APPROVAL SHEET

Title of Dissertation: A Compressive Sensing Approach to Hyperspectral Image Classification

Name of Candidate: Charles Della Porta

Doctorate of Philosophy, 2019

Dissertation and Abstract Approved: Chein-I Chang

Professor Chein-I Chang
Remote Sensing Signal and Image Processing Lab
Department of Electrical Engineering, UMBC

Date Approved: July 4, 2019

ABSTRACT

Title of Document: **A COMPRESSIVE SENSING APPROACH TO HYPERSPECTRAL IMAGE CLASSIFICATION**

Charles Della Porta, Doctorate of Philosophy

Conferred August 2019

Directed By: Dr. Chein-I Chang

Remote Sensing Signal and Image Processing Lab

Department of Electrical Engineering

University of Maryland, Baltimore County

Hyperspectral imaging (HSI) technology has found success in a variety of applications; however, its use is often still limited due to size, weight and power (SWaP) constraints. In this dissertation, compressive sensing (CS) is proposed as an enabling technology to reduce the high spectral band count, through the creation of compressively-sensed bands (CSBs). A CS model based on the universality of random sensing is proposed for the analysis of hyperspectral classification in the compressed domain. Specifically, the utility of the support vector machine (SVM) in the compressed domain is evaluated through both mathematical analysis and empirical experimentation. This work shows that is indeed possible to achieve full band classification performance in the compressed domain. The experiments also demonstrate that the minimum number of CSBs is scene dependent, requiring additional algorithms to provide a full solution. Two supervised algorithms based on a feature selection framework are proposed for

estimating the minimum lower bound on the required number of CSBs. The first algorithm is based on feature filtering techniques and the second algorithm is based on classifier wrapping. Finally, an unsupervised algorithm is presented, based on progressive band processing, that is able to adaptively determine the required number of CSBs *in-situ*. The contributions of this dissertation provide the fundamental foundation for compressed classification of hyperspectral images and identify several new opportunities for future research.

A COMPRESSIVE SENSING APPROACH TO HYPERSPECTRAL IMAGE CLASSIFICATION

By

Charles Della Porta

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, Baltimore County, in partial fulfillment
of the requirements for the degree of
Doctorate of Philosophy
2019

Dedication

This dissertation is dedicated to my parents Richard and Laura, my wife, Tatiana, and my beautiful children Isabella and Eric. Your love and support is what has made this work a possibility.

Acknowledgements

This work would not have been possible without the mentorship of my advisor Dr. Chein-I Chang, department of electrical engineering at The University of Maryland Baltimore County. His advice and insights have contributed greatly to the direction of my research. I would also like to acknowledge the support of my peers Adam Bekit and Bernard Lampe. In addition, I am grateful for the feedback and insights provided by the dissertation committee members Dr. Kalpakis, Dr. Zhu, Dr. Bradley, and Dr. Safavi.

Table of Contents

Nomenclature:

- Hyperspectral image (3D): $\boldsymbol{R} \in \Re^{N_x \times N_y \times L}$
- Hyperspectral image (2D): $\boldsymbol{R} \in \Re^{N_R \times L}$
- Number of pixels in the x dimension: N_x
- Number of pixels in the y dimension: N_y
- Total number of pixels in the image: $N_R = N_x * N_y$
- Number of spectral bands: L
- Hyperspectral band image: $\boldsymbol{R}_l \in \Re^{N_x \times N_y}$
- Hyperspectral pixel vector: $\mathbf{r} \in \Re^{L \times 1}$
- Hyperspectral pixel element: $r \in \Re^{1 \times 1}$
- Sampling matrix: $\boldsymbol{\Phi} \in \Re^{m \times L}$
- Representation matrix: $\boldsymbol{\Psi} \in \Re^{L \times L}$
- Compressed measurement: $\mathbf{y} \in \Re^{m \times 1}$
- Compressively-sensed band index $m \in \{1,2,3, \ldots L\}$
- Recovered pixel vector $\tilde{\mathbf{r}}_s \in \Re^{L \times 1}$
- Order of sparsity k
- Restricted isometry constant δ_k
- Class index $p \in \{0,1,2, \ldots, P\}$
- Number of samples in class p N_p
- Pixel vectors in class p $R_p \in \Re^{L \times N_p}$

- Class mean pixel vector $\boldsymbol{\mu}_p \in \Re^{L\times 1}$
- Classification map: $\boldsymbol{C} \in \Re^{N_x \times N_y}$
- Individual classification accuracy $P_{CA}(p)$
- Individual classification precision $P_{CP}(p)$
- Overall classification accuracy P_{OA}
- Average classification accuracy P_{AA}
- Average classification precision P_{AP}
- Overall classification accuracy efficacy $P_{OA_{Eff}}$
- Average classification accuracy efficacy $P_{AA_{Eff}}$
- Average classification precision efficacy $P_{AP_{Eff}}$
- Optimal number of compressed bands m_{opt}
- Individual progression accuracy P_{PA}
- Individual progression precision P_{PP}
- Overall progression accuracy P_{OPA}
- Average progression accuracy P_{APA}
- Average progression precision P_{APA}
- Individual Tanimoto coefficient P_{TC}
- Overall Tanimoto coefficient P_{OTC}
- Average Tanimoto coefficient P_{ATC}

In general, lower case bold face characters are used to designate vectors and upper-case bold face characters are used to designate matrices.

Chapter 1: Introduction and Fundamental Concepts

Hyperspectral sensing (HS) technologies have found success in a variety of applications ranging from agricultural land cover and land use mapping, food inspection, environmental monitoring to medical imaging, law enforcement and military reconnaissance and surveillance. Although hyperspectral technology has continued to improve over the years, its applications are still limited due to size, weight and power (SWaP) constraints. One of the challenging requirements is the need to sample a large number of very fine spectral bands which require very fast and expensive analog-to-digital converters (ADCs), high capacity on-board storage and optimized computational hardware and software to allow for real-time processing. Such requirements limit the utility of many applications and preclude the use of hyperspectral technologies in many applications.

Compressive sensing (CS) has recently developed as a promising approach in hyperspectral data analysis. CS is based on the concepts of signal sparsity and incoherence, and allows for data to be acquired at Sub-Nyquist rates, with little or no loss of information. By taking advantage of CS, the burdens imposed by high sampling rates (band rates in terms of band acquisition) can be significantly reduced, i.e., by leveraging the fact that the spectral bands are redundant. Such a compressive sampling approach can be applied to each hyperspectral pixel vector by reducing a very large number of spectral bands to a small number of compressively sensed bands (CSBs) that need to be collected. However, to fully exploit this benefit, hyperspectral processing should be able to be applied directly in the compressively sensed band domain (CSBD) without an appreciable loss of performance.

This dissertation presents a compressive sensing (CS) based approach to the classification of hyperspectral images (HSI). The document consists of six chapters. Chapter 1 discusses previous work, introduces hyperspectral technologies, defines the task of classification, and describes the datasets that are used for the included experiments. Chapter 2 describes a mathematical model for the compressive sensing of HS data and provides the framework for subsequent analysis. Chapter 3 introduces the proposed compressive sensing classification approach, along with a mathematical error analysis and experimental results. Chapter 4 introduces two supervised approaches for estimating the minimum number of CSBs required for optimal performance. Chapter 5 introduces a progressive band extension to compressive sensing classification, along with the supporting experimental results. Finally, Chapter 6 provides summary discussion. For consistency of notation, all vectors and matrices are bold face with lower case and upper case used to differentiate between them, respectively.

Related Work and Dissertation Contributions

Sparsity is an enabling concept that has been leveraged in many different ways to support various applications in hyperspectral data exploitation. Most commonly, the concept of sparsity is applied as an additional constraint to expand existing hyperspectral imaging algorithms such as linear spectral unmixing (Jordache, Bioucas-Dias and Plaza 2011) (Bioucas-Dias and Figueiredo 2010), band selection (Du, Bioucas-Dias and Plaza 2012), feature extraction (Zhong and Wang 2008) (Tuia, Flamary and Courty 2015), and classification (Chen, Nasrabadi and Tran 2011) (Chen, Nasrabadi and Tran 2013). In such approaches, a desired sparsity level is typically imposed on the HS data as a constraint allowing for more unique mathematical

solutions to existing HS algorithms. In contrast, compressive sensing (Candes and Wakin 2008) provides a different approach in which, rather than incorporating sparsity as a constraint, the image acquisition process is fundamentally changed to account for signal sparsity and to maximize incoherence (increase information) between sparse measurements. The fundamentals of compressed sensing have been researched in great detail in the literature and a more detailed discussion on the essential concepts is provided in Chapters 2, where mathematical models are introduced.

Early works in CS theory have addressed the concept compressed classification and evaluated the expected performance limits for various classifiers. One of the earliest works was the smashed filter (Davenport, et al. 2007) which was an extension of the matched filter to matched embeddings in the compressed domain. The use of more complicated classifiers such as the support vector machine (SVM) in the compressed domain were also investigated (Calderbank, Jafarpour and Schapire 2009); however, the work was limited to the linear kernel and focused on fairly trivial synthetic classification problems. The work by Hahm (Hahm, Rosenkranz and Zoubir 2014) introduced the concept of compressed classification to hyperspectral data; however, the sampling strategy was based on an adaptively optimized sensing paradigm which was not easily implementable in hardware. Additionally, a method for choosing the appropriate number of compressed bands and the effects of individual scene complexity on compressed performance were not addressed.

In this work, several major contributions are made to advance the state-of-the-art in compressed classification of hyperspectral images. First of all, a universal compressive sensing model is identified that can be implemented without creation of

new, complex sensing hardware. While the concept of universal sampling is not new to CS, the implications of using such a model in the context of hyperspectral classification are in fact new and critical for achieving realizable hardware designs. Second, error expressions between the full band domain and CSBD are derived for the linear kernel and radial basis function-based kernel-based SVM. These error expressions show, analytically, that full band classification performance is indeed achievable in CSBD for sufficient band sampling conditions. Third, feature selection approaches are developed to estimate a minimum bound on the required number of CSBs. Fourth, a progressive band classification approach is introduced that allows for an *in-situ* determination of how many compressed bands need to be acquired. Finally, a series of experiments are performed to demonstrate hyperspectral classification in CSBD. The experiments include multiple images of various scenes from different sensors and illustrate the effects of scene complexity.

Hyperspectral Imaging (HSI)

A hyperspectral image is a three dimensional data cube composed of two spatial dimensions and a wavelength (spectral band) dimension. Contrary to multispectral images which are typically composed of just three bands (red, green, and blue), an HS image is composed of many finely spaced, contiguous, spectral bands. HS data provided spatial, spectral, and radiometric (intensity) information about an imaged scene and have found success in numerous applications. A simple illustration of a Hyperspectral data cube is shown in Figure 1, where several of the Hyperspectral channels has been plotted in an exploded view.

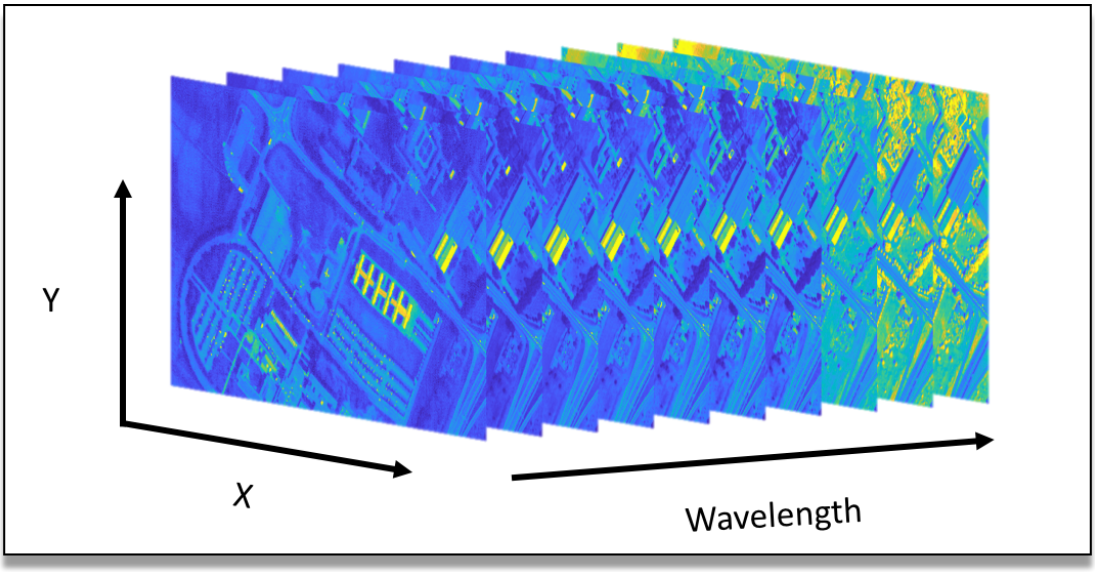

Figure 1: Hyperspectral image cube.

Given the high dimensionality of HS data, the 3D images are often referred to and processed along different dimensions. Within this document, several conventions have been adopted and are maintained for consistency. An image cube, $\boldsymbol{R} \in \Re^{N_x \times N_y \times L}$, is defined as the full hyperspectral image (all pixels and bands), consisting of $N_x \times N_y$ spatial pixels and L spectral bands. A band image, $\boldsymbol{R}_l \in \Re^{N_x \times N_y}$, is defined to include all pixels for a single band, l. A pixel vector, $\mathbf{r} \in \Re^{L \times 1}$, is defined to include all bands for a single pixel. Finally, a pixel element, $r \in \Re^{1 \times 1}$, is defined to be a single pixel from a single band. All illustration for each of the aforementioned definitions is provided in Figure 2.

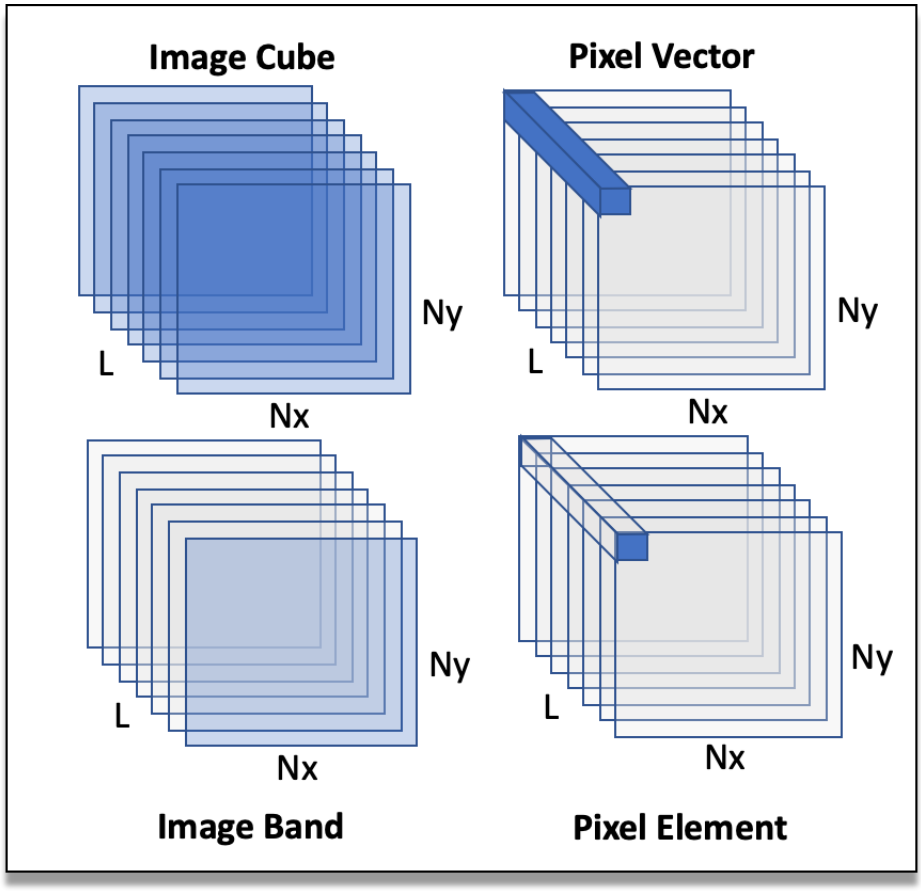

Figure 2: Hyperspectral image dimensions.

HSI sensors can operate in the visible (VIS), near-infrared (NIR), short-wave infrared (SWIR), medium-wave infrared (MWIR), and long-wave infrared (LWIR) spectral wavelengths. Depending on the spectral bands acquired, HSI sensors are tuned to either reflected or emitted electromagnetic radiation. The images considered in this work are limited to the VIS and NIR portion of the spectrum which is dominated by reflections from solar radiation. Figure 3 shows an example of several individual pixel vectors plotted as a function of their spectral bands, in the unitless reflectance ratio. This simple example illustrates the intrinsic discriminability that is available within a single pixel.

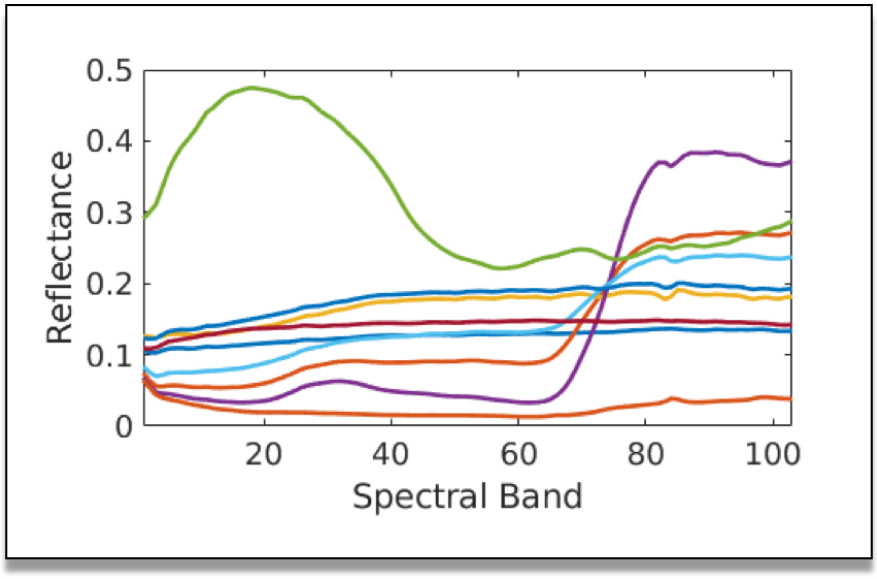

Figure 3: Hyperspectral pixel vector.

Multi-Class Classification of Hyperspectral Images

For an HSI, $\boldsymbol{R} \in \Re^{N_x \times N_y \times L}$, where each pixel belongs to one of P possible classes, a classifier is a function, $f(\boldsymbol{r})$, that assigns a label, $p \in \{1,2,\ldots,P\}$, to each pixel, $\boldsymbol{r}$, resulting in a classification map, $\boldsymbol{C} \in \Re^{N_x \times N_y}$. An example three-class problem, with two features, is illustrated in Figure 4. The green 'x's, blue '+'s, and orange 'o's represent samples from three different classes plotted as a function of feature x_1 and feature x_2. The boundaries defined by the classifier, $f(\boldsymbol{r})$, are illustrated as shaded regions with matching colors for each respective class.

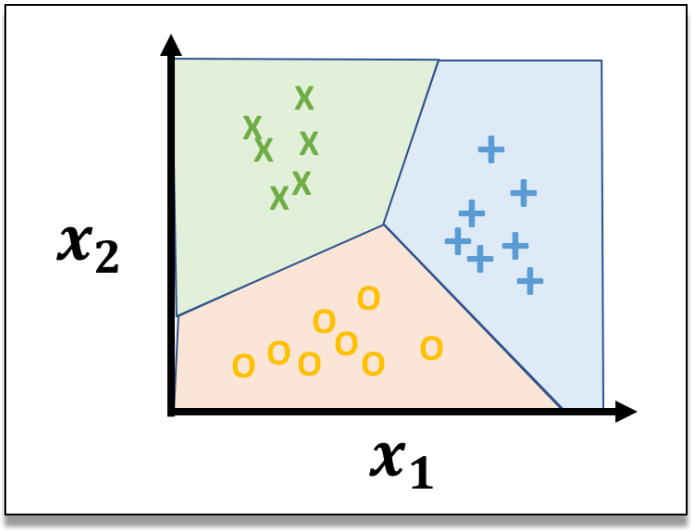

Figure 4: An example of a three-class dataset with two features.

The spectral diversity intrinsically available in hyperspectral data can be leveraged directly as classification features to perform pixel-wise classification. Additionally, spatial correlation from neighboring pixels can be included to further improve classification accuracy. Hyperspectral classification continues to be a very active area of research utilizing state-of-the-art classifiers, such as deep neural network architectures (Chen, Lin, et al. 2014), (Li, Zhang and Zhang 2014), kernel-based discriminators (Camps-Valls and Bruzzone 2005), (Camps-Valls, Gomez-Chova, et al. 2006), statistical learning theory (Camps-Valls, Tuia, et al. 2014), and complex feature strategies based on both spatial and spectral features (Fauvel, Benediktsson, et al. 2008), (Fauvel, Tarabalka, et al. 2013). For this work, the spectral-spatial classifier proposed in (Kang, Li and Benediktsson 2014) is adopted for all experiments. The algorithm is described in detail in Chapter 3.

There are several classification performance metrics that are commonly reported in hyperspectral literature. Within this work, the class accuracy, class

precision and overall accuracy are reported for all experiments. Each of these metrics can be most easily understood by first referring to a confusion matrix, as shown in Figure 5. A confusion matrix is constructed by doing a pair-wise comparison of the ground truth and predicted label for every combination of classes. For the confusion matrix in Figure 5, the columns represent the true values, the rows represent predicted values, and the elements represent the number of pixels that fall into that particular case. A confusion matrix for a perfect classifier would be an identity matrix, since the predicted values (rows) would always match the true values (columns).

