbf{x}' = g(\mathbf{y}) = \mathbf{W}'\mathbf{y} + \mathbf{b}_\mathbf{y}$$

with the decoder's learned parameters being $\mathbf{W}'$ and $b_\mathbf{y} \in \mathbb{R}^m$.

According to Baldi and Hornik [6], a linear single layer Autoencoder is similar to principal component analysis (PCA). With this similarity, non-linear Autoencoders were introduced to expand the capabilities of Autoencoders to learn more useful features in a non-linear space [49]. This modification allows us to reformulate the Autoencoder function to be

$$\mathbf{y} = s(f(\mathbf{x}))$$
$$\mathbf{x}' = s(g(\mathbf{y}))$$

where s is an activation functions, often the sigmoid or identity functions (allowing for linear projections) [87].

To learn the parameters $\Theta = (\mathbf{W}, \mathbf{W}', \mathbf{b_x}, \mathbf{b_y})$, an Autoencoder minimizes the reconstruction loss on the given data $\mathbf{x}$ with the objective function being

$$\Theta = \min_{\Theta} L(\mathbf{x}, \mathbf{x}') = \min_{\Theta} L(\mathbf{x}, g(f(\mathbf{x})))$$

where L is any desired loss function. One of the common loss functions is squared error loss (L_s) [15]:

$$L_s(\Theta) = \sum_{i=1}^{n} ||\mathbf{x}_i - g(f(\mathbf{x}_i))||^2$$

Since its introduction in the 1980s by Rumelhart *et al.*[106], backpropagation has become a common training algorithm for neural networks. To perform backpropagation, we need to define a loss function that we want to minimize that is based on the outputs of the network and the given target values. Then, the algorithm computes the gradient of the loss function with respect to the weights of the network. The gradients are computed one layer at a time, iterating backward from the last layer. Each weight is updated proportionally

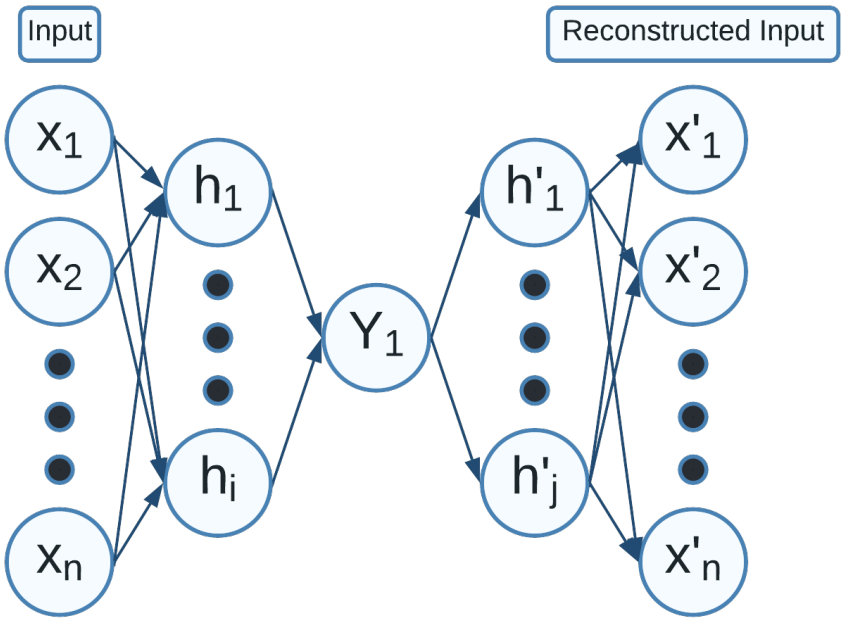


Figure 2.2: Example Stacked Auto Encoder

to the gradient of the loss function with respect to the current weight. One of the risks of using backpropagation is that when the computed gradient approaches zero, it prevents the weights of the network from changing values. This is known as the vanishing gradient problem [116]. On the other hand, when the gradient is too large it can cause the weights to be updated too far and leads to the explosion of weight values that cannot easily be reduced. This is known as the exploding gradient problem [105].

The Stacked Autoencoder (SAE) extends standard Autoencoders allowing for many hidden layers between the input and output layers [68]. Figure 2.2 is an example of the SAE architecture. This allows for the generation of differing levels of new features. SAE allows a deep neural network representation of the encoding and decoding process by applying the autoencoding process between each layer. To demonstrate, given an l-layer SAE we can

design an encoder such that:

$$Y = f_l(f_{l-1}(\dots(f_1(X))))$$

with a corresponding decoding function:

$$X' = g_l(g_{l-1}(\dots(g_1(Y))))$$

Bengio *et al.*, observing that stacking Restricted Boltzmann Machines (a type of Autoencoder) can be used to build Deep Belief Networks, introduced the greedy layer-wise training strategy that allows for the stacking of many Autoencoders while removing the difficult training process [10]. For each hidden layer, train the layer as a standard Autoencoder with inputs from the previous layer and connect a new output layer to reconstruct the input. When the layer is trained to minimize reconstruction error, the parameters of the encoding process are saved and the new output layer is thrown away. To fine-tune the results, backpropagation is often applied to the whole network.

2.3 Sequential Data Processing

For many fields, sequential data is a critical consideration for classification tasks. Given a data point $\mathbf{x}$, $\mathbf{x}$ would be considered sequential data if the features within the dataset are dependent on the other points in the dataset and where the order of the data points matter [20]. Due to the dependency on order, sequential data is often referenced as having temporal elements where each feature is referred to as a time step, but time is not a requirement for sequential data. x can then be represented as $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T)$ where $\mathbf{x}_t$ is the feature vector of data at the tth time step.

For the analysis of HSI, the captured wavelengths are stored as an ordered set of bands with each band representing a wavelength range. Figure 2.3 demonstrates the line plot of a

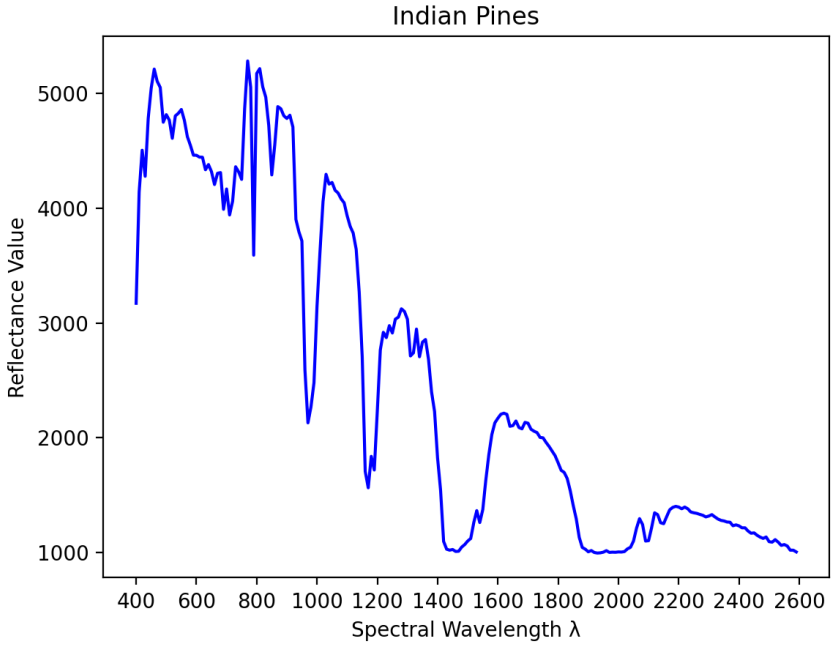


Figure 2.3: Example spectral curve from the Indian Pines dataset.

single captured spectrum. The plot of the captured spectrum demonstrates that HSI data can be seen as a set of ordered continuous values. With this representation, we can represent the bands of hyperspectral images as sequential data as demonstrated by Mou *et al.*[94].

2.3.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are artificial neural networks that allow previous outputs to be used as inputs. They are most commonly used to learn sequential data [128]. This is accomplished by the incorporation of recurrent hidden states whose nonlinear activation is dependent upon the previous step [104]. For sequence data $\mathbf{x}$, an RNN will update the current step t's output $\mathbf{o}_t$ by the following function:

$$\mathbf{o}_t = \begin{cases} 0, & \text{if } t = 0 \\ s(\mathbf{o}_{t-1}, \mathbf{x}_t), & \text{otherwise} \end{cases}$$

where s is the nonlinear activation function, commonly either a logistic sigmoid function or a hyperbolic tangent function [94]. With this formulation, we can modify the update rule of a hidden state as follows:

$$\mathbf{o}_t = s(\mathbf{W}_t\mathbf{x}_t + \mathbf{U}_{t-1}\mathbf{o}_{t-1})$$

where W_t is the weight matrix for the input at the current step and U_{t-1} is the activation of the recurrent hidden units of the previous step.

One of the benefits of this hidden state representation is the flexibility of the applications of the RNN structure. For example, an RNN is capable of either outputting the full sequence $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T)$ or the final step $\mathbf{y}_T$.

Another application of an RNN is the ability to model a probability distribution over the next element of the sequence data given its present state h_t dependent on the previous states [109]. The sequence probability $p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ can be decomposed into

$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1)\ldots p(\mathbf{x}_T|\mathbf{x}_1, \ldots, \mathbf{x}_{T-1})$$

The probability distributions can be obtained from the output directly with the following representation:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = s(o_t)$$

These networks are often trained using a backpropagation method called "backpropagation through time" [96] which usually "unfolds" the recurrent elements of the network. This process is twofold, a forward pass and a backpropagation pass. For the forward pass, the loss function over the entire time series is defined as:

$$L(x, y, w) = \frac{1}{T}\sum_{t=1}^{T} l(y_t, o_t)$$

where l is the designated loss function looking at the discrepancies between the output o_t

and the true label y_t for each time step or the whole sequence if that is the output decision.

Backpropagation through time expands the backpropagation calculations to capture the dependencies among the model's variables and parameters. To calculate the gradients with regards to the parameters w_h of the objective function L, where w_h captures the weights tied to the hidden layer h_t with the following equation:

$$\frac{\partial L}{\partial w_h} = \frac{1}{T}\sum_{i=1}^{T}\frac{\partial l(y_t, o_t)}{\partial w_h}$$

When broken out by the chain rule, the equation is decomposed into the following equation:

$$\frac{\partial L}{\partial w_h} = \frac{1}{T}\sum_{i=1}^{T}\frac{\partial l(y_t, o_t)}{\partial o_t}\frac{\partial g(h_t, w_o)}{\partial h_t}\frac{\partial h_t}{\partial w_h} \tag{2.1}$$

where o_t is the output of a node, and $g(h_t, w_o)$ is the transformation function applied to the output with respect to the output of the hidden state [99, 126].

Calculating Equation 2.1 can be expensive, so modifications to the backpropagation through time algorithms have worked to reduce the computational complexity. The truncated backpropagation through time is a modification where the sequence is processed one-time step at a time and periodically applies the backpropagation through time update back for a fixed number of time steps [127]. Another approach to reducing the cost of backpropagation through time is the Real-Time Recurrent Learning (RTRL) approach [129]. RTRL is a gradient descent algorithm that applies online learning to remove the requirement of storing the previous inputs and network states.

2.3.2 Long Short Term Memory Cells

While RNNs have shown promising results in many tasks, it has been observed that they have a major weakness if a sequence is too long. The backpropagation through time can lead to the vanishing gradient problem, with a long enough sequence leading to prior

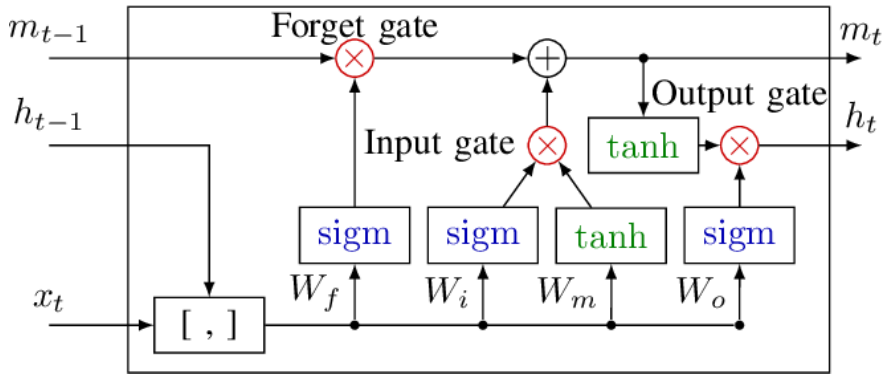


Figure 2.4: Example of a LSTM cell [100]

time steps eventually getting left behind [132].

A modification on the RNN architecture was proposed by Hochreiter and Schmidhuber to overcome the disadvantages of the RNN called the Long Short Term Memory (LSTM) [43]. In an LSTM, each node in the hidden layer is replaced with a memory cell, as seen in Figure 2.4. Figure 2.4, the LSTM cell receives input value $\mathbf{x}_t$, hidden layer output $\mathbf{h}_t$, and memory output of $\mathbf{m}_t$. The sigm represents a sigmoid function and tanh represents a hyperbolic tangent where $[\mathbf{x}_t, \mathbf{h}_t]$ in the bottom left is a concatenation operation of the input $\mathbf{x}$ and hidden layer $\mathbf{h}_{t-1}$. In Figure 2.4, the $\bigoplus$ symbol represents the gate that performs element-wise summation and $\bigotimes$ represents the gate that performs element-wise multiplication. Each gate is labeled to represent the incoming and outgoing operations.

In each cell, there is a memory state m_t, the hidden state of the cell h_t, and the input value of the sequence x_t. There are three regulatory gates: the input gate, the output gate, and the forget gate, with the representations $\mathbf{i}_t$, $\mathbf{o}_t$, $\mathbf{f}_t$ respectfully. Given the previous cell's memory state $\mathbf{m}_{t-1}$, hidden state $\mathbf{h}_{t-1}$, and input vector $\mathbf{x}_t$, the values of each of the gates

is calculated with the following equations:

$$\mathbf{i}_t = \text{sigm}(\mathbf{W}_i \odot [\mathbf{x}_t, \mathbf{h}_{t-1}]) \odot \tanh(\mathbf{W}_m \odot [\mathbf{x}_t, \mathbf{h}_{t-1}]) \tag{2.2}$$

$$\mathbf{o}_t = \tanh(\mathbf{i}_t + \mathbf{f}_t) \odot \text{sigm}(\mathbf{W}_o \odot ([\mathbf{x}_t, \mathbf{h}_{t-1}])) \tag{2.3}$$

$$\mathbf{f}_t = \mathbf{m}_{t-1} \odot \text{sigm}(\mathbf{W}_f \odot [\mathbf{x}_t, \mathbf{h}_{t-1}]) \tag{2.4}$$

$$\mathbf{m}_t = \mathbf{i}_t + \mathbf{f}_t \tag{2.5}$$

The input gate manages the input flow to the memory cell to prevent the negative effects that unrelated inputs can cause. Equation 2.2 specifies how the input gate regulates the input state and hidden state. The output gate regulates the signal to filter output states and keep the output within selected bounds. Equation 2.3 specifies the combination of the input gates, output gates, and the original inputs to regulate the output signal. The forget gate manages the information that is to be remembered or removed from the memory cell. Equation 2.4 takes the memory from the previous cell and combines it with the input value and hidden state value to determine which information is to be forgotten. The cell state ($\mathbf{m}_t$) is specified by Equation 2.5 as a combination of the input gates and the forget gates.

2.4 LSTM Autoencoder

A Long Short Term Memory Autoencoder is the integration of LSTM networks into an Autoencoder structure. An LSTM Autoencoder network is capable of dealing with sequential data as input in contrast to regular Autoencoders. These networks are most commonly utilized to predict future flow problems [7, 31, 125].

Figure 2.5 demonstrates the integration of LSTM cells into an Autoencoder structure. This network contains a single node LSTM network compressing a time-series sequential data set. A single node LSTM network contains a series of n LSTM cells that associate to n time steps. The encoder saves the output of the last LSTM cell while the decoder returns

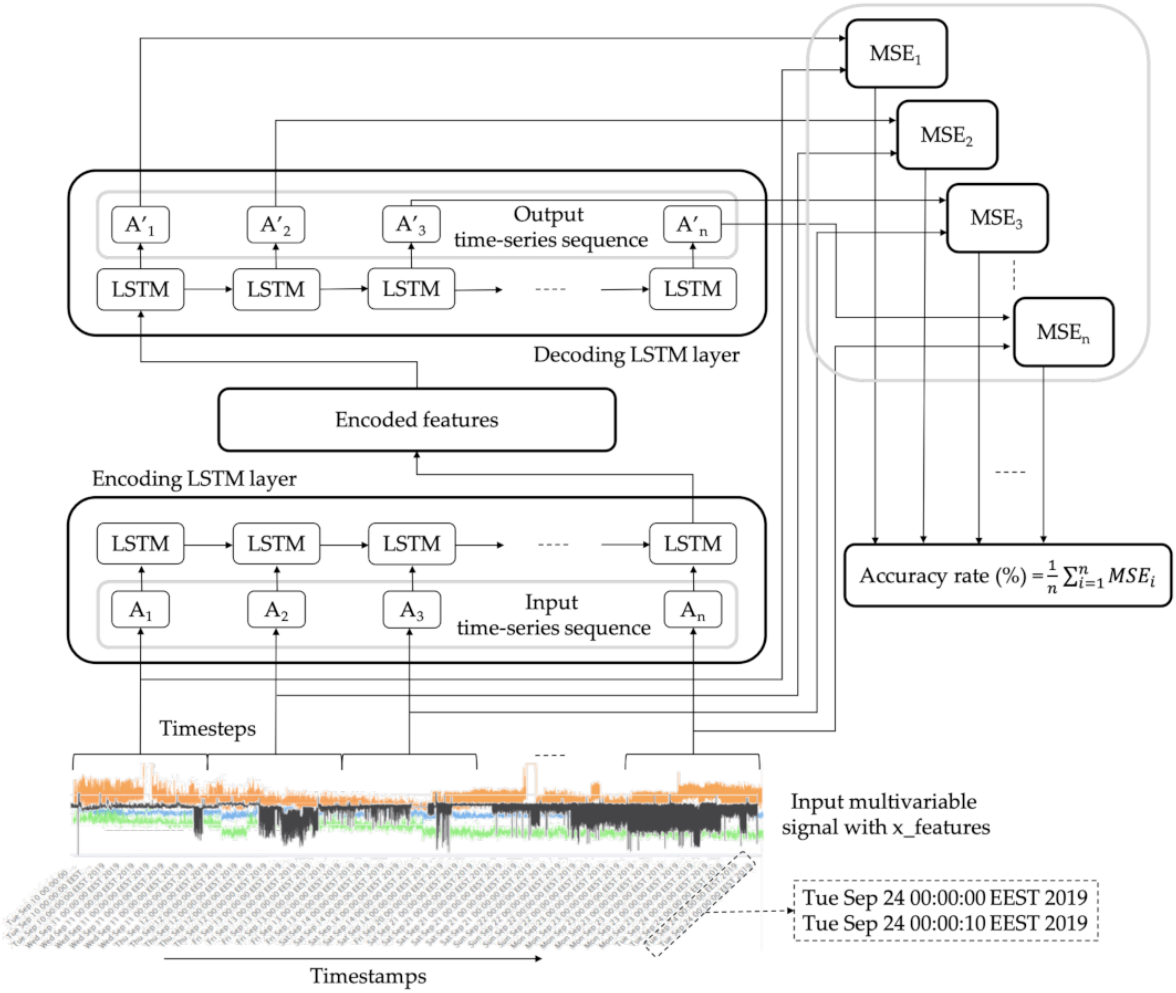


Figure 2.5: Example of a single node LSTM Autoencoder that contains a single encoding and decoding layers [7].

the output of all LSTM cells.

Autoencoders rarely consist of a single node in each layer, they often utilize multiple nodes in each layer with an encoding layer consisting of fewer nodes than the previous layers [117]. As such, for an LSTM Autoencoder to contain multiple nodes, each node is treated as separate time series of LSTM cells. To incorporate additional layers under a Stacked Autoencoder structure, all layers except the final encoding step duplicate the output of all of the LSTM cells contained within each given node. The decoding layer in Figure 2.5

demonstrates this process. At each layer, each nodes' outputs are combined as a feature set at each timestamp.

To decode the feature maps obtained from the encoder layers, a repeat vector layer is applied. The layer duplicates the final output vector from the encoding layer to generate n length time step data. With this process, a decoder can reconstruct the original time sequence at the correct length [136].

To train the network, the standard backpropagation through time algorithm is applied to the network as a whole as defined by Equation 2.1. The loss function for LSTM Autoencoders is typically the mean squared reconstruction error (MSRE) function defined as:

$$MSRE(\Theta) = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i - g(f(\mathbf{x}_i))||^2$$

where $\mathbf{x}$ is an input sequential data set, $f(\mathbf{x})$ is the encoder function, and $g(\mathbf{x})$ is the decoder function.

Additionally, an LSTM Autoencoder can incorporate other neural network strategies into the Autoencoder structure to expand the model's flexibility, such as convolutional neural networks or fully connected layers. Time distributed layers are incorporated to apply a layer to every temporal slice within the input sequence [123]. The network applied is usually the same instance of the layer, so the weights remain static when applied to each temporal slice in a sequential input. This allows an expansion of the capabilities of the LSTM Autoencoder not only to incorporate the different strategies but also to allow for an increased variety of nodes per layer. In particular, the incorporation of a fully connected layer with a TimeDistributed layer is commonly used during the reconstruction process to restructure the sequence to be of the proper dimension.

2.5 Convolutional Neural Networks

A Convolutional Neural Network (CNN), is an artificial neural network that specializes in processing data that contains a local structure among the input examples, such as image processing or 1D sequential data [113]. With this capability, a wide range of computer vision and pattern recognition applications have utilized CNNs for classification tasks [55].

According to Goodfellow *et al.*[34], CNNs utilize mathematical *convolution* operation instead of the traditional general matrix multiplication. A convolution is defined as:

$$(\mathbf{x} * \mathbf{w})(d) = \int_{-\infty}^{\infty} \mathbf{x}(a)\mathbf{w}(d-a)da$$

where $\mathbf{x}$ is the input, $\mathbf{w}$ is the kernel or filter, along dimension d. For a discrete $\mathbf{x}$ and $\mathbf{w}$, we can simplify the operation to be:

$$(\mathbf{x} * \mathbf{w})(d) = \sum_{a=-\infty}^{\infty} \mathbf{x}(a)\mathbf{w}(d-a).$$

This operation can be extended for $\mathbf{x}$ and $\mathbf{w}$ with multiple dimensions. For example, let $\mathbf{I}$ be a 2-D image and $\mathbf{K}$ be a 2-D filter with the convolutional operation be:

$$S(i,j) = (\mathbf{I} * \mathbf{K})(i,j) = \sum_{m}\sum_{n} \mathbf{I}(m,n)\mathbf{K}(i-m, j-n).$$

S can be considered as a feature map where modifying the filter values $\mathbf{K}$ allows for a differing representation of the features extracted from $\mathbf{I}$.