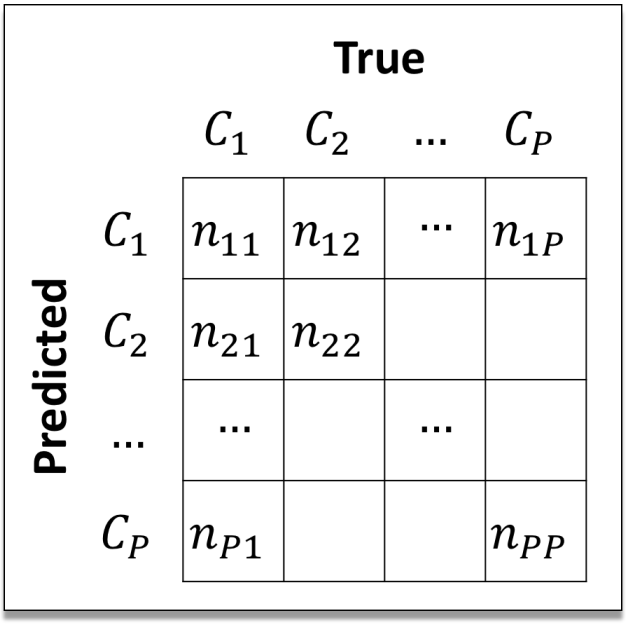

Figure 5: A classification confusion matrix.

The class accuracy for class p, $P_{CA}(p)$, can be computed from the confusion matrix by dividing the total number of correct classifications for a given class, $C_{p,p}$, by the total pixel count from that particular class, as shown in Equation (1). The class precision for class p, $P_{CP}(p)$, can be computed from the confusion matrix by dividing

the total number of correct classifications for a given class, $C_{p,p}$, by the total count of predictions made for that particular class, as shown in Equation (2). Finally, the overall accuracy, P_{OA}, can be computed from the confusion matrix by summing the diagonal elements and dividing by the total number of pixels, as shown in Equation (3).

$$P_{CA}(p) = \frac{n_{pp}}{\sum_{j=1}^{P} n_{jp}} \tag{1}$$

$$P_{CP}(p) = \frac{n_{pp}}{\sum_{j=1}^{P} n_{pj}} \tag{2}$$

$$P_{OA} = \frac{\sum_{j=1}^{P} n_{jj}}{N} \tag{3}$$

In addition to referring to P_{CA} and P_{CP} for each class, it is also convenient to the consider the average class accuracy, $P_{AA} = \frac{1}{P}\sum_{1}^{P} P_{CA}(p)$, and the average class precision, $P_{AP} = \frac{1}{P}\sum_{1}^{P} P_{CP}(p)$. These metrics are also reported in the experiment sections.

Hyperspectral Datasets for Classification Evaluation

The Airborne Visible / Infrared Imaging Spectrometer (AVIRIS) (Green, et al. 1998) is an aircraft based hyperspectral sensor with 224 contiguous bands over the 400-2500 nm spectrum. AVIRIS images are well-known in the hyperspectral literature and prolific in algorithm benchmarks. Two AVIRIS images are examined throughout this work: Indian Pines and Salinas. The Indian Pines (Baumgardner, Biehl and Landgrebe

2015) image is a well-known benchmark dataset for Hyperspectral classification. The image has 20 m spatial resolution and consists of 200 spectral channels, after removal of the water absorption bands. The image was collected over Purdue's Indiana Indian Pines test site and is composed of a mixture of agriculture and forestry. The provided ground truth consists of 17 classes, including the background, and is shown in Figure 6.

The Salinas image (Universidad del Pais Vasco Grupo de Inteligencia Computacional n.d.) is another well-known benchmark dataset for Hyperspectral classification. The image has 3.7 m spatial resolution and consists of 200 spectral channels, after removal of the water absorption bands. The image was collected over Salinas Valley, California and is composed of a mixture agriculture. The ground truth consists of 17 classes, including the background, and is shown in Figure 7.

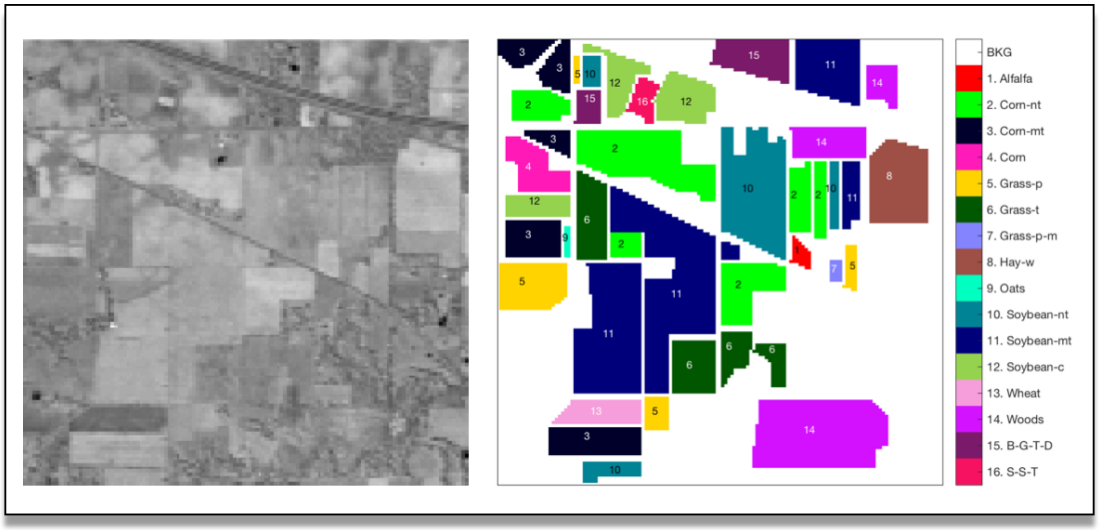

Figure 6: AVIRIS Indian Pines image and ground truth.

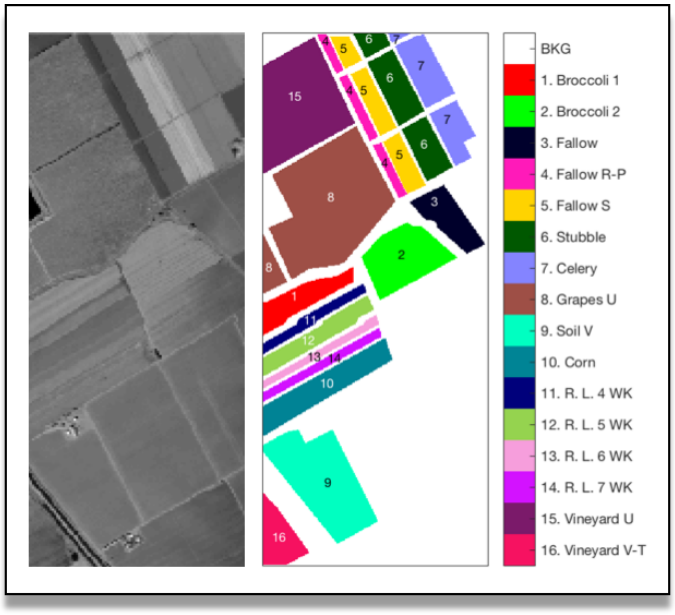

Figure 7: AVIRIS Salinas image and ground truth.

The Reflective Optics System Imaging Spectrometer (ROSIS) (Holzwarth, et al. 2003) is an aircraft based hyperspectral sensor with 115 contiguous bands over the 430-860 nm spectrum. The sensor was originally designed for resolving fine spectral structures and has produced several images that have been widely used within the literature. Two ROSIS images are examined within this work: Pavia University and Pavia Centre.

The Pavia University and Pavia Centre image scenes (Universidad del Pais Vasco Grupo de Inteligencia Computacional n.d.) are well-known benchmark datasets for Hyperspectral classification. The images have 1.3 m spatial resolution and consists of 102 spectral channels, after removal of the water absorption bands. The images were collected over Pavia, northern Italy and are composed of urban scenes. The provided ground truths, for each image, consists of 9 different classes and are shown in Figure 8 and Figure 9.

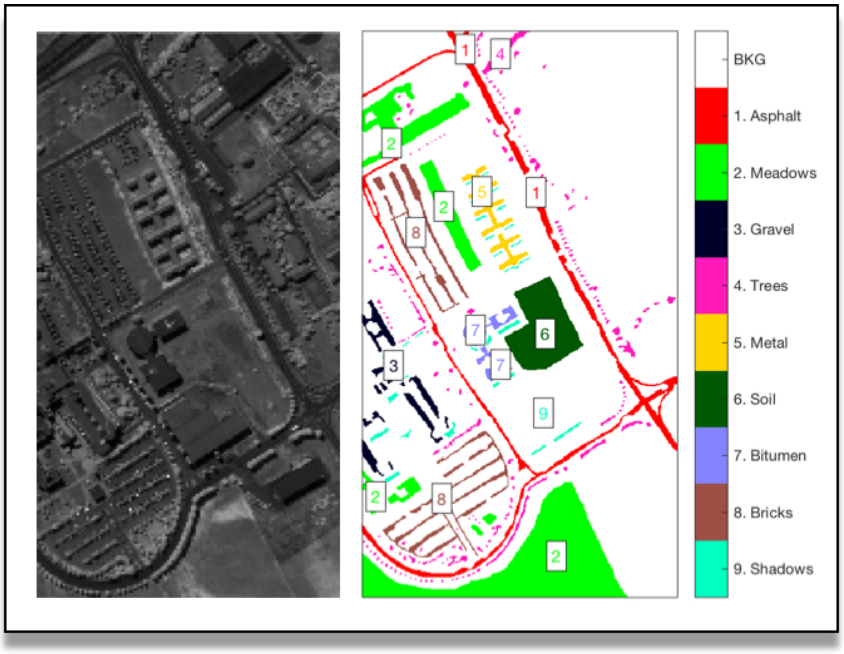

Figure 8: ROSIS Pavia University image and ground truth.

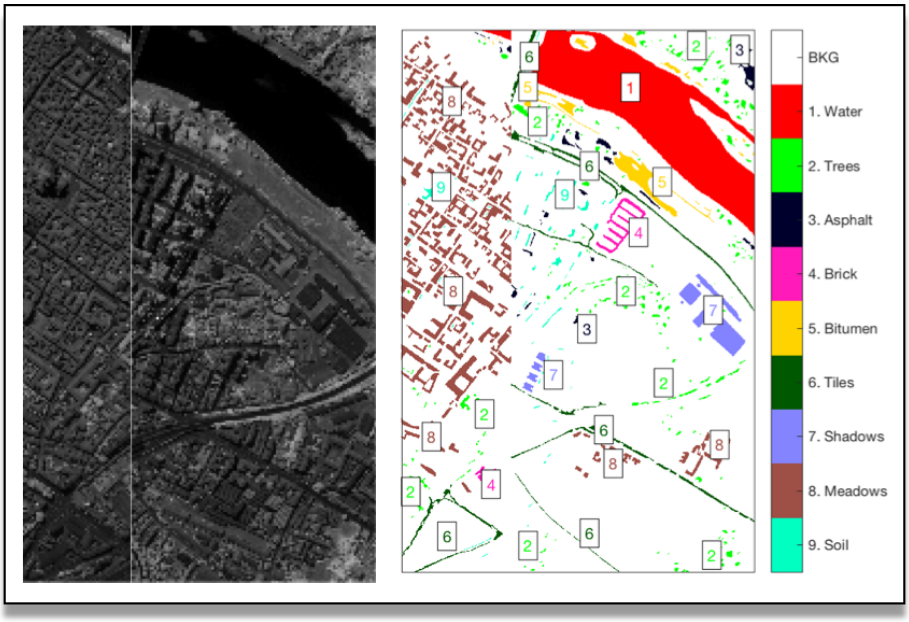

Figure 9: ROSIS Pavia Centre image and ground truth.

Conclusion

This chapter introduced the concept of performing classification within a compressive sensing framework, as well as provided some motivation for the application of such a technique in the field of hyperspectral image processing. A brief summary of the related work was provided and contrasted with the major contributions of the presented work. Hyperspectral images were introduced and common notation was established to provided consistency throughout the document. The task of classification of hyperspectral data was also introduced and a brief summary of the current-state-of-the art was provided. A set of performance metrics were defined for hyperspectral classification that are used in subsequent chapters to evaluate the merit of the proposed approach. Finally, four real hyperspectral images, from two different sensors, were introduced and described. These images are evaluated and analyzed in all of the included experimental sections.

Chapter 2: A Universal Sampling Model for Compressive Sensing

Introduction

In this chapter, a topic in mathematics referred to as compressive sensing (CS) is introduced. The notion of a sub-sampled system is described for both the general case and in the context of a hyperspectral imaging sensor. An alternative version of the hyperspectral system model is proposed in which data analysis is performed directly in the compressed domain. The potential benefits and use cases for such a system are outlined and discussed.

The fundamental concepts of sparsity and incoherence are introduced and applied to hyperspectral pixel vectors. A mathematical model for compressively-sensed hyperspectral images is derived from the common CS model and is used to define a compressive encoding and decoding process. The concept of *universality* is introduced and used to define a new universal sampling model that guarantees the ability to achieve optimal performance in the compressed domain. The chapter concludes with a brief experimental section that illustrates the presented concepts.

Sub-Sampled Systems

A Compressive Sensing System

In a typical digital signal processing system, signals are sampled according to the Shannon-Nyquist sampling theorem (Shannon 1949), which states that a uniformly spaced set of samples must be collected, at a sufficient rate, from a bandlimited signal

to ensure perfect reconstruction. Furthermore, the sampling rate is directly bounded by the signal bandwidth. In a CS system, signals are sub-sampled well below the Shannon-Nyquist rate, without loss of information, by leveraging some intrinsic sparsity within the data. The fully-sampled signals can later be recovered from the compressively-sampled signals and processed without modification of the original analysis algorithms.

A CS system offers potential reductions in the SWaP requirements of the sensing platform, as well as potential reductions in bandwidth requirements for data that must be transmitted between systems. A simple motivating example is that of a space-borne system where minimizing the required onboard data storage and communication bandwidth are critical. Compressively sensed data can be captured and transmitted back to a ground station, where the fully-sampled data can recovered and analyzed. CS enables flexibility within a larger communication system, by shifting high data rate and high computation tasks to more SWaP-tolerant components of the overall system.

A typical CS system consists of four general stages: sensing, encoding, decoding and analysis. A block diagram illustrating these four general stages is shown in Figure 10. The sensing stage includes the acquisition of the analog signal, before it is digitally sampled. In the encoding stage, the analog signal is compressively sampled resulting in a digital signal acquired at sub-Nyquist rate, i.e. a compressed digital signal. The compressed digital signal is then typically moved to some type of storage for processing at a later time, or is transmitted to a different system for offboard processing. During the decoding stage, a sparse recovery algorithm is used to extract the fully-

sampled digital signal from the compressively-sampled digital signal. Finally, traditionally analysis algorithms can be applied to the recovered fully-sampled signal.

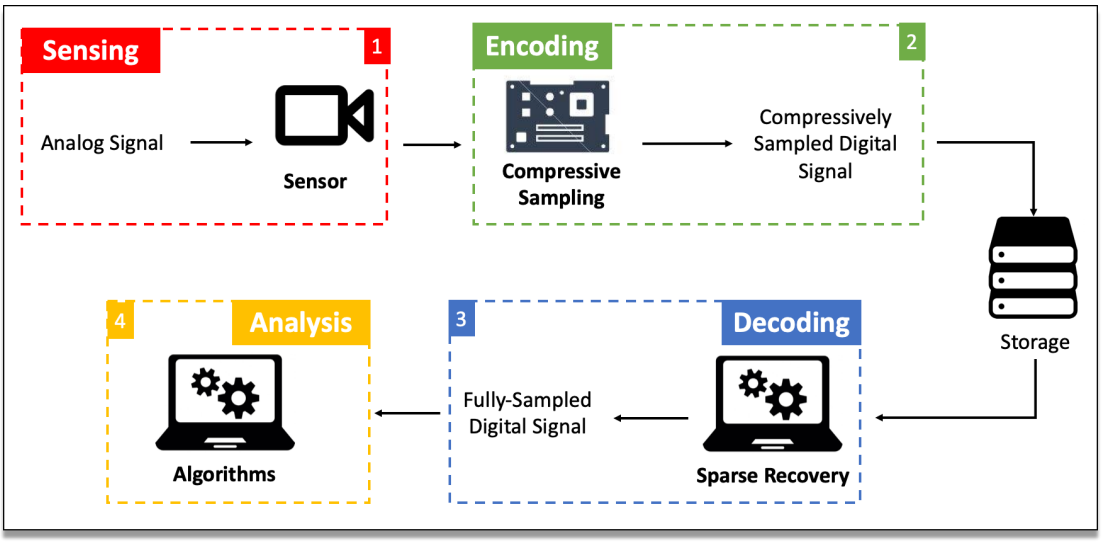

Figure 10: Common CS system block diagram.

A Compressive Hyperspectral Imaging System

Imaging systems where one of the earliest applications of CS techniques. The single pixel camera (Duarte, et al. 2008) demonstrated that it was possible to form a full 2D image through a series of compressively-sampled single-pixel measurements. In this approach, the spatial data of the image are sub-sampled into a single pixel that consists of a mixture of the individual pixels. A number of these single-pixel measurements are aggregated together to form a single compressed measurement. This imaging system concept is a good example of the common CS model shown in Figure 10. The sensing stage consists of traditional image photonic elements that capture the light sources. The compressive sampling stage consists of a digital micromirror that mixes the light sources and an analog-to-digital converter to digitally sample the mixtures. The

compressed measurement is then stored, where it can later undergo a sparse recovery and then be analyzed using standard image processing algorithms.

A similar hyperspectral imaging system concept can be imagined, where rather than compressing the spatial information, the large number of spectral bands are compressively-sampled. In this approach, the spatial information is fully sampled but the spectral channels are sub-sampled. An illustration of a CS HSI system following the common CS model is shown in Figure 11. In the sensing stage, the full hyperspectral image cube is acquired before it is digitally sampled. The image bands are then compressively-sampled, resulting in an image cube with full spatial samples but a reduced number of spectral bands. Following the same logic as before, this compressed measurement can then be stored for sparse recovery and processing.

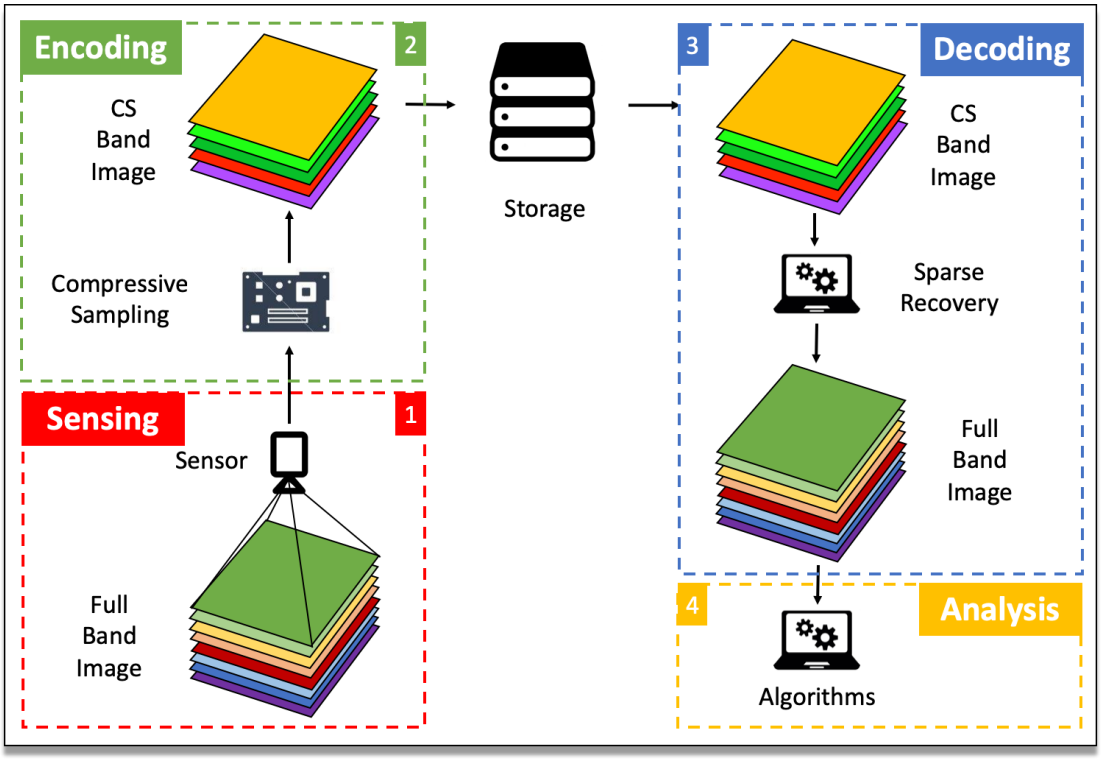

Figure 11: An HSI concept for the common CS system model.

This CS HSI concept will serve as the starting point of discussion; however, it must be modified to account for compressed analysis and to better accommodate on-board / real-time analysis of compressed HS data. A proposed updated system concept based on compressed analysis is shown in Figure 12. In this approach, the decoding process has been completely removed, and the analysis algorithms are applied directly in the compressed domain. One of the benefits of this approach, is that analysis can be performed on-board enabling real-time decisions to be made based on *in-situ* analysis. Some example use cases, for an unmanned aerial vehicle (UAV), might include decisions to: switch to an alternate route, re-survey an area to reduce uncertainty, or to change missions entirely. Similarly for a space-borne platform, *in-situ* analysis may include decisions to: enable or disable additional sensors, determine which data should be saved, or prioritize which data should be transmitted back to the base station for immediate review. This model also maintains the ability to store the compressed data and re-analyze at a later time, after the data have been sparsely recovered.