CNNs commonly integrate many feature maps and layers to extract as much information as possible. This is possible by implementing each filter on the input data separately and stacking the outputs to recombine the data from the filters. With this recombining stage, CNN layers can then be stacked into a deep neural network structure [59]. Figure 2.6 is a

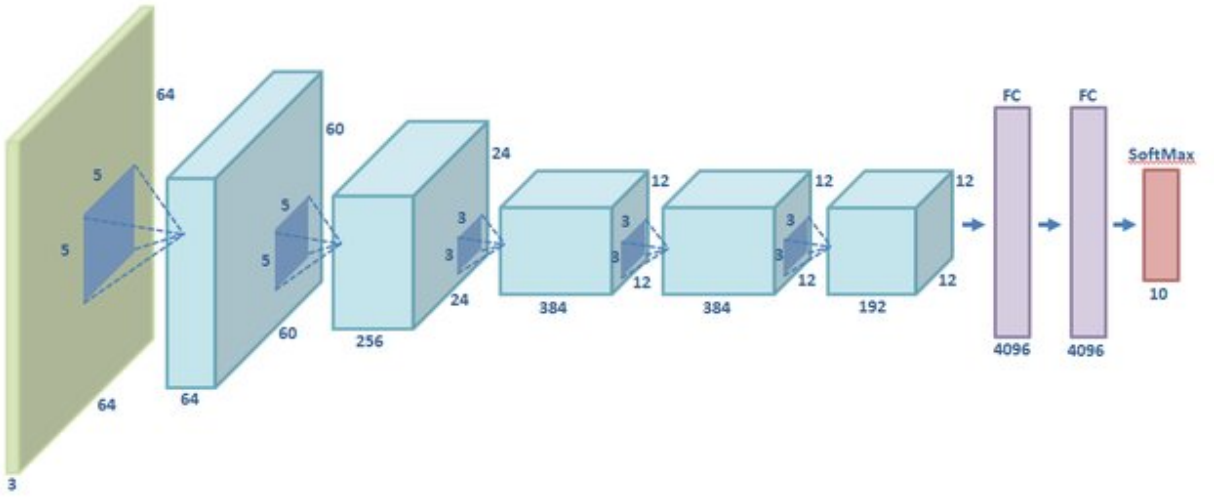


Figure 2.6: AlexNet architecture [76]

visual representation of a deep convolutional neural network. Like Stacked Autoencoders, convolutional neural networks use backpropagation to train the parameters of the network. These networks also suffer from the vanishing gradient problem and must find alternatives to utilize a deep neural network structure.

One solution is the utilization of the rectified linear unit activation function (ReLU) defined by $ReLU(x) = max(0, x)$ [66]. ReLU functions help to prevent the vanishing gradient problem by slowing the degradation of the parameters since the derivative of ReLU is 1 if the input value is positive or 0 if the input value is negative. This additionally helps prevent the exploding gradient problem since a gradient of value 1 multiplied by 1 many times is still 1.

CNNs also benefit from the inclusion of batch normalization layers [47]. Batch normalization is implemented during training and it standardizes the inputs to a layer using the mean and standard deviation of said inputs per mini-batch. Let $\mathbf{B}$ be a mini-batch of the data where $\mathbf{B} = \mathbf{z}_1, \ldots, \mathbf{z}_m$ of m samples long. Each sample $\mathbf{z}_i$ is normalized to have

zero mean and unit variance such that:

$$\hat{\mathbf{z}}_i = \frac{\mathbf{z}_i - \mu_{\mathbf{B}}}{\sqrt{\sigma_{\mathbf{B}}^2 + \epsilon}}$$

where $\mu_{\mathbf{B}}$ is the mean of $\mathbf{B}$, $\sigma_{\mathbf{B}}^2$ is the variance of $\mathbf{B}$, and ϵ is a small number that is used to avoid inconsistencies. We can then adjust $\hat{\mathbf{z}}_i$ using the learnable parameters λ and β to obtain the output $\tilde{\mathbf{z}}_i$ such that:

$$\tilde{\mathbf{z}}_i = \lambda \hat{\mathbf{z}}_i + \beta.$$

During the training process, the batch normalization layer calculates a running mean μ and running variance σ^2 instead of sampling each of the mini-batches for these values. When using batch B, μ and σ^2 are updated as follows:

$$\mu = \alpha * \mu + (1 - \alpha) * \mu_B,$$
$$\sigma^2 = \alpha * \sigma^2 + (1 - \alpha) * \sigma_B^2.$$

where α, commonly referred to as momentum, is the tunable hyperparameter which is the importance given to the previous μ. Batch normalization further reduces the risk of the vanishing gradient problem by providing some regularization to the training process.

2.6 Dimensionality Reduction

Working with hyperspectral images presents a challenge due to their high spectral and spatial resolution, leading to high levels of dimensionality. This high dimensionality often introduces redundant or irrelevant information, which increases the size and complexity of the feature space. This phenomenon is known as the "curse of dimensionality" [122] or the "Hughes phenomenon" [112] which refers to the inherent sparseness of high dimensional spaces, implying that the amount of data needed to learn a model grows exponentially with

respect to the number of dimensions, or features [90].

With this limitation, hyperspectral images benefit from dimensionality reduction methods that reduce the number of spectral bands while retaining the most useful information for specific applications [22, 120]. Dimensionality reduction methods can be categorized as feature extraction methods and feature selection methods.

Feature selection determines which features contain the most important levels of information. The identified important features are then selected as the feature set for classification and the remaining features are often discarded. This process can be very beneficial to the capabilities of classifiers analyzing hyperspectral images, but the discarding of features means that compression algorithms cannot incorporate feature selection techniques.

Feature extraction methods apply linear or non-linear transformations to extract composite features from the original data. The objective of such transformations is to obtain a very similar non-redundant feature space that facilitates the learning process and, in some cases, human interpretation. Feature extraction methods can be subdivided into two classes: unsupervised and supervised. The former aims to find natural groupings from data according to some criterion, while the latter takes advantage of labeled datasets aiming to find new data representations that maximize differentiation between labels or classes.

Feature extraction tools are often utilized for compression purposes due to the ability to 'recover' the original features from the composite features. Since this thesis focuses on feature extraction processes for hyperspectral image processing, we will be discussing the most common feature extraction methods that are most utilized within the hyperspectral image compression literature. In particular, we will be discussing Principal Component Analysis and Wavelet Transformation due to their prevalence in the hyperspectral image compression field.

2.6.1 Principal Component Analysis

Principal Component Analysis (PCA) is one of the most widely used feature extraction methods. It is defined as an orthogonal linear transformation that transforms the data into a new orthogonal basis, whose components are uncorrelated and ordered so that the first few retain most of the variation present in all of the original variables [108].

Let $\mathbf{X}$ be a set of samples where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ where $\mathbf{x}_i \in \mathbb{R}^d$ such that $\mathbf{x}_i$ can be represented as a d-dimensional vector $\mathbf{x}_i = (\mathbf{x}_{i1}, \mathbf{x}_{i2}, \ldots, \mathbf{x}_{id})$. For conciseness, we consider $\mathbf{X}$ to have zero-mean. Thus, the sample variance of $\mathbf{X}$, used to measure the spread of the data along each axis, is given by:

$$v = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i)^2.$$

We can verify how correlated two features a, b in the d-dimensional data based on their covariance $v_{a,b}$ value calculated as:

$$v_{ab} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_{ia}\mathbf{x}_{ib}.$$

The covariance is used to build a $d \times d$ sample covariance matrix, calculated as $\mathbf{X}^\top\mathbf{X}$. To obtain the diagonal matrix Λ, a transformation matrix $\mathbf{U}$:

$$\Lambda = \mathbf{U}^\top\mathbf{X}^\top\mathbf{X}\mathbf{U}$$

where the rows of vectors of $\mathbf{U}$. These rows of vectors called the principal components are the eigenvectors of $\mathbf{X}^\top\mathbf{X}$. Further, the diagonal elements of Λ are the square root of the eigenvalues representing the variances of the transformed data along with the new components.

The ordering of the eigenvalues allows us to obtain the components by order of significance. To reduce the dimensionality of the dataset, we select the first k eigenvectors

that contain the greatest eigenvalues and ignore the rest. The data can then be projected onto the selected eigenvectors using the $n \times k$ matrix $\mathbf{U}_k$ representing the first k eigenvectors. We then obtain the transformed dataset $\mathbf{P}$ as follows:

$$\mathbf{P} = \mathbf{X}^\top \mathbf{U}_k$$

where the matrix $\mathbf{P}$ is a new data representation of $\mathbf{X}$ with reduced dimensionality with decorrelated components. To recover the original data we can invert Equation 2.6.1 as follows:

$$\mathbf{X} = \mathbf{P}\mathbf{U}_k$$

As an effective feature extraction technique that has benefited machine learning models, it has been adapted to hyperspectral image compression.

2.6.2 Wavelet Transformation

Wavelet transformation has been a popular feature extraction technique for signal and image data [141]. A wavelet is a function that allows the analysis of signals in terms of scales and resolutions. In particular, wavelets represent a wave-like oscillation that is localized in time.

Discrete wavelet transformation returns a vector of the same length as the dimensionality of the input. The contents of the vector indicate the scaling and translating coefficients from a known wavelet to the current wavelet. The main strength of the discrete wavelet transform is the data contained in the vector represents the corresponding translations and scaling that the wavelet went from a known wavelet, commonly referred to as the 'mother wavelet'. To perform dimensionality reduction, an additional process reduces the dimensions of the vector representation. This is beneficial because the vector representation is easier to identify patterns to reduce the dimensions than the raw data.

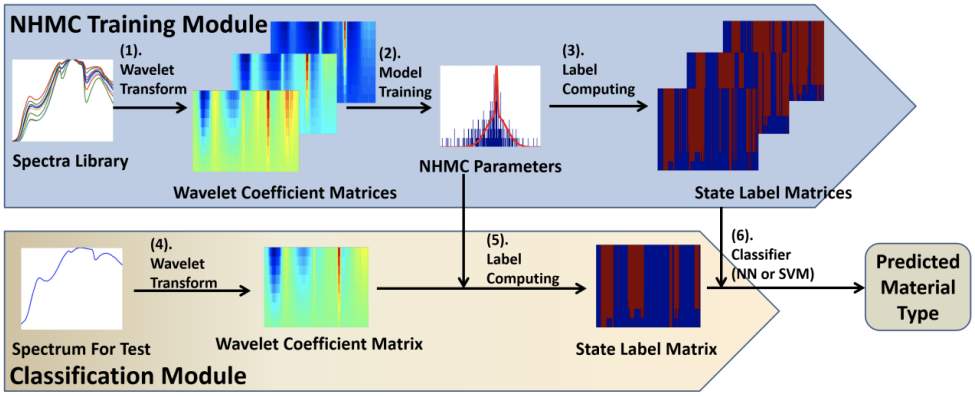


Figure 2.7: Example of Wavelet Transformation being utilized for hyperspectral image compression using a non-homogeneous hidden Markov chain [29].

Discrete wavelet transformation gained popularity in the hyperspectral image compression community because of the capability to extract both local spectral and temporal information simultaneously.

Wavelets are functions generated from one single function, or mother wavelet, ψ modified through scaling and translating the function defined as:

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right)$$

where a is the scaling variable, b is the translation variable, and where ψ must satisfy $\int_{-\infty}^{\infty}\psi(x)dx = 0$ [130].

Wavelet transformation divides the given signal into a set of wavelets that are separately computed for different frequency information. Let the function f be a continuous square-integrable function, its wavelet transformation is calculated as:

$$W[f(a,b)] = \langle f, \psi_{a,b}\rangle = \int_{-\infty}^{\infty} f(x)\psi_{a,b}(x)dx$$

where ψ, a, b, represent the wavelet family with the scale and translation coefficients

as defined in Equation 2.6.2, and $W[f(a,b)]$ is the wavelet coefficient matrix. To perform discrete wavelet transformation, a and b are set to powers of two such that $a = 2^m$ and $b = 2^n$.

The corresponding wavelet coefficients can be used to represent the translation of the 'mother wavelet' required to match the original wavelet. This effective feature extraction method has been adapted as an efficient method to represent discrete functions in a redundant form that is more efficient to compress, as demonstrated by Figure 2.7.

CHAPTER THREE

EXPERIMENTAL IMAGE DATASETS

In this chapter, we describe the datasets we will use in Chapter 4 to design and validate effective spectral hyperspectral image compression. We also describe the data pre-processing that is used in both Chapters 4 and 5. These datasets were also used in Chapter 5 to validate our method for compressed hyperspectral image classification with an efficient convolutional neural network.

In our experiments, we used five well known remote sensing HSI datasets: Indian Pines (IP) [9], Pavia Center (PC) [4], Kennedy Space Center (KSC) [40], Salinas (SA) [36], and Botswana (BSW) [18]. Indian Pines was acquired by the Airborne Visible Infrared Imaging Spectrometer (AVIRIS) sensor, Pavia Center was acquired by the Reflective Optics System Imaging Spectrometer (ROSIS) sensor, Kennedy Space Center was acquired by AVIRIS, Salinas was acquired by AVIRIS, and Botswana was acquired by the Hyperion sensor. The Hyperion and AVIRIS sensor bands both have a resolution of 10nm with AVIRIS ranging from 400nm - 2400nm and Hyperion ranging from 400 - 2500nm. The ROSIS sensor has a 5nm resolution ranging from 430 - 960nm. The datasets are all available from [80]. The spectral and spatial dimension of each of the datasets is given in Table 3.1.

3.1 Dataset Description

3.1.1 Kennedy Space Center

KSC, visualized in Figure 3.1, was captured by the NASA AVIRIS sensor in 1996, which captured the Merrit Island National Wildlife Refuge, the NASA Shuttle Landing Facility, and elements of the nearby town of Titusville. The 512×614 pixel image was taken with 224 captured spectral bands; however, 48 bands were removed due to them capturing water

Dataset	Rows	Columns	Bands
Indian Pines	145	145	220
Pavia Center	1096	1096	102
KSC	512	614	176
Salinas	512	217	224
Botswana	1476	256	145

Table 3.1: Summary of the 5 selected images datacube sizes.

absorption and containing low signal-to-noise ratios. There are 13 classes identified for the ground truth. This image was chosen because of the split between the heavily developed Titusville and sparse Merrit Island National Wildlife Refuge. The image was also chosen to examine how robust the models against the missing spectral information will be by observing its behavior during the compression process.

3.1.2 Botswana

Botswana, visualized in Figure 3.2, is a scene captured by the NASA EO-1 Hyperion imager. The 1476×256 image captures 242 bands of the Okavango Delta in Botswana. The original image was preprocessed by the University of Texas Center for Space Research, which removed 97 bands to reduce the number of water absorption features. The ground truth consists of 14 identified classes. This image was selected both because the image consists of swamps and drier woodlands. Additionally, the image being from a satellite will allow experimentation with the differences between airplane-mounted sensors and satellite-mounted sensors.

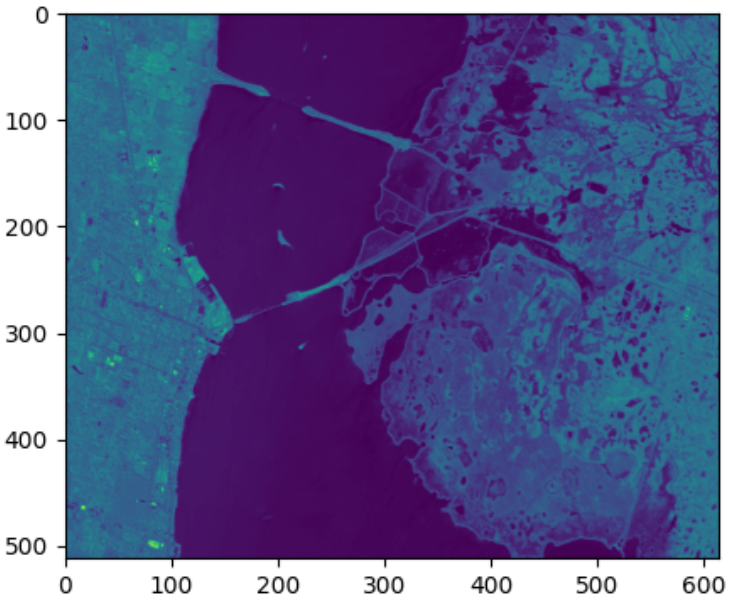


Figure 3.1: Kennedy Space Center

3.1.3 Pavia Center

Pavia Center, visualized in Figure 3.3, was one of two scenes acquired by the ROSIS sensor during a flight over Pavia in northern Italy. Pavia Center captures the city itself in a 1096×1096 pixel image with 102 spectral bands. The image was preprocessed by researchers at the Telecommunications and Remote Sensing laboratory to remove the rectangular region of the pixels that contained no spectral information, which reduced the image into a 1096×715 pixel image. There are 9 classes in the ground truth. Pavia Center was chosen because most of its pixels contain developed features with some water and mixed plant elements.

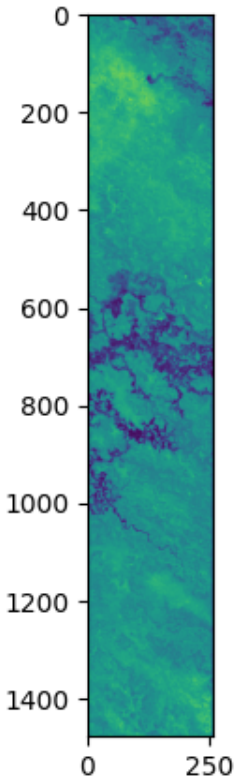


Figure 3.2: Botswana

3.1.4 Salinas

Salinas, visualized in Figure 3.4, is a scene captured by the AVIRIS sensor during a flight over the Salinas Valley in California. The 512×217 pixel image captures 224 bands of spectral reflectance. This image had 20 bands removed manually from the original image due to capturing water absorption. This image was chosen both for diverse vegetation coverage as well as containing a higher level of spatial resolution. The Salinas ground truth contains 16 classes.

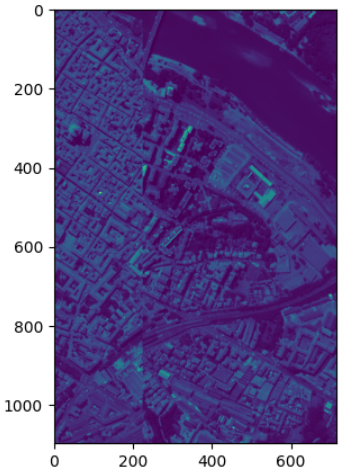


Figure 3.3: Pavia Center

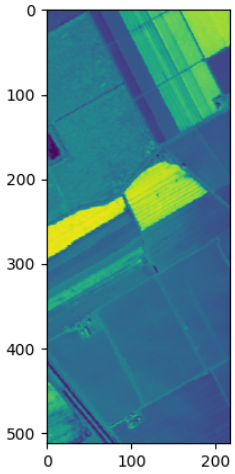


Figure 3.4: Salinas

3.1.5 Indian Pines

Indian Pines, visualized in Figure 3.5, was captured from a test site in North-western Indiana by the AVIRIS sensor in 1995. The 145×145 pixel image captured 224 bands of spectral reflectance; however, 24 bands were removed that covered the region of water absorption. The image contains a mixture of agriculture, forest, or other natural perennial vegetation, and a small number of pixels contain developed elements. This image was chosen due to the low spatial resolution causing higher levels of noise in the spectral dimension. The ground truth contains 16 classes.

3.2 Data Preprocessing

The pre-processing step differs depending on the type of task performed: hyperspectral image compression or hyperspectral image classification. That is, the pre-processing step used in Chapter 4 mainly consists of reshaping the data before passing it to a compression algorithm; whereas, in Chapter 5 it consists of generating patches to train the classifier as

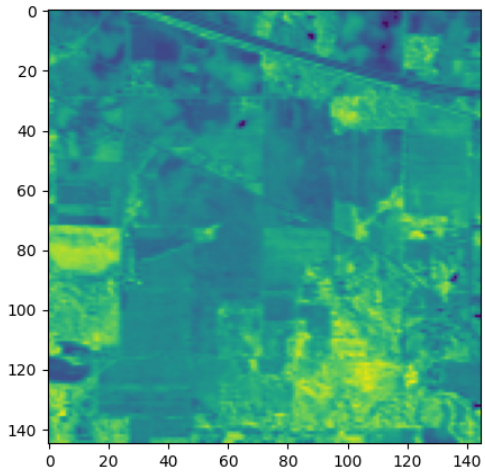


Figure 3.5: Indian Pines

well as normalizing the data for accelerated learning.

For Chapter 4, we performed little data preprocessing before compressing the images. All images contain 16-bit spectral reflectance values. Since we are utilizing sigmoid and hyperbolic tangent functions in the neural networks, we normalized the data from the range $[0, 65535]$ to $[0, 1]$ by dividing the values by 65535. Aside from this initial normalization, the compression algorithm applied no other data preprocessing.

Since our proposed network requires a sequence of pixels, we reshaped the data by merging the 2-D spatial dimension into a 1-D collection of pixels with each pixel containing the spectral bands of each of the pixels. Additionally, since our proposed network can process multiple images with a variety of spectral band lengths, the pixels were padded to be of uniform size. For the datasets selected, this led to a size of 224. When compressing an image, the original length of the spectral bands is saved alongside the compressed image to recover the original band size when decompressing.

For Chapter 5, to train the classifiers, we generated small overlapping patches such that each patch represented a single class. For each of the images, 5×5-pixel patches around

each pixel were extracted and labels assigned. Furthermore, the only patches collected were around those pixels with an assigned label. The Indian Pines dataset generated 10,249 patches, the Salinas dataset generated 54,129 patches, the Pavia Center dataset generated 7,456 patches, the Kennedy Space Center dataset generated 5,211 patches, and the Botswana dataset generated 3,248 patches. The spectral depth of each image corresponds with the number of features in the compressed image. We performed this patch generation for all of the classification experiments.

CHAPTER FOUR

ROBUST SPECTRAL BASED COMPRESSION

In this chapter, we use the concepts introduced in Section 2.3 to design a Long Short Term Memory Network Autoencoder (LSTM-AE) for hyperspectral image compression. This network was designed to meet comparable state-of-the-art compression performance while decreasing the reconstruction error. We begin with an overview of the related work, outline the compression method in detail, describe the modifications required to increase the robustness of the model, and provide experimental results.