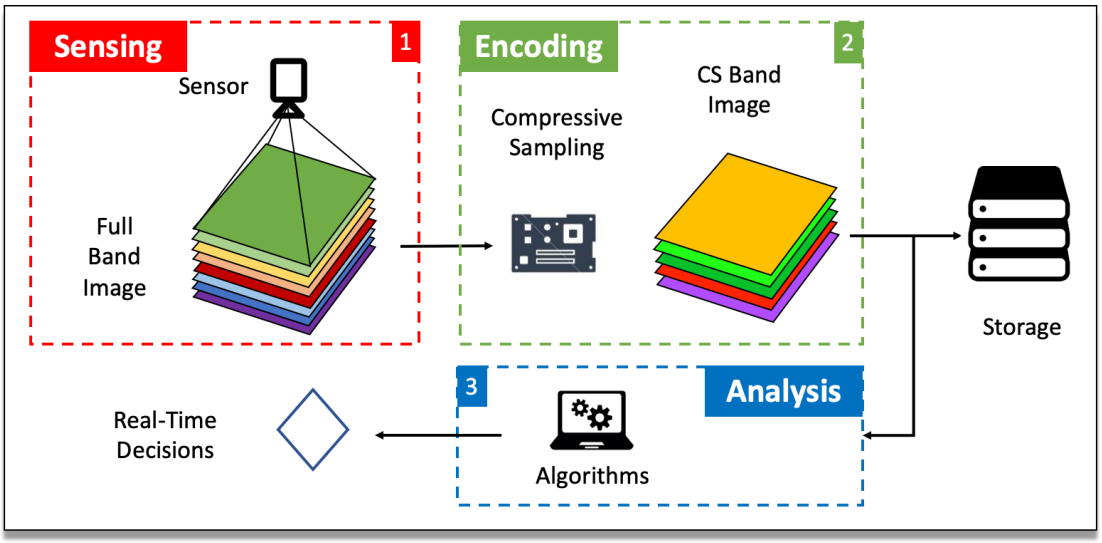

Figure 12: A Universal HSI system model.

Realization of such a system will, of course, require the development of new analysis algorithms that are capable of operating on the compressed data. The development of such techniques are the central theme of this thesis. More specifically, the task of performing hyperspectral image classification in the compressed domain is addressed in the remaining chapters.

A General Mathematical Model for Compressive Sensing

A large body of work exists in the mathematical and, more recently, engineering communities describing various approaches to implement a compressive sensing system. The focus of this section is specifically on the encoding and decoding stages, which include detailed descriptions of the compressive sampling and sparse recovery processes. The analysis stage is the central theme of this thesis and is covered in great detail in later chapters. A detailed discussion on the sensing stage and related hardware modifications is beyond the scope of this work and left for future discussion.

A complete mathematical model for the general compressive hyperspectral imaging system, illustrated in Figure 11, is provided in this section. The adopted approach assumes that the spatial information is fully sampled but the spectral channels are sub-sampled. To make this distinction clear, the sparse measurements will be referred to as compressively-sensed bands (CSBs) to eliminate any potential ambiguities between spatial and spectral samples.

Compressive Sampling

A common mathematical model for the compressive sampling process is shown in equation (4), where $\boldsymbol{y} \in \mathbb{R}^{\mathrm{m}\times 1}$ is the compressed measurement vector, $\boldsymbol{\Phi} \in \mathbb{R}^{\mathrm{m}\times \mathrm{L}}$ is the sampling matrix, $\boldsymbol{\Psi} \in \mathbb{R}^{\mathrm{L}\times \mathrm{L}}$ is the sparsifying representation basis, $\boldsymbol{r} \in \mathbb{R}^{\mathrm{L}\times 1}$ is a hyperspectral pixel vector, $\boldsymbol{n} \in \mathbb{R}^{\mathrm{m}\times 1}$ is a noise vector and m represents the number of CSBs that are acquired in a single measurement.

$$\boldsymbol{y} = \boldsymbol{\Phi}\,\boldsymbol{\Psi}\boldsymbol{r} + \boldsymbol{n} \tag{4}$$

In this general framework, the noise vector, $\boldsymbol{n}$, can represent multiple noise sources, such as, environmental noise and various forms of sensor and operation noise. The theoretical discussion, within this work, is limited to the noise free case; however, many works in the literature have demonstrated the robustness of CS to noise (Candes and Wakin 2008). Additionally, noise robustness is demonstrated empirically through the use of real hyperspectral images in experimental analysis.

The success of the model in (4) depends on two specific concepts: sparsity and incoherence. Fundamentally, the signal of interest must be sparse in some representation basis, $\boldsymbol{\Psi}$, for any sub-sampling strategy to be effective. The order of sparsity, k, of a signal is often defined as the number of non-zero values and is denoted as the l_0 norm, $\|\boldsymbol{r}\|_0$. However, this strict definition is impractical for most real signals, especially in the presence of noise. A more qualitative definition is adopted where a signal is said to be sparse or *compressible* if the sorted coefficients of the absolute value of the signal follows an exponential decay. A simple illustration of this qualitative

definition is shown in Figure 13. In this cartoon, the orange line corresponds to a sparse signal where a majority of the points are exactly zero, and the green line corresponds to a compressible signal where the absolute value of the coefficients follows an exponential decay toward zero.

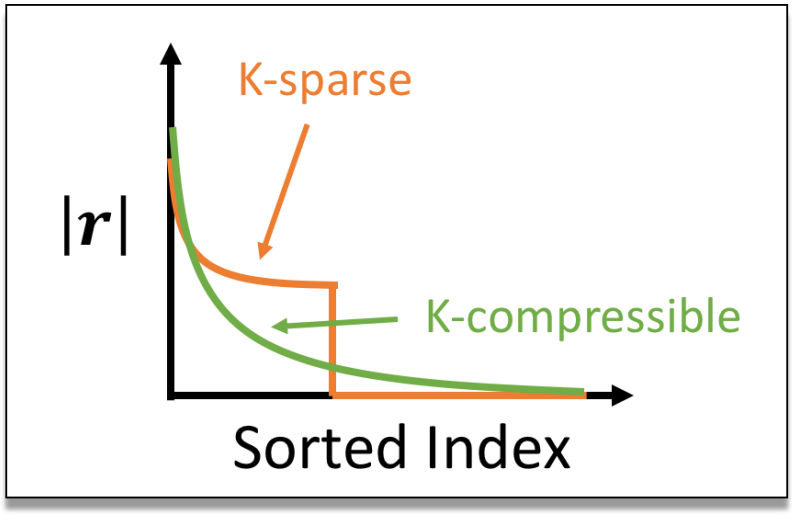

Figure 13: Simple illustration of a sparse or compressible signal.

The second key requirement for the compressive sampling model in equation (4), is that the data must be sampled incoherently. This requirement is motivated by the fact that the compressive sampling process is signal-independent and it is desirable that each compressed sample contain approximately the same amount of information. This ensures that for any given number of CSBs the amount of information acquired will be, with high probability, approximately equivalent. In particular, for the given model, the sampling matrix, $\mathbf{\Phi}$, should be maximally incoherent with the representation basis, $\mathbf{\Psi}$. As was the case for sparsity, many definitions have also been proposed for the coherence between two matrices. The adopted convention in this work is to define the

coherence measure, $\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi})$, as the largest correlation between any two elements of $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ (Candes and Wakin 2008). This definition of coherence is described mathematically in equation (5), where the measure is bounded as $\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi}) \in \left[1, \sqrt{L}\right]$.

$$\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi}) = \sqrt{L} \max_{1 \leq k, j \leq L} \left| \langle \phi_k, \psi_j \rangle \right| \tag{5}$$

Sparse Recovery

To discuss the sparse recovery process, a more formal consideration of the concept of information loss is required and a key notion of CS known as *restricted isometries* must be introduced. The *restricted isometry property* (RIP) (Baraniuk, et al. 2008) defines the restricted isometry constant δ_k of a matrix $\boldsymbol{\Phi}$ as the smallest number such that equation (6) holds for all k-sparse vectors, where k can be any positive integer.

$$(1 - \delta_k)\|\boldsymbol{r}\|_{l_2}^2 \leq \|\boldsymbol{\Phi r}\|_{l_2}^2 \leq (1 + \delta_k)\|\boldsymbol{r}\|_{l_2}^2 \tag{6}$$

The matrix $\boldsymbol{\Phi}$ can be said to satisfy the RIP of order k if δ_k is not too close to one (Candes and Wakin 2008). This also implies that all subsets of k columns taken from $\boldsymbol{\Phi}$ will be approximately orthogonal.

In CS, a general convention is to define a lossless compression as one in which the Euclidean distance between any two vectors is preserved. This can be described mathematically by the RIP of order 2, since the maximum dimensionality of the difference of two k-sparse vectors will be $2k$. Equation (6) can be written in terms of the difference of two vectors as shown in equation (7).

$$(1-\delta_{2k})\|\boldsymbol{r}_1-\boldsymbol{r}_2\|_{l_2}^2 \leq \|\boldsymbol{\Phi r}_1-\boldsymbol{\Phi r}_2\|_{l_2}^2 \leq (1+\delta_k)\|\boldsymbol{r}_1-\boldsymbol{r}_2\|_{l_2}^2 \qquad (7)$$

The fact that the Euclidean distance is maintained, guarantees that the k-sparse vectors cannot be in the null space of the matrix $\boldsymbol{\Phi}$ and that a sparse recovery is indeed possible. The sparse recovery process can now be framed as the minimization shown in equation (8), where $\tilde{\boldsymbol{r}}_s$ represents the sparse signal that has been recovered from the compressed measurement $\boldsymbol{y}=\boldsymbol{\Phi r}$.

$$\min_{\tilde{\boldsymbol{r}}\in\mathbb{R}^L}\|\tilde{\boldsymbol{r}}_s\|_{l_1} \quad s.t. \quad \boldsymbol{\Phi}\tilde{\boldsymbol{r}}_s=\boldsymbol{y}=\boldsymbol{\Phi r} \qquad (8)$$

Orthogonal matching pursuit (OMP) (Tropp and Gilbert 2007) and basis pursuit (BP) (Chen, Donoho and Saunders 2001) are two of the most common sparse recovery algorithms; however, there are a number of ways to solve this problem algorithmically in practice. A more detailed discussion on sparse recovery algorithms is beyond the scope of this work and the interested reader is referred to (Fornasier and Peter n.d.) for a survey of the available algorithms.

Sparse Representation of Hyperspectral Images

A logical starting point is to determine a representation basis in which hyperspectral pixel vectors are indeed sparse or compressible. Previous works on hyperspectral sparsity have suggested the use of the discrete cosine transform (DCT) (Amhed, Natarajan and Rao 1974), wavelet transforms (Antonini, et al. 1992), and dictionary learning approaches (Li, et al. 2013). Figure 14 illustrates the effectiveness of the DCT

and the Haar wavelet (Haar 1910) as sparsifying transforms on all four hyperspectral test images. For each pixel in an image, the minimum number of coefficients required to represent 99% of the total signal power is reported, where the total signal power is inner product of the pixel vector in the specified domain, $Pwr(\boldsymbol{r}_i) = (\boldsymbol{\Psi r}_i)^T(\boldsymbol{\Psi r}_i)$. The minimum number of coefficients for all pixels, in each image, are summarized as normalized histograms for the original basis ($\boldsymbol{\Psi} = \boldsymbol{I}_{LxL}$), the DCT basis and the Haar basis. The original basis (shown in blue) requires nearly all of the coefficients to represent 99% of the signal power, suggesting that the data are not sparse in the original domain. Both the DCT (red) and the Haar (orange) transforms show consistently lower number of required coefficients to represent 99% of the signal power, suggesting that the data are indeed compressible in these domains.

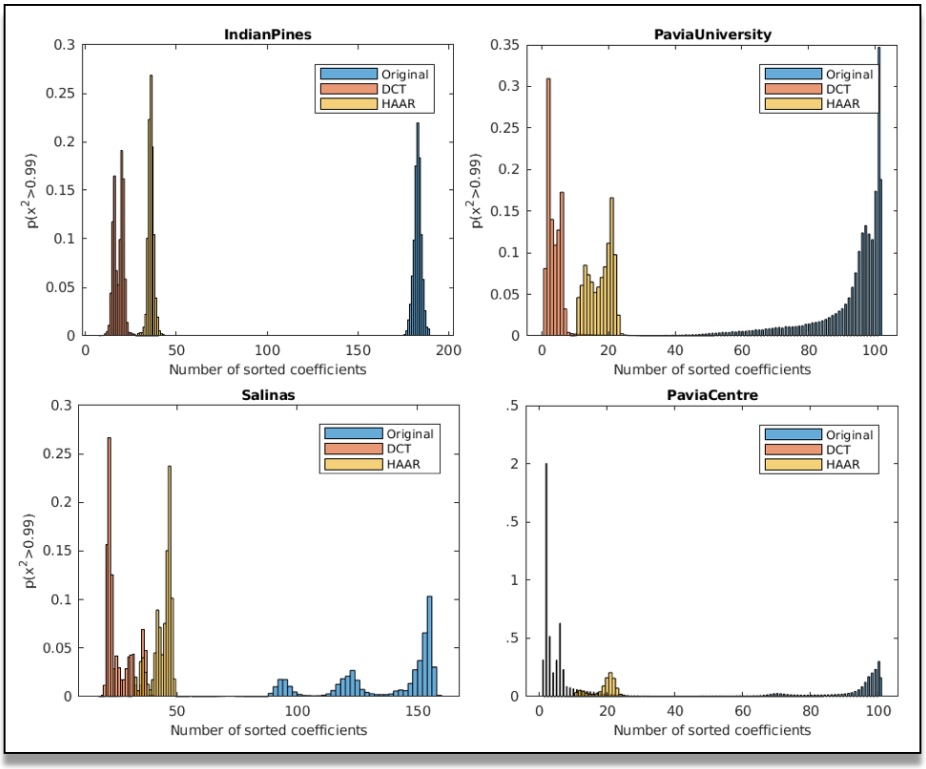

Figure 14: Distribution of the total number of sorted coefficients required to represent 99% signal energy.

It is important to re-iterate that there are many other transformations in which hyperspectral data are sparse and that the DCT is likely not an optimal sparsifying domain. However, as it will be shown in the following section, there is in fact no need to find an optimal sparsifying basis for algorithms that will operate directly on the compressed measurements. The sparsifying transform need only be known during the reconstruction process.

Random Compressive Sampling

As mentioned previously, it is critical in a CS system that the measurements are made incoherently with respect to the sparse domain, more strictly, $\mu(\boldsymbol{\Phi}, \boldsymbol{\Psi})$ should be minimized. It is natural then to design the sampling matrix, $\boldsymbol{\Phi}$, based on the assumed sparse representation, $\boldsymbol{\Psi}$; however, this is not common in practice as it is not easy to design a sampling matrix that is simultaneously incoherent with the sparse representation and able to satisfy the RIP. Fortunately, for most signals a random sampling strategy has been shown to be sufficient, if not optimal (Candes and Wakin 2008). In other words, sampling matrices drawn from certain random distributions can be shown to satisfy both RIP and incoherent sampling with respect to the sparse representation, $\boldsymbol{\Psi}$. That is, for most practical systems, a random sensing matrix is indeed a nearly optimal choice regardless of the sparse representation. This is critical for designing real-world systems as it makes the sampling system agnostic to the sensing modality and is exactly the main reason that random sampling strategies have become ubiquitous in the CS literature.

More specially, the sparse acquisition model in equation (4) can be implemented by constructing the sampling matrix, $\mathbf{\Phi}$, from any distribution that satisfies the *concentration of measure inequalities* (Baraniuk, et al. 2008). For such a family of sampling matrices, it can be shown that RIP is valid provided that equation (9) is satisfied, where the number of compressively sensed bands, m, and the true sparsity order of k are related by a constant, c, and the total number of spectral bands, L. For the case of reconstruction, c will be approximately equal to a value of two to ensure the Euclidean distance is preserved.

$$m \geq ck \log \frac{L}{k} \tag{9}$$

While many distributions have been proven to satisfy the inequalities, the Gaussian distribution is very commonly adopted due to its simplicity in analysis and modeling. To construct a Gaussian sampling matrix, each element of the matrix, ϕ_{ij}, is drawn from a normal distribution with a variance scaled by the number of dimensions, as shown in equation (10). The column vectors of the sampling matrix are next normalized to ensure the matrix is orthogonal. The resulting sampling matrix will satisfy RIP and will be nearly incoherent with any sparse representation, achieving both of the desired objectives.

$$\phi_{ij} \sim \mathcal{N}\left(0, \frac{1}{L}\right) \tag{10}$$

A Universal Compressive Sensing Model for Hyperspectral Imaging

This sub-section presents a property referred to as *universality*, which is an enabling concept in the proposed CS classification approach. It is a universal sensing model which leverages the fact that the sparse basis matrix, $\boldsymbol{\Psi}$, need not be known at the time of signal acquisition, but only required during sparse recovery. More specifically, by adopting a random sampling strategy, the sparse acquisition model in equation (4) can be simplified to the form shown in equation (11), where $\boldsymbol{\Phi}_{\mathrm{R}}$ represents a random sampling matrix, that satisfies the *concentration of measure inequalities* and m is selected such that equation (9) is satisfied.

$$y = \boldsymbol{\Phi}_{\mathrm{R}}\boldsymbol{r} + \boldsymbol{n} \tag{11}$$

Furthermore, the constraint of the sparse recovery process in equation (8) can be modified by including the sparse representation matrix $\boldsymbol{\Psi}$ to yield the form shown in equation (12).

$$\min_{\tilde{\boldsymbol{r}}\in\mathbb{R}^{L}} \|\widetilde{\boldsymbol{r}_s}\|_{l_1} \quad s.t. \quad \boldsymbol{\Phi}_{\mathrm{R}}\boldsymbol{\Psi}\widetilde{\boldsymbol{r}_s} = \boldsymbol{y} \tag{12}$$

By combining equations (11) and (12), a universal sensing model is established and clearly demonstrates that knowledge of the sparse matrix is only required for the signal reconstruction. This has the extremely important implication that, for algorithms performed in CSBD, optimal performance can be achieved without the need of identifying the sparse matrix $\boldsymbol{\Psi}$.

The universality property also has very important implications in the design of physically realizable compressively-sensed hyperspectral systems. Given the fact that the sparsifying transform, $\boldsymbol{\Psi}$, is no longer a part of the acquisition process, there is no need for a specific transform to be included in the hardware design. This means that new compressively sampled hyperspectral systems can be designed using existing hardware with the addition generic linear mixing hardware, to create the random mixtures. As a result, it reduces the overall cost of new systems. In addition, it also facilitates the process of retrofitting existing hyperspectral sensors to be able to compressively sample bands.

Experiments

An experiment was conducted to illustrate the concept of universality for the task of reconstruction. For each image, 1,000 pixels were randomly selected and sparsely sampled using both of the sensing models described in equations (4) and (11). The sampling matrix was randomly generated from a normal distribution, as shown in equation (10), where each column was normalized such that $\|\boldsymbol{\phi}_v\|_{l_2} = 1$ for any column v. For the sensing model in equation (4), the DCT was chosen as the sparsifying transformation. In both cases, the number of CSBs, m, was varied from 40 to $0.7L$, where the total number of bands, L, varied for each image. The minimum and maximum number of CSBs were selected to reduce computation time, and were chosen based on the expected sparsity levels in the DCT domain, shown in Figure 14. Recall, that the minimum number of CSBs required to satisfy the RIP condition is approximately twice the true sparsity level. Each of the CSB vectors were then reconstructed using the Orthogonal Matching Pursuit (OMP) (Tropp and Gilbert 2007) reconstruction

algorithm with the appropriate constraints shown in equations (8) and (12), respectively. A high level description of the experimental steps for a single pixel is provided in Table 1.

Table 1: Experiment procedure for illustrating universal compressive sampling

Universal Random Sampling Experiment
Input: A hyperspectral pixel vector $\boldsymbol{r} \in \Re^{1\times L}$ 1. Generate a random sampling matrix, $\phi_{ij} \sim \mathcal{N}\left(0, \frac{1}{L}\right)$, and normalize the columns such that $\sum_{i=1}^{m} \phi_{ij} = 1$, for $j \in \{1,2,\dots,L\}$. 2. Model a compressively-sampled band vector sampled in the DCT domain, $\boldsymbol{y}_1 = \boldsymbol{\Phi\Psi}_{DCT}\boldsymbol{r}$, and directly in the spectral domain, $\boldsymbol{y}_2 = \boldsymbol{\Phi r}$. 3. Use OMP to minimize the objective $\min_{\tilde{\boldsymbol{r}}\in\mathbb{R}^L} \|\tilde{\boldsymbol{r}}\|_{l_1}$ such that $\boldsymbol{\Phi}\tilde{\boldsymbol{r}}_1 = \boldsymbol{y}_1$ and $\boldsymbol{\Phi\Psi}\tilde{\boldsymbol{r}}_2 = \boldsymbol{y}_2$. 4. Perform an inverse DCT transform on the reconstructed pixel vectors to obtain estimates of the original pixel vector. **Output**: Reconstructed estimates of the pixel vector, $\tilde{\boldsymbol{r}}_1 \in \Re^{1\times L}$, $\tilde{\boldsymbol{r}}_2 \in \Re^{1\times L}$

To assess the consistency of both sampling approaches, the root-mean-square (RMS) errors between the original pixel vectors and the pixel vectors that were recovered during sparse reconstruction were calculated. The mean RMS error over all pixels was calculated as a function of m, and is shown in Figure 15 for all four images. For the red lines, the pixel vectors were sampled directly in the spectral domain, and for the blue lines, the pixel vectors were sampled in the DCT domain. In all cases, the

mean error vectors align exceptionally well, which suggests that, on average, both approaches provide the same achievable reconstruction error.

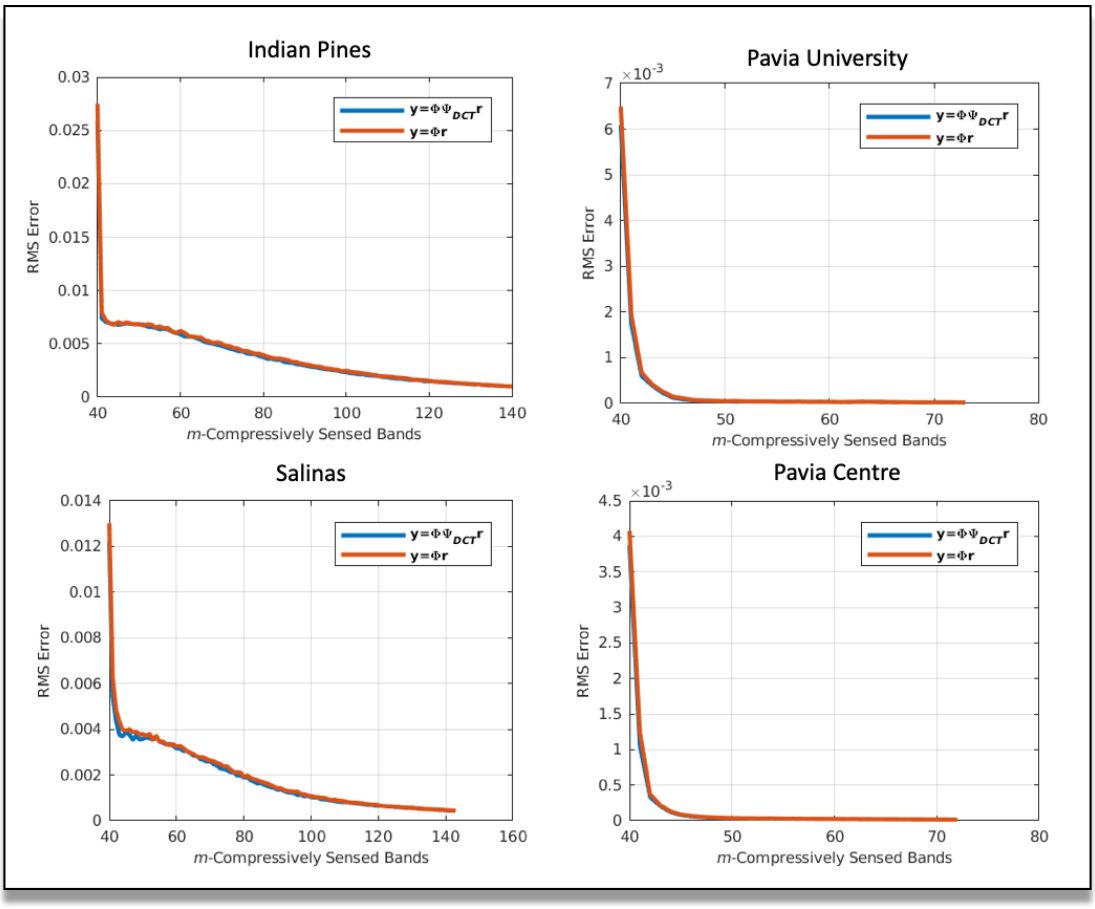

Figure 15: Mean RMS reconstruction error for all images

To probe a little bit further, a metric based on the individual reconstruction error of each pixel was devised. An error threshold was first determined based on the statistics of the mean RMS error curves. More specifically, for each image, an empirical cumulative distribution function (CDF) was estimated from the mean RMS curves and then the 50th percentile RMS error was selected as a global threshold. Finally, the minimum number of CSBs required to achieve an RMS error less than or equal to the

50th percentile threshold was calculated for each pixel. The minimum CSBs required for each of the individual pixel vectors were used to generate normalized histograms for all the images, and they are shown in Figure 16. As before, the blue represents pixel vectors sampled in the DCT domain and red represents pixel vectors directly sampled. The resulting distributions show excellent agreement. There are small differences between the distributions; however, given the relatively few test pixels, and the probabilistic nature of CS, these slight perturbations are to be expected.

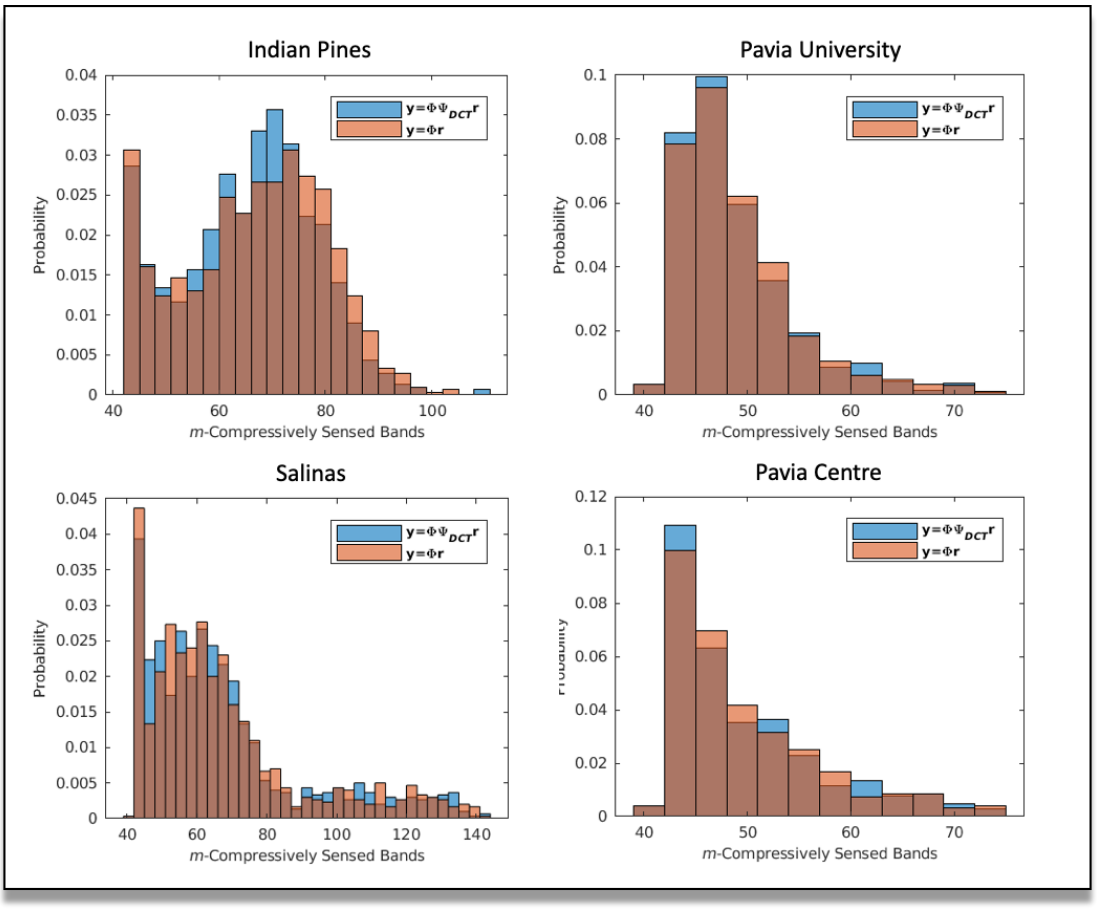

Figure 16: Minimum CSB distribution for all images

Conclusion

The notion of a sub-sampled system was introduced and a compressed hyperspectral imaging system model was proposed. The fundamentals of compressive sensing were reviewed and the two key concepts of sparsity and incoherence were introduced and defined. A typical mathematical model for compressive sampling and sparse recovery were presented and used to derive a compressively-sensed hyperspectral model. A random sampling strategy was proposed and an argument was made to leverage the universality afforded by this sampling strategy, when operating directly on the compressed measurements, since it removes the need to determine the sparse representation. An experiment was performed to illustrate the universality of random sampling and to confirm that all of the required information is embedded in the sparse measurement regardless of which orthogonal basis it is acquired from.

Chapter 3: Hyperspectral Image Classification via Compressive Sensing

Introduction

In this Chapter, the task of hyperspectral classification in the compressively-sensed band domain (CSBD) is considered. A joint spectral-spatial classifier based on an Edge Preserving Filter (EPF) is presented. Spectral classification based on the Support Vector Machine (SVM) is considered. The specific contribution of this chapter is to evaluate the utility of the spectral classifier in CSBD through both mathematical analysis and empirical experimentation. Expected error bounds on classification performance between the full band and CSBD are derived for the linear and radial basis function (RBF) SVM. A set of experiments are performed to validate the mathematical analysis and theoretical discussion using the four real-world images described in Chapter 1. Finally, the concept of scene complexity is investigated through a comparison between the different hyperspectral sensors and the individual image content. A new metric is proposed, based on performance efficacy and a compressively-sensed band ratio, that allows for direct comparisons among the images. The work presented in this chapter ultimately demonstrates both analytically and experimentally that classification in the CSBD is possible, while maintaining sufficient performance for nearly all practical classification tasks.

Support Vector Machines

The support vector machine (SVM) (Cortes and Vapnik 1995) is a popular classifier that has been successfully applied in hyperspectral applications and is well understood

in the machine learning and pattern recognition communities. The SVM is a binary discriminant function that defines an l-dimensional hyperplane separating two classes described by L features. The linear SVM can be defined as the discriminant function shown in Equation (13), where the weight vector ω bias b are maximized using a maximum margin objective function, subject to $\|\boldsymbol{\omega}\| = 1$.

$$f(\boldsymbol{r}) = \boldsymbol{\omega}^T \boldsymbol{r} + b \tag{13}$$

A hard classification of unknown pixel vectors is performed by observing the sign of the output, where positive values $+1$ belong to one class and negative values -1 belong to the other. Although, the SVM is natively a binary classifier, it is often extended to multi-class classification problems through the use of a binary extension strategy, such as one-vs-rest or one-vs-one (Bishop 1995). The one-vs-one approach is specifically adopted in this work.

To make SVMs more mathematically convenient, the Representer Theorem (Schölkopf, Herbrich and Smola 2001) is often leveraged. The theorem states that, with appropriate constraints, $\boldsymbol{\omega}$ can always be written as a linear combination of the training data, $\boldsymbol{\omega}^T = \sum_{j=1}^{N_{Train}} \alpha_j d_j \boldsymbol{r}_j^T$, where $d_j \in \{-1,1\}$ is the training label, $\alpha_j \in [0,1]$ is a linear coefficient and N_{train} is the total number of training samples. Substituting this relationship, the linear SVM discriminant function can also be written as:

$$f(\boldsymbol{r}) = \sum_{j=1}^{N_{Train}} \alpha_j d_j \boldsymbol{r}_j^T \boldsymbol{r} \tag{14}$$

where the bias term is implicitly included in the weight vector. In general, most values of α_j will be zero and the non-zero values are referred to as the support vectors. In this form, the SVM is completely described by the choice of the support vectors, which are typically solved for iteratively.

In many practical classification problems, the data are not linearly separable in the original feature space. A solution to this problem is to transform the feature vectors into a high dimensionality feature space where the data are linearly separable. For an arbitrary transformation, $\boldsymbol{\phi}(\boldsymbol{r})$, the linear discriminant function can be rewritten, more generally, as shown in Equation (15).

$$f(\boldsymbol{r}) = \sum_{j=1}^{N} \alpha_j d_j \boldsymbol{\phi}(\boldsymbol{r})_j^T \boldsymbol{\phi}(\boldsymbol{r}) \tag{15}$$

In practice, transforming all of the data to the new dimension is computationally prohibitive and is avoided using what is referred to as the kernel trick. A property of Mercer's theorem (Scholkopf and Smola 2002) is used to directly calculate the required dot products without ever mapping the data into the non-linear feature space. The dot products in Equation (15) are replaced with an arbitrary kernel operator, $K(\boldsymbol{r}^T, \boldsymbol{r}) = \boldsymbol{\phi}(\boldsymbol{r}^T)\ \boldsymbol{\phi}(\boldsymbol{r})$, as shown in equation (16), where the Kernel is carried out on the original data.

$$f(\boldsymbol{r}) = \sum_{j=1}^{N} \alpha_j d_j\ K(\boldsymbol{r}_j, \boldsymbol{r}) \tag{16}$$

Many different types of kernels have been proposed for use with support vector machines; however, one of the most common kernels is the radial basis function (RBF) kernel. The RBF is defined as $K(\boldsymbol{r}^T, \boldsymbol{r}) = \exp\left(-\frac{\|\boldsymbol{r}-\boldsymbol{r}\|^2}{2\sigma^2}\right)$, where σ is a tuneable hyperparameter. Substituting the kernel definition into Equation (16), the RBF SVM discriminant function is written as shown in Equation (17).

$$f(\boldsymbol{r}) = \sum_{j=1}^{N} \alpha_j d_j \exp\left(-\frac{\|\boldsymbol{r}_j - \boldsymbol{r}\|^2}{2\sigma^2}\right) \tag{17}$$

Edge Preserving Filters

Spatial correlation between neighboring pixels must be taken into account to achieve state-of-the-art hyperspectral classification performance. To capture the spatial contextual information, the edge preserving filter (EPF) proposed in (Kang, Li and Benediktsson 2014) has been included in the proposed compressed, spectral-spatial classifier. In this approach the initial pixel class probabilities are determined by a spectral classifier and then the spectral classification map is further refined by a follow-up EPF. The spatial filters are performed using guidance images derived from the principal components of the input image, which preserves the major features in the scene. A summary of the approach is presented in Table 2.

Table 2: Edge Preserving Filter Classification Algorithm

Edge Preserving Filtering Classification Algorithm
Input: A hyperspectral image $\boldsymbol{R} \in \Re^{N_x \times N_y \times L}$

1. Generate a classification map, $C_{SPC} \in \Re^{N_x \times N_y}$, using a spectral classifier.
2. Generate an initial binary probability map, $P_{i,p} \in [0,1]$, by assigning the pixelwise class to each respective channel as $P_{i,p} = \begin{cases} 1, \ if \ c_i = l \\ 0, \ otherwise \end{cases}$
3. Optimize the probability map by applying an edge preserving filter, $\acute{P}_{i,p} = \sum_j W_{i,j}(I) P_{i,p}$, where the weights, $W_{i,j}$, depend on the choice of EPF and the guidance image I.
4. Generate the final classification map by choosing the class with the highest probability $C_{EPF} = arg \max_{1 \le n \le P} \acute{P}_{i,p}$

Output: A final classification map, $C_{EPF} \in \Re^{N_x \times N_y}$

The original work presents two approaches for calculating the weights of EPF: the joint bilateral filter and the guided filter. Both approaches provide increased classification performance over the pixelwise classifier and are suitable for the proposed experiments. Specifically, the guided filter was chosen for this analysis and the definition is shown in (18), where ω_i and ω_j are local windows around pixel i and j, μ_k and σ_k^2 are the mean and variance of the local window, and $|\omega|$ is the number of pixels in the window. Regarding the guidance image, I, principal component analysis (PCA) (Wold, Esbensen and Geladi 1987) is used to define a binary or color guidance image by selecting the one or three of the first principal components, respectively. The color guidance image has been selected for use in this analysis.

$$W_{i,j}(I) = \frac{1}{|\omega|^2} \sum_{k \in \omega_i, k \in \omega_j} \left(1 + \frac{(I_i - \mu_k)(I_j - \mu_k)}{\sigma_k^2 + \epsilon}\right) \tag{18}$$

To illustrate the effectiveness of this approach, a comparison between classification maps generated using SVM and SVM-EPF is shown in Figure 17, for the Indian Pines dataset. It is clear that the spatial filtering is able to leverage the local structure of the scene and provide a much more accurate class estimate. The large amount of "speckle" resulting from the spectral-only classifier, is removed by the spatial filtering process.

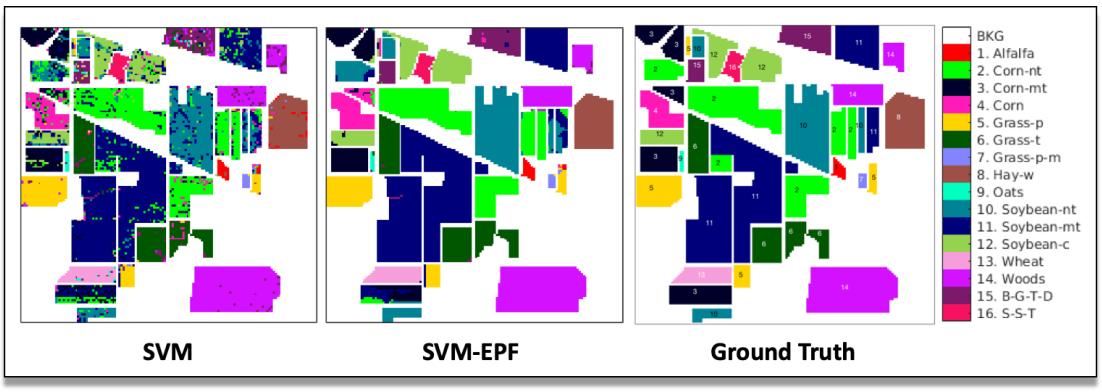

Figure 17: Comparison of SVM and SVM-EPF classification maps for Indian Pines.

Support Vector Machines in the Compressively-Sensed Band Domain

Classification in CSBD can be represented using a modification of the discriminant function that includes the sparse acquisition matrix. The compressed discriminant function then takes the form shown in Equation (19), where the subscript s, for α, denotes the fact that the SVM in the CSBD is not guaranteed to use the same support vectors or bias.

$$f_s(\boldsymbol{r}) = \sum_{j=1}^{N} \alpha_{s_j} d_j \left(\boldsymbol{\Phi r}_j\right)^{\mathbf{T}} \boldsymbol{\Phi r} \tag{19}$$

To examine the performance of the support vector in the classified domain, an error function can be constructed between the full band and CSB domains. Let $\epsilon = |f(\boldsymbol{r}) - f_s(\boldsymbol{\Phi r})|$ denote the absolute error between the fully sampled and the sparsely sampled classifiers.

$$\epsilon = \left| \sum_{j=1}^{N} \alpha_j d_j \boldsymbol{r}_j^T \boldsymbol{r} - \sum_{j=1}^{N} \alpha_{s_j} d_j \left(\boldsymbol{\Phi r}_j\right)^{\mathbf{T}} \boldsymbol{\Phi r} \right| \tag{20}$$

In general, it is impossible to derive a closed form solution for this expression since the support vectors are solved iteratively in practice. Additionally, there is no guarantee of a unique solution when solving for these model parameters. To simplify the analysis, the support vectors are chosen, without guarantee of optimality, to be the same in the sparsely sampled domain. This implies that $\alpha_{s_j} = \alpha_j, \forall\, j$, and the error function in Equation (20) can be reduced to

$$\epsilon = \left| \sum_{j=1}^{N} \alpha_j d_j \left[\boldsymbol{r}_j^T \boldsymbol{r} - \left(\boldsymbol{\Phi r}_j\right)^{\mathbf{T}} \boldsymbol{\Phi r} \right] \right| \tag{21}$$

Furthermore recognizing that $\boldsymbol{r}_j^T \boldsymbol{r} = \left\| \boldsymbol{r}_j \right\|_2^2$, the classification error can be defined as

$$\epsilon = \left| \sum_{j=1}^{N} \alpha_j d_j \left(\left\| \boldsymbol{r}_j \right\|_2^2 - \left\| \boldsymbol{\Phi} \boldsymbol{r}_j \right\|_2^2 \right) \right| \tag{22}$$

Given that the sampling matrix, $\boldsymbol{\Phi}$, satisfies the RIP in Equation (6), the error can be re-written as an inequality in terms of the RIC, δ_k, as

$$\epsilon \leq \left| \sum_{j=1}^{N} \alpha_j d_j \left(\left\| \boldsymbol{r}_j \right\|_2^2 - (1 - \delta_k) \left\| \boldsymbol{r}_j \right\|_2^2 \right) \right| \tag{23}$$

Expanding the RIC term and simplifying, the final error can be written as

$$\epsilon \leq \left| \sum_{j=1}^{N} \alpha_j d_j \left\| \boldsymbol{r}_j \right\|_2^2 \delta_k \right| \tag{24}$$

which indicates that the classification error is directly bounded by the RIC and will go to zero for all m that satisfy the condition shown in Equation (6), as δ_k goes to zero. This derivation demonstrates that it is indeed possible to achieve full classification accuracy in the CSBD, provided that sufficient sampling conditions are satisfied.

Kernel Support Vector Machines in the Compressively-Sensed Band Domain

Following a similar approach to the linear SVM, the performance of the RBF SVM in CSBD can be assessed by analyzing the error between the full band and CSB domains. Choosing, again, to maintain the same support vectors in the CSBD, the error function for the RBF SVM is described as shown in Equation (25).

$$\epsilon = \left| \sum_{j=1}^{N} \alpha_j d_j \left[\exp\left(-\frac{\left\| \boldsymbol{r}_j - \boldsymbol{r} \right\|^2}{2\sigma^2} \right) - \exp\left(-\frac{\left\| \boldsymbol{\Phi r}_j - \boldsymbol{\Phi r} \right\|^2}{2\sigma^2} \right) \right] \right| \tag{25}$$

For simplicity, let $\boldsymbol{z} = \boldsymbol{r}_j - \boldsymbol{r}$ and substitute it into Equation (25) to yield

$$\epsilon = \left| \sum_{j=1}^{N} \alpha_j d_j \left[\exp\left(-\frac{\left\| \boldsymbol{z}_j \right\|^2}{2\sigma^2} \right) - \exp\left(-\frac{\left\| \boldsymbol{\Phi z}_j \right\|^2}{2\sigma^2} \right) \right] \right| \tag{26}$$

Knowing that the sampling matrix, $\boldsymbol{\Phi}$, satisfies the RIP in Equation (6), the error can be re-written as an inequality in terms of the RIC, δ_k, as

$$\epsilon \leq \left| \sum_{j=1}^{N} \alpha_j d_j \left[\exp\left(-\frac{\left\| \boldsymbol{z}_j \right\|^2}{2\sigma^2} \right) - \exp\left(-\frac{\left\| (1-\delta_k) \boldsymbol{z}_j \right\|^2}{2\sigma^2} \right) \right] \right| \tag{27}$$

Recognizing that $(1 - \delta_k)$ is a scalar quantity, the expression can be further reduced by expanding the RIC term and factoring out the exponential, resulting in

$$\epsilon \leq \left| \sum_{j=1}^{N} \alpha_j d_j \left[\exp\left(-\frac{\left\| \boldsymbol{z}_j \right\|^2}{2\sigma^2} \right) - \exp\left(-\frac{\left\| \boldsymbol{z}_j \right\|^2}{2[\sigma/(1-\delta_k)]^2} \right) \right] \right| \tag{28}$$

Interestingly, the error is no longer bounded linearly by the RIC, but rather by a difference in the kernel space. In this particular case, both terms are identical with the exception of the RBF hyperparameter, which is scaled by $(1 - \delta_k)^{-2}$. Just as in the linear case, the RBF classification error will go to zero for all m that satisfy the condition shown in Equation (6), as δ_k goes to zero.