4.1 Related Work

Many of the classic HSI compression approaches have focused on a combined spectral-spatial compression, but there has been little analysis of the capabilities of only spectral-based compression. The spectral-spatial compression strategy can be broken into two parts; spectral compression augmented by a separate spatial map and methods that compress the spectral and spatial dimensions simultaneously. Methods that have traditionally implemented a spectral strategy further compressed by a spatial strategy often suffer from a significant increase in reconstruction error the further compressed an image is [23, 30, 53, 141]. The combined compression of an HSI's spatial and spectral dimensions simultaneously has shown some improvements with regards to the weaknesses of the previous strategies, but they still suffer from the same trends [14, 25, 108]. In the following sections, we will be looking separately at the methods previously presented for both lossless compression and lossy compression.

4.1.1 Lossless Compression

Lossless compression of HSI is powerful because of the guarantee of no information loss. This approach to HSI compression is often carried out by removing the spectral correlation and the spatial correlation. With the high level of spectral correlation, traditional image compression methods, JPEG compression, for example, cannot directly compress a given HSI [121].

One of the most prevalent approaches to lossless compression is the predictive coding approach. Predictive coding makes a prediction of the current sample using previous samples where the prediction results dictate the probabilities of the predicted sample. These probabilities then dictate that the most likely values will be encoded with the smallest bit representation and the least likely values will be encoded with the longest bit representation. The mapping of values to the bit representation is often referred to as code words. In 2005, the consultative committee for space data systems (CCSDS) released the CCSDS 121.0-B-1 standard for lossless onboard satellite hyperspectral image compression that utilizes predictive coding for entropy-based coding [110]. This standard has also been extended to incorporate machine learning models to better predict the coding parameters to improve the compression rate [52]. Lookup tables have also been proposed to optimize the compression rate of predictive coding [88]. Recursive least-squares filtering techniques that offer more adaptive coder predictors have also been proposed as alternative methods due to their speed [54].

Another common approach is the transformation of the spectral information into a smaller de-correlated set where a standard two-dimensional image compressor can be applied. Discrete Wavelet Transformation (DWT) was used to break the existing spectral correlation into sub-bands for a JPEG-2000 compressor to process [119]. Karhunen-Loève Transform algorithms are also a spectral de-correlation process that also has error-correcting capabilities [83].

Vector quantization (VQ) was also used to break the spectral bands into a set of vectors to quantify for compression. Given that the generation of a codebook for HSI is expensive, mean-normalized vector quantization was applied to reduce the size of the codebook [107].

4.1.2 Lossy Compression

One of the biggest downsides of lossless compression is the reduction in the compression ratio with recent methods rarely surpassing a compression ratio of 25% of the original size [8]. For specific applications, this might be a worthwhile trade-off, but for most general applications, lossy compression algorithms are acceptable due to the significantly higher compression rates and acceptable loss of information. Lossy compressors often take inspiration from the dimensionality reduction field. Common methods include principal component analyses, wavelet transformation, and learning-based compression which include neural networks and regression-based learning [98, 141].

As one of the most common approaches, dimensionality reduction looks to represent the hyperspectral image in a smaller dimension. The typical process for these images is to de-correlate the dimensions and represent the image with a reduced representation of the de-correlated dimensions. An example of this process is the implementation of PCA to be used in conjunction with a JPEG compressor [22]. Non-iterative factorized tensor decomposition has also been successful in further compressing the images from the PCA algorithm [108].

DWT, which is one of the bases of the JPEG-2000 compression standard [48], has been incorporated into lossy compression elements to further compress HSI [102]. With a spectral de-correlation process, JPEG-2000 can also successfully compress HSI [62].

Recently, machine learning and deep learning techniques have been utilized in the hyperspectral image compression process to improve the compression rates as well as the reconstruction error. One such example is a DWT-SVM proposed by Zikiou et al. [141] where the support vector regression model is used to further reduce the most important

spectral bands. The authors of DWT-SVM use it to better Also, a Convolutional Neural Network Autoencoder (CNN-AE) has been proposed to compress the images directly [25].

4.1.3 LSTM classifying Hyperspectral Images

There has been no work investigating the applicability of LSTM networks in the hyperspectral compression literature. There has been literature published with regards to LSTM networks performing hyperspectral image classification. In this section, we will be analyzing the published literature investigating the application of LSTM networks in both hyperspectral image classification and feature extraction.

Recent work with Long Short Term Memory Networks to learn the spectral information of hyperspectral images has been focused on the LSTM being a feature extractor [95]. Given the high correlation of the spectral bands [73], an LSTM processes the pixels of hyperspectral images through a sequential perspective instead of the perspective of them as feature vectors that capture the intrinsic sequential data structure of hyperspectral bands. This approach has been further expanded by incorporating a deep neural network structure by Hang *et al.*[41]. They have demonstrated that layering LSTM networks into a deep neural network structure have the capability of improving classification results.

LSTMs have also been demonstrated to extract both the spectral and spatial information to classify a given pixel [139]. This is accomplished by incorporating the surrounding pixels, in a similar strategy to generating patches for convolutional neural networks, as additional features for each time step. The results of this classification task demonstrated the effectiveness of using the LSTM network as a spectral feature extraction process.

Bidirectional LSTMs have demonstrated an increase in performance for the classification of pixels when incorporated with convolutional neural networks. [71]. This was reinforced by Yin *et al.*[133] where the Bidirectional LSTM is able more effectively extract features for a CNN classifier.

Further optimization of LSTMs on classifying HSI has shown some potential that grouping bands have some increase in classification performance [131]. Xu *et al.*proposed the round-robin approach for band grouping. Round-robin grouping pairs look at grouping every n bands together to form shorter time steps instead of the traditional approach of adjacent bands being grouped together. These results demonstrated that the bands are still highly correlated when not the direct neighbors. This also demonstrated that the LSTM has the potential to not be sensitive to changes in the spectral resolution.

4.2 The LSTM Autoencoder Compression

Recent work in hyperspectral compression techniques demonstrates that a spatial and spectral approach may not be necessary to achieve similar results. In particular, recent work with Long Short Term Memory Networks to learn the spectral information and autoencoders performing feature extraction of hyperspectral images indicate that a combined strategy may be able to represent the bands and extract enough feature information from the bands to allow for a competitive HSI compression strategy. [69, 71, 86, 140].

4.2.1 LSTM-AE Architecture

We present an HSI compression method using a Long Short Term Memory Autoencoder (LSTM-AE) architecture. Since a given pixel can be represented as a sequence of correlated spectral bands, we consider each pixel as an independent sequential data set. As a consequence, we will use our LSTM-AE architecture to compress the spectral dimension pixel-by-pixel with a lightweight model.

The LSTM-Autoencoder combines the feature learning of an autoencoder with the sequential context representation of an LSTM. This is accomplished by having each layer of the autoencoder contain a series of single-layer LSTM models. At each layer is a designated number of LTSM networks representing the nodes of a traditional stacked autoencoder.

Layer Number	Layer Name	Input Shape	Output Shape	Parameters
1	LSTM	(224, 1)	(224, 20)	1760
2	LSTM	(224, 20)	(224, 5)	520
3	LSTM	(224, 5)	(1)	28
4	Repeat Vector	(1)	(224, 1)	0
5	LSTM	(224, 1)	(224, 5)	140
6	LSTM	(224, 5)	(224, 20)	2080
7	Time Distributed + Dense	(224, 20)	(224, 1)	21

Table 4.1: Example LSTM-AE layers with input dimensions, output dimensions, and the number of parameters per layer. Layers 1-3 are the encoder and layers 4-7 are the decoder.

Table 4.1 represents an example LSTM-AE model for an image that has 224 spectral bands.

Given an input band at a pixel $\mathbf{x}$, the autoencoder extracts the features from the sequence. The intermediate layers pass the full sequence to the next layer while the final encoding layer outputs the n^{th} sequence position of each node. This continues until the encoding layer where the last iteration of the sequence is output. This encoding process is demonstrated by Figure 4.1

The decoder takes the reduced representation and reconstructs the original sequence. In addition to the LSTM layers, a Repeat Vector Layer and a Time Distributed Layer are utilized to resize the sequences. The Repeat Vector layer takes the compressed value and duplicates the input values to build the original sequence length. The LSTM layers reconstruct the original sequence values of the pixel. The Time Distributed + Dense layer applies a standard feedforward neural network layer for each time step for each node in layer 6. This dense layer condenses outputs of the nodes into the original spectral sequence of the pixel.

The autoencoder structure is trained using the Adam optimizer introduced by Kingma

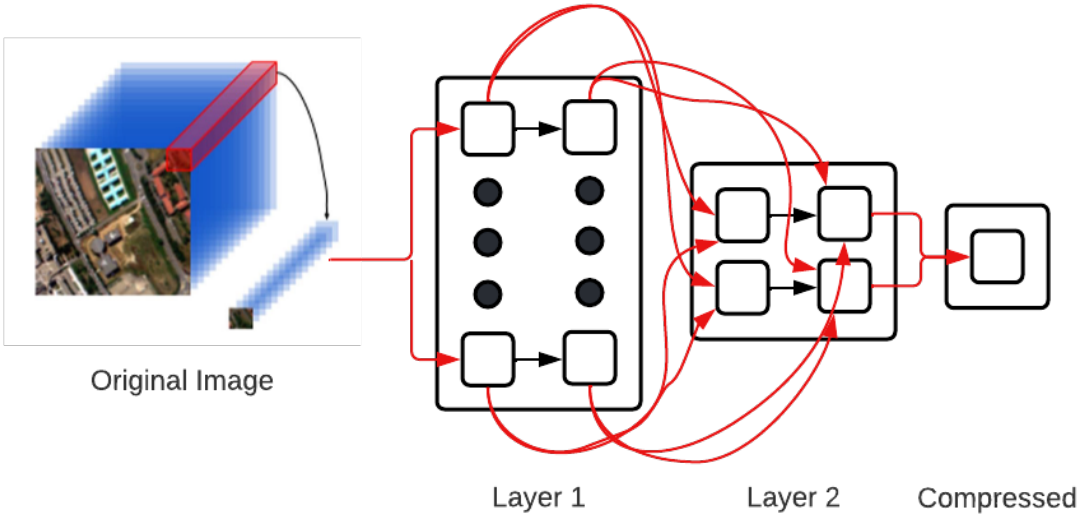


Figure 4.1: Example of LSTM-AE Compression on HSI band. Red arrows represent autoencoder connections and boxes represent an LSTM network.

[57]. The Adam optimizer is a stochastic gradient descent method that employs an adaptive learning rate strategy and calculates an exponential moving average of the gradient and the squared gradients. These properties allow Adam to require fewer tunable inputs while still being able to handle noisy and some non-convex stochastic gradient descent problems [44].

4.2.2 Bidirectional Modification

The Bidirectional LTSM Autoencoder (BLSTM-AE) is a modification of the LTSM Autoencoder that includes an extra layer that reverses the order of the input sequence. Let x be sequential data where $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$. A Bidirectional LSTM node contains two groups of hidden layers. The first hidden layer contains a standard LSTM node that will process the sequence starting at $\mathbf{x}_1$ and the final sequence being $\mathbf{x}_t$. The other hidden layer inverts the order of the sequence so that it will process the sequence starting at $\mathbf{x}_t$ and the final sequence being $\mathbf{x}_1$. In our approach, we used the Time Distributed + Dense

Layer Number	Layer Name	Input Shape	Output Shape	Parameters
1	Bidirectional LSTM	(224, 1)	(224, 40)	3520
2	Bidirectional LSTM	(224, 40)	(224, 10)	1840
3	Bidirectional LSTM	(224, 10)	(2)	96
4	Repeat Vector	(2)	(224, 2)	0
5	Bidirectional LSTM	(224, 2)	(224, 10)	320
6	Bidirectional LSTM	(224,10)	(224, 40)	4960
7	Time Distributed + Dense	(224, 40)	(224, 1)	41

Table 4.2: BLSTM-AE layers with input dimensions, output dimensions, and number of parameters per layer.

layer to recombine both the groups of hidden layers into a single band. With this structure, the Bidirectional LSTM network can capture the time dependencies within a sequence in a forward and a backward manner. While the structure of the network does not change fundamentally from the LSTM Autoencoder, the number of nodes contained within each of the layers in the network increases due to the capturing of the outputs of both of the hidden layers. This is shown in the network structures between Table 4.1 and Table **??**. The overall structure is the same except for the number of nodes in each layer, and subsequently the number of features passed between each layer, is doubled.

4.2.3 Robustness Modifications

Given that the LSTM structure only considers the spectral bands as a sequence of integers, we can integrate the information from multiple images at the same time without requiring the selected images to contain the same spectral resolution and range. This approach, which we will refer to as the multi-image modification, considers the network's performance when training on a set of pixels sampled from multiple hyperspectral images.

To incorporate the information of multiple images, we combine multiple images' pixels as independent samples and generate the training and validation sets through a proportional uniform random selection. This means that the training and validation sets have a proportional number of pixels sampled in comparison with the original image. The test set was each of the original images being compressed separately.

Since this is combining the pixels from multiple images, the pixels were padded to be of uniform size (i.e., 224 bands) for training the model. With the LSTM not recognizing the wavelength values behind the bands and viewing the bands as a standard sequential set, we did not have to adjust the bands for differing spectral resolutions and wavelength ranges for the bands. This process allows us to save overall training time since a single model with the same dimensions but a larger training set will take significantly less time to train than models trained on individual images.

With the relaxation of the requirement to retrain models for new scenes, we experiment with the generality of a trained model to compress an image that was not included in the training data. This generalized modification was trained and tested similarly to the multi-image modification, but it employed a "leave one out" strategy where the set of training images only contained 4 out of the 5 images. The image that was not used to train the model was instead used entirely as the test data for the trained network. This modification requires that the verification process is modified such that each of the models was trained using the "leave one image out" strategy and resampled 5 times to validate the results.

4.3 Experimental Approach

In this section, we provide an overview of the procedures that we applied to test our proposed LSTM-AE and BLSTM-AE models for single image hyperspectral image compression. We experimented with the single image approach to compression as well as analyzing our proposed LSTM-AE and BLSTM-AE models' robustness for the applicability

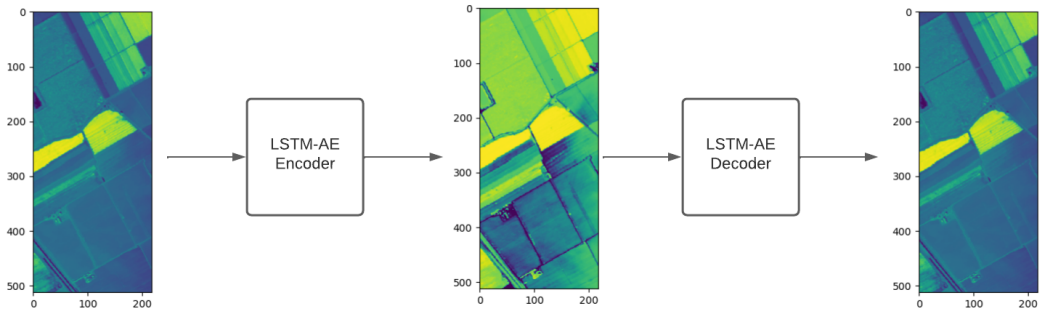


Figure 4.2: Single image compression approach.

of performing multi-image compression. We also experimented with the generality of the model by analyzing our proposed LSTM-AE and BLSTM-AE models' performance on single image compression with images not seen during the training process.

4.3.1 Single Image Compression

The traditional approach for image compression using a machine learning model is to train the model on a subset of pixels from the image that is going to be compressed than to compress the entirety of the same image using the trained model. With a spectral-spatial strategy, the trained model is reused on images that have very similar spatial elements. These deployed models require frequent retraining to minimize the reconstruction error.

For the first set of experiments, we followed this approach of training on the image to be compressed. Given a subset of pixels for training data, we trained our proposed LSTM-AE compression model to minimize the reconstruction error of the compressed features with the additional emphasis on minimizing the compression rate. We then compressed the full image and analyzed the compression results on that image.

Figure 4.2 demonstrates the compression process used where the input Salinas image is sent to the encoder, and then the decoder recovers the original image. The image between

the encoder and decoder is a false-color representation of the extracted features.

We implemented both the LSTM-AE and BLSTM-AE approaches to compress the images. We tested to see if the increase in parameters the BLSTM-AE approach incorporates had a significant impact on the reconstruction error in comparison to a standard LSTM-AE. We expected degradation in the compression rate that coincides with the doubling of the number of nodes that the BLSTM-AE approach has. We hypothesize that the BLSTM-AE will have a reduction in the reconstruction error across the board.

We compared the algorithm to three state-of-the-art compression approaches, with each of them taking different algorithms to hyperspectral image compression: linear transformation, wavelet transformation, and non-linear transformation. Our hypothesis for this experiment was that we could achieve comparable compression rates while improving the reconstruction error.

After the experiments with single image compression, we investigated how reducing the number of pixels sampled for training impacted our proposed algorithm's compression reconstruction performance. We iteratively reduced the training set that the model used to train to observe the impact that the reduced data set had.

We applied this reduction in the number of training samples to both the LSTM-AE and BLSTM-AE algorithms. Since the comparison algorithms require the entire image either to both directly compress as well as to train on, we compared the reduced training set experiments directly from the previous single image compression experiment's performance for both LSTM-AE and BLSTM-AE. To be able to compare the models directly as well to identify the impacts that the reduced dataset had on the proposed model more reliably, we took the standard network's tuned hyperparameters and reused them for the training process of the reduced set. We hypothesize that with the high levels of spectral information in the pixels and with a sufficient representation of the variety of pixels in the training set, the reduced training set will not have a significant reduction in the reconstruction error rates.

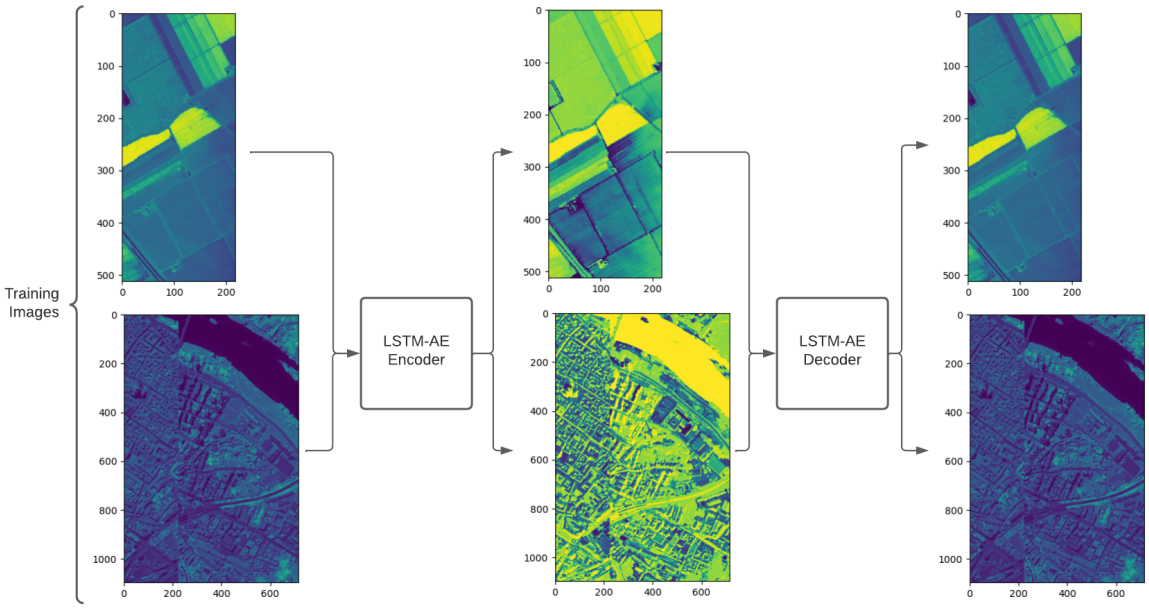


Figure 4.3: Generalized Image Compression.

4.3.2 Robust Image Compression

One of the benefits of a spectral-only strategy is the relaxation of the spatial dependency of a given pixel. This relaxation allows our proposed LSTM-AE and BLSTM-AE models both to not require retraining when the spatial information significantly changes, but we also may be able to generalize models for the compression of multiple scenes using an existing trained model. We experimented with the proposed framework's ability to generalize a single trained model to compress many hyperspectral images. For this, we generated a training data set by sampling a set of pixels from each of the images. To ensure that randomization did not select proportionally more pixels from one image over another, we used a proportional uniform random selection process to ensure that each image provided a proportional level of pixels related to the size of the image. To ensure uniformity in the spectral data, we padded the band data until all of the spectral dimensions were of uniform length.

Since the proposed network is only taking into consideration the spectral band information, we hypothesize that if the contents of the image have been previously seen in the training data then the compressor will be able to compress the image successfully with low reconstruction error rates. Additionally, since the proposed framework is using an LSTM cell to process the spectral information as a numerical sequential data point, we do not have to consider the spectral resolution or the wavelength size when combining the information across images. This further demonstrates the significance of the LSTM-AE framework since most hyperspectral image compression models are designed to compress a specific sensor.

Figure 4.3 demonstrates the compression process used where both the Salinas and Pavia Center datasets are compressed. The training dataset of the encoder and decoder are sampled from both of the images. To compress, each image is sent into the encoder where a compressed representation is saved for each image. Alongside the compressed image is information about how long the original bands were. The decoder can then take each compressed image and reconstruct the original image using the compressed representation as well as the saved length of the spectral dimension.

Given that the comparison algorithms perform spatial-spectral compression, they can not utilize this approach. Therefore, we compared the same LSTM-AE model selected for the standard compression to be used to test this multi-image compression approach. To be able to directly compare the models as well to more reliably identify the impacts that the new generalized dataset had on the proposed model, we took the standard network's tuned hyperparameters and reused them for the training process of the reduced set. In particular, based on the results of the reduced set, we compared against the reduced approach to take advantage of the improved runtime. We hypothesize that the increased levels of generality will not have a significant change in the performance of the reconstruction of the compressed image.