Experiments

Simulated experiments were performed comparing performance between the full band and CSB hyperspectral data. Algorithm performance is measured by three common metrics: overall accuracy (P_{OA}), average accuracy (P_{AA}), and average precision (P_{AP}) as described in Chapter 1. Additionally, the individual class accuracies were also considered to further understand the results. Finally, scene complexity was evaluated with the introduction of a new metric based on performance efficacy and a compressively sensed band ratio (CSBR).

Experimental Setup

The sparse hyperspectral pixel vectors were modelled using random Gaussian sensing matrices with m compressively sensed bands taken directly from the original pixel domain (i.e. no sparsifying transformation was applied before compressively sampling the pixel vectors). The number of CSBs, m, was varied from five up to the total number of bands for each image. The RBF-based SVM was used as the spectral classifier, with the tunable kernel parameter set to 0.72, 0.64, 0.75, and 0.59 for Indian Pines, Salinas, Pavia University, and Pavia Centre, respectively. The edge preserving filtering process was based on a guided filter with a color guidance image, based on the first 3 principal components, i.e., EPF-G-c (Kang, Li and Benediktsson 2014).

For each experiment, the pixel vectors were randomly partitioned into training and validation sets following the same set-up described in (Kang, Li and Benediktsson 2014) for Indian Pines, Salinas and Pavia University. The Pavia Centre scene was partitioned in a similar fashion to Pavia University with 285 training samples from each class. The exact number of training and validation samples are tabulated in Tables 3-6.

Table 3: Indian Pines Training and Test Samples

Class Name	Training Samples	Test Samples
Alfalfa	25	21
Corn-notil	83	1345
Corn-mintil	78	752
Corn	68	169
Grass-pasture	79	404
Grass-trees	78	652
Grass-mowed	14	14
Hay-windrowed	66	412
Oates	10	10
Soybean-notil	81	891
Soybean-mintil	99	2356
Soybean-clean	73	520
Wheat	70	135
Woods	90	1175
Buildings	65	321
Stone-Steele Towers	46	47
Background	0	10,076

Table 4: Salinas Training and Test Samples

Class Name	Training Samples	Test Samples
Broccoli 1	67	1942
Broccoli 2	67	3659
Fallow	67	1909
Fallow Rough Plow	69	1325
Fallow Smooth	67	2611
Stubble	67	3892
Celery	68	3511
Grapes Untrained	69	11202
Soil Vineyard	68	6135
Corn	68	3210
Lettuce 4 Week	68	1000
Lettuce 5 Week	67	1860
Lettuce 6 Week	67	849
Lettuce 7 Week	67	1003
Vineyard Untrained	70	7198
Vineyard VT	67	1740
Background	0	56,975

Table 5: Pavia University Training and Test Samples

Class Name	Training Samples	Test Samples
Asphalt	286	6345
Meadows	286	18363
Gravel	285	1814
Trees	285	2779
Painted Metal Sheets	285	1060
Bare Soil	285	4744
Bitumen	285	1045
Self-blocking Bricks	285	3397
Shadows	285	662
Background	0	164,624

Table 6: Pavia Centre Training and Test Samples

Class Name	Training Samples	Test Samples
Water	285	65686
Trees	285	7313
Asphalt	285	2805
Self-blocking Brick	285	2400
Bitumen	285	6299
Tiles	285	8963
Shadows	285	7002
Meadows	285	42541
Bare Soil	285	2578
Background	0	635,488

The same randomly selected set of training samples in Tables 3-6 were used for experiments in both the full band and CSB domains for the EPF-G-c classifier. Each experiment was repeated 20 times and the average results are reported. For all of the plots in this section, the dashed and solid lines show the results of each classification measure produced by EPF-G-c on the full band pixels, i.e., data without compressive sensing and the CSB pixels, respectively.

Classification Accuracy Analysis

The P_{OA} and P_{AA} for Purdue's Indian Pines are plotted in Figure 18, with m ranging from 1 to 220. Note that the plot had a large jump around 30 CSBs, and then began to flatten after 100 CSBs with near full band performance. As can be seen in the figure the maximum performance was achieved using approximately 50% of the total number of CSBs. The range of accuracy for this image is significant with just over a 20% difference in accuracy between the minimum number of CSBs and using the maximum number of CSBs. This particular image has the most imbalanced classes and was also the most difficult one among the four images that were tested. The presented results, in the CSBD, are in line with many of the performance measures reported in the literature.

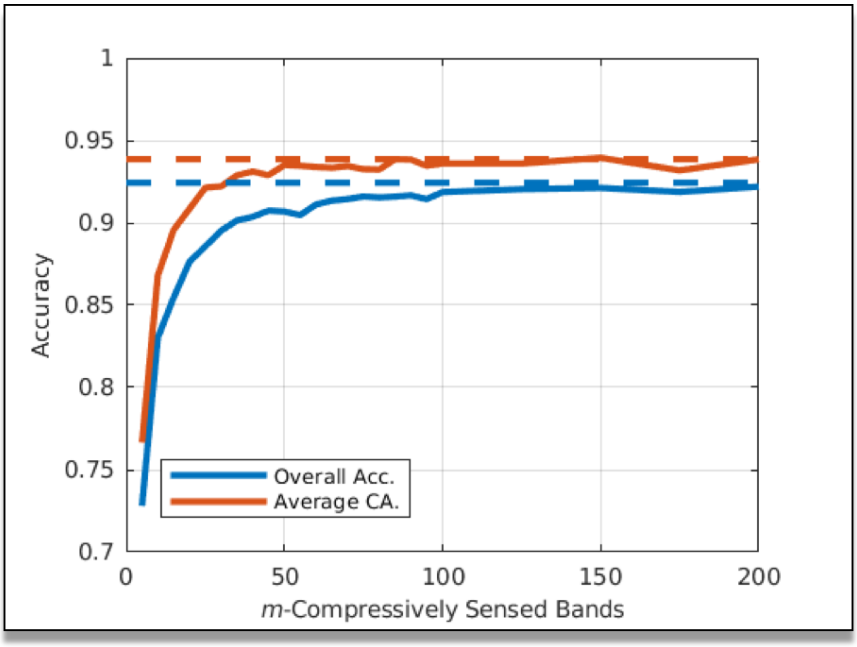

Figure 18: Indian Pines accuracy.

The P_{OA} and P_{AA} for Salinas are plotted in Figure 19. Similar to the experiments conducted for the Purdue Indian Pines, the largest improvement in classification accuracy occurs within the first 30 CSBs and continues to improve slowly until the accuracy flattens after 100 CSBs. Contrary to the Indian Pines image, the range of accuracy values is quite small with just about a 6% difference in performance between the minimum and maximum achievable accuracy. For the minimum case it required only 5 CSBs for EPF-G-c to achieve an overall accuracy and an average accuracy of 92% and 95%, respectively. Remarkably, even with just 2.5% of the total number of CSBs, the achieved accuracy level is acceptable for many practical applications.

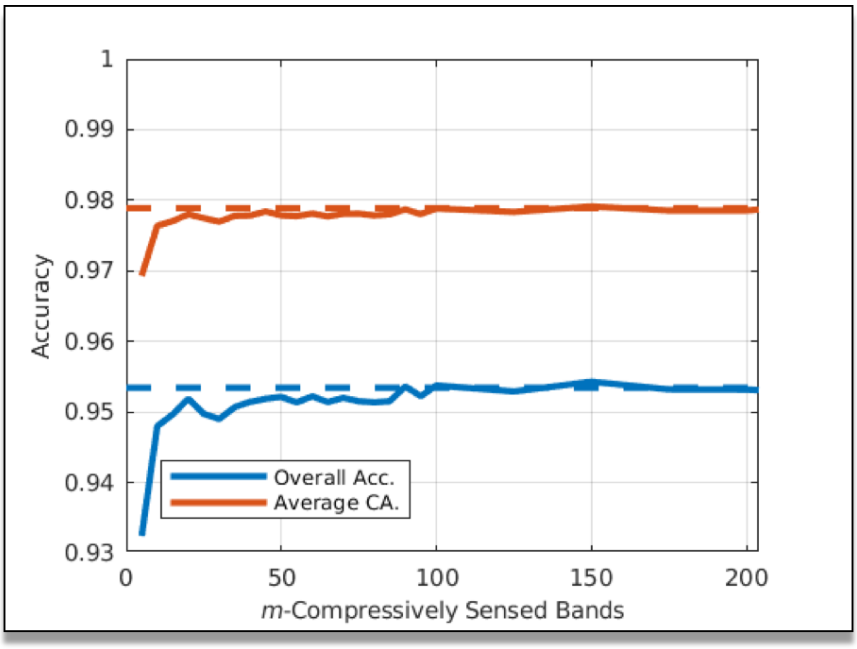

Figure 19: Salinas accuracy.

The P_{OA} and P_{AA} for Pavia University are plotted in Figure 20. Both accuracies converged quickly between 20 and 40 CSBs. There is an exponential increase in performance for the lower number of CSBs. This ROSIS image showed a similar trend

to the AVIRIS Indian Pines image, where there was a significant difference in the minimum and maximum performance achieved for a different number of CSBs. The range of accuracy values between the fewest CSBs and the full band performance is quite significant with a total difference of about 15% for the overall accuracy and 20% for the average accuracy.

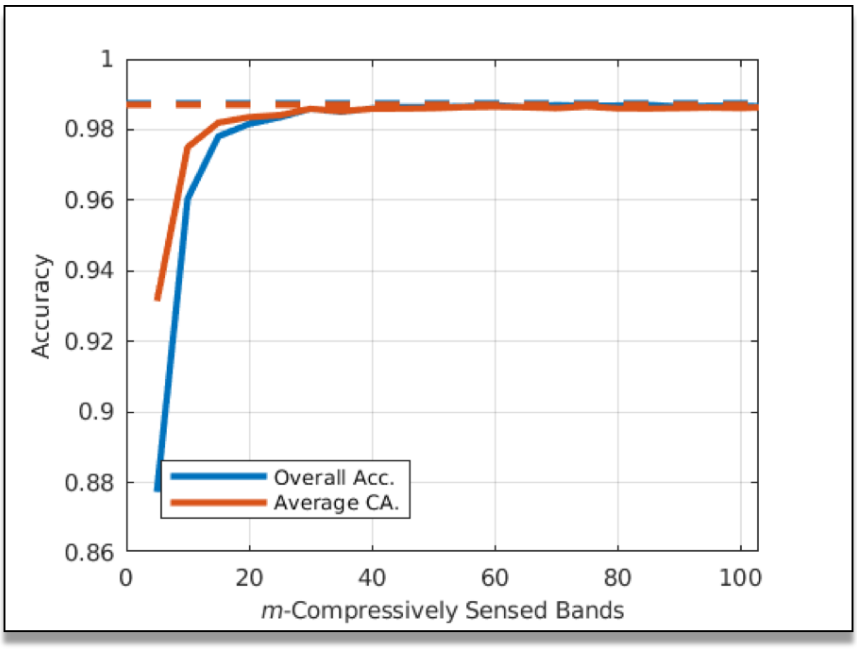

Figure 20: Pavia University accuracy.

The P_{OA} and P_{AA} for Pavia Centre are plotted in Figure 21. This image is somewhat unique because both accuracies appeared to converge at a slightly different number of CSBs where P_{OA} converged with about 20 CSBs and the P_{AA} converged with around 30-50 CSBs. The reason for this will become more apparent when the individual class accuracies are discussed in later experiments. The range of accuracies for this image was exceptionally small. The difference in accuracy between using 5 CSBs and the full band image was just 2.5% for P_{AA} and less than 1% for P_{OA}. This image is

another example which shows accuracy required for practical applications can be achieved with small fractions of the total number of bands.

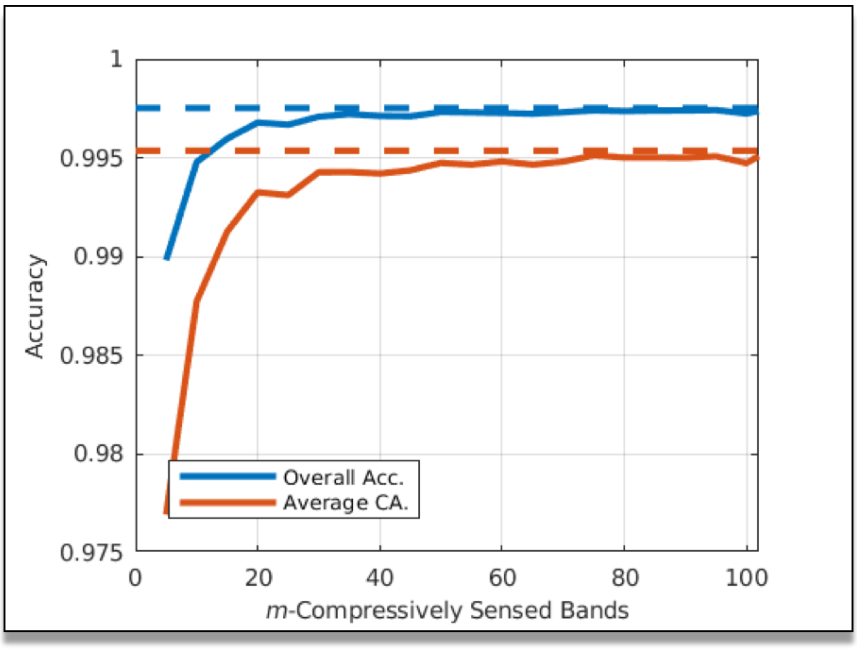

Figure 21: Pavia Centre accuracy.

Interestingly, the classification accuracy achievable in the CSBD did not appear to be correlated with either the scene types (agricultural and urban) or the sensors (AVIRIS and ROSIS) used for experiments. However, for all images acceptable accuracy can be achieved directly in CSBD with a significant reduction in the number of CSBs

Classification Precision Analysis

The P_{AP} is considered in this section and has received little interest in hyperspectral image classification. However, its practical impact is quite important since it is the only measure that accounts for background data samples for classification. To address

background issue in classification P_{AP} is calculated with and without inclusion of the BKG samples in the image scenes.

The P_{AP} for the Purdue Indian Pines is shown in Figure 22. Near full precision performance is achieved with only 50 CSBs with and without the inclusion of the BKG samples. The range of precision values between the fewest CSBs and the maximum CSBs is much smaller than the spread in accuracy, with only about a 10% difference. Following a similar trend as with accuracy, Indian Pines is again the most difficult of the four images tested.

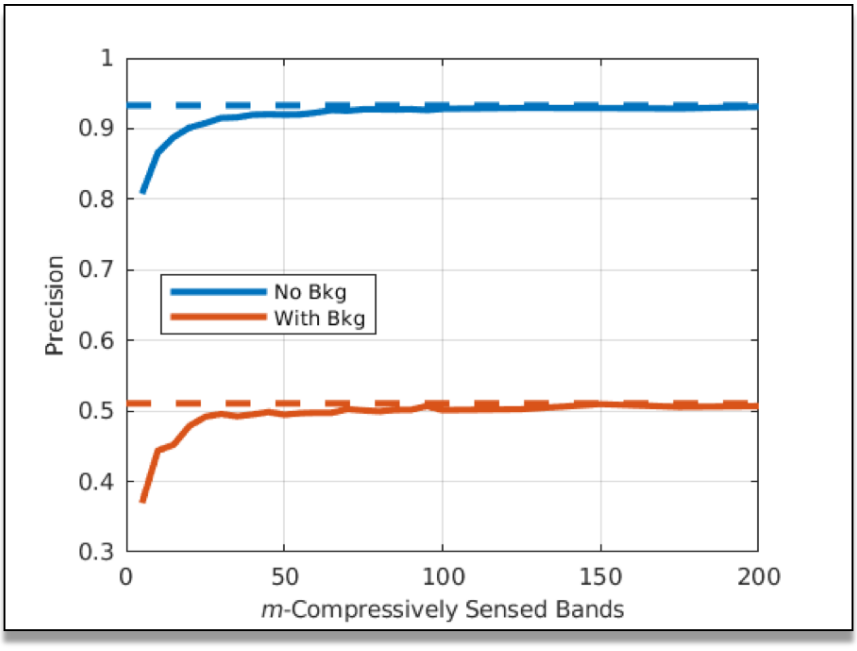

Figure 22: Indian Pines precision.

The results of P_{AP} for Salinas, Pavia University, and Pavia Centre are shown in Figure 23-Figure 25, respectively. For these images, full band domain performance could be achieved with using only 5-10 CSBs! This is quite interesting given the fact that both Indian Pines and Salinas were collected from the same sensor and consist of

similar agricultural classes. Since the relative performance holds for the cases with and without precision, it is not believed to be due to a bias in the background signatures. One possible cause could be due to the fact that the Purdue data contains many imbalanced classes with 4 classes less than 100 data samples. The relatively low training samples for these classes could introduce some instability into the SVM classifiers. Given the many potential influencing factors and availability of ground truthing, further investigation into this difference is was not believed to be warranted.

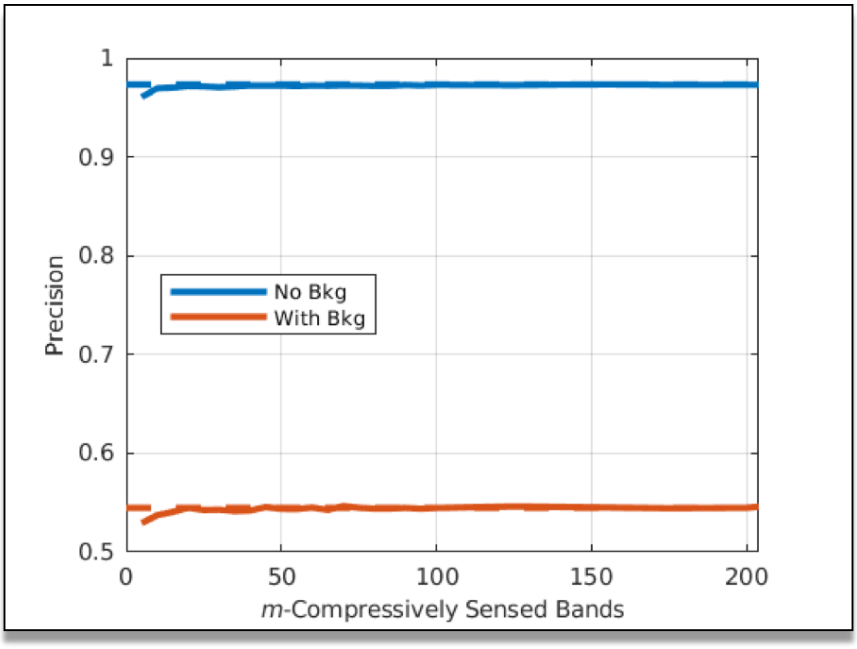

Figure 23: Salinas precision.

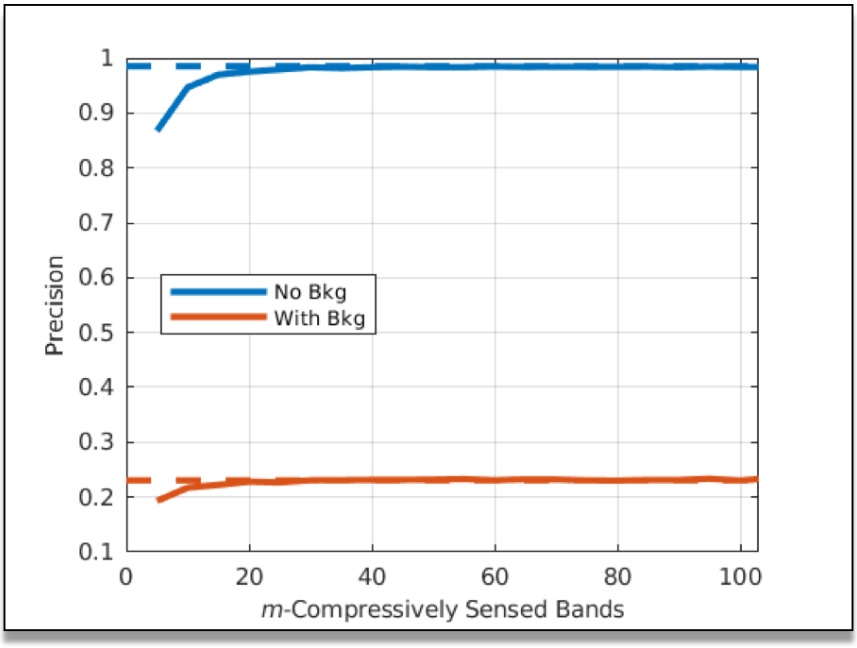

Figure 24: Pavia University precision.

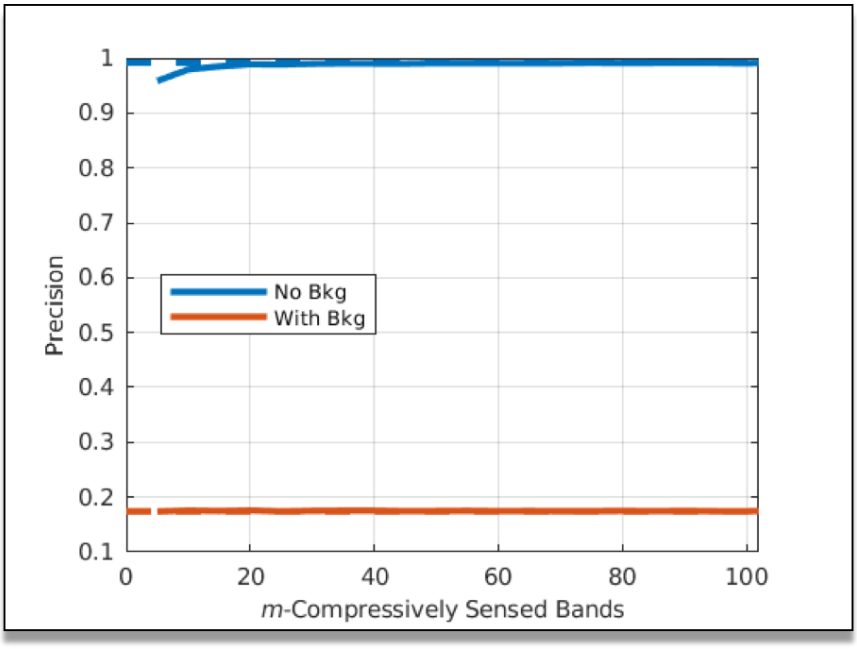

Figure 25: Pavia Centre precision.

In general, the precision is less affected by the compressive sampling for all images. In all cases, full band performance could be achieved with 25% or less than

the total number of available CSBs. For Salinas and Pavia Centre, full band precision performance could be achieved while using only 5% of CSBs. The precision performance was also fairly consistent with and without the inclusion of BKG samples. In agreement with the accuracy performance results, the classification in the CSBD was capable of achieving acceptable precision performance with just fractions of the total number of CSBs.

Individual Class Accuracy Analysis

To further probe into the performance of the four images, the individual class accuracy is reviewed. By observing how the individual classes are affected by the number of CSBs, it is possible to gain some intuition on what really determines the maximum achievable performance in the CSBD, as well as explain some of the differences that have been noted between P_{OA} and P_{AA}.

The individual class accuracies for Indian Pines are plotted in Figure 26. Interestingly, many of the classes converged to full band performance with as few as 10 CSBs, i.e., only 5% of the spectral bands! This is in stark contrast to the 100 CSBs needed before the overall accuracy converged. Accuracy performance in both the full band domain and CSBD is clearly limited by specific classes.

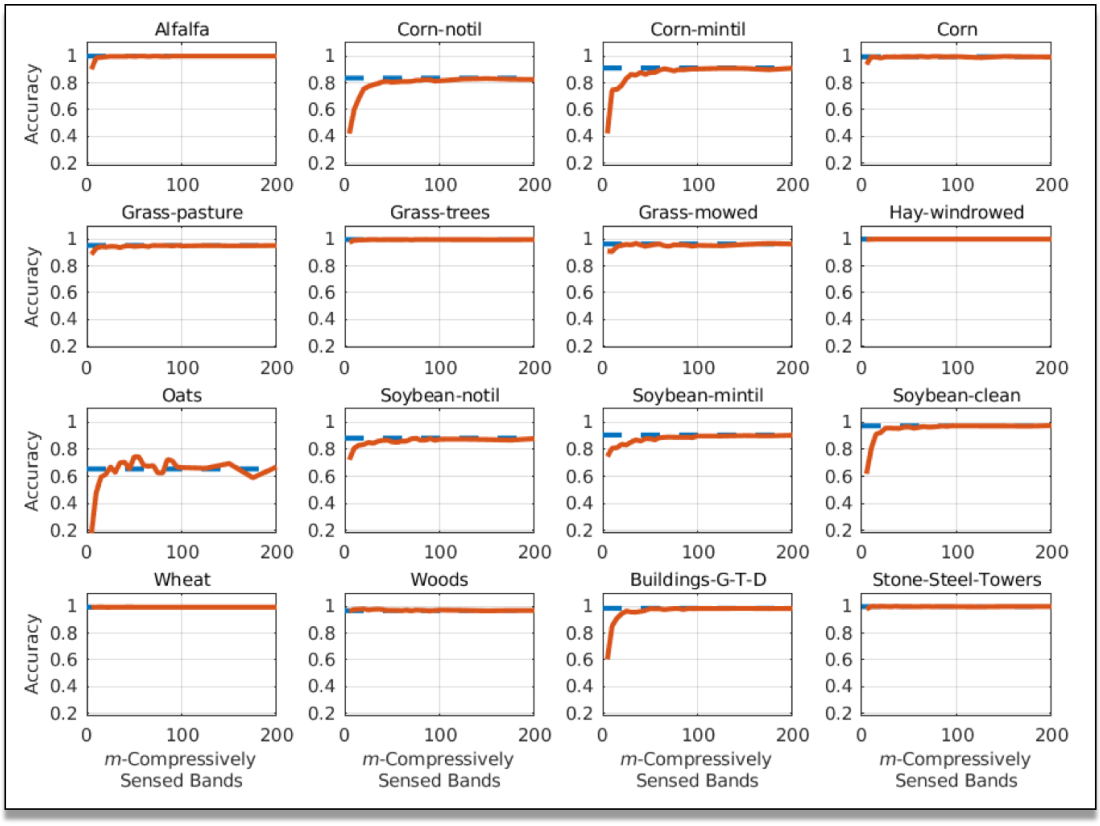

Figure 26: Indian Pines individual class accuracy.

The individual class accuracies for Salinas are shown in Figure 27. Remarkably, in comparison with the Purdue data all of the classes with the exception of "vineyard untrained" converged to full band performance with only 10 CSBs or less! This explains why P_{AA} is higher than P_{OA}. This suggests that the data set is almost completely separable, and shows a clear correlation between class separability and the viability of compressed classification.

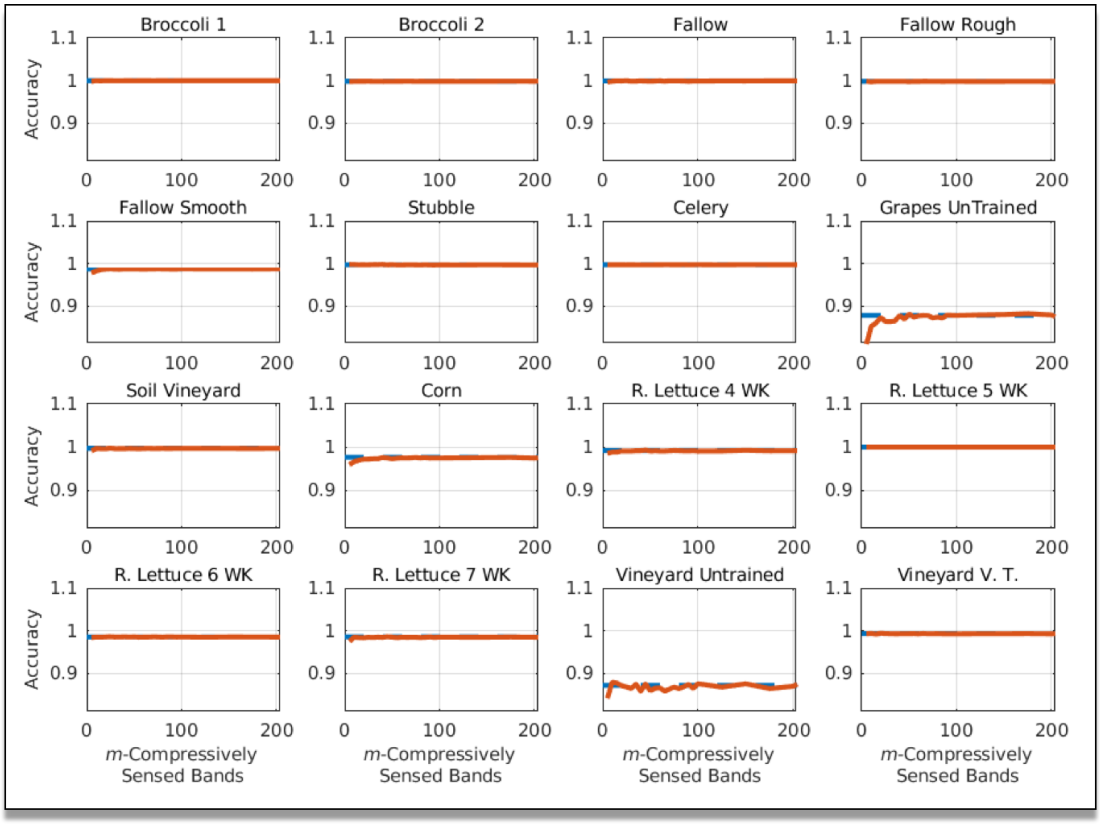

Figure 27: Salinas individual class accuracy.

Similar performance was also observed for the ROSIS images. The individual class accuracies for Pavia University are shown in Figure 28. For this image, half of the classes converged with just 5 CSBs and the remaining classes converged at various numbers of CSBs. All of the classes with the exception of "Gravel" converged to full band classification accuracy.

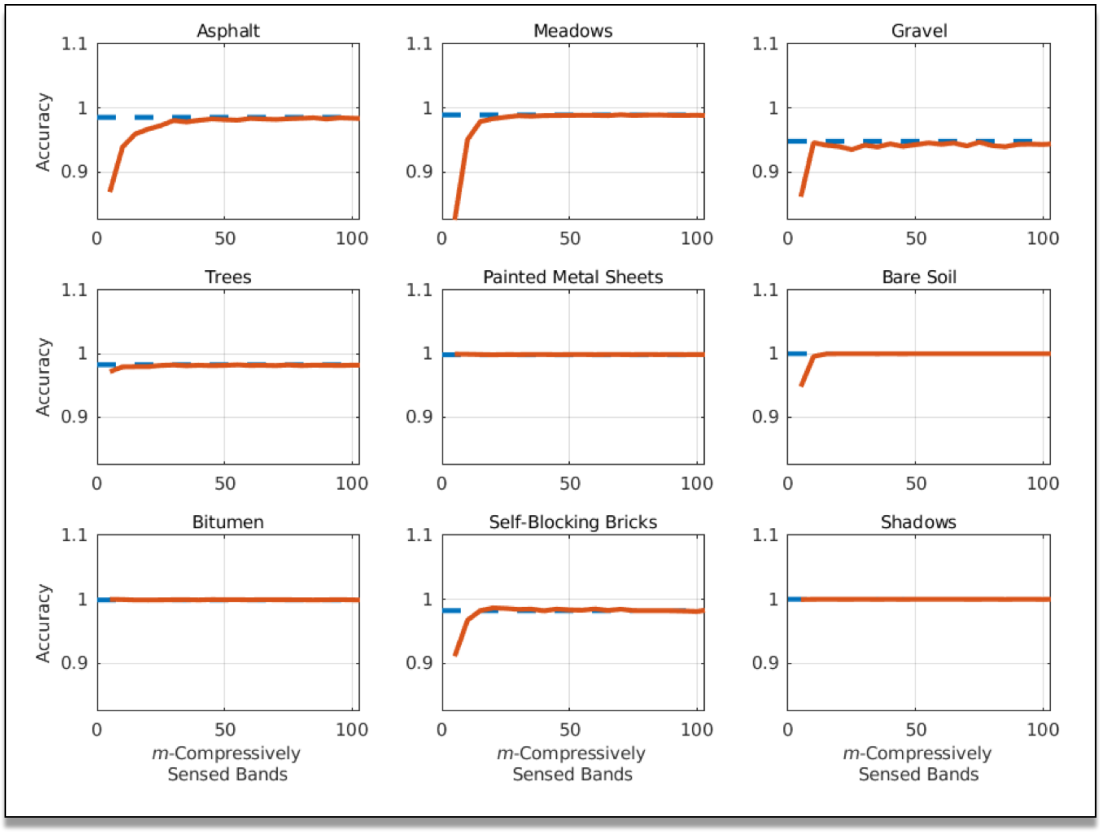

Figure 28: Pavia University individual class accuracy.

The individual class accuracies for Pavia Centre are shown in Figure 29. A similar trend was observed with half of the classes converging with only 5 CSBs and the remaining classes converging at various numbers of CSBs. Two of the classes that did not converge immediately, "asphalt" and "shadows", are classes that never achieved full band classification accuracy. This also reinforces the correlation between class separability and viability of compressive sensing classification.

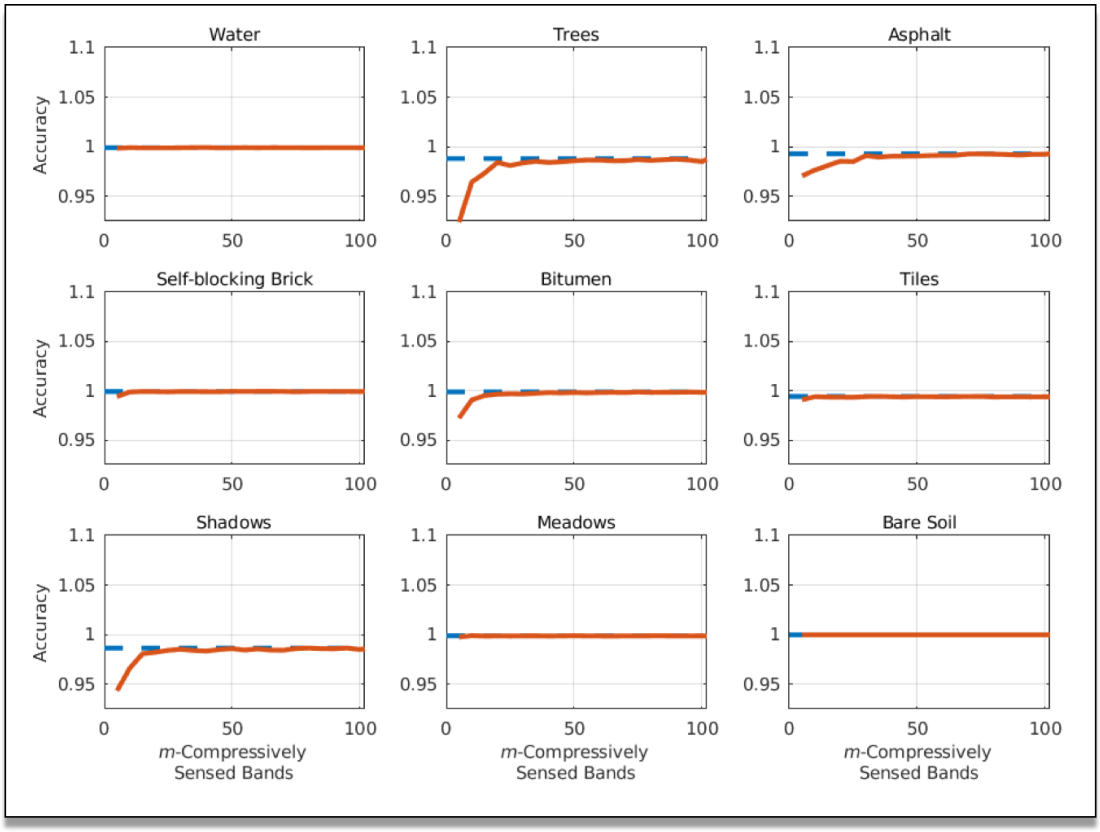

Figure 29: Pavia Centre individual class accuracy.

Scene Complexity

One of interesting results from the above experiments is variability in the number of required CSBs to achieve maximum performance for each of the different scenes. This observation was true for both AVIRIS and ROSIS sensors, suggesting that it was indeed scene complexity that limits effectiveness of CSBs. To investigate this concept further, an efficacy criterion is defined for the overall accuracy, $P_{OA_{Eff}}$, which calculates the ratio of P_{OA} in CSBD to P_{OA} in the original data domain, i.e., full band domain. Similarly, the compressively sensed band ratio (CSBR), $CSBR = \frac{m}{L}$, defined as the number of CSBs divided by the total number of bands, L, is introduced to

normalize the number of spectral bands among images. Using these criteria makes it possible to directly compare the performance for all four images. The overall accuracy efficiency is plotted as a function of CSBR in Figure 30. For both Salinas and Pavia Centre, nearly full band performance can be achieved with a CSBR of only 10%. For the more difficult images, Pavia University required a CSBR of approximately 20% and Indian Pines required approximately 50%.

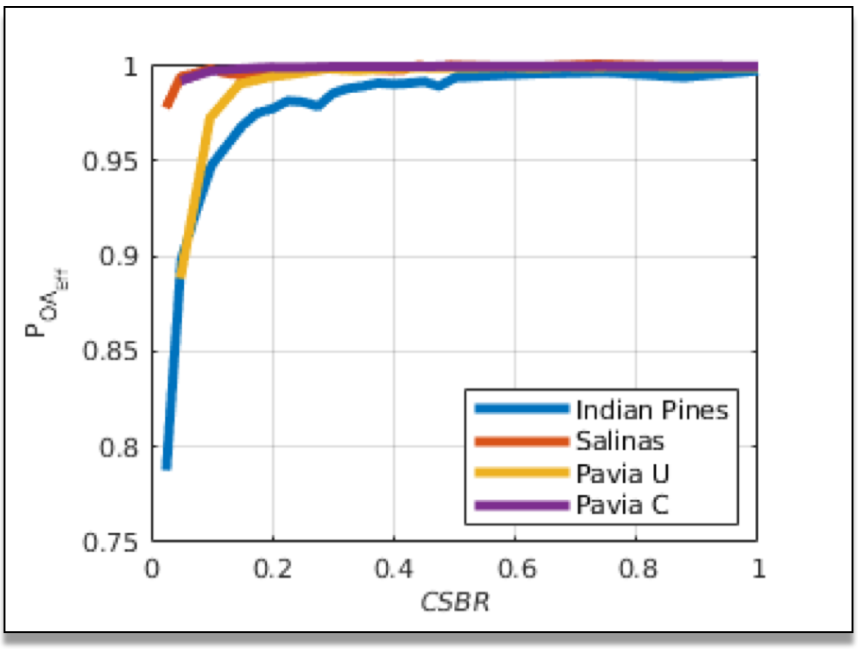

Figure 30: Overall accuracy efficacy for all four images.

Similarly, the efficacy of precision, $P_{AP_{Eff}}$, calculates a ratio of precision in the CSBD to precision in the original full band domain and its results are plotted in Figure 31. As shown in the previous section, precision is generally better than accuracy. The efficacy of precision confirmed this fact with the worst case performance never being lower than 90% of the full band precision performance after a CSBR greater than 20%.

The efficacy of precision also showed the same trend as the $P_{OA_{Eff}}$ with the Indian Pines and Pavia University images being more difficult than the other images.

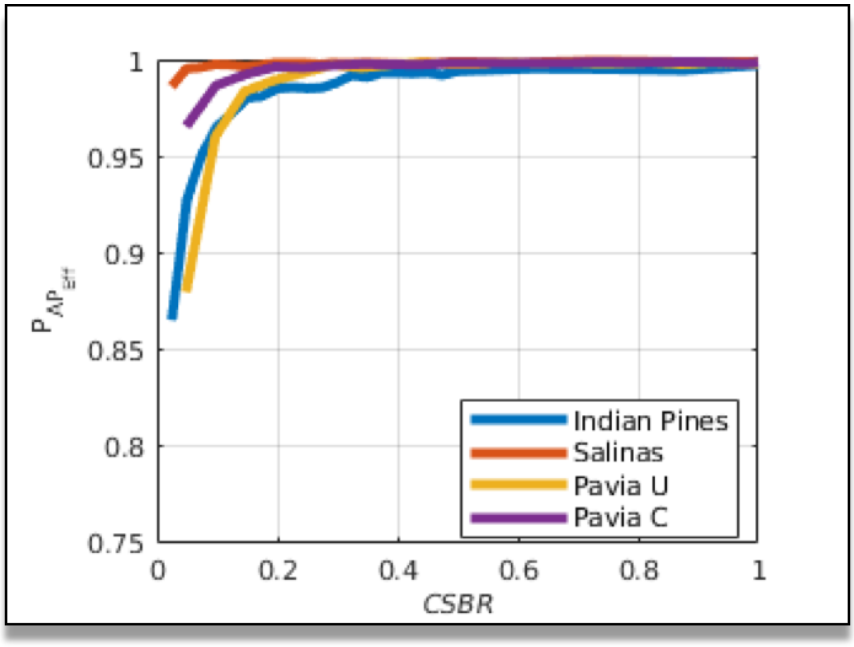

Figure 31: Average precision efficacy for all four images.

Conclusion

Hyperspectral image classification in the CSBD was explored in this chapter with the motivation of enabling low-cost, low-SWAP hyperspectral designs. A CSBD spectral-spatial classifier was proposed based on a RBF-SVM spectral classifier and an EPF spatial filter that is applied after pixelwise classification. A mathematically error analysis was performed for SVMs in the CSBD that showed classification is indeed possible under sufficient sampling conditions. The empirical analysis consisted of a set of experiments performed on four real hyperspectral images. Observing the individual class accuracies, it was noted that nearly all of the classes converge to full performance with a few number of CSBs (10-20) and performance is typically limited by a select

few classes. The results across two sensors and four images (two per sensor) were compared using the proposed algorithm efficacy and compressively sensed band ratio metrics. This comparison confirmed that scene complexity directly limits the maximum amount of compressed performance that can be obtained. The variability from scene to scene suggests that a method to predict or adaptively adjust the number of CSBs is needed and is the focus of chapter 4.

Chapter 4: Estimating A Measurement Bound for Compressed Hyperspectral Classification Via Feature Selection

Introduction

One of the practical challenges associated with implementing a compressed classification system, is deciding on an appropriate number of compressed bands to collect. As it was shown in Chapter 3, the required number of CSBs can vary across scenes and sensors. Selecting too few CSBs will result in poor classification efficacy. Conversely, selecting too many CSBs will result in wasted storage, increased bandwidth, and higher computational costs. An approach is desired that can estimate the optimal number of CSBs required to achieve high classification efficacy, while minimizing the total number of CSBs that are acquired. For convenience, the symbol, m_{opt}, is defined as the lower bound required to achieve near optimal performance. The term "optimal" is not strictly defined, since the exact meaning may vary based on the desired application.