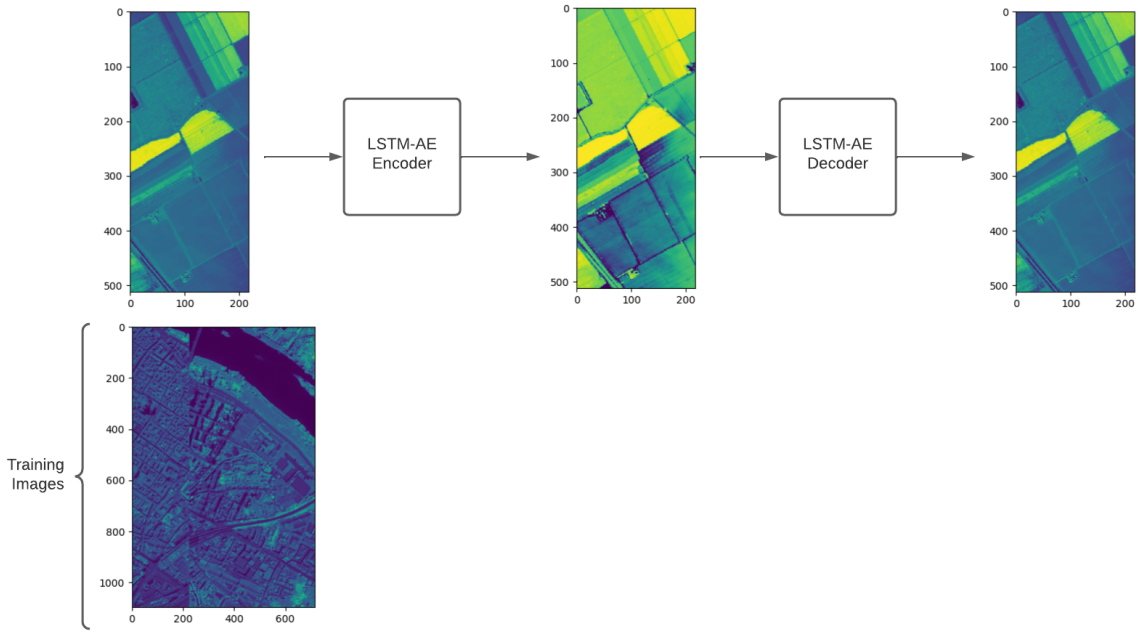


Figure 4.4: Transfer image compression approach.

With the multi-image compression results, we experimented with the capabilities of the proposed model to be utilized to compress images that were left out of the training set. For this, we generated a training data set by selecting a subset of the pixels from a set of images in the same manner as the generalized approach. The difference is that we left one of the images out of the training set. After training the model, we then tested its compression performance by compressing the image that was left out of the training set. This demonstrates the robustness of the model to relax the requirement further to retrain significantly different scenes. Since we also use different sensors with a variety of platforms and applications, this also demonstrates the proposed model's capabilities to compress images of differing spectral resolutions and independent of the wavelengths behind the bands. Further, this also reduces computational requirements since the model can be trained in a data center and then reused to compress the images of a variety of sensors used in the field.

Figure 4.4 demonstrates the compression process used where the Salinas dataset is compressed. The training dataset is generated from the Pavia Center dataset to train the model. Then, the Salinas dataset is compressed with the LSTM-AE encoder where the compressed representation is saved. The original band length is saved alongside the compressed image. The decoder can then take each compressed image and reconstruct the original image using the compressed version as well as the saved length of the spectral dimension.

We compared the same LSTM-AE model selected for the single image compression to be used to test this generalized compression approach. To be able to compare the models directly as well to identify the impacts that the unseen image had on the proposed model more reliably, we took the standard network's tuned hyperparameters and reused them for the training process of the reduced set. As with the previous experiments, we also compared against the reduced approach to take advantage of the improved runtime performance. We hypothesize that the further increase in the levels of generality will not have a significant change in the performance of the reconstruction of the compressed image unless patterns previously unseen by the network are in the image to be compressed.

4.4 Experimental Design

In this section, we will describe the experimental design that each of the experiments applied. We will be discussing the methods utilized to select the compression model, discuss how we sampled the data, and discuss how we validated our results.

4.4.1 Tuning

To tune the model, we performed a random search optimization [3] process due to the large number of hyperparameters to tune. The random search hyperparameter algorithm utilized is defined in Algorithm 4.1 where the hypersphere is the range of values that can be

sampled.

Algorithm 4.1 Random Search Algorithm Inspired by Villalobos-Arias *et al.*[124]

Require: $t \geq 0$ ▷ Number of steps before termination

Initialize $\lambda \in \mathbb{R}^{|\lambda|}$

Evaluate model with parameters λ

while $t \geq 0$ **do**

Sample a new λ' from hypersphere(λ) and evaluate

if $\lambda' > \lambda$ **then**

$\lambda \in \lambda'$

end if

$t \leftarrow t - 1$

end while

The following hyperparameters were tuned: autoencoder network depth, number of nodes at each layer, dropout rate, size of mini-batch, number of epochs, initial learning rate, and the activation function used. While we use the Adam method which uses an adaptive learning rate, we found that modifying the initial learning rate did impact convergence rates. We pre-determined the number of samples from the dataset for training, the number of samples reserved for validation, and the encoded layer size. Since we found no significant reconstruction error benefits for networks with a lower compression rate, we set the encoding layer to be a single node to maximize the compression rate. We predetermined that the autoencoder will have the same network shape on both the encoder and decoder side as well as only selecting networks that were under-complete, or had a reduction in parameters at the encoding layer. To evaluate the model, we calculated the Mean Squared Reconstruction Error ($MSRE$) of the validation set to compare the loss functions of the models.

We defined the hypersphere λ in Algorithm 4.1 to include the following sampling

strategies. Through experimentation, we found that sampling from the uniform distribution allowed for the most effective hyperparameter exploration for our set of hyperparameters. We set the size of the hypersphere to be: 1 for the autoencoder network depth with a minimum network depth of 2, 5 for the number of nodes at each layer with a minimum layer size of 2, 0.1 for the dropout rate, 64 for the size of the mini-batch process, 64 for the number of epochs, 0.0005 for the learning rate, and we randomly switched between the sigmoid and hyperbolic tangent activation functions. To find the optimal network, we set t to be 1000 networks to be tuned.

The optimal values that the random selection identified was a network with the shape $20 \times 5 \times 1 \times 5 \times 20$, with the dropout rate being 0.1, a mini-batch size of 128, 120 epochs, and 0.0009 for the learning rate. We also found the hyperbolic tangent activation function performed the best for minimizing reconstruction error. These values were consistently found to converge quickly to the minimal reconstruction error rate.

4.4.2 Data Selection

To select the pixels to be used in the training set, we performed a uniform random sampling process. For the single image compression approach, we randomly selected 70% of the data to be training data and 15% of the data to be reserved as the validation set. Figure 4.5 demonstrates the pixels selected from the Indian Pines image for generating the training set where the yellow pixels are the selected pixels.

For the reduced experiments, we applied a grid search process to iteratively observe the impacts of reducing the training set size of the proposed LSTM-AE model. We reduced the number of pixels sampled from the images for the training set from 70% to 5% of the image being selected for training. Figure 4.6 demonstrates the number of pixels selected from the Indian Pines image to generate the training set. The white pixels are the selected pixels used for training.

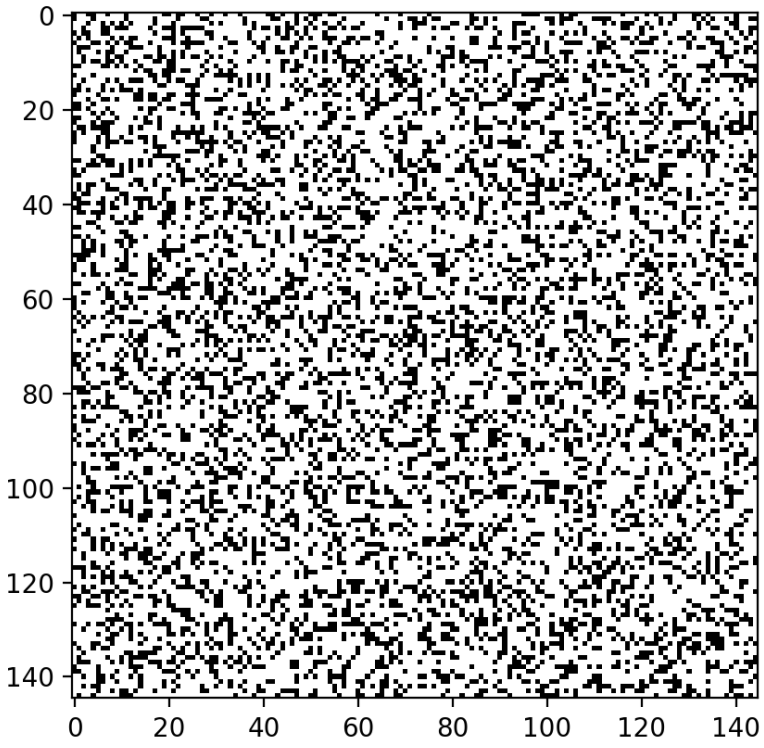


Figure 4.5: Visual representation of pixels selected from the Indian Pines dataset for training. The white regions are the selected pixels.

For the robustness experiments, we used a proportional uniform random selection process to select 5% of the total number of pixels from the set of all pixels. The proportional uniform random selection process sampled the number of pixels from each image proportional to the number of pixels in the image compared to the total number of pixels in the images to be used for training.

4.4.3 Validation

Since we are training on a subset of the image and then compressing the whole image as our reported test results, we could not perform k-fold stratified cross-validation on the

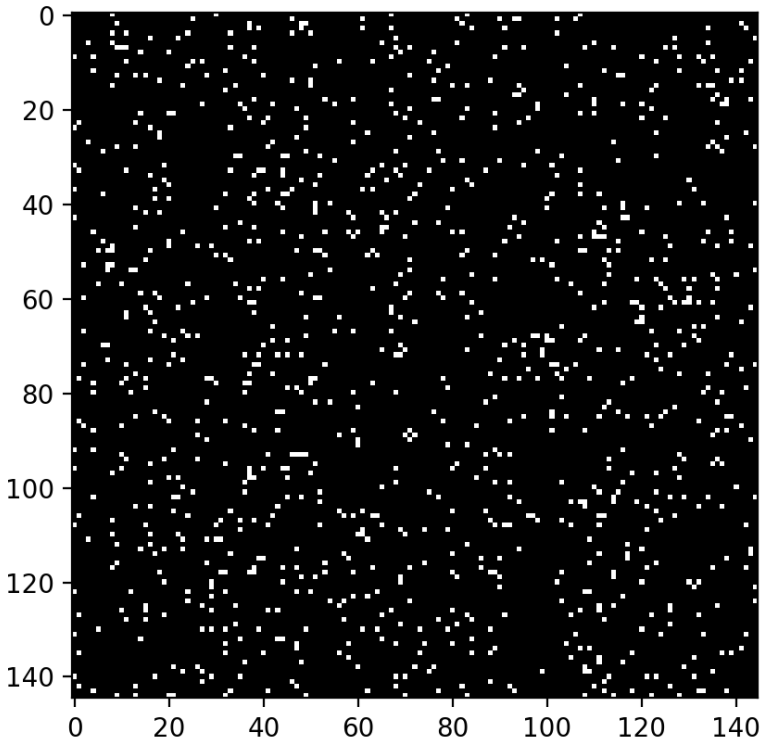


Figure 4.6: Visual representation of pixels selected from the Indian Pines dataset for training and validation. The white regions are the selected pixels.

outcomes of the models. This is a common practice for hyperspectral image compression tasks since the goal of the algorithm is to compress the whole image [25] except for numerical compression processes that do not train a model such as PCA [108].

Instead, we resampled training data for each of the images five times and for each resampling, we compressed the whole image again. For each of the experiments, we set a seed to be able to replicate the samples across each of the experiments to better analyze the performance of the models. This allowed us to negate the impact random chance had on modifying the performance of the proposed models. This was applied to both the single image and multi-image compression tasks.

For the generalized LSTM experiments, we performed a "leave one out" strategy where for each resampling round we sampled the training set on all of the images excluding a single image. This process was applied so that each of the images is used as a test image. Each image has the training and compression process applied five times so that each image is tested a total of five times.

To validate the significance of all of the experiments, we performed a permutation-based paired t-test to determine the level of significance between the reconstruction error and compression rate. The permutation-based paired t-test allows us to provide a more reliable analysis since we are utilizing only 5 images to compare the performance of our proposed algorithm. The reported p-value will be able to give us some insight into how much of a change there is between each of the compared methods.

4.5 Results

The evaluation metrics that were used include compression rate (CR), peak signal-noise ratio ($PSNR$), and Mean Squared Reconstruction Error ($MSRE$). The compression rate is defined as the ratio of the total number of the compressed image bits to the original image bits given by the following:

$$CR = \frac{\text{\#bytes to store compressed image}}{\text{\#bytes to store original image}}$$

To measure the levels of reconstruction error in the compression of hyperspectral images, we utilize $MSRE$. This metric is defined as:

$$MSRE = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2$$

where n is the number of pixels in an image, $\mathbf{x}_i$ is the original sequence of bands in pixel i, and $\hat{\mathbf{x}}_i$ is the decompressed sequence of bands in pixel i.

A more common measurement of the reconstruction error rates of hyperspectral images is $PSNR$. This metric measures the ratio of the maximum power of the signal to the noise affecting the signal. The calculation of $PSNR$ is based on $MSRE$ for all pixels in an image with an inverse logarithm scale where the higher the $PSNR$ the lower the reconstruction error rates. This is used to represent the quality of the compression process.

$$PSNR = 10 \cdot log_{10} \left(\frac{R^2}{MSRE} \right)$$

where R is the maximum possible reflectance value of any given pixel. For the images we have selected, $R = 65535$. The main advantage of calculating $PSNR$ instead of reporting only the $MSRE$ for image compression quality is that it allows for better comparisons for images with differing numbers of encoding bits as well as differing reflectance values. The objective of compression is to minimize both the compression rate and the reconstruction error rates, thus maximizing $PSNR$.

A random search optimization was applied for each experiment to find the optimal parameters for the LSTM-AE models averaged for all 5 of the images. The hyperparameter tuning process balanced the CR and $PSNR$. We found that all of the experiments were consistent for the following sets of parameters: the number of epochs of 120, a dropout rate of 0.1, a learning rate of 0.0009, and a mini-batch size of 128. Figure 4.7 demonstrates the convergence of the model towards an optimal solution. The models were retrained on a uniformly random resampled set of pixels a total of 5 times to validate the results on the training set.

With an LSTM using sigmoid and hyperbolic tangent functions, we normalized the reflectance values from [0, R] to $[0, 1]$. With each image having differing maximum possible

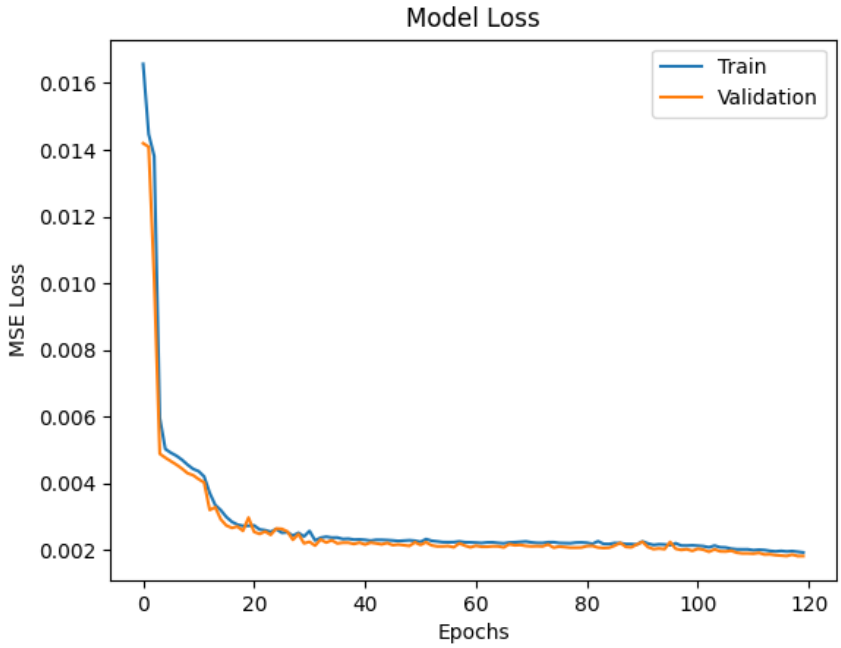


Figure 4.7: Example training performance of the LSTM autoencoder on the Salinas dataset

values for the bands, we will keep the original and decompressed sequences normalized for the calculation of $MSRE$. To save computation time, the networks were trained with the loss function being $MSRE$ since the $PSNR$ is dependent on this value.

4.5.1 Compared Methods

The results of LSTM-AE and its modifications were compared to the results of three spatial-spectral compression methods: Discrete Wavelet Transform Support Vector Machine (DWT-SVM) [141], non-iterative factorized tensor decomposition and principal component analysis (NFTD+PCA) [108] with 16 principal components to balance the compression rate with an acceptable $PSNR$ value, and a Convolutional Neural Network Autoencoder (CNN-AE)[25].

DWT-SVM incorporated the discrete wavelet transformation compression process with a support vector regression model. DWT-SVM was selected for this comparison because it

Models	LSTM-AE 20x5x1	BLSTM-AE 20x5x1	LSTM-AE 15x3	LSTM-AE 100x20x6	CNN-AE
Indian Pines	0.14542	0.29084	0.07128	3.31672	29.98272
Pavia Center	0.067422	0.134844	0.033048	1.537752	29.98272
KSC	0.116336	0.232672	0.057024	2.653376	29.98272
Salinas	0.148064	0.296128	0.07276	3.377024	29.98272
Botswana	0.095845	0.19169	0.04698	2.18602	29.98272

Table 4.3: The MFLOPs required for each network to perform compression on each image.

is a compression method that incorporates a combined spatial-spectral compression process where the DWT approximation sparsely represented the spatial dimension and the support vector regression weights are then used to obtain coefficients of the spectral dimension for compression.

Non-iterative factorized tensor decomposition (NFTD) decomposes an $N \times N$ matrix into two $1 \times N$ matrices. The authors of NFTD+PCA apply PCA on the hyperspectral image to get a lower-dimensional spatial-spectral data cube and apply DWT to acquire coefficients for the NFTD to further compress the PCA results. NFTD+PCA was selected due to the combined transformation approaches from both PCA on the whole image and the tensor decomposition to further compress the PCA process.

CNN-AE utilizes each convolutional layer to generate a lower-dimensional spatial-spectral data cube in a similar architecture to an Autoencoder. To decode the compressed image, the authors utilized transpose convolution layers to reverse the encoding convolutions. CNN-AE was selected due to the combined spectral-spatial approach as well as being the closest method to the proposed implementation.

To determine if the difference in performance scores was statistically significant between

Models	**LSTM-AE 20x5x1**	**BLSTM-AE 20x5x1**	**LSTM-AE 15x3**	**LSTM-AE 100x20x6**	**CNN-AE**
Parameters	4,549	9,078	1,336	53,621	594,976

Table 4.4: The number of parameters that are utilized by each network.

LSTM-AE and the other networks, a permutation-based paired t-test comparing the means of both $PSNR$ and the CR was performed. For this, we made sure the datasets sampled the data the same way using fixed random seeds for the CNN-AE and LSTM-AE. Since the DWT-SVM and NFTD+PCA used the full image to compress and are deterministic, we did not have to fix the random seeds. In this case, the null hypothesis is that the calculated $PSNR$ and CR were drawn from the same distribution at significance level α.

To compare the runtime performance, we report the number of training parameters required and we calculated the number of mega floating-point operations needed (MFLOPs) to compress each pixel for LSTM-AE and CNN-AE. Table 4.4 shows the number of parameters and Table 4.3 shows the number of MFLOPS for each of the Deep Neural Network compressors. In the table, we list each of the different neural network sizes utilized in this thesis. During testing, the memory consumed by a model is used to store the outputs of intermediate layers as well as the parameters of the model. In addition, the number of parameters is related to the complexity of the model and the amount of data needed to train the model effectively. On the other hand, the number of MFLOPS is related to the time complexity of processing the model.

4.5.2 Results on Single Image Compression

The average CR, $PSNR$, and $MSRE$ were reported on the Indian Pines, Pavia Center, KSC, Salinas, and Botswana datasets are reported in Table 4.5 and Table 4.6. We report the results for both the LSTM-AE modification and the BLSTM-AE. We also experimented with

	LSTM-AE		BLSTM-AE		3D DWT +SVR		NFTD +PCA		CNN-AE	
Dataset	$PSNR$	$MSRE$	$PSNR$	$MSRE$	$PSNR$	$MSRE$	$PSNR$	$MSRE$	$PSNR$	$MSRE$
Indian Pines	62.07	0.0020	**64.84**	**0.0015**	42.37	0.0145	49.92	0.0068	54.85	0.0041
Pavia Center	61.24	0.0022	**72.73**	**0.0007**	43.01	0.0136	46.78	0.0093	55.17	0.0040
KSC	78.70	0.0004	**78.80**	**0.0004**	42.84	0.0138	48.16	0.0081	54.15	0.0044
Salinas	65.40	0.0014	**66.73**	**0.0013**	41.32	0.0161	46.78	0.0093	54.68	0.0042
Botswana	89.53	0.0001	**91.40**	**0.0001**	44.92	0.0112	49.48	0.0071	57.29	0.0033

Table 4.5: Comparison between each of the methods for each of the images with results from the full single image compression. The bold values indicate highest average performance.

seeing how far we could reduce the training set without a significant loss in the compression performance. Through a grid search optimization process, we reduced the training set to be 5% of the image's pixels and a validation set of 5% of the pixels to be used for training. Table 4.7 reports the average CR, $PSNR$, and $MSRE$ of the 5% training set for both LSTM-AE and BLSTM-AE.

4.5.3 Results on Robustness

We experimented with seeing how the models performed when the training set was sampled from multiple images to create a single compression model to be used for all of the images. Table 4.8 reports the average CR, $PSNR$, and $MSRE$ calculated from the results of models with training sets generated by uniformly selecting 5% of pixels from all data sets. Specifically, the values captured represent the compression performance for when each of the images was compressed.

We further experimented with the robustness of our approach through training a generalized model but leaving one of the images out to be compressed without being seen previously. We report the average CR, $PSNR$, and $MSRE$ of each of the images in Table 4.9. Specifically, the values captured represent the compression performance for when each

	LSTM-AE	**BLSTM-AE**	**3D DWT +SVR**	**NFTD +PCA**	**CNN-AE**
Dataset	CR	CR	CR	CR	CR
Indian Pines	**0.0045**	0.0090	0.037	0.0077	0.0078
Pavia Center	0.0098	0.0195	0.039	0.0073	**0.0063**
KSC	0.0057	0.0114	0.038	0.0043	**0.0027**
Salinas	0.0045	0.0090	0.046	0.0037	**0.0022**
Botswana	0.0069	0.0138	0.043	0.0052	**0.0032**

Table 4.6: Comparison of the compression rates between each of the methods for each of the images with results from the full single image compression. The bold values indicate highest average performance.

of the images were compressed during the "leave one out" validation strategy.