In this chapter, the problem of selecting the appropriate number of CSBs is framed as a special case of a feature selection problem. Supervised approaches based on estimating m_{opt} directly from the training data, are explored. A brief overview of feature selection is provided and its applicability is discussed. Two different approaches are proposed based on the feature selection framework. Finally, a set of experiments are performed to evaluate the utility of the explored approaches.

Feature Selection

Feature extraction and feature selection are two well researched topics, within the machine learning and pattern recognition communities. While there are some overlapping themes between these two areas of research, it is important to clearly define them both. Ultimately, feature selection will be adopted as the underlying framework for estimating m_{opt}.

Feature Extraction

Feature extraction is defined here as an approach in which existing features, or raw data, are manipulated to produce new, more powerful features. This was done historically by projecting existing features into new spaces that are lower dimensional mixtures of the original features; such as principal component analysis (PCA) (Wold, Esbensen and Geladi 1987), linear discriminant analysis (LDA) (Bishop 1995), or canonical correlation analysis (CCA) (Hadoon, Szedmak and Shawe-Taylor 2004). More recently, deep learning approaches, such as deep belief networks (Boureau and Cun 2008), auto-encoding networks (Vincent, et al. 2008), and deep classifiers (Krizhevsky, Sutskever and Hinton 2012) have found great success in extracting features directly from the raw data. In many of these cases, the transformations depend on data-adaptive learning procedures. Given the signal-independent nature of the compressive sampling process, feature extraction techniques are not directly applicable and are not discussed any further.

Feature Selection

Feature selection is defined as an approach in which a subset of existing features are selected with the goal of removing redundancy or maximizing relevance. In a typical feature selection problem, there are L disparate classification features that must be chosen from, each of which are independent and have varying, unknown levels of discriminatory power. The challenge is to determine the appropriately sized subset of features and, specifically, which combination of features are the most effective for classification. This becomes an NP-Complete problem where each of the subsets must be searched exhaustively. Given the difficulty of this problem, it has been heavily researched and there are many existing algorithms that can be leveraged. A detailed review of feature selection algorithms can be found in (Tang, Alelyani and Liu 2014).

There are three general categories of supervised feature selection algorithms: filter methods, wrapper methods, and embedded methods. Filter methods are independent of a particular classifier and are based on characteristics of the dataset. They are typically fast to run because classifiers do not need to be trained on the various sets of features. Wrapper methods are a brute force approach in which a particular classifier is trained using various sets of features and performance metrics such as accuracy or precision are used to select the optimal feature set. This approach is typically very slow for general feature selection and often results in only a partial sampling of the parameter space. Finally, embedded methods are a combination of filter and wrapper methods, where data characteristics are used to limit the features and then wrapper methods are applied to ensure that maximum performance is achieved.

Embedded methods are often applied to achieve the accuracy of wrapper methods but with a reduced search time.

The task of estimating m_{opt} can be framed as a special case of a feature selection problem. For this task all of the features can be considered to have equal discriminatory power, given that the compressed bands are sampled incoherently. More specifically, we can state the nature of the incoherent sampling will cause all possible subsets of a fixed size to be approximately equivalent. This greatly reduces the complexity of the problem since it means that only the number of features needs to be determined rather than a specific subset. This reduction in the problem complexity removes a lot of the computational burdens that would otherwise limit the practicality of the more exhaustive techniques. In this chapter, discussion is focused on filter and wrapper methods. It will be shown that wrapper approaches are indeed computationally tractable, without requiring the compromises associated with an embedded variation.

Filter Method Approach

Filter methods are based on calculating measures of effectiveness for classification features, without directly applying them within a classifier. This offers the potential of identifying features that are universally effective for any type of classifier. However, this approach also raises the challenge of identifying appropriate measures of effectiveness for each feature. The choice of such a measure is often dependent upon the type of data, and relating these measures back to the ability to perform classification is not always straightforward.

In general, filter methods can be further classified into univariate and multivariate. In the univariate approach, each feature is considered one at a time,

making implementation simple and reducing the overall search space. A typical univariate filter approach consists of individually scoring each of the features and then choosing the highest ranked features for use with classification. The disadvantage to the univariate approach is that it is incapable of identifying the combined discriminatory power of multiple features, since they are only considered one by one. Additionally, univariate approaches are unable to recognize redundancy between the features.

The multivariate approach considers features in batches, providing the ability to solve both of the challenges faced by a univariate approach. Estimation of m_{opt} can be thought of as a multivariate filter problem, where the batch size, alone, is varied and the individual feature scores are irrelevant. The pixel training data, $\boldsymbol{R}_{Train} \in \Re^{N_R \times L}$, can be grouped by the class labels, $\boldsymbol{c} \in \Re^{N_R \times 1}$, and summarized into a single, representative class statistic, $\boldsymbol{Z} \in \Re^{P \times L}$, for each of the P classes. The class statistics can then be compressively sampled, $\boldsymbol{Z}_m = \boldsymbol{\Phi} Z^T$, with m CSBs, and a similarity measure can be calculated between all unique combinations of classes, $\boldsymbol{\zeta} \in \Re^{P(P-1) \times 1}$. This process can then be repeated for all values of m, and m_{opt} can then be selected by observing the point at which adding additional features (i.e. CSBs) no longer significantly affects the similarity measure. This general approach is summarized in Table 7.

Table 7: Feature Selection Filtering Algorithm

General Feature Selection Filtering Approach
Input: Training pixel vectors $\boldsymbol{R}_{Train} \in \Re^{N_R \times L}$, pixel class label vector $\boldsymbol{c} \in \Re^{N_R \times 1}$

1. Summarize each class into representative P statistic vectors, $\boldsymbol{Z} \in \Re^{P \times L}$, one for each class.
2. Choose an initial number of CSBs, $m = m_{min}$ and a CSB step size δ_m.
3. Randomly generate a compressive sampling matrix $\boldsymbol{\Phi} \in \Re^{L \times m}$.
4. Compressively sample the summary statistic vectors, $\boldsymbol{Z}_m = \boldsymbol{\Phi} Z^T$
5. Calculate a measure of similarity, $\boldsymbol{\zeta} \in \Re^{P(P-1) \times 1}$, between all unique pairs of class statistic vectors.
6. Set $m = m + \delta_m$ and repeat steps 3 through 5 until $m \geq L$.
7. Chose the value of m where the similarity measure converges.

Output: The estimated minimum number of CSBs: m_{opt}.

Class Statistics

Given the large number of training samples that are available for most HSI datasets, each class must be summarized in some manner to make computations tractable. A natural starting place is to consider the mean pixel vector over all pixels within a single class. The class mean pixel vector, $\boldsymbol{\mu}_p \in \Re^{L \times 1}$, for class p, is defined in equation (29), where the index, *i*, represents the *i*th spectral band. $\boldsymbol{R}_p \in \Re^{L \times N_p}$ is a matrix containing all the pixel vectors within the class and N_p denotes the total number of pixels in class p.

$$\mu_p(i) = \frac{1}{N_p} \sum_{j=1}^{N_p} \boldsymbol{R}_p(j, i) \tag{29}$$

Although it is not explicitly shown, the class statistics can also be derived for the compressed vectors, where the number of bands, L, simply becomes the number of CSBs, m.

Similarity metrics

Once each class has been reduced to a representative pixel vector, a measure of similarity[1], $\zeta(\boldsymbol{\mu}_1, \boldsymbol{\mu}_2) = \zeta_{12}$, can be calculated between any combination of two individual classes. This concept is most easily visualized as a similarity matrix, where, for each element, the index of the rows and columns correspond to the class index, and the value is the measure of similarity between those classes. The diagonal of the similarity matrix will necessarily be zero, since a pixel vector will always be perfectly similar to itself. Furthermore, if the similarity metrics are symmetric, $\zeta_{ij} = \zeta_{ji}$, then the matrix will be also be symmetric, resulting in $P * (P - 1)$ measures of similarity, where P is the total number of classes. A class similarity matrix is illustrated in Figure 32.

[1] The term similarity here is used to maintain generality; however, it should be noted that the metrics proposed in this work take the form of a distance measure. This is why the convention of 0 designating complete similarity (0 distance) has been adopted.

	C_1	C_2	...	C_P
C_1	0	ζ_{12}	...	ζ_{1P}
C_2	ζ_{21}	0		
...	...		...	
C_P	ζ_{P1}			0

Figure 32: Class similarity matrix.

To further reduce the similarity down to a single, per-class, metric, the average class similarity, $\bar{\zeta}_p$, for class p, is defined in equation (30). This metric provides a simple way of observing the effect of varying the number of CSBs.

$$\bar{\zeta}_p = \frac{1}{P-1}\sum_{i \neq p} \zeta_{ip} \tag{30}$$

Hyperspectral similarity metrics are needed to quantify the impact of increasing the number of CSBs. Since the dimensionality will be increasing, it is imperative that these metrics be normalized to allow for a relative comparison. Three different similarity metrics are specifically considered: normalized squared Euclidean distance (NSED), spectral angler mapper (SAM), and spectral information divergence (SID).

The Euclidean distance between class means is a suitable choice for a similarity metrics since it provides an easily understood measure of class separation. Unfortunately, the standard Euclidean distance is unbounded and will potentially grow monotonically as the number of dimensions are increased. To consider the Euclidean

distance as a tractable similarity metric, it must first be normalized to ensure bounded outputs. The normalized squared Euclidean distance measure is proposed as shown in equation (31). Here, $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ represent arbitrary pixel vectors, μ_1 and μ_2 represent the mean over the pixel vectors and $\|\cdot\|_2^2$ represents the squared L2 norm.

$$NSED(\boldsymbol{r}_1, \boldsymbol{r}_2) = \frac{1}{2} \frac{\|(\boldsymbol{r}_1 - \boldsymbol{r}_2) - (\mu_1 - \mu_2)\|_2^2}{\|\boldsymbol{r}_1 - \mu_1\|_2^2 + \|\boldsymbol{r}_2 - \mu_2\|_2^2} \tag{31}$$

Spectral angle mapper (Kruse, et al. 1993) is a metric that assumes that the spectral signature of hyperspectral pixel vectors are points lying in an L-dimensional space, where L is the total number of spectral bands. Spectral similarity is quantified as the angle between two vectors, or signatures, in the L-dimensional space. SAM is either reported as an angle, $\alpha(\boldsymbol{r}_1, \boldsymbol{r}_2)$, on the interval $\left[0, \frac{\pi}{2}\right]$, or as the cosine of the angle, $\cos \alpha(\boldsymbol{r}_1, \boldsymbol{r}_2)$, lying on the interval [0,1]. The inherent normalization of SAM, lends itself straightforwardly as a multivariate filter metric. The angle definition based on the zero-mean pixel vectors[2] is adopted as shown in equation (32).

$$SAM(\boldsymbol{r}_1, \boldsymbol{r}_2) = \cos^{-1}\left(\frac{(\boldsymbol{r}_1 - \mu_1) \cdot (\boldsymbol{r}_2 - \mu_2)}{\|\boldsymbol{r}_1\|_2 \, \|\boldsymbol{r}_2\|_2}\right) \tag{32}$$

Spectral information divergence (Chang 1999) is an information-theoretic approach to quantifying spectral similarity. Unlike the NSED or SAM which treat the

[2] This particular formulation of SAM has also been referred to as spectral correlation mapper (De Carvalho and Meneses 2000); however, for simplicity, the name SAM is maintained throughout the text.

pixel vectors as points in an L-dimensional space, SID views each pixel vector as a random variable and estimates differences between the distributions of the spectral bands. A normalized band probability is defined for a pixel vector, $\boldsymbol{r}$, as $p_{(j)} = \frac{r(i)}{\sum_{i=1}^{L} r(j)}$, where $j \in [1, L]$. Furthermore, the Kullback-Leibler (KL) divergence between two arbitrary pixel vectors, $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, is defined as $D(\boldsymbol{r}_1||\boldsymbol{r}_2) = \sum_{i=1}^{L} p_1(i) \log \frac{p_1(i)}{p_2(i)}$ and $D(\boldsymbol{r}_2||\boldsymbol{r}_1) = \sum_{i=1}^{L} p_2(i) \log \frac{p_2(i)}{p_1(i)}$, where the information measures are not symmetric. Finally, the SID is defined as the sum of both KL-divergences, as shown in equation (33). Note, that the SID is a symmetric metric since it accounts for the KL-divergence in both directions.

$$SID(\boldsymbol{r}_1, \boldsymbol{r}_2) = \sum_{i=1}^{L} p_1(i) \log \frac{p_1(i)}{p_2(i)} + \sum_{i=1}^{L} p_2(i) \log \frac{p_2(i)}{p_1(i)} \tag{33}$$

In the original form, SID cannot be readily used as a similarity metric, due to the unnormalized output of the KL-divergence. A simple adjustment can be made by adding a factor of $\frac{1}{L}$ to the SID definition, to account for the total number of bands. The resulting metric will be an average estimate of how correlated high probability events are between both pixel vectors, and more importantly, will provide a normalized upper bound on the metric.

Statistical Robustness

One of the unique aspects to using feature selection for estimation of m_{opt}, is the innately probabilistic nature of the random projections that occur during sparse

acquisition. Unlike wrapper methods which adaptively train in the CSD, the filter approaches are based on similarity measures that are fixed and are therefore potentially much more sensitive to the random projections. Given this probabilistic nature, it is important to assess how sensitive the similarity metrics will be to any particular random draw. To examine this further, the class mean pixel vectors from the Indian Pines image, were compressively sampled according to the model in equation (11). The number of CSBs were varied from 5 to 220. The average similarity metric defined in equation (30), was calculated between the Alfalfa class and all other classes. The similarity measures based on NSED, SAM and SID measure are shown in Figure 33, for 10 random trials. The average over all trials is also shown as the thicker black line.

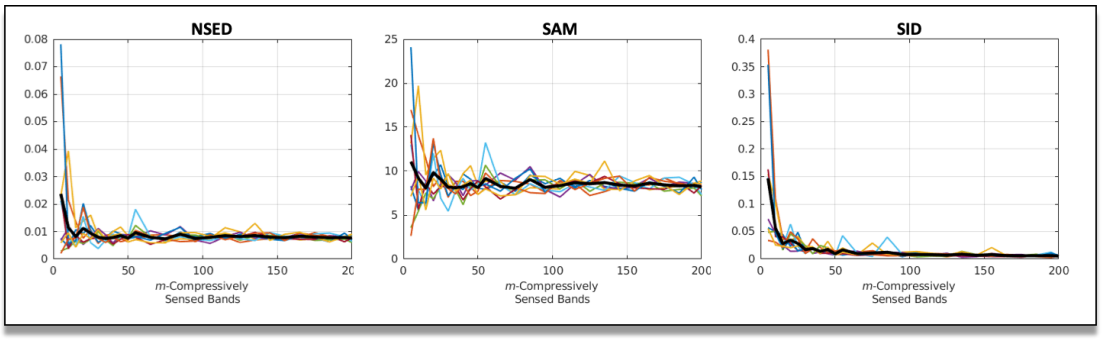

Figure 33: Average similarity for the alfalfa class from the Indian Pines image.

Each of the different similarity measure appears to asymptote relatively quickly, less than 50 CSBs, which shows good promise for utility in estimating a lower bound. However, there is clearly a lot of variance in each of the individual estimates due to the randomness of the sparse projections. This variance may potentially introduce some difficulty in developing an automated method for selecting the appropriate bound. One simple approach is to average multiple trials together and

operate on a single smoothed similarity measure, to provide a single hard estimate of m_{opt}. Unfortunately, for reasonable batch sizes, the variance in the compressed similarity metric manifests as perturbations in the bound estimate. To capture this variability, multiple estimates can be made and then be combined into a distribution of possible values of m_{opt}, providing a soft bound estimate. This approach is further illustrated in the experiment section.

Wrapper Method Approach

Given that only the number of compressed bands need to be determined, rather than a specific combination of bands, this problem is well-suited as feature selection wrapper method. Wrapper methods often show the best performance but tend to be intractable given the combinations of features that must be exhaustively searched. Fortunately, for estimating m_{opt}, all of the features can be considered to have approximately equal discriminatory power and all fixed size sets will be equivalent. This severely reduces the parameter space that must be searched, and therefore reduces the total number of times the classifiers must be trained.

Therefore a, straightforward and effective algorithm for estimating m_{opt} can be designed by observing the performance metrics of a classifier, as m is increased. A compressed classifier is trained for a range of CSBs and the classifier performance is evaluated at each step. The minimum number of CSBs can then be determined by observing the point at which the classifier performance asymptotes, or reaches a desired minimum performance level. The feature selection wrapper algorithm is summarized in Table 8.

Table 8: Feature Selection Wrapper Algorithm

Feature Selection Wrapper Algorithm
Input: A hyperspectral training dataset $\boldsymbol{R}_N \in \Re^{N \times L}$ and sampling matrix $\boldsymbol{\Phi} \in \Re^{Lxm_{max}}$ 1. Choose an initial number of CSBs, $m = m_{min}$ and a CSB step size δ_m. 2. Form a valid sampling matrix, $\boldsymbol{\Phi} \in \Re^{Lxm}$ and project all of the training pixels into the CSBD, $\boldsymbol{R}_{CSBD} = \boldsymbol{R\Phi}$. 3. Train the classifier $f_m(\boldsymbol{r})$ and calculate a performance metric. 4. Increment the number of compressed bands, $m = m + \delta$. 5. Set $m = m + \delta_m$ and repeat steps 2 through 4 until $m \geq L$. 6. Choose the value of m where the performance metric asymptotes or when a desired performance level has been met. **Output**: The estimated minimum number of CSBs, m_{opt}, and a set of trained classifiers $F = \{f_m(\boldsymbol{r}), f_{m+\delta}(\boldsymbol{r}), f_{m+2\delta}(\boldsymbol{r}), \dots, f_{m_{max}}(\boldsymbol{r})\}$.

Automatic Bound Selection

In the previous sections, two feature selection algorithms were proposed: however, a specific rule for determining convergence was not defined. A change-point detection algorithm is proposed for adaptively determining CSB convergence. The proposed change-point algorithm can be combined with the feature selection algorithms described in Table 7 and Table 8, to create fully automated approaches.

Optimal Partition Change-point Detection

Change-point detection is the task of determining a point within a data sequence at which the signal characteristics abruptly change. The field has been well researched and large number of approaches have been proposed (Basseville and Nikiforov 1993). Optimal partitioning (Lavielle 2005) is a global approach to change-point detection where all possible change points are simultaneously detected by minimizing a single cost function. A cost function, $J(\boldsymbol{h})$, is constructed as a function of a data sequence, $\boldsymbol{h} \in \Re^{1\times T}$. For a given candidate change-point, $\tau \in [1, T]$, the data are partitioned into a lower segment, $\boldsymbol{h}_{Lower} = \boldsymbol{h}[1, \dots, \tau]$, and an upper segment, $\boldsymbol{h}_{Upper} = \boldsymbol{h}[\tau + 1, \dots, T]$. A summary characteristic is calculated for segments and then summed together to form the associated cost for that particular change-point, $J(\boldsymbol{h}[\tau])$. The optimal change-point is then determined by choosing the value of τ that minimizes the cost function. The general form of the optimal partitioning algorithm is summarized in Table 9.

Table 9: Change-point detection via optimal partitioning

Optimal Partitioning Change-point Detection
Input: A data sequence $\boldsymbol{h} \in \Re^{1\times T}$ 1. Choose a candidate change-point, τ. 2. Divide the data sequence into lower, $\boldsymbol{h}_{Lower} = \boldsymbol{h}[1, \dots, \tau]$, and upper, $\boldsymbol{h}_{Upper} = \boldsymbol{h}[\tau + 1, \dots, T]$, segments. 3. Calculate the summary statistics for the lower and upper data segments. 4. Calculate the total cost, $J(\boldsymbol{h}[\tau]) = J_{lower}(\boldsymbol{h}_{Lower}) + J_{upper}(\boldsymbol{h}_{Upper})$.

5. Repeat steps 1 through 4 for all candidate change-points, $\tau \in [1, T]$.
6. Chose the value of τ that minimizes $J(\boldsymbol{h})$.

Output: The optimally partitioned change-point, τ_{opt}.

In general, the summary statistic calculated during step 3, can take any form; however, in this work the partition mean and partition standard deviation are specifically considered. To understand the differences between these approaches, two simple simulations were created, where the points of convergence where known. Each of the two simulations were modeled in an attempt to the represent the filter and wrapper methods. For the filter methods, the similarity metrics always tended to converge quickly; however, the overall variance seemed to reduce at a slower rate. For the classifier performance observed in Chapter 3, there was lower amount of variance, but the point of convergence spanned a larger range of values.

For both simulations, a logarithmic convergence was modeled. The data sequence, $\boldsymbol{h}$, was calculated using a limited log function of the form shown in (34). In this form, the scale factor, β, can be used to control how quickly the data sequence will converge. The limiting index, τ_{max}, corresponds to the point of convergence, for a unity scale factor. The true point of convergence, for the general case, can be calculated by multiplying the limiting index by the inverse of the scale factor, $\tau_{converge} = \frac{\tau_{max}}{\beta}$.

$$h(\tau) = \min(\log(\beta\tau), \log(\tau_{max})) \tag{34}$$

Figure 34 shows the results for the filter method with $\tau_{max} = [5,10,15,20]$, $\tau \in [1,200]$, and $\beta = 1$. To account for the slower reduction in variance, zero-mean, Gaussian noise with a linearly decreasing standard deviation, from 4 to 0.8, was added directly to the data sequence. The thin gray lines represent 10 individual random trials, and the thick green line is the mean over all trials. The dashed blue and red lines represent the optimal partitioning estimate for the partition mean and the partition standard deviation summary statistics, respectively. The black dashed line represents the true convergence index. For this case, the mean statistic provides a fairly robust estimate of the true convergence point. The standard deviation metric is clearly biased by the increased variance, and is unable to provide a robust estimate.