4.5.4 Statistical Analysis

With a total of 25 results on each model for each of the experiments, there are too few samples to effectively utilize a paired t-test to compare the experimental results. To resolve this, we utilized a permutation-based paired t-test analysis for all of the experiments to analyze the differences between each of the models. Since we have 25 results for each model, it is not computationally feasible to calculate all possible permutations. With this, we opted to use a sampled permutation t-test that generated 5000 permutation samples. Each sample represents a permutation of the results from which a t-statistic can be calculated. We then captured the p-value to be proportional to the number of simulated t-values that are as or more extreme than the one observed in the non-permuted data. With the reduced reliability that the sampled permutation t-test incurs with it not being an exact statistical hypothesis test, we state the results are significantly different if the p-value is less than 0.1.

	LSTM-AE			BLSTM-AE		
Dataset	CR	$PSNR$	$MSRE$	CR	$PSNR$	$MSRE$
Indian Pines	**0.0045**	**63.27**	**0.0018**	0.0090	62.25	0.0020
Pavia Center	**0.0098**	66.38	0.0013	0.0196	**72.70**	**0.0007**
KSC	**0.0057**	78.56	0.0004	0.0114	**78.69**	**0.00038**
Salinas	**0.0045**	**67.28**	**0.0012**	0.0090	66.73	0.0013
Botswana	**0.0069**	**92.20**	**9.90e-5**	0.0138	89.63	0.0001

Table 4.7: Comparison results between LSTM-AE and BLSTM-AE for each of the images with results from the reduced training set. The bold values indicate highest average performance.

We first compared the compression rates from the proposed LSTM-AE and BLSTM-AE models against the spatial-spectral algorithms. The p-values are reported in Table 4.10. Each element in Table 4.10 represents the p-value when comparing the model labeled at the given row and column.

We then expanded our comparison to the reconstruction error with the $PSNR$ metric. We compared not only the state-of-the-art spatial-spectral algorithms to the single image LSTM-AE and BLSTM-AE models, but we also compared our proposed single image compression methods against our proposed modifications. In particular, we compared the results of reducing the dataset, the results of performing multi-image compression, and the results of generalizing the model against the single image model. The reported p-values are reported in Table 4.11.

4.6 Discussion

As shown in Table 4.5, the $PSNR$ ratio performance of the LSTM-AE and BLSTM-AE models was consistently over a 15% increase over the CNN-AE, with the largest increase

	LSTM-AE			BLSTM-AE		
Dataset	CR	$PSNR$	$MSRE$	CR	$PSNR$	$MSRE$
Indian Pines	0.0045	84.64	0.0002	0.0090	**86.19**	0.0002
Pavia Center	0.0098	91.78	0.0001	0.0196	**93.35**	8.83e-5
KSC	0.0057	78.89	0.0004	0.0114	**79.13**	0.0004
Salinas	0.0045	88.28	0.0001	0.0090	**88.44**	0.0001
Botswana	0.0069	90.19	0.0001	0.0138	**93.01**	9.13e-5

Table 4.8: Comparison between LSTM-AE and BLSTM-AE for each of the images with results from the multi-image compression experiment. The bold values indicate highest average performance.

being 60% in the Botswana image. Table 4.11 demonstrates that the increases found in the reconstruction error rates has evidence supporting a significant increase in performance over the state-of-the-art methods. It is also important to note that since the $PSNR$ metric is a logarithmic scaled function, this increase leads to a large reduction in the calculated mean squared error rate. Furthermore, as shown in Table 4.10, the significant improvement in reconstruction rates was not at a significant loss in CR when compared to the NFTD+PCA and CNN-AE, whereas the DWT-SVM consistently had the worst compression rate. This suggests that not only can the proposed spectral LSTM-AE compression approach achieve comparable results, supporting H1 as stated in Section 1.4. Since this is a spectral-only compression method, there is potential that the image might be further compressed spatially. It also suggests that we do not need to sacrifice the relevant spatial information of hyperspectral images, allowing us to directly classify the compressed image.

In Tables 4.5 and Table 4.7, we can see that when comparing the whole image to the reduced dataset, there was little change in performance between the images for LSTM-AE except in the case of the Pavia Center image, which saw an 8% increase in $PSNR$. This

	LSTM-AE			**BLSTM-AE**		
Dataset	CR	$PSNR$	$MSRE$	CR	$PSNR$	$MSRE$
Indian Pines	**0.0045**	79.90	0.0003	0.0090	**82.92**	**0.0003**
Pavia Center	**0.0098**	86.95	0.0002	0.0196	**89.17**	**0.0001**
KSC	**0.0057**	78.16	0.0004	0.0114	**78.85**	**0.0004**
Salinas	**0.0045**	87.29	0.0002	0.0090	**88.33**	**0.0001**
Botswana	**0.0069**	85.17	0.0002	0.0138	**87.92**	**0.0002**

Table 4.9: Comparison between LSTM-AE and BLSTM-AE for each of the images with results from the generalized compression experiment. The bold values indicate highest average performance.

	LSTM-AE	**BLSTM-AE**
DWT-SVM	0.121	0.4214
CNN-AE	0.5766	0.1836
NFTD-PCA	0.7416	0.2502

Table 4.10: The calculated p-values from the permutation t-tests for comparing the compression rates.

is verified by 4.11 where the p-value being over 0.4 shows there is very little evidence that this is a significant change in performance. The small increase in compression performance can be explained best as overfitting the full image due to having too much of the image be seen during the training process. In the case of BLSTM-AE, the reduction in the number of pixels selected for training saw a reduction from the full image compression approach across all of the images. This reduction in performance might be due to the BLSTM being a network twice as large as the LSTM-AE. It is important to note that this difference is not statistically significant, so the reduction in training time might be more beneficial. With

	S-LSTM-AE	**S-BLSTM-AE**
DWT-SVM	0.0578	0.0618
CNN-AE	0.0648	0.0635
NFTD-PCA	0.0612	0.0568
Reduced	0.7456	0.4998
Multi-Image	0.1163	0.1176
Generalized	0.1192	0.1200

Table 4.11: p-values from the permutation t-tests for comparing the $PSNR$ values. The S-LSTM-AE and S-BLSTM-AE refers to the single image compression experiments.

this, both models support H2 as stated in Section 1.4.

As shown in Tables 4.5, 4.7, 4.8, and 4.9 the $PSNR$ improved significantly for most of the image with Indian Pines, Pavia Center, and Salinas, with an over 33% increase over the full image compression $PSNR$. As summarized in Table 4.11, there was not a statistically significant difference with both the multi-image and generalized LSTM-AE and BLSTM-AE models. This in large part is due to the lack of performance improvements with the Kennedy Space Center images when compared to the single image compression experiment. With these results, the differences found in Tables 4.7 and 4.8 support our H3 as stated in Section 1.4. Additionally, the differences in Tables 4.7 and 4.9 support our H4. Unlike in the reduced training set experiments, the BLSTM-AE was consistently an improvement over the LSTM-AE, even if not by a statistically significant amount. The lack of statistical significance in the performance of the BLSTM-AE model across all experiments suggests that hypothesis H5 is not supported by the results of the experiment.

It is interesting to note that the models trained on the KSC and Botswana datasets did not see the same rise in $PSNR$ values. This may be due to the loss of a large number of bands captured by the sensors that were removed. M. Graña stated that 21% of the original

bands in KSC and 40% of the original bands in Botswana were removed [80]. Due to a large number of bands removed, there may not have been enough unique spectral information to be learned from other datasets. This also demonstrates the relationship between spectral information and the model's ability to maximize $PSNR$.

Comparing Tables 4.5, 4.7, and 4.8, it is demonstrated that the proposed LTSM-AE approach and the modification of BLSTM-AE are both capable of robust image compression. This is significant since we can train a single compression model to be used across many scenes with changing subjects in the image without retraining. We can also transfer that model for compression of not only differing scenes but across different sensors with different captured spectral information. This is not possible with the spectral-spatial approach due to the dependency on the spatial information which requires periodic retraining and the models to be trained for a specific sensor.

Table 4.4 demonstrates the required number of parameters and Table 4.3 demonstrates the amount of floating-point operations required to perform compression tasks. The proposed LSTM-AE requires fewer training parameters and operations than the modern CNN-AE approach for compression tasks. This supports our hypothesis H6 in Section 1.4. With the additional robustness of the model, we can further reduce the computational cost of the proposed approach to compress hyperspectral images.

4.7 Conclusions

In this chapter, we presented a Long Short Term Autoencoder model to compress HSI in the spectral dimension. Many previous approaches focus on a spectral-spatial compression strategy, which increases the risk of the reconstruction error increasing by combining sources of loss in both the spectral dimension and the spatial dimension. Our experimental results demonstrate that our low computational cost framework cannot only reduce reconstruction error, but we do not sacrifice compression rates. Our experiments also demonstrate the

novelty of this framework where the model is robust enough to both be able to be trained on multiple images to be able to reuse a trained model for many scenes. We also demonstrated that a trained model can be generalized to scenes that capture different subjects without needing to be retrained. Our experimental results demonstrated that the increased network size with utilizing a Bidirectional Long Short Term Memory Autoencoder model does not lead to a further reduction in the reconstruction error rates.

In Chapter 5, we perform one of the first analyses to consider the effectiveness of directly classifying compressed hyperspectral images to further reduce the computational cost of analyzing the data contained within hyperspectral images. We will also be analyzing the impacts that relaxing the retraining requirements for hyperspectral image compression algorithms has on classifying compressed images.

CHAPTER FIVE

CLASSIFYING COMPRESSED HYPERSPECTRAL IMAGES

In this chapter, we perform one of the first analyses on the effectiveness of classification on the compressed HSI. Utilizing the images compressed by LSTM-AE in Chapter 4, we train a Convolutional Neural Network on the compressed images. We begin with an overview of the related work, describe the classification model utilized, and provide experimental results.

5.1 Related Work

The increased feature space of HSI can lead to powerful, high accuracy classifiers, but the large level of spectral and spatial information captured can make training the classifiers slow and susceptible to overfitting. HSI bands are also highly correlated, so the level of redundancy and noise within images can be large, reducing their applicability to users with smaller resources available [24]. With these challenges, methods to reduce the size and processing requirements of HSI have the potential to open up the technology to even more applications.

Feature extraction is the process of taking the features from a dataset to reduce dimensionality, thereby representing them as a smaller set of new features. For HSI, the most common form is with latent variables selected from Deep Neural Networks (DNN) [60]. In particular, recent works have integrated a spatial dimension reduction process that then passes the data to a Convolutional Neural Network (CNN) [137]

Several spectral feature extraction techniques have been applied in the past with the primary goal to condense the spectral information into fewer features for HSI classification. This reduces the time complexity when classifying the images, as well as sometimes improves a classifier's performance [137]. Feature extraction does not necessarily lead to dimensionality

reduction, but many feature extraction methods have made great strides in reducing the dimensions of HSI [19, 64, 138].

Dimensionality reduction is looking for low-dimensional representations of HSIs [137]. Spectral analysis dimensionality reduction methods can be classified into either unsupervised or supervised. Unsupervised dimensionality reduction does not utilize labeled information to extract low-dimensional data patterns. One of the most common of these methods, principal component analysis (PCA), removes the redundancy in the spectral data and transforms the data into a smaller dimension by mapping to the k principal components [120].

Supervised dimensionality reduction uses the labeled information to separate the data points. Another common dimensionality reduction algorithm is Partial Least Square-Discriminant Analysis (PLS-DA) [13]. Partial Least Square regression expresses latent variables between two matrices $\mathbf{X}$ and $\mathbf{Y}$ by modeling the covariance structure between the data and the classes. PLS-DA is an extension of the PLS algorithm that is utilized when $\mathbf{Y}$ is categorical. By maximizing the covariance between the independent variables $\mathbf{X}$ and the classes $\mathbf{Y}$, a linear subspace of explanatory variables can be used to perform predictions. For hyperspectral feature extraction, PLS-DA extracts features by finding a linear regression model from the data points to the class label of that data point. This model is then used to perform classification on the most beneficial features.

Deep learning has been growing in popularity recently for supervised feature extraction. For example, some of the popular methods utilize stacked autoencoders (SAE) and convolutional neural networks (CNN) [60]. These methods can generate both high-level spatial and spectral features. SAEs extract deep features in a hierarchical manner, which allows them to be connected directly to other models for classification [17]. CNNs have been successful at reducing the required level of parameters in comparison to SAE while being expandable to discovering spatial features in addition to spectral features [16].

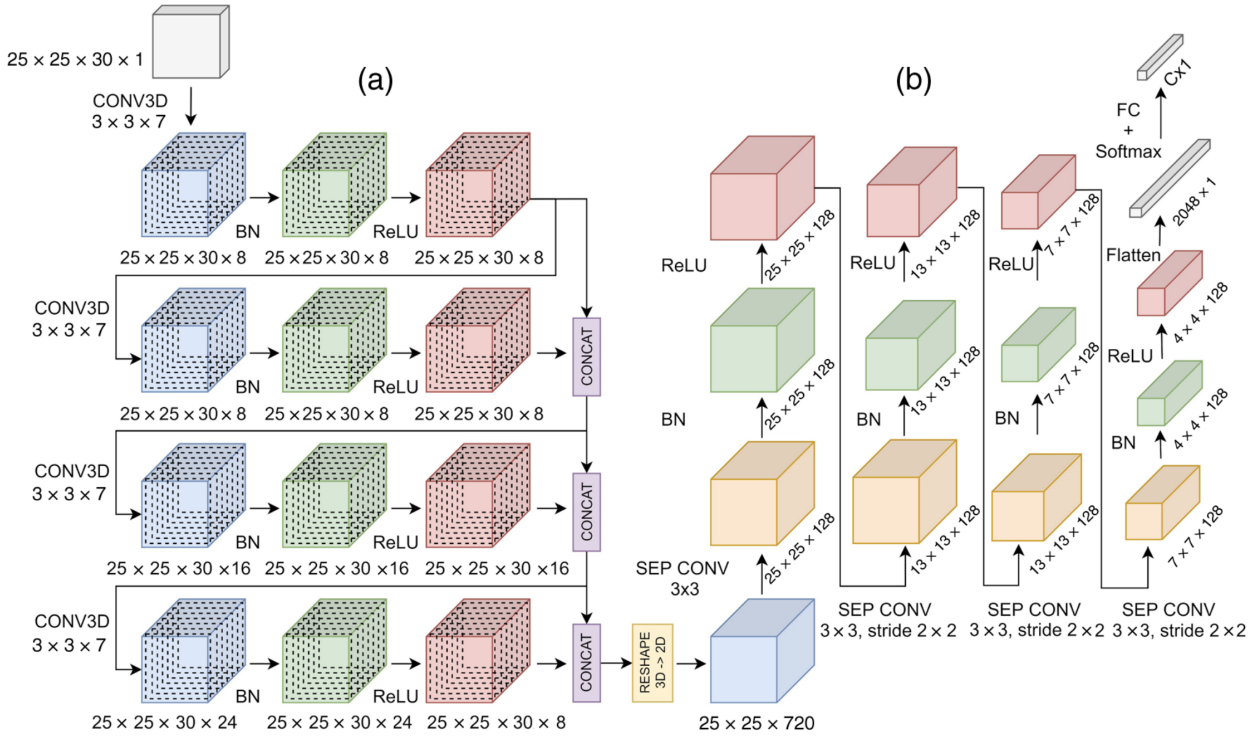


Figure 5.1: Hyper3DNet Architecture [93]

5.2 Compressed Image Classification

Autoencoders have been demonstrated to be an effective method of feature extraction [68]. Given the LSTM-AE's compressed images are extracted features, the compressed images can be input to a classifier to perform classification on the image. In the following, we will discuss the classification model that we selected to perform classification on the images we have compressed from Chapter 4. The selected network has demonstrated strong classification performance when classifying both the original images as well as after performing substantial dimensionality reduction on the image.

5.2.1 Hyper3DNet

The classifier that we have selected is the 3D-2D CNN Hyper3DNet architecture proposed by Morales *et al.*[93]. This architecture uses layers of 3D and 2D convolutional filters to classify the images. The 3-D convolutional layers perform feature extraction on a data cube with the shape $W \times W \times D$ where the width and height W represents the spatial information of the pixels and spectral depth D represents the bands of the compressed image. The 2-D CNN layers perform spatial encoding. Figure 5.1 shows the first iteration of the Hyper3DNet architecture with its 3-D feature extraction and 2-D spatial encoder.

The feature extractor consists of a 4-layer densely connected block where each layer contains a $3 \times 3 \times 7$ 3-D convolution layer that contains eight filters (denoted in Figure 5.1 as "CONV3D") a batch normalization layer, and a rectified linear unit activation layer (ReLU). Concatenation is used to ensure the output of the feature extraction process is a stack of 32 data cubes.

The spatial encoder, shown on the right of Figure 5.1, is used to perform 2-D convolutional operations on the data. The input tensor coming from the feature extractor is reshaped from a 4-D stack of data cubes into a 3-D tensor. The spatial encoder takes the 3-D tensor and gradually compresses it into an encoded feature vector. This is accomplished by performing 3×3 separable convolutions with 128 filters. After the convolutions are performed, a batch normalization layer and ReLU activation layer are utilized to reduce the risk of the vanishing gradient problem. To obtain the classification results, the output is reshaped into a 1-D tensor by layering all values from each channel into a vector. Then, a fully connected layer with a softmax activation estimates a multinomial probability distribution over the classes.

5.2.2 Hyper3DNetLite

Hyper3DNetLite is a modification of the Hyper3DNet architecture that is optimized for data that contains a reduced number of non-contiguous bands [92]. Hyper3DNetLite is a simplification of Hyper3DNet to reduce the required number of training parameters the network used. The main difference between the two networks is that Hyper3DNetLite consists of two 3-D convolutional layers instead of the 4-layer densely connected block and the 2-D spatial encoder contains three convolutional layers instead of the original four. Another difference is that the flatten operation used after the last "SepConv2D" layer is replaced with a "GlobalAveragePooling" operation. This is used to transform the 3-D tensor into a 1-D vector by averaging the values from each channel. With the example of a tensor with dimensions $7 \times 7 \times 256$ pixels, the "GlobalAveragePooling" operation would transform the tensor into a vector with 256 elements by averaging the 49 values from each channel while the flatten operation would obtain a vector with 12,544 elements.

The modifications on the network relax the required number of parameters to train for the network. This simplification allows for more efficient classification, especially with hyperspectral images that have dimensionality reduction techniques applied to them. This also reduces the risk of overfitting the data by reducing the overparameterization of the Hyper3DNet architecture.

Table 5.1 shows the architecture utilized for Hyper3DNetLite. Note that the stride size of the last two "SepConv2D" layers was set to (1,1) instead of the (2,2) as defined by Morales *et al.*[92]. This was done to avoid dimensionality inconsistencies with how many features are selected. In addition, with the networks compressed to be a single feature, the filter size was reduced to a (3, 3, 1) for the "Conv3D" layers, also to prevent dimensionality inconsistencies.

Hyper3DNetLite has demonstrated a high classification performance on low dimensional data as well as being a low-cost classifier. Morales *et al.*introduced an Inter-

Layer Name	**Kernel Size**	**Stride Size**	**Output Size**
Input	—	—	(25, 25, N, 1)
Conv3D + ReLU	(3, 3, 3)	(1, 1, 1)	(25, 25, N, 16)
Conv3D + ReLU	(3, 3, 3)	(1, 1, 1)	(25, 25, N, 16)
Reshape	—	—	(25, 25, 16N)
SepConv2D + ReLU	(3, 3)	(1, 1)	(25, 25, 320)
SepConv2D + ReLU	(3, 3)	(1, 1)	(13, 13, 256)
SepConv2D + ReLU	(3, 3)	(1, 1)	(7, 7, 256)
GlobalAveragePooling	—	—	256
Dense + Softmax	—	—	# classes

Table 5.1: Hyper3DNet-Lite architecture.

Band Redundancy Analysis and Greedy Spectral Selection (IBRA-GSS) method alongside the Hyper3DNetLite architecture to perform feature selection on hyperspectral images. Hyper3DNetLite demonstrated high classification performance with as few as 5 features. This demonstrates that Hyper3DNetLite is robust enough to the features our proposed LSTM-AE model generates to compress while not sacrificing classification performance.

5.3 Experiment Approach

In this section, we provide an overview of the procedures that we applied to test the compressed images generated by our proposed LSTM-AE for hyperspectral image classification using Hyper3DNetLite. We experimented with the single image compression model as well as the robust modifications introduced in Chapter 4. We generated three different networks, each with different compression rates. The models we analyzed compressed the images to 1, 3, and 6 features.

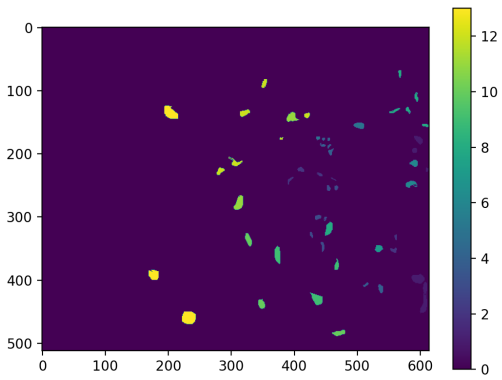


Figure 5.2: KSC Classification Map

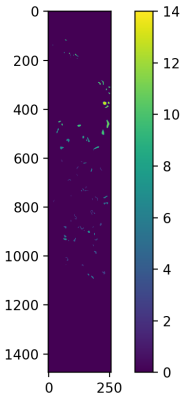


Figure 5.3: Botswana Classification Map

5.3.1 Single Image Compression

The traditional approach for image compression uses a machine learning model is to train the model on a subset of pixels from the image being compressed and then compress the image using the trained model. This process is closest to existing machine learning feature extraction methods that train on the data set that the method will perform dimensionality reduction on. Similar to feature learned feature extraction methods, if the scene changes significantly enough, then the model has to be retrained.

For these experiments, we trained the Hyper3DNetLite network on the images that were compressed using this approach. Given a subset of pixels for training data, we trained our proposed LSTM-AE compression model to minimize the reconstruction error. We then compressed the full image. We took the compressed image and input the compressed image into the Hyper3DNetLite training process as a reduced feature set.

For training, the Hyper3DNetLite framework generates 5×5 pixel-sized overlapping patches across all of the labeled pixels. We selected the training and testing set from these patches. We specified selecting the labeled pixels for our training and test set because some of the images that we have selected have few labeled pixels that cover their entire spatial

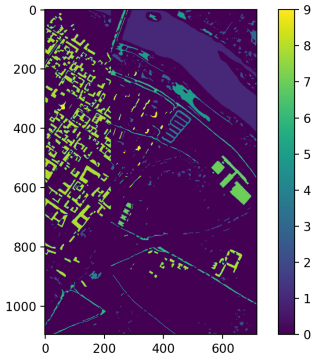


Figure 5.4: Pavia Classification Map

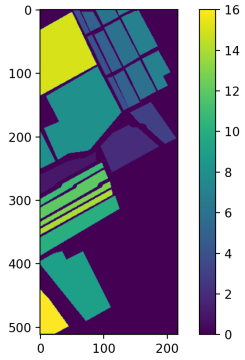


Figure 5.5: Salinas Classification Map

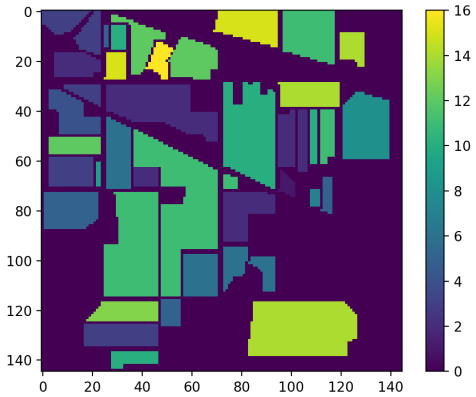


Figure 5.6: Indian Pines Classification Map

regions. For example, Figures 5.2 and 5.3 demonstrate that both the Kennedy Space Center and Botswana datasets few labeled pixels that contain the entire region of the subject being classified. The Pavia Center dataset improves on this issue, but as Figure 5.4 demonstrates, there are still vast regions that are unlabeled. The agricultural datasets, Salinas and Indian Pines are the exception and have all of their classes fully labeled within the images as shown in Figures 5.5 and 5.6.

We compared the performance of the LSMT-AE's compressed images as extracted

features to the IBRA-GSS feature selection approach. We selected this algorithm due to the level of features extracted yielding high classification results. We also selected the IBRA-GSS algorithm to analyze the interactions between the features generated by compression. It should be noted that few modern spectral feature extraction methods have been experimented on with fewer than 6 spectral features [103]. We also noticed that the LSTM-AE compression to a single feature was similar to a grayscale image, so we also compared the single feature compression LSMT-AE model to a grayscale representation of the hyperspectral image. e hypothesize that the feature extraction process used to compress the hyperspectral images can effectively be used to perform classification on the compressed images directly. We expect that the IBRA-GSS will perform better than our compression model due to the supervised nature of IBRA-GSS.

We also experimented with how reducing the compression rate, thus increasing the number of features in the dataset that the Hyper3DNetLite algorithm used to classify an image. We selected the number of features that were analyzed to be one, three, and six. For a fair comparison, we compared the number of bands in the IBRA-GSS algorithm to be the same as the number of features in the compressed image. We expect that increasing the number of features will lead to an improvement in classification performance.

5.3.2 Robust Compression

Since the robust modifications of the LSTM-AE framework were demonstrated to be effective, we expanded our analysis of the generated features that the robust modifications generated to assess their performance for classification. The robust modifications allowed us to compress not only many images with a single classifier, but also images that were not seen in the training process, thus, removing the requirement to retrain the feature extractor for each different scene.

For these experiments, we trained the Hyper3DNetLite network on the images that

were compressed using the robust approach. We looked at both the multi-image compression model as well as the generalized model for these experiments. Since we are analyzing the impacts the increased levels of robustness had on the classification performance, we compared the robust models to the single image compression model. We also used the same network structures that were tuned to be the most optimal networks for the standard compression approach for a fair comparison.

For the multi-image compression model, we generated our subset of pixels for the training data from all of the images that were to be compressed to train our proposed LSTM-AE compression model to minimize the reconstruction error of the compressed features. Next, we compressed each of the images separately. We then took those compressed images and input them into the Hyper3DNetLite training process as the reduced feature set for each image. We hypothesize that utilizing a single model to compress multiple images will not lead to a degradation in the classification performance on the compressed hyperspectral image.

For the generalized model, we generated our subset of pixels for the training data from all of the images except the image to be compressed to train our proposed LSTM-AE compression model. Next, we compressed the image that was left out. We took the compressed image and used it as part of training the Hyper3DNetLite network. We hypothesize that generalizing the model to compress unseen images will also not lead to a degradation in the classification performance on the compressed image.

5.3.3 Correlation Analysis

When looking at feature extraction, it is well documented that the reduction in the number of features can lead to a degradation in the classification performance due to the loss of information, especially when the dimensionality reduction generates too few features [22, 63, 92]. This leads to the trade-off between compression rates and the classification

performance where, in general, reducing the compression rate is required to improve the classification performance. With the results found in Chapter 4, we found that a model compressing an image into a single feature was able to encode and decode a hyperspectral image with low levels of reconstruction error. The results bring into question how strong the relationship is between the compression rate, reconstruction error rate, and classification performance.

Given the multiple objectives that are tested between the LTSM-AE compression and the classification analysis, analyzing the correlation between the compression rates, $PSNR$, and classification performance may be beneficial in better understanding the trade-off between the optimization of the LSTM-AE hyperspectral image compression approach and the optimization hyperspectral image classification on the compressed image. We can use correlation analysis to aid in the identification of the strength of the assumed trade-off. This additionally may be beneficial in achieving a better understanding of the information extracted from the generated features.

For this analysis, we used the Spearman rank correlation method to identify the association between the classification rate, the compression rate, and the reconstruction error rates. The Spearman rank correlation is a non-parametric test used to measure the degree of association between two variables. We selected the Spearman rank correlation over the Pearson correlation method due to potential nonlinearity in the results between the single image compression approach and the generalized compression approach.

Since the Spearman rank correlation is the correlation between two values, we compared each of the metrics to each of the other metrics. We then calculated a correlation strength, as well as the likelihood of the found correlation strength, being significant. We also wanted to see if the correlation trends carried over to each of the differing levels of robustness in the proposed LTSM-AE compression. We performed the same correlation calculations as previously performed, but we segregate the data by the different approaches used to generate

the test results.

The correlation coefficient, ρ, varies between -1 and 1, where 0 implies no correlation. ρ is a metric of the strength of the correlation. As an example, a ρ value of 1 implies an exact monotonic relationship between variables x and y, where as x increases, so does y. The p-value roughly indicates the likelihood of correlation [58]. Due to the lower number of samples from these experiments, the ρ and p-values may not be reliable but can provide insight into the relative correlations of the compression and classification processes.

5.4 Experimental Design

In this section, we will describe the experimental design that each of the experiments applied. We will be discussing the methods utilized to select the classification model, discuss how we sampled the data, and discuss how we validated our results. We will also address the changes found when generating the different compression rates.

5.4.1 Tuning

To tune the model, we performed a random search optimization process due to the large number of hyperparameters to tune. The random search hyperparameter algorithm utilized is shown in Algorithm 4.1 where the hypersphere is the range of values that can be sampled.

The following hyperparameters were tuned for the LSTM-AE models: autoencoder network depth, number of nodes at each layer, dropout rate, size of mini-batch, number of epochs, learning rate, and the activation function used. We pre-determined the number of samples from the dataset for training to be 5% due to the high compression performance while improving the training time as well as the encoded layer size. We specified that the autoencoder had the same network shape on both the encoder and decoder side as well as only selecting networks that were under-complete. We also found that using the Adam optimizer for the LSTM-AE model was led to lower reconstruction error rates. To evaluate

the model, we calculated the $MSRE$ of the validation set to compare the loss functions of the models. We used the same hypersphere sampling limitations as described in Section 4.4.1 The optimal values that the random selection identified for the network with the shape $20 \times 5 \times 1 \times 5 \times 20$ were found to have a dropout rate of 0.1, a mini-batch size of 128, 120 epochs, and 0.0009 for the initial learning rate. The optimal values for the network with the shape $15 \times 3 \times 3 \times 15$ were found to have a dropout rate of 0.1, a mini-batch size of 32, 140 epochs, and 0.001 for the initial learning rate. The optimal values for the network with the shape $100 \times 20 \times 6 \times 20 \times 100$ were found to have a dropout rate of 0.1, a mini-batch size of 64, 180 epochs, and 0.0008 for the initial learning rate.

For the Hyper3DNetLite model, we used a grid search to find the optimal mini-batch size as well as the number of epochs to train the model. We found that a mini-batch size of 128 and 350 epochs maximized the classification performance. We also found that the Adadelta optimizer [134], a gradient descent method based on an adaptive learning rate, was more optimal for training than the Adam optimizer for Hyper3DNet-Lite. We found that an initial learning rate of 0.1 leads to increased classification performance.

5.4.2 Data Selection

In this section, we discuss the data selection process used for both compression and classification. We discuss both to identify the data used to train the compressed image as well as the variations that the Hyper3DNet model requires to train.

For compression, we opted to use the reduced selection set described in Section 4.4.2 because we can reduce the training time for the proposed LSTM-AE model without significantly reducing our compression performance. To select the pixels to be used in the training set, we performed a uniform random sampling process. For the single-image compression model, we uniformly randomly selected 5% of the data to be training data and 5% of the data to be reserved as the validation set. For the robustness experiments, we

used a proportional uniform random selection process to select 5% of the total number of pixels from the set of all pixels. The proportional uniform random selection process sampled the number of pixels from each image proportional to the number of pixels in the image compared to the total number of pixels in the images to be used for training.

For classification, we generated the 5×5 patches from the compressed images. We used the ground truth information to identify the pixels that contained labels to center all of the patches on. We then treated each of the patches as a data point during the classification process.

We used a 5×2 stratified cross-validation design, where we randomly generated two equal-sized bins of patches that contain equivalent ratios of classes on the selected pixels as there are in the ground truth. We trained Hyper3DNetLite on one bin and tested the model with the other bin. Then, we reversed the roles of the bins and repeated the training and testing process. This process is then repeated four more times. Furthermore, the stratification indicates that each bin has the same proportion of samples of each given class.

5.4.3 Validation

In this section, we will discuss the validation methods utilized to validate the classification performance. For the compressed image generation, we repeated the validation approaches discussed in Section 4.4.3.

As stated in Section 5.4.2, we used a 5×2 stratified cross-validation design to validate the performance of the classification model. We generate five compressed images on each approach, compression rate, and on each image. Since each image is treated as a dataset, we performed the 5×2 stratified cross-validation design to validate the performance of the classification model on each of the compressed images. For each of the images that used the same network and the same approach, we combined the classification results to generate the final reported results.

To analyze the performance across all of the networks, the macro-averaged $F1$ score was calculated. The $F1$ score is the harmonic mean of precision and recall. The equations for these metrics are given below:

$$Precision = \frac{1}{\#Classes} \sum_{c=1}^{\#Classes} \frac{TP_c}{TP_c + FP_c}$$

$$Recall = \frac{1}{\#Classes} \sum_{c=1}^{\#Classes} \frac{TP_c}{TP_c + FN_c}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

where TP_c, FP_c, and FN_c represents the true positive of classe c, the false positive of class c, and the false negative of class c, respectively.

To determine if the difference in performance scores was statistically significant between LSTM-AE and the other networks, a permutation-based paired t-test was performed for both $PSNR$ and the CR. We performed this test on the results of the single image LSTM-AE compression model, the multi-image LSTM-AE compression model, and the generalized LSTM-AE compression model as well as the IBRA-GSS method. In this case, the null hypothesis was that the samples' $F1$ scores were drawn from the same distribution at significance level α.

5.5 Results

We also introduce two variations of the LSTM-AE models from Chapter 4. We tuned networks that compress the image into a 3 feature and 6 feature representation. The new networks, of layer size $15 \times 3 \times 15$ and $100 \times 20 \times 6 \times 20 \times 100$, will be referred to as LSTM-AE 3 and LSTM-AE 6 respectively. The network tuned in Chapter 4 will be referred to as LSTM-AE 1. The network parameter and MFLOPs requirements for all compression models are found in Table 4.4.

Compared Method	S-LSTM-1	S-LSTM-3	S-LSTM-5
Gray Scale	0.012	0.556	0.637
IBRA-GSS	0.005	6.32e-05	4.21e-04
Multi-Image	0.015	0.063	0.042
Generalized	0.002	0.017	0.002

Table 5.2: Maximum p-values for all images when comparing the classification performance of the single image LSTM compression model to the other methods.

For conciseness, we report the maximum p-value across all of the datasets for each model in Table 5.2. In the table, S-LSTM represents the single image LSTM Compression model. The multi-image and generalized rows represent the multi-image compression model and generalized compression model, respectively. For each row, except the Gray Scale row, the compared model reported is classified with the same number of features as the single image LSTM-AE compression model in a given column.

5.5.1 Compared Methods

In Chapter 4, we demonstrated the LSTM-AE model's ability to compress an HSI to a single band; therefore, we will compare the LSTM-AE adaptation to a standard grayscale image. We also generate a baseline grayscale image by transforming each HSI into a false RGB image. The three bands used for false RGB were selected by dividing the whole spectrum into three equivalent intervals and selecting the mean value [2]. The false RGB image was then transformed into grayscale using $(b_1 + b_2 + b_3)/3$, where b_i represents the mean value of the bands in the ith interval in the false RBG image.

We first validated that the reconstruction error that the LSTM-AE compression model created during the compression of a hyperspectral image did not impact the classification results. We compared the decompressed image of a single image compression model against

Dataset	Uncompressed Image	Decompressed Image	p-value
Indian Pines	99.253 ± 0.586	98.314 ± 0.473	0.893
Salinas	99.687 ± 0.372	97.992 ± 0.583	0.466
Pavia Center	99.431 ± 0.133	99.021 ± 0.303	0.739
KSC	99.543 ± 0.264	99.105 ± 0.376	0.973
Botswana	99.689 ± 0.114	99.425 ± 0.153	0.579

Table 5.3: $F1$ Scores for uncompressed image and the decompressed image along with the p-values associated with the results of the images.

Feature Size	Generality	Indian Pines	Pavia Center	KSC	Salinas	Botswana
1 Band	**Single Image LSTM-AE**	71.373 ± 1.876	83.957 ± 1.051	66.459 ± 1.84	85.262 ± 0.663	84.077 ± 1.655
	Gray Scale	63.416 ± 1.866	80.551 ± 0.711	57.895 ± 4.141	83.262 ± 1.016	77.799 ± 1.083
	IBRA-GSS	62.813 ± 1.461	74.140 ± 584	57.498 ± 2.434	76.972 ± 1.449	79.546 ± 1.043
3 bands	**Single Image LSTM-AE**	69.126 ± 1.348	79.906 ± 0.350	66.638 ± 1.828	80.492 ± 1.701	74.66 ± 1.615
	IBRA-GSS	87.84 ± 0.41	98.034 ± 0.094	84.85 ± 1.112	98.584 ± 0.089	99.411 ± 0.221
6 Bands	**Single Image LSTM-AE**	66.881 ± 1.433	76.031 ± 0.055	63.378 ± 2.928	76.719 ± 0.411	74.152 ± 0.771
	IBRA-GSS	98.24 ± 0.387	98.904 ± 0.107	98.85 ± 0.57	99.54± 0.04	99.31 ± 0.42

Table 5.4: $F1$ Scores for 1, 3, and 6 compressed features, compared to gray scale and IBRA-GSS

the original image.

To compare the results of the LSTM-AE model to a dimensionality reduction approach, the Inter-Band Redundancy Analysis and Greedy Spectral Selection (IBRA-GSS) method was selected due to its state-of-the-art performance [92]. This feature selection gives us the baseline to compare the features we generate from the selected most informative bands in the original image. This also has the benefit of allowing us to analyze the benefits of training on the compressed image in comparison to just decompressing the image before classifying it.

Models	Generality	Indian Pines	Pavia Center	KSC	Salinas	Botswana
LSTM-AE 1	**Single Image LSTM-AE**	71.373 ± 1.876	83.957 ± 1.051	66.459 ± 1.84	85.262 ± 0.663	84.077 ± 1.655
	Multi-Image LSTM-AE	79.635 ± 1.877	98.702 ± 0.148	72.693 ± 1.680	90.025 ± 1.839	88.33 ± 0.837
	Generalized LSTM-AE	86.24 ± 1.115	99.185 ± 0.071	81.520 ± 0.877	95.434 ± 0.354	91.161 ± 1.982
LSTM-AE 3	**Single Image LSTM-AE**	69.126 ± 1.348	79.906 ± 0.350	63.638 ± 1.828	80.492 ± 1.701	74.66 ± 1.615
	Multi-Image LSTM-AE	76.893 ± 1.673	97.801 ± 0.167	68.732 ± 1.824	84.623 ± 1.663	79.456 ± 0.736
	Generalized LSTM-AE	82.48 ± 1.284	98.034 ± 0.094	77.573 ± 1.112	91.844 ± 0.484	82.552 ± 2.139
LSTM-AE 6	**Single Image LSTM-AE**	66.881 ± 1.433	76.031 ± 0.055	63.378 ± 2.928	76.719 ± 0.411	74.152 ± 0.771
	Multi-Image LSTM-AE	74.764 ± 1.133	95.744 ± 0.118	68.035 ± 1.735	81.442 ± 1.584	79.860 ± 0.966
	Generalized LSTM-AE	82.38 ± 1.331	97.947 ± 0.090	78.114 ± 1.547	89.998 ± 0.285	83.011 ± 1.903

Table 5.5: $F1$ Scores for LSTM-AE's Single Image, Multi-Image, and Generalized

5.5.2 Results on Single Image Compression

We experimented with seeing how the compression process impacted the classification performance of the LSTM-AE model. We selected the single image LSTM-AE model due to its lowest compression rates shown in Chapter 4. Table 5.3 reports the average and standard deviations of the classification performance on each image. The table demonstrates the impact that performing the encoding and decoding steps of the LSTM-AE model in comparison to the original image. The results show a reduced performance when comparing the uncompressed image to the decompressed image, though this reduction is small with an overlap in the standard deviation between the images.

We then experimented with comparing the single image compression to the state-of-the-art IBRA-GSS algorithm. We performed this experiment over three separate feature sizes for the models. Table 5.4 reports the average and standard deviations of the performance on each of the compressed images and the grayscale images when they were classified by Hyper3DNetLite. In Table 5.2, we can see that the single image LSTM-AE compression for a single feature was significantly different from the compared models. For a single feature, we found that it was a significant improvement over the grayscale and IBRA-GSS models. For

Models	Generality	Indian Pines		Pavia Center		KSC		Salinas		Botswana	
		$PSNR$	CR	$PSNR$	CR	$PSNR$	CR	$PSNR$	CR	$PSNR$	CR
LSTM-AE 1	**Single Image LSTM-AE**	63.27	0.0045	66.38	0.0098	78.56	0.0057	67.28	0.0045	92.20	0.0069
	Multi-Image LSTM-AE	84.64	0.0045	91.78	0.0098	78.89	0.0057	88.28	0.0045	90.19	0.0069
	Generalized LSTM-AE	79.90	0.0045	86.95	0.0098	78.16	0.0057	87.29	0.0045	85.17	0.0069
LSTM-AE 3	**Single Image LSTM-AE**	63.11	0.0135	74.89	0.0294	78.61	0.0171	64.83	0.0135	87.96	0.0207
	Multi-Image LSTM-AE	64.97	0.0135	75.84	0.0294	78.93	0.0171	66.32	0.0135	89.11	0.0207
	Generalized LSTM-AE	83.53	0.0135	92.04	0.0294	79.04	0.0171	87.92	0.0135	89.03	0.0207
LSTM-AE 6	**Single Image LSTM-AE**	61.69	0.027	75.16	0.0588	78.53	0.0342	65.39	0.027	87.93	0.0414
	Multi-Image LSTM-AE	62.14	0.027	78.24	0.0588	78.64	0.0342	65.98	0.027	89.26	0.0414
	Generalized LSTM-AE	82.88	0.027	92.33	0.0588	78.44	0.0342	86.91	0.027	88.94	0.0414

Table 5.6: $PSNR$ and CR for each of the LSTM-AE networks.

both the three and six feature models, we see a degradation in the classification performance where the single image compression led to being on par with the grayscale image and being significantly worse in performance over the IBRA-GSS method.

5.5.3 Results on Robust Compression

For this experiment, we evaluated how the increased levels of robustness in the LSTM-AE compression model impacted the classification performance. Table 5.5 presents the average $F1$ score for each model variation when the compressed images were classified using Hyper3DNetLite. We performed this experiment across 1, 3, and 6 compressed features.

Table 5.2 shows the p-values for comparing the single image compression models to the multi-image compression models as well as the generalized compression model. The single feature multi-image and generalized compression models significantly improved classification performance. We still see significant improvement with the generalized compression model, but we have a lower significance rate for the multi-image compression model when comparing the p-values between each of the single image LSTM-AE Compressors.

	Value	$PSNR$-CR	$PSNR$-$F1$	$F1$-CR
Single Image LSTM-AE	ρ	0.425	0.037	0.634
	p-value	0.11	0.89	0.011
Multi-Image LSTM-AE	ρ	0.423	-0.179	0.557
	p-value	0.112	0.523	0.03
Generalized LSTM-AE	ρ	0.195	0.5	0.717
	p-value	0.487	0.058	0.003

Table 5.7: Spearman correlation of compression and classification, showing the correlation coefficient ρ and associated p-values of that correlation.

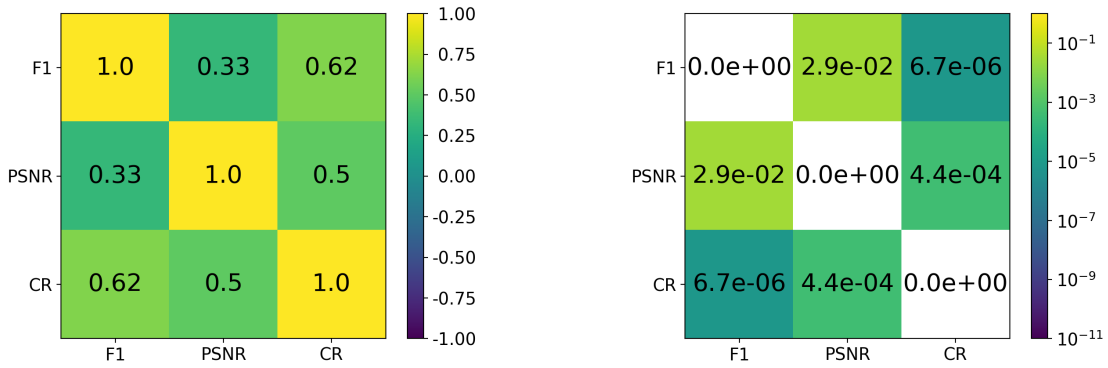
5.5.4 Correlation Results

Table 5.6 shows the compression results for each of the networks. Figure 5.7a shows a heat map of the Spearman correlation coefficient when comparing the compression rates, the $PSNR$, and the $F1$ scores. Figure 5.7b shows a heat map of the Spearman correlation p-values for comparing the compression rates, the $PSNR$ reconstruction error, and $F1$ scores.

Table 5.7 reflects each experiment's Spearman correlation ρ and p-values for comparing the compression rates, the $PSNR$, and the $F1$ scores. The table represents the ρ and p-values for each of the LSTM-AE network's training modes and how the results correlate with the classification performance. We examined each experiment separately because, while it further reduced the number of samples, it also provided a more representative view of the patterns seen during the experiments.

5.6 Discussion

From Table 5.4 and Table 5.5, we can conclude that not only can we successfully perform hyperspectral image classification using the compressed images, but this also has



(a) Spearman correlation coefficient of compression and classification.

(b) The p-values comparing compression and classification.

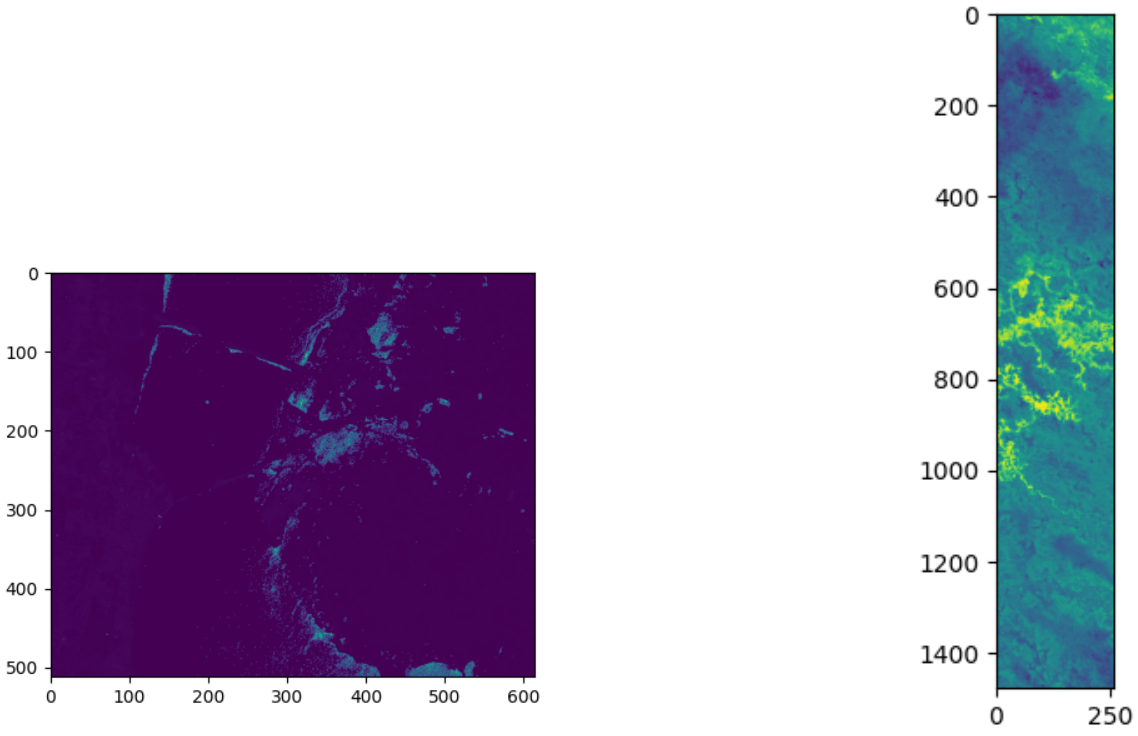
Figure 5.7: Spearman correlation coefficient and associated p-values.

the potential to achieve high classification performance. All of the features learned from the LSTM-AE outperformed the baseline grayscale images, which supported hypotheses H7, H8, and H9 as described in Section 1.4. We also found that by expanding the generality of the compression model, there was an increase in classification performance across each of the networks, whereas, the increasing number of bands lead to a reduction.

Since the IBRA-GSS method was optimized for band selection and not compression, it makes sense that the IBRA-GSS algorithm performed much better than the unsupervised LSTM-AE compression approach. What is interesting is that the single feature LSTM-AE compression was more effective for classification than selecting a single band. What this suggests is that the original bands are not as informative by themselves as they are in a group whereas the single compression band integrates the information of several bands into a single value.

Figures 5.4 - 5.6 present maps of the classification labels of the hyperspectral images. We expected that the images with a smaller set of labels would perform significantly worse than the images with a larger proportion of the pixels labeled. When reviewing the classification

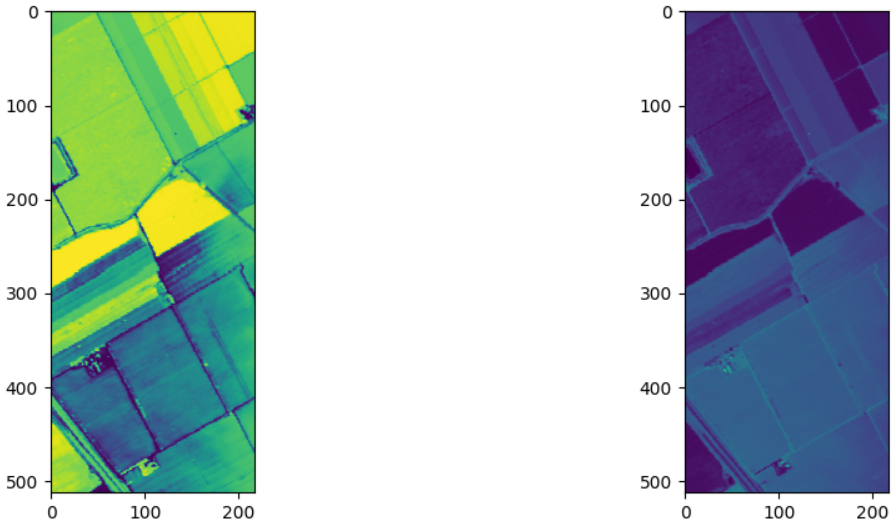
performance and the map of the classification labels of Pavia Center (Figure 5.4) and Salinas (Figure 5.5), we found that this assumption may not have been accurate since Pavia Center performed on par with Salinas. This expectation continued to be demonstrated false with the Kennedy Space Center (Figure 5.2), but Botswana (Figure 5.3) consistently performed better than Kennedy Space Center while having a much smaller proportion of pixels labeled. Botswana also consistently performed better than Indian Pines (Figure 5.6) even though Indian Pines had more than half of its pixels labeled.



(a) Kennedy Space Center (b) Botswana

Figure 5.8: False color view of the compressed Kennedy Space Center image and the compressed Botswana image.

From Table 5.4, we do see a general pattern that, as the number of compressed

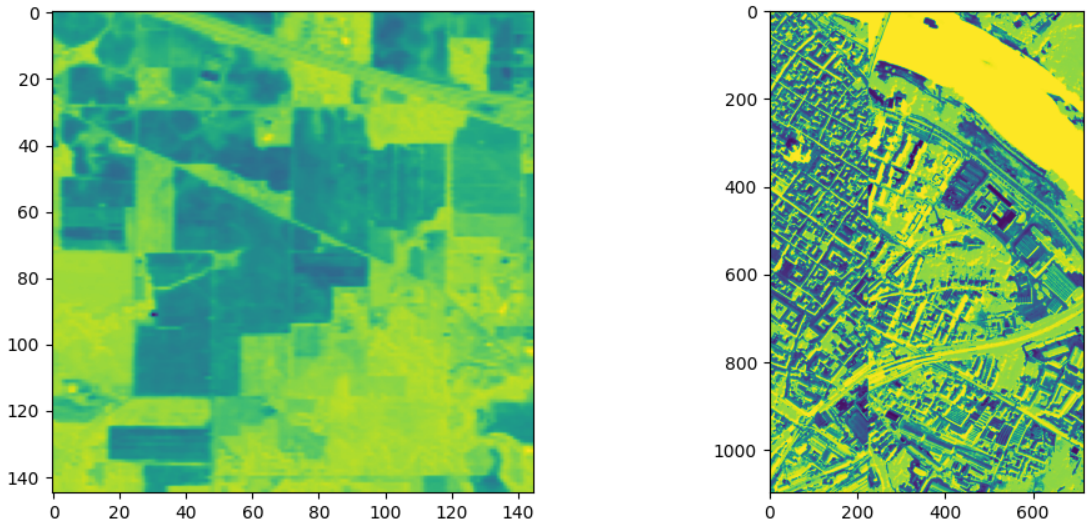


(a) Salinas by Single Image LSTM-AE (b) Salinas by Generalized LSTM-AE

Figure 5.9: False color view of the Single Image LSTM-AE compressed Salinas vs the Generalized LSTM-AE compressed Salinas.

features saved increases, so does the classification performance. Alternatively, Table 5.5 shows a pattern whereas the level of generality of the compression model increases, and the classification performance increases slightly.

One observation is that the images that perform better have features that often contain a wider range of values for features. This observation is most prominent in Figure 5.8 where the differences in the values of the features in the compressed KSC dataset are hardly noticeable to the visible eye and the classification performance increased along with the increased range of values in the compressed image. Figure 5.9 is another demonstration of a counterexample of this phenomenon where the performance increased alongside a reduction in the range of values between the same image with different compression implementations. The Generalized implementation performed better than the Standard implementation, which translates better to the observation that the model tends to perform better when distinct separations occur



(a) Indian Pines (b) Pavia Center

Figure 5.10: False color view of the compressed Indian Pines image and the compressed Pavia Center image.

between the classes of the image. As another example, Figure 5.10 demonstrates that while the range of values for the features of both images is relatively large, there is very little separation between the classes in the Indian Pines image compared to the Pavia Center image. In particular, misclassifications of the Indian Pines image occurred the most where the fields visually appeared to be the same.

Figures 5.7a and 5.7b reveal that there is evidence of a positive correlation between compression performance and classification performance. In particular, there is evidence that as the level of compression is reduced, the level of classification performance increases. This is intuitive, but it also provides a basis for determining an appropriate level of compression when classification is also an end goal. In addition, being able to tie a positive correlation between $PSNR$ and the $F1$ score would allow for further improvements when selecting optimal network sizes, which would further reduce the required resources to train the models.

Table 5.7 provides further evidence for some form of correlation between the compression rate and the $F1$ score. What we found as a trend in this table was that, the more generalized the compression model, the greater the evidence for a correlation between compression and classification performance.

5.7 Conclusion

In this chapter, we performed an analysis of the capabilities of performing HSI classification tasks on images compressed by LSTM-AE. Experimental results show that while there is significant loss between IRBA-GSS and the features learned by LSTM-AE, we are still able to successfully perform classification tasks on the compressed images. The results also demonstrated that the increase in generality led to an increase in classification performance.

CHAPTER SIX

CONCLUSION

Hyperspectral imaging has become a reliable source of spatial and spectral information. The spectral information in particular has been central to the analytical power of this technology. The capabilities of hyperspectral imaging have been utilized across many critical applications, but the image sizes are still a bottleneck for general use.

Traditional compression approaches to deal with the storage problem of hyperspectral images often combine a spectral and spatial strategy to maximize the compression rate the models can achieve. Conversely, dimensionality reduction techniques to deal with the processing limitations of hyperspectral images are often calculated for each image without being able to be reused and the reduced features cannot be recovered if additional details are required.

We presented a hyperspectral image compression framework that focuses on a spectral-only strategy. The framework utilizes a Long Short Term Memory Autoencoder (LSTM-AE) architecture to compress the spectral bands at each pixel into a reduced set of features. Results showed that LSTM-AE not only is competitive with respect to compression rates but can greatly reduce reconstruction error. In addition to this, we demonstrated the framework is lightweight for the high level of compression performance.

Further, the network also demonstrated the capability of generalizing a single model to be able to compress many images. The experimental results additionally demonstrated a benefit over the standard compression approach with regards to reconstruction error. This allows for the further reduction in the required computation time to compress images since we can train a model on many test scenes that may be encountered during the data collection process of a hyperspectral imager without requiring retraining the model for each of the test

scenes and changing subjects in the images.

We also demonstrated that this generality can be applied to applying trained models to also compress unseen images. This level of robustness is not possible with a spectral-spatial strategy because of the dependence on the spatial dimension. Therefore, we conclude that the spectral only strategy allows a robust compressor that can perform on par, if not better, than state-of-the-art compression algorithms.

We then performed one of the first analyses on the capability of performing classification on compressed hyperspectral images. If a compressed image can be classified directly, then users can see a reduction in the computational resources required to utilize hyperspectral imagery. We also demonstrated a lightweight framework in which to more efficiently generate features for hyperspectral image classification.

From our experimental results, we demonstrated an ability to classify the images compressed by the LSTM-AE framework. The experiments also demonstrated that increasing the number of features led to an increase in classification performance. They also demonstrated, as expected, that the IBRA-GSS algorithm was significantly better for classification than the compressed images except in the case of a single band.

We also demonstrated that performing compression with the robust modifications was effective, further reducing the computational requirement of utilizing hyperspectral imagery. It was demonstrated experimentally that generalizing a model across multiple images had an increase in classification performance over the standard compression approach. Experiments also showed that compressing an unseen image with a trained classifier did not reduce the classification performance significantly in comparison to the generalized approach, suggesting that our proposed model does not require retraining when an imager is re-tasked.

To better predict the classification performance of the compressed images, we demonstrated a strong correlation between the compression rate and the classification rate as well as a moderately strong correlation between the reconstruction error and classification rate.

This analysis demonstrates that as the number of features saved by the proposed LSTM-AE model increases, we increase the classification performance. With this analysis, we showed that the level of compression is directly competing against the classification rate, suggesting that a user can fine-tune the model to the acceptable levels of compression and classification performance.

6.1 Contributions

To summarize, the main contributions presented in this thesis include the following:

- We present a low-cost Long Short Term Memory Autoencoder model for hyperspectral image compression. We show that this design accomplishes lower reconstruction error without reducing the compressive capabilities. We also demonstrate that our model requires few trainable parameters and few training samples without detriment to its performance.

- We present two modified training variations that expand the generality of the compression model to eliminate the requirement to retrain the compressor for new scenes. We demonstrate that this increase in generality can reduce the reconstruction error compared to the standard training process.

- We analyzed the capabilities of the compressed image being utilized for classification tasks. We demonstrate that the compressed image can effectively be utilized for classification.

- We demonstrate that increasing the generality of the compressor does not have a significant reduction in the classification performance on the compressed image.

Our contributions serve to provide efficient compression without sacrificing either the reconstruction error or the compression rate. Overall, the main reduction in the

computational cost of the approach is the re-usability of a trained model. The computational efficiency was enhanced by the robustness of this approach by reducing the retraining on sufficiently different scenes often found with other compression methods. Our analysis of the ability to perform compressed classification demonstrates the re-usability of the compressed images, further demonstrating the benefits of a spectral-only approach.

6.2 Future Work

As initial future work, we would like to analyze the capabilities of a spatial compressor to further compress the images that the LSTM-AE model acquires. Given that the LSTM-AE model performs comparably to existing state-of-the-art spatial-spectral compression models, it would be interesting to analyze the impacts that a spatial compressor has on the compressed spectral dimension.

The layering of multiple deep neural networks for classification makes the model's behavior unclear and difficult for users to understand why misclassifications have occurred. In the future, we plan on incorporating explainable AI to explore the relationship between the generated features and the data further. Incorporating explainable AI components would work towards improving the trust in the performance of the model with the users that would most benefit from the reduction in the resource requirements.

For additional future work, we also want to expand the interpretability of the outputs. Higher levels of interpretability allow users to more rapidly identify when misclassifications have occurred and tie the features that were used to the predicted output. The incorporation of interpretive processes would further work towards improving the trust in the model's performance.

In real word HSI applications, the observed HSI is degraded by different sources, related to the applied imaging technology, system, environment, etc. With the low error rates in reconstruction in combination with the generality of the model, it may be possible that the

model can be adapted to remove some of this noise. In the future, we plan on modifying the LSTM autoencoder to perform noise reduction on HSI. This would further expand the applicability of the model for users in the field.

REFERENCES CITED

[1] Aamir Naveed Abbasi and Mingyi He. Convolutional neural network with PCA and batch normalization for hyperspectral image classification. In *IEEE International Geoscience and Remote Sensing Symposium*, page 959–962, Jul 2019.

[2] José Manuel Amigo, Hamid Babamoradi, and Saioa Elcoroaristizabal. Hyperspectral image analysis. A tutorial. *Analytica Chimica Acta*, 896:34–51, Oct 2015.

[3] Sigrún Andradóttir. An overview of simulation optimization via random search. *Handbooks in operations research and management science*, 13:617–631, 2006.

[4] Nicolas Audebert, Bertrand Le Saux, and Sébastien Lefevre. Generative Adversarial Networks for Realistic Synthesis of Hyperspectral Samples. In *IEEE International Geoscience and Remote Sensing Symposium*, pages 4359–4362, 2018.

[5] Nicolas Baghdadi and Mehrez Zribi. *Optical Remote Sensing of Land Surface: Techniques and Methods*. Elsevier, September 2016. Google-Books-ID: Ine0CwAAQBAJ.

[6] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, Jan 1989.

[7] Xanthi Bampoula, Georgios Siaterlis, Nikolaos Nikolakis, and Kosmas Alexopoulos. A deep learning model for predictive maintenance in cyber-physical production systems using LSTM autoencoders. *Sensors*, 21(3):972, 2021.

[8] Yubal Barrios, Alfonso Rodríguez, Antonio Sánchez, Arturo Pérez, Sebastián López, Andrés Otero, Eduardo de la Torre, and Roberto Sarmiento. Lossy Hyperspectral Image Compression on a Reconfigurable and Fault-Tolerant FPGA-Based Adaptive Computing Platform. *Electronics*, 9(10):1576, October 2020.

[9] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe. 220 Band AVIRIS Hyperspectral Image Data Set: June 12,1992 Indian Pine Test Site 3. 1992.

[10] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-Wise Training of Deep Networks. In *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

[11] Kavita Bhosle and Vijaya Musande. Evaluation of deep learning CNN model for land use land cover classification and crop identification using hyperspectral remote sensing images. *Journal of the Indian Society of Remote Sensing*, 47(11):1949–1958, Nov 2019.

[12] Larry Biehl. HYDICE image of washington DC mall. `https://engineering.purdue.edu/~biehl/MultiSpec/hyperspectral.html`, 1995. Accessed: 2022-3-27.

[13] Giuseppe Bonifazi, Giuseppe Capobianco, and Silvia Serranti. Hyperspectral imaging and hierarchical PLS-DA applied to asbestos recognition in construction and demolition waste. *Applied Sciences*, 9(21):4587, 2019.

[14] Daniel Báscones, Carlos González, and Daniel Mozos. An Extremely Pipelined FPGA Implementation of a Lossy Hyperspectral Image Compression Algorithm. *IEEE Transactions on Geoscience and Remote Sensing*, 58(10):7435–7447, October 2020.

[15] Minmin Chen, Kilian Weinberger, Fei Sha, and Yoshua Bengio. Marginalized Denoising Auto-encoders for Nonlinear Representations. In *Proceedings of the 31st International Conference on Machine Learning*, page 1476–1484. PMLR, Jun 2014.

[16] Yushi Chen, Hanlu Jiang, Chunyang Li, Xiuping Jia, and Pedram Ghamisi. Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks. *IEEE Transactions on Geoscience and Remote Sensing*, 54(10):6232–6251, Oct 2016.

[17] Yushi Chen, Zhouhan Lin, Xing Zhao, Gang Wang, and Yanfeng Gu. Deep Learning-Based Classification of Hyperspectral Data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(6):2094–2107, Jun 2014.

[18] Mingmin Chi, Rui Feng, and Lorenzo Bruzzone. Classification of hyperspectral remote-sensing data with primal SVM for small-sized training dataset problem. *Advances in Space Research*, 41(11):1793–1799, Jan 2008.

[19] P. Deepa and K. Thilagavathi. Feature extraction of hyperspectral image using principal component analysis and folded-principal component analysis. In *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, pages 656–660, 2015.

[20] Thomas G. Dietterich. Machine Learning for Sequential Data: A Review. In *Structural,Syntactic,and Statistical Pattern Recognition*, Lecture Notes in Computer Science, page 15–30. Springer, 2002.

[21] Qian Du and James E. Fowler. Hyperspectral Image Compression Using JPEG2000 and Principal Component Analysis. *IEEE Geoscience and Remote Sensing Letters*, 4(2):201–205, Apr 2007.

[22] Qian Du and James E. Fowler. Hyperspectral Image Compression Using JPEG2000 and Principal Component Analysis. *IEEE Geoscience and Remote Sensing Letters*, 4(2):201–205, April 2007.

[23] Qian Du and James E. Fowler. Low-Complexity Principal Component Analysis for Hyperspectral Image Compression. *The International Journal of High Performance Computing Applications*, 22(4):438–448, November 2008.

[24] Yaman Dua, Vinod Kumar, and Ravi Shankar Singh. Comprehensive review of hyperspectral image compression algorithms. *Optical Engineering*, 59(9):090902, September 2020.

[25] Yaman Dua, Ravi Shankar Singh, Kshitij Parwani, Smit Lunagariya, and Vinod Kumar. Convolution Neural Network based lossy compression of hyperspectral images. *Signal Processing: Image Communication*, 95:116255, July 2021.

[26] Rui Dusselaar and Manoranjan Paul. Hyperspectral image compression approaches: opportunities,challenges,and future directions: discussion. *JOSA A*, 34(12):2170–2180, Dec 2017.

[27] Josef Maria Eder. *X. The Life of Johann Heinrich Schulze*, page 64–83. Columbia University Press, Mar 1945.

[28] Gamal ElMasry and Da-Wen Sun. *CHAPTER 1 - Principles of Hyperspectral Imaging Technology*, page 3–43. Academic Press, Jan 2010.

[29] Siwei Feng, Yuki Itoh, Mario Parente, and Marco F Duarte. Wavelet-based semantic features for hyperspectral signature discrimination. *arXiv preprint arXiv:1602.03903*, 2016.

[30] Wei Fu, Shutao Li, Leyuan Fang, and Jón Atli Benediktsson. Adaptive Spectral–Spatial Compression of Hyperspectral Image With Sparse Representation. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2):671–682, February 2017.

[31] A Gensler, J Henze, B Sick, and N Raabe. Deep learning for solar power forecasting—an approach using autoencoder and LSTM neural networks, systems, man, and cybernetics (smc). In *2016 IEEE International Conference on*, pages 2858–2865.

[32] Pedram Ghamisi, Naoto Yokoya, Jun Li, Wenzhi Liao, Sicong Liu, Javier Plaza, Behnood Rasti, and Antonio Plaza. Advances in Hyperspectral Image and Signal Processing: A Comprehensive Overview of the State of the Art. *IEEE Geoscience and Remote Sensing Magazine*, 5(4):37–78, Dec 2017.

[33] Elham Kordi Ghasrodashti, Azam Karami, Rob Heylen, and Paul Scheunders. Spatial Resolution Enhancement of Hyperspectral Images Using Spectral Unmixing and Bayesian Sparse Representation. *Remote Sensing*, 9(6):541, Jun 2017.

[34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[35] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. In *Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005*, Lecture Notes in Computer Science, pages 799–804, Berlin,Heidelberg, 2005. Springer.

[36] J Anthony Gualtieri, Samir R Chettri, Robert F Cromp, and LF Johnson. Support vector machine classifiers as applied to AVIRIS data. In *Proc. Eighth JPL Airborne Geoscience Workshop*, pages 8–11, 1999.

[37] Raúl Guerra, María Díaz, Yubal Barrios, Sebastián López, and Roberto Sarmiento. A Hardware-Friendly Algorithm for the On-Board Compression of Hyperspectral Images. In *9th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, page 1–5, Sep 2018.

[38] Driss Haboudane, John R Miller, Elizabeth Pattey, Pablo J Zarco-Tejada, and Ian B Strachan. Hyperspectral vegetation indices and novel algorithms for predicting green LAI of crop canopies: Modeling and validation in the context of precision agriculture. 90:337–352, Apr 2004.

[39] Jürgen Hahn, Christian Debes, Michael Leigsnering, and Abdelhak M. Zoubir. Compressive sensing and adaptive direct sampling in hyperspectral imaging. *Digital Signal Processing*, 26:113–126, Mar 2014.

[40] J. Ham, Yangchi Chen, M.M. Crawford, and J. Ghosh. Investigation of the random forest framework for classification of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):492–501, 2005.

[41] Renlong Hang, Qingshan Liu, Danfeng Hong, and Pedram Ghamisi. Cascaded Recurrent Neural Networks for Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 57(8):5384–5394, 2019.

[42] Jingping He and Isabel Barton. Hyperspectral remote sensing for detecting geotechnical problems at Ray mine. *Engineering Geology*, 292:106261, 2021.

[43] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[44] Liam Hodgkinson and Michael Mahoney. Multiplicative Noise and Heavy Tails in Stochastic Optimization. In *International Conference on Machine Learning*, pages 4262–4274. PMLR, July 2021.

[45] Mehdi Hosseinzadeh. 4 - Robust control applications in biomedical engineering: Control of depth of hypnosis. In *Control Applications for Biomedical Engineering Systems*, pages 89–125. Academic Press, January 2020.

[46] Ryan Hruska, Jessica Mitchell, Matthew Anderson, and Nancy F. Glenn. Radiometric and Geometric Analysis of Hyperspectral Imagery Acquired from an Unmanned Aerial Vehicle. *Remote Sensing*, 4(9):2736–2752, Sep 2012.

[47] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[48] ISO/IEC:15444-1. JPEG 2000 image coding systems. Standard, International Organization for Standardization, London, United Kingdom, 2019.

[49] Nathalie Japkowicz, Stephen José Hanson, and Mark A. Gluck. Nonlinear Autoassociation Is Not Equivalent to PCA. *Neural Computation*, 12(3):531–545, 03 2000.

[50] Jianxin Jia, Yueming Wang, Jinsong Chen, Ran Guo, Rong Shu, and Jianyu Wang. Status and application of advanced airborne hyperspectral imaging technology: A review. 104:103115, Jan 2020.

[51] Jun Jiang, Dengyu Liu, Jinwei Gu, and Sabine Süsstrunk. What is the space of spectral sensitivity functions for digital color cameras? In *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, page 168–179, Jan 2013.

[52] Zhuocheng Jiang, W. David Pan, and Hongda Shen. Universal Golomb–Rice Coding Parameter Estimation Using Deep Belief Networks for Hyperspectral Image Compression. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(10):3830–3840, October 2018.

[53] Worku Jifara, Feng Jiang, Bing Zhang, Huapeng Wang, Jinsong Li, Aleksei Grigorev, and Shaohui Liu. Hyperspectral image compression based on online learning spectral features dictionary. *Multimedia Tools and Applications*, 76(23):25003–25014, December 2017.

[54] Ali Can Karaca and Mehmet Kemal Güllü. Lossless hyperspectral image compression using bimodal conventional recursive least-squares. *Remote Sensing Letters*, 9(1):31–40, Jan 2018.

[55] Teja Kattenborn, Jens Leitloff, Felix Schiefer, and Stefan Hinz. Review on Convolutional Neural Networks (CNN) in vegetation remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing*, 173:24–49, 2021.

[56] Muhammad Jaleed Khan, Hamid Saeed Khan, Adeel Yousaf, Khurram Khurshid, and Asad Abbas. Modern trends in hyperspectral image analysis: A review. *IEEE Access*, 6:14118–14129, 2018.

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

[58] Stephen Kokoska and Daniel Zwillinger. *CRC Standard Probability and Statistics Tables and Formulae,Student Edition*. CRC Press, Mar 2000.

[59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[60] Brajesh Kumar, Onkar Dikshit, Ashwani Gupta, and Manoj Kumar Singh. Feature extraction for hyperspectral image classification: a review. *International Journal of Remote Sensing*, 41(16):6248–6287, 2020.

[61] Junichi Kurihara, Tetsuro Ishida, and Yukihiro Takahashi. *Unmanned Aerial Vehicle (UAV)-Based Hyperspectral Imaging System for Precision Agriculture and Forest Management*, page 25–38. Springer International Publishing, 2020.

[62] H.S. Lee, N.H. Younan, and R.L. King. Hyperspectral image cube compression combining JPEG-2000 and spectral decorrelation. In *IEEE International Geoscience and Remote Sensing Symposium*, volume 6, pages 3317–3319 vol.6, June 2002.

[63] Mengmeng Li, Haofeng Wang, Lifang Yang, You Liang, Zhigang Shang, and Hong Wan. Fast hybrid dimensionality reduction method for classification based on feature selection and grouped feature extraction. *Expert Systems with Applications*, 150:113277, 2020.

[64] Qianming Li, Bohong Zheng, Bing Tu, Jinping Wang, and Chengle Zhou. Ensemble EMD-based spectral-spatial feature extraction for hyperspectral image classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13:5134–5148, 2020.

[65] Shutao Li, Weiwei Song, Leyuan Fang, Yushi Chen, Pedram Ghamisi, and Jon Atli Benediktsson. Deep learning for hyperspectral image classification: An overview. *IEEE Transactions on Geoscience and Remote Sensing*, 57(9):6690–6709, 2019.

[66] Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with relu activation. *Advances in neural information processing systems*, 30, 2017.

[67] Yunsong Li, Jing Hu, Xi Zhao, Weiying Xie, and JiaoJiao Li. Hyperspectral image super-resolution using deep convolutional neural network. *Neurocomputing*, 266:29–41, Nov 2017.

[68] Zhouhan Lin, Yushi Chen, Xing Zhao, and Gang Wang. Spectral-spatial classification of hyperspectral image using autoencoders. In *2013 9th International Conference on Information,Communications Signal Processing*, page 1–5, Dec 2013.

[69] Bing Liu, Xuchu Yu, Anzhu Yu, and Gang Wan. Deep convolutional recurrent neural network with transfer learning for hyperspectral image classification. *Journal of Applied Remote Sensing*, 12(2):026028, 2018.

[70] Bing Liu, Xuchu Yu, Pengqiang Zhang, Anzhu Yu, Qiongying Fu, and Xiangpo Wei. Supervised Deep Feature Extraction for Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 56(4):1909–1921, Apr 2018.

[71] Qingshan Liu, Feng Zhou, Renlong Hang, and Xiaotong Yuan. Bidirectional-Convolutional LSTM Based Spectral-Spatial Feature Learning for Hyperspectral Image Classification. *Remote Sensing*, 9(12):1330, December 2017.

[72] Ting Liu, Zhongren Li, Chunxia Yu, and Yuhua Qin. NIRS feature extraction based on deep auto-encoder neural network. *Infrared Physics & Technology*, 87:124–128, 2017.

[73] Xiaobo Liu, Chaochao Zhang, Zhihua Cai, Jianfeng Yang, Zhilang Zhou, and Xin Gong. Continuous particle swarm optimization-based deep learning architecture search for hyperspectral image classification. *Remote Sensing*, 13(6):1082, 2021.

[74] Yuwei Liu, Hongbin Pu, and Da-Wen Sun. Hyperspectral imaging technique for evaluating food quality and safety during various processes: A review of recent applications. *Trends in Food Science & Technology*, 69:25–35, November 2017.

[75] Yuwei Liu, Hongbin Pu, and Da-Wen Sun. Hyperspectral imaging technique for evaluating food quality and safety during various processes: A review of recent applications. *Trends in Food Science & Technology*, 69:25–35, Nov 2017.

[76] Jose Llamas, Pedro M Lerones, Roberto Medina, Eduardo Zalama, and Jaime Gómez-García-Bermejo. Classification of architectural heritage images using deep learning techniques. *Applied Sciences*, 7(10):992, 2017.

[77] Riley Donovan Logan. Calibration and characterization of a VNIR hyperspectral imager for produce monitoring. Master's thesis, Montana State University-Bozeman, 2020.

[78] Bing Lu, Phuong D. Dao, Jiangui Liu, Yuhong He, and Jiali Shang. Recent Advances of Hyperspectral Imaging Technology and Applications in Agriculture. *Remote Sensing*, 12(16):2659, Jan 2020.

[79] Guolan Lu and Baowei Fei. Medical hyperspectral imaging: a review. *Journal of Biomedical Optics*, 19(1):010901, Jan 2014.

[80] M. Graña, M.A. Veganzons, and B. Ayerdi. Hyperspectral Remote Sensing Scenes. `http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes`, May 2011. Accessed: 2022-3-27.

[81] Marena Manley. Near-infrared spectroscopy and hyperspectral imaging: non-destructive analysis of biological materials. *Chemical Society Reviews*, 43(24):8200–8214, 2014.

[82] D. Manolakis and G. Shaw. Detection algorithms for hyperspectral imaging applications. *IEEE Signal Processing Magazine*, 19(1):29–43, Jan 2002.

[83] Nor Rizuan Mat Noor and Tanya Vladimirova. Parallelised fault-tolerant integer KLT implementation for lossless hyperspectral image compression on board satellites. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013)*, pages 115–122, 2013.

[84] Mbongowo Mbuh. *Use of Hyperspectral Remote Sensing to Estimate Water Quality*. IntechOpen, October 2019.

[85] Shaohui Mei, Ruituo Jiang, Xu Li, and Qian Du. Spatial and Spectral Joint Super-Resolution Using Convolutional Neural Network. *IEEE Transactions on Geoscience and Remote Sensing*, 58(7):4590–4603, Jul 2020.

[86] Shaohui Mei, Xingang Li, Xiao Liu, Huimin Cai, and Qian Du. Hyperspectral image classification using attention-based bidirectional long short-term memory network. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–12, 2021.

[87] Qinxue Meng, Daniel Catchpoole, David Skillicom, and Paul J. Kennedy. Relational autoencoder for feature extraction. In *International Joint Conference on Neural Networks (IJCNN)*, page 364–371, May 2017.

[88] Jarno Mielikainen and Bormin Huang. Lossless Compression of Hyperspectral Images Using Clustered Linear Prediction With Adaptive Prediction Length. *IEEE Geoscience and Remote Sensing Letters*, 9(6):1118–1121, 2012.

[89] Puneet Mishra, Mohd Shahrimie Mohd Asaari, Ana Herrero-Langreo, Santosh Lohumi, Belén Diezma, and Paul Scheunders. Close range hyperspectral imaging of plants: A review. *Biosystems Engineering*, 164:49–67, Dec 2017.

[90] Matthieu Molinier and Jorma Kilpi. Avoiding Overfitting When Applying Spectral-Spatial Deep Learning Methods on Hyperspectral Images with Limited Labels. In *IEEE International Geoscience and Remote Sensing Symposium*, pages 5049–5052, 2019.

[91] Gerald K. Moore. What is a picture worth? A history of remote sensing. *Hydrological Sciences Bulletin*, 24(4):9, 1979.

[92] Giorgio Morales, John W. Sheppard, Riley D. Logan, and Joseph A. Shaw. Hyperspectral Dimensionality Reduction Based on Inter-Band Redundancy Analysis and Greedy Spectral Selection. *Remote Sensing*, 13(18):3649, January 2021.

[93] Giorgio Morales, John W. Sheppard, Bryan Scherrer, and Joseph A. Shaw. Reduced-cost hyperspectral convolutional neural networks. *Journal of Applied Remote Sensing*, 14(3):036519, Sep 2020.

[94] Lichao Mou, Pedram Ghamisi, and Xiao Xiang Zhu. Deep Recurrent Neural Networks for Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(7):3639–3655, Jul 2017.

[95] Lichao Mou, Pedram Ghamisi, and Xiao Xiang Zhu. Deep recurrent neural networks for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(7):3639–3655, 2017.

[96] M. Mozer. A Focused Backpropagation Algorithm for Temporal Pattern Recognition. *Complex Syst.*, 1989.

[97] Paul W. Nugent, Joseph A. Shaw, Prashant Jha, Bryan Scherrer, Andrew Donelick, and Vipan Kumar. Discrimination of herbicide-resistant kochia with hyperspectral imaging. *Journal of Applied Remote Sensing*, 12(1):016037, Mar 2018.

[98] M. E. Paoletti, J. M. Haut, J. Plaza, and A. Plaza. A new deep convolutional neural network for fast hyperspectral image classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 145:120–147, November 2018.

[99] Fernando J Pineda. Generalization of back-propagation to recurrent neural networks. *Physical review letters*, 59(19):2229, 1987.

[100] Aida Mehdipour Pirbazari, Ekanki Sharma, Antorweep Chakravorty, Wilfried Elmenreich, and Chunming Rong. An Ensemble Approach for Multi-Step Ahead Energy Forecasting of Household Communities. *IEEE Access*, 9:36218–36240, 2021.

[101] Shen-En Qian. Hyperspectral Satellites,Evolution,and Development History. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:7032–7056, 2021.

[102] Kala Rajan and Vasuki Murugesan. Hyperspectral image compression based on DWT and TD with ALS method. *International Arab Journal of Information Technology (IAJIT)*, 13(4), 2016.

[103] Behnood Rasti, Danfeng Hong, Renlong Hang, Pedram Ghamisi, Xudong Kang, Jocelyn Chanussot, and Jon Atli Benediktsson. Feature Extraction for Hyperspectral Imagery: The Evolution From Shallow to Deep: Overview and Toolbox. *IEEE Geoscience and Remote Sensing Magazine*, 8(4):60–88, 2020.

[104] Paul Rodriguez, Janet Wiles, and Jeffrey L Elman. A recurrent neural network that learns to count. *Connection Science*, 11(1):5–40, 1999.

[105] Matías Roodschild, Jorge Gotay Sardiñas, and Adrián Will. A new approach for the vanishing gradient problem on sigmoid activation. *Progress in Artificial Intelligence*, 9(4):351–360, 2020.

[106] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct 1986.

[107] M.J. Ryan and J.F. Arnold. The lossless compression of AVIRIS images by vector quantization. *IEEE Transactions on Geoscience and Remote Sensing*, 35(3):546–550, 1997.

[108] Jeyakumar S and Sudha S. Hybrid hyperspectral image compression technique for non-iterative factorized tensor decomposition and principal component analysis: application for NASA's AVIRIS data. *Computational Geosciences*, 23(5):969–979, October 2019.

[109] Achille Salaün, Yohan Petetin, and François Desbouvries. Comparing the Modeling Powers of RNN and HMM. In *18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 1496–1499, 2019.

[110] Lucana Santos, Ana Gómez, and Roberto Sarmiento. Implementation of CCSDS Standards for Lossless Multispectral and Hyperspectral Satellite Image Compression. *IEEE Transactions on Aerospace and Electronic Systems*, 56(2):1120–1138, 2020.

[111] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10(5):1299–1319, 1998.

[112] B.M. Shahshahani and D.A. Landgrebe. The effect of unlabeled samples in reducing the small sample size problem and mitigating the Hughes phenomenon. *IEEE Transactions on Geoscience and Remote Sensing*, 32(5):1087–1095, Sep 1994.

[113] Hoo-Chang Shin, Holger R. Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald M. Summers. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, 35(5):1285–1298, 2016.

[114] Juha Suomalainen, Niels Anders, Shahzad Iqbal, Gerbert Roerink, Jappe Franke, Philip Wenting, Dirk Hünniger, Harm Bartholomeus, Rolf Becker, and Lammert Kooistra. A lightweight hyperspectral mapping system and photogrammetric processing chain for unmanned aerial vehicles. *Remote Sensing*, 6(11):11013–11030, 2014.

[115] Nicolaas Tack, Andy Lambrechts, Philippe Soussan, and Luc Haspeslagh. A compact,high-speed,and low-cost hyperspectral imager. In *Silicon Photonics VII*, volume 8266, pages 126 – 138. International Society for Optics and Photonics, SPIE, 2012.

[116] Hong Hui Tan and King Hann Lim. Vanishing gradient mitigation with deep learning neural network optimization. In *2019 7th international conference on smart computing & communications (ICSCC)*, pages 1–4. IEEE, 2019.

[117] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017.

[118] Prasad S. Thenkabail and John G. Lyon. *Hyperspectral Remote Sensing of Vegetation*. CRC Press, Apr 2016.

[119] Behçet Uğur Töreyın, Ozan Yilmaz, Yakup Murat Mert, and Fethi Türk. Lossless hyperspectral image compression using wavelet transform based spectral decorrelation. In *7th International Conference on Recent Advances in Space Technologies (RAST)*, pages 251–254, 2015.

[120] Md. Palash Uddin, Md. Al Mamun, and Md. Ali Hossain. Effective feature extraction through segmentation-based folded-PCA for hyperspectral image classification. *International Journal of Remote Sensing*, 40(18):7190–7220, 2019.

[121] Diego Valsesia and Enrico Magli. Fast and lightweight rate control for onboard predictive coding of hyperspectral images. *IEEE Geoscience and Remote Sensing Letters*, 14(3):394–398, 2017.

[122] Michel Verleysen and Damien François. The Curse of Dimensionality in Data Mining and Time Series Prediction. In *Computational Intelligence and Bioinspired Systems*, Lecture Notes in Computer Science, page 758–770. Springer, 2005.

[123] Ishan Verma, Rahul Ahuja, Hardik Meisheri, and Lipika Dey. Air pollutant severity prediction using bi-directional LSTM network. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 651–654, 2018.

[124] Leonardo Villalobos-Arias and Christian Quesada-López. Comparative study of random search hyper-parameter tuning for software effort estimation. In *Proceedings of the 17th International Conference on Predictive Models and Data Analytics in Software Engineering*, pages 21–29, 2021.

[125] Wangyang Wei, Honghai Wu, and Huadong Ma. An autoencoder and LSTM-based traffic flow prediction method. *Sensors*, 19(13), 2019.

[126] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[127] Ronald J Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501, 1990.

[128] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[129] Ronald J Williams and David Zipser. Experimental analysis of the real-time recurrent learning algorithm. *Connection science*, 1(1):87–111, 1989.

[130] Panrong Xiao. *Image compression by wavelet transform*. East Tennessee State University, 2001.

[131] Yonghao Xu, Bo Du, Liangpei Zhang, and Fan Zhang. A Band Grouping Based LSTM Algorithm for Hyperspectral Image Classification. In *Computer Vision*, Communications in Computer and Information Science, pages 421–432, Singapore, 2017. Springer.

[132] Ozal Yildirim. A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification. *Computers in Biology and Medicine*, 96:189–202, May 2018.

[133] Junru Yin, Changsheng Qi, Qiqiang Chen, and Jiantao Qu. Spatial-spectral network for hyperspectral image classification: A 3-d CNN and Bi-LSTM framework. *Remote Sensing*, 13(12):2353, 2021.

[134] Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv:1212.5701 [cs]*, Dec 2012. arXiv: 1212.5701.

[135] Hongyan Zhang, Wei He, Liangpei Zhang, Huanfeng Shen, and Qiangqiang Yuan. Hyperspectral Image Restoration Using Low-Rank Matrix Recovery. *IEEE Transactions on Geoscience and Remote Sensing*, 52(8):4729–4743, Aug 2014.

[136] Yue Zhang, Chuan Qin, Anurag K. Srivastava, Chenrui Jin, and Ratnesh K. Sharma. Data-driven day-ahead PV estimation using autoencoder-LSTM and persistence model. *IEEE Transactions on Industry Applications*, 56(6):7185–7192, 2020.

[137] Wenzhi Zhao and Shihong Du. Spectral–Spatial Feature Extraction for Hyperspectral Image Classification: A Dimension Reduction and Deep Learning Approach. *IEEE Transactions on Geoscience and Remote Sensing*, 54(8):4544–4554, Aug 2016.

[138] Wenzhi Zhao and Shihong Du. Spectral–Spatial Feature Extraction for Hyperspectral Image Classification: A Dimension Reduction and Deep Learning Approach. *IEEE Transactions on Geoscience and Remote Sensing*, 54(8):4544–4554, 2016.

[139] Feng Zhou, Renlong Hang, Qingshan Liu, and Xiaotong Yuan. Hyperspectral image classification using spectral-spatial LSTMs. *Neurocomputing*, 328:39–47, February 2019.

[140] Peicheng Zhou, Junwei Han, Gong Cheng, and Baochang Zhang. Learning Compact and Discriminative Stacked Autoencoder for Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 57(7):4823–4833, July 2019.

[141] Nadia Zikiou, Mourad Lahdir, and David Helbert. Support vector regression-based 3D-wavelet texture learning for hyperspectral image compression. *The Visual Computer*, 36(7):1473–1490, July 2020.

[142] R. J. Zomer, A. Trabucco, and S. L. Ustin. Building spectral libraries for wetlands land cover classification and hyperspectral remote sensing. *Journal of Environmental Management*, 90(7):2170–2177, May 2009.