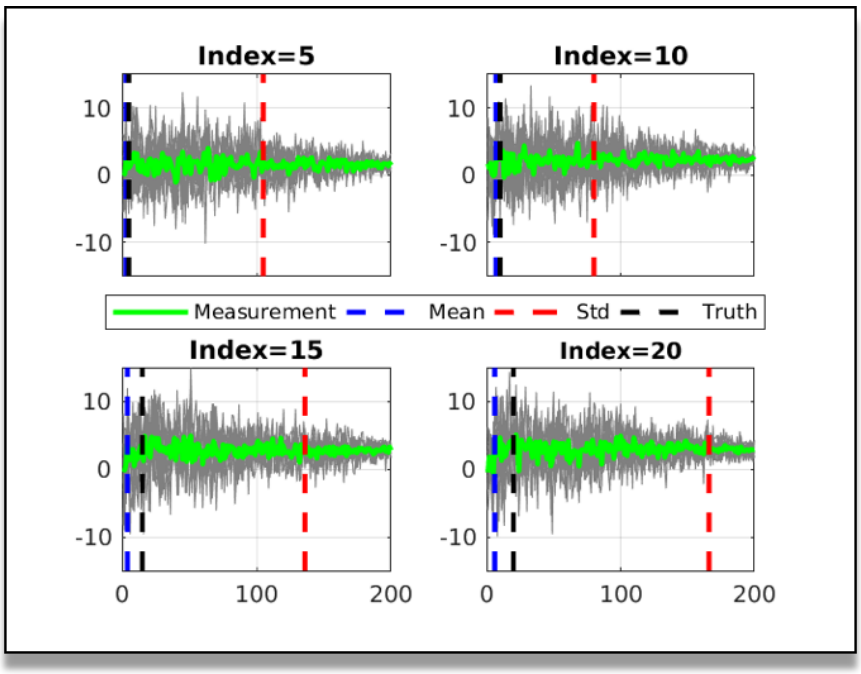

Figure 34: Optimal partitioning simulation for the filter method.

Figure 35 shows the results for a wrapper method simulation with $\tau_{max} = 100$, $\tau \in [1,200]$, and $\beta = [1, 1.5,2,4]$. To introduce some uncertainty, zero-mean, Gaussian

random noise, with a standard deviation of 0.8, was added directly the sequence. The thin gray lines represent 10 individual random trials, and the thick green line is the mean over all trials. The dashed blue and red lines represent the optimal partitioning estimate for the partition mean and the partition standard deviation summary statistics, respectively. The black dashed line represents the true convergence index. In general, both statistics underestimate the true convergence; however, they both appear to perform better for sequences that converge quickly. The standard deviation statistic consistently provides a better estimate than the mean statistic.

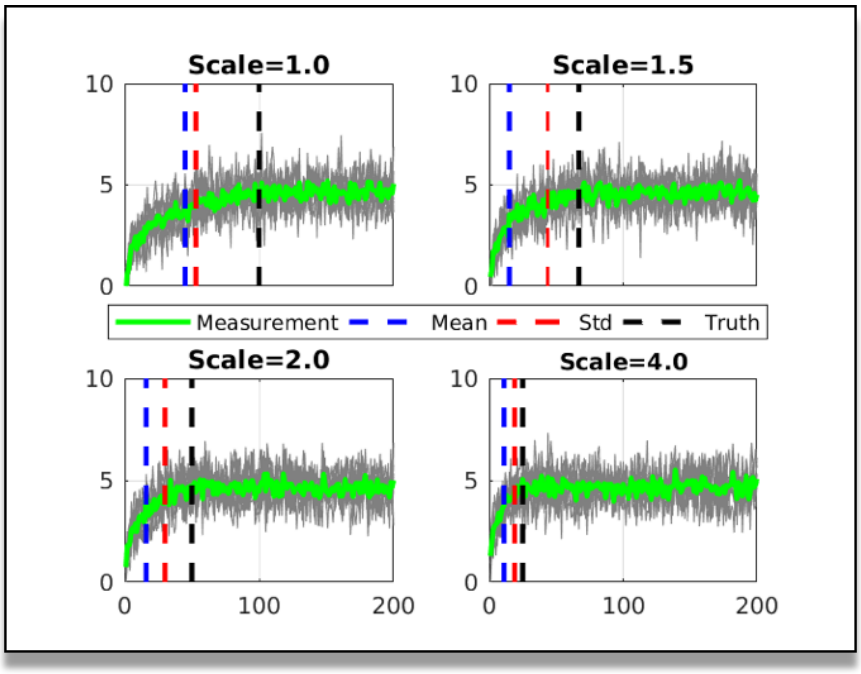

Figure 35: Optimal partitioning simulation for the wrapper method.

Based on these simple experiments, the optimal partitioning algorithm with a mean summary statistic will be used to automate the filter approach. Conversely, the optimal partitioning algorithm with a standard deviation statistic will be used to automate the wrapper approach.

Class-based Adaptive Compression

The ability to estimate a class-specific m provides a unique opportunity to further compress the acquired images before storing them to disk. For most hardware implementations, the number of CSBs that must be collected will be decided by the average or worst case bound among all classes. However, it is possible to further compress some of the pixels based on the individual lower bounds of each class. The classification map, $\boldsymbol{C}$, can be combined with a class specific CSB bound, $m_{opt}(i)$, to save each pixel with the appropriate number of compressed bands. More specifically, all pixels belonging to class i, will be stored with $m_{opt}(i)$ compressed bands.

To measure the effectiveness of this approach, a new metric must be introduced to account for the variable number of compressed bands among the pixels. The compressively-sensed band ratio (CSBR), introduced in Chapter 3, can be generalized into the compressively-sensed element ratio (CSER), where individual compressed elements are considered rather than compressed bands. In this context, the compressively-sensed element corresponds to a single compressed measurement and is analogous to the pixel element of the full band image, as defined in chapter 1. The CSER takes the form shown in equation (35), where $n(i)$ represents the number of pixels in the *i*th class.

$$CSER = \frac{\#\ of\ compressed\ elements}{\#\ of\ pixel\ elements} = \frac{1}{LN}\sum_{i=1}^{P} m(i)n(i) \qquad (35)$$

Furthermore, it can be easily shown that the CSBR is simply a special case of the CSER when $m\ (i) = m, \forall i$. In this case, m can be factored out of the summation resulting in

$\sum_{i=1}^{P} n(i) = N$. Applying these simplifications to equation (35), the CSBR can be derived from the CSER as shown in equation (36).

$$CSBR = \frac{m}{LN}\sum_{i=1}^{P} n(i) = \frac{mN}{LN} = \frac{m}{L} \tag{36}$$

Another convenient property of the CSER metric, is that it can easily be related to the storage requirements. This can be accomplished by multiplying an element count by the desired bit-depth. In general, the element count is calculated as shown the numerator in equation (35), $element\ count = \sum_{i=1}^{P} m(i)n(i)$. For the case of a single number of CSBs, this reduces to $m * N$. Similarly, for the case of the full band image, this reduces to $L * N$. As an example, assume there is an *2*-class image with $L = 100$ bands, and $N = 200{,}000$ pixels, that is stored with single precision (4 bytes per element). The required number of bytes to store the full band image would be $100 * 200{,}000 * 4\ bytes = 80\ MB$.[3] Further assume that a CSB lower bound has been estimated to be $m(1) = 5$ and $m(2) = 10$, and that exactly half of the pixels belong to each of the classes. The required number of bytes to store the compressed image would be $\left(5 * \frac{200{,}000}{2} + 10 * \frac{200{,}000}{2}\right) * 4\ bytes = 6\ MB$. The storage savings in this example would simply be $\frac{80-6}{80} = 92.5\%$. Alternatively, this can be directly calculated using the CSER, as $(1 - CSER) = \left(1 - \frac{7.5}{100}\right) = 92.5\%$. Thus the CSER provides a normalized metric for comparing multiple images and is directly related to storage requirements.

[3] Here a gigabyte (MB) is simply assumed to be 1,000,000 bytes rather than the formal 1,024 kilobytes.

Experiments

Filter Experiments

An experiment was performed to explore the effectiveness of the filter-based bound estimation algorithm. Given the probabilistic nature of the random projections, the experiment was setup in a Monte Carlo fashion, and repeated for 2,000 trials to provide representative statistics for each of the proposed similarity measurements. In each trial, the class mean vector was calculated for each class and then projected into the compressed domain following the model in equation (11). The number of CSBs, m, was varied from 5 to 55 in steps of 5 and then from 65 to L in steps 10. Average class similarities, from equation (30), were calculated at each value of m, based on NSED, SAM, and SID, as described in equations (31), (32), and (33), respectively. Finally, m_{opt} was estimated by applying the optimal partitioning algorithm, with a summary statistic based on the mean partition statistic.

The estimated lower bound from each trial has been summarized into a separate probability distribution for each of the similarity metrics: NSED, SAM and SID. The experiment results for both of the AVIRIS images, Indian pines and Salinas, are shown in Figure 36 and Figure 37, respectively. The NSED and SAM distributions show maximum probability with 5 CSBs or less, which is in line with the classification results shown in Chapter 3. The SID distributions show a peak probability between 5 and 10 CSBS; however, the distribution is very tightly bound. For all of the metrics, both images resulted in very similar distributions. Given that the types of image scenes and classes are very similar, this result is not surprising.

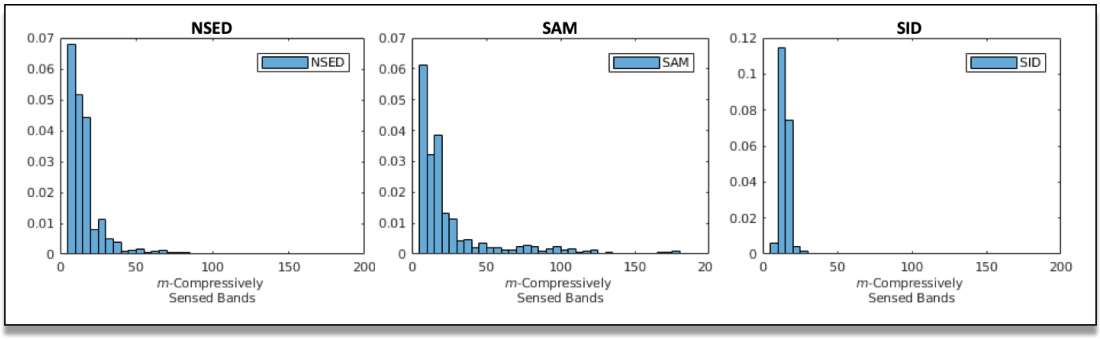

Figure 36: Indian Pines filter results for the mean-based optimal partition algorithm.

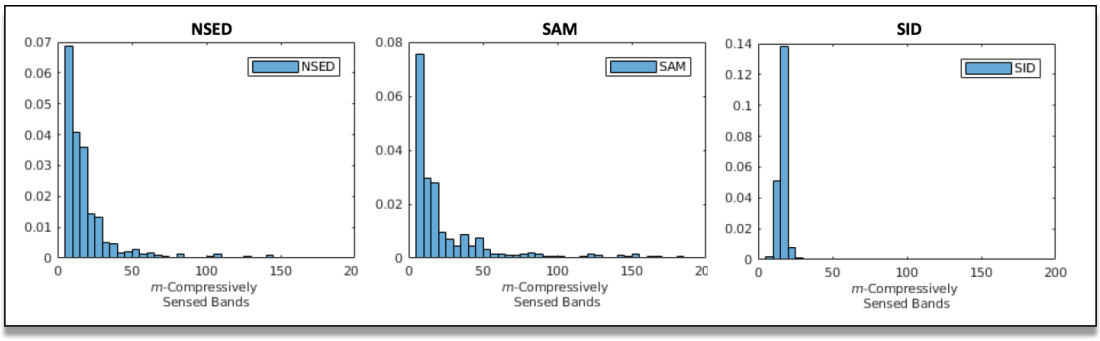

Figure 37: Salinas filter results for the mean-based optimal partition algorithm.

The results for the ROSIS images, Pavia University and Pavia Centre, are shown in Figure 38 and Figure 39, respectively. All of the similarity metrics show distributions with highest probabilities occurring within the first 15 CSBs. For NSED and SAM, the distributions show maximum probabilities with 5 or less CSBs, which is in line with the performance that was observed for nearly all of the classes in Chapter 3. Similar to the AVIRIS images, both ROSIS images resulted in very similar distributions. This is again attributed to the fact that both scenes contain similar content.

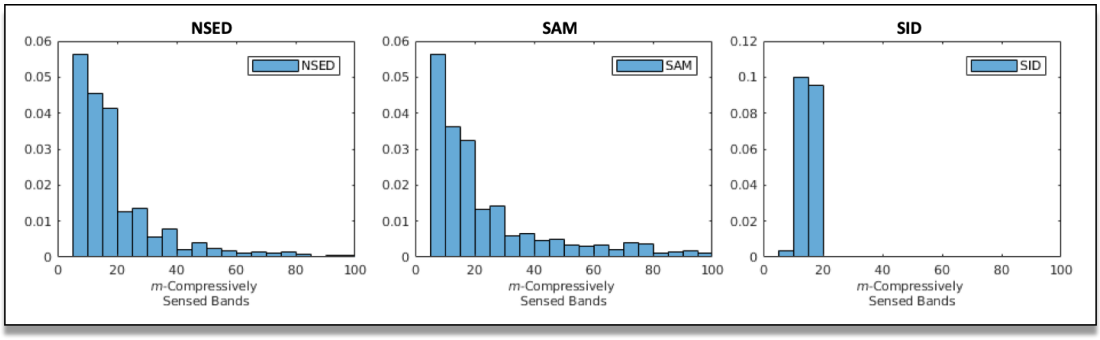

Figure 38: Pavia University filter results for the mean-based optimal partition algorithm.

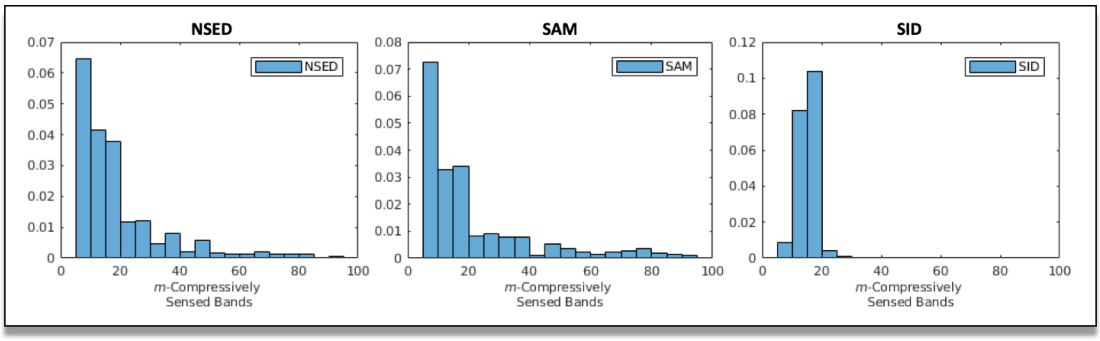

Figure 39: Pavia Centre filter results for the mean-based optimal partition algorithm.

In general, the NSED and SAM similarities gave comparable results. This is not surprising considering that they are both based on a geometric interpretation of hyperspectral similarity, while SID is based on a probabilistic one, making it more unique. The SID distributions showed lower amounts of variance, but were biased to be higher than what was observed in Chapter 3. It is also important to note that these results are all sensitive to the specific change-point algorithm that is implemented, as well as the averaging sizes that are used in the experiment.

Wrapper Experiments

The first consideration that should be made for a wrapper approach is to determine if the value of m_{opt} is consistent between the training data and the test data. That is, can the training data alone provide a robust estimate of m_{opt}. While it is well

known that the classification performance of the training dataset may potentially be biased compared to the test dataset, it turns out that m_{opt} is indeed unbiased. This is demonstrated empirically by comparing the average accuracy, P_{AA}, between the training samples and the test samples. In this case, the test samples represent the unobserved pixels that will be encountered in a real-world application.

Figure 40 shows an example of the training P_{AA} and test P_{AA} for the Indian Pines image scene. Notice that while there is approximately 2% difference between the training and test P_{AA}, however, the relative effect from varying the number of CSBs is consistent. Similarly, training and test comparisons for Salinas, Pavia University, and Pavia Centre are shown in Figure 41, Figure 42, and Figure 43, respectively. A strong correlation exists between the training and test P_{AA} for all of these image scenes as well. While they do show varying amounts of bias in the reported P_{AA}, the relative behavior as a function of the number of CSBs is again consistent.

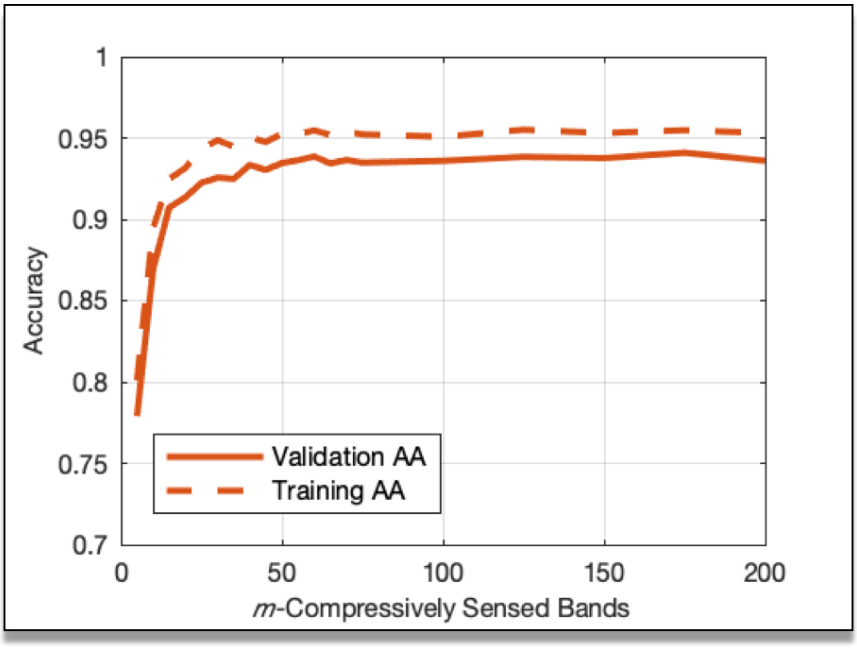

Figure 40: Comparison of training and test P_{AA} for Indian Pines.

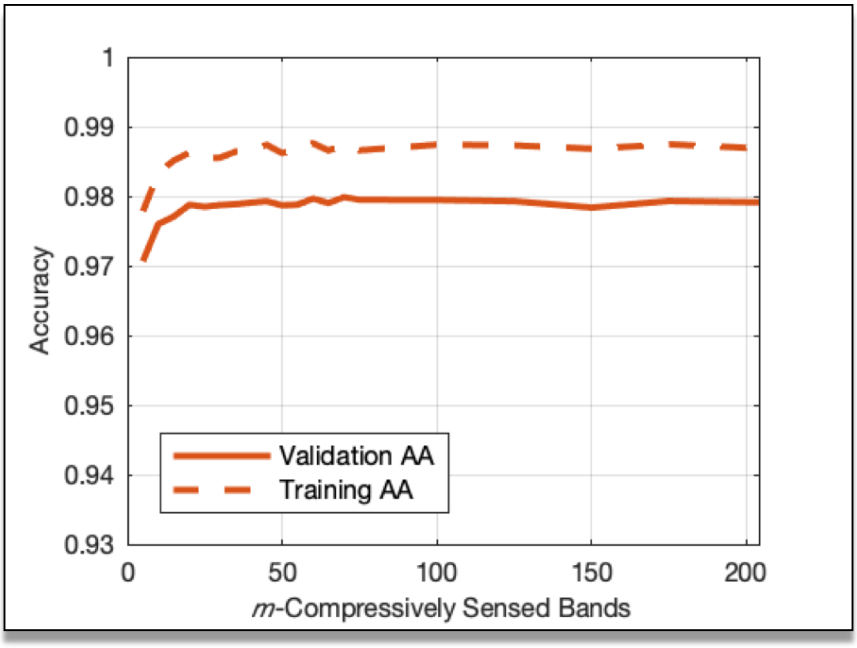

Figure 41: Comparison of training and test P_{AA} for Salinas.

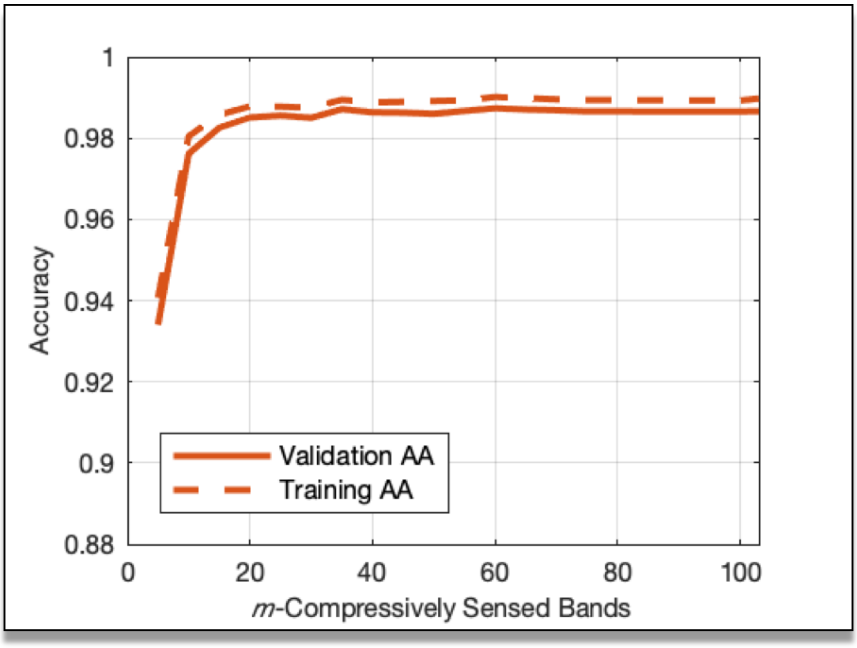

Figure 42: Comparison of training and test P_{AA} for Pavia University.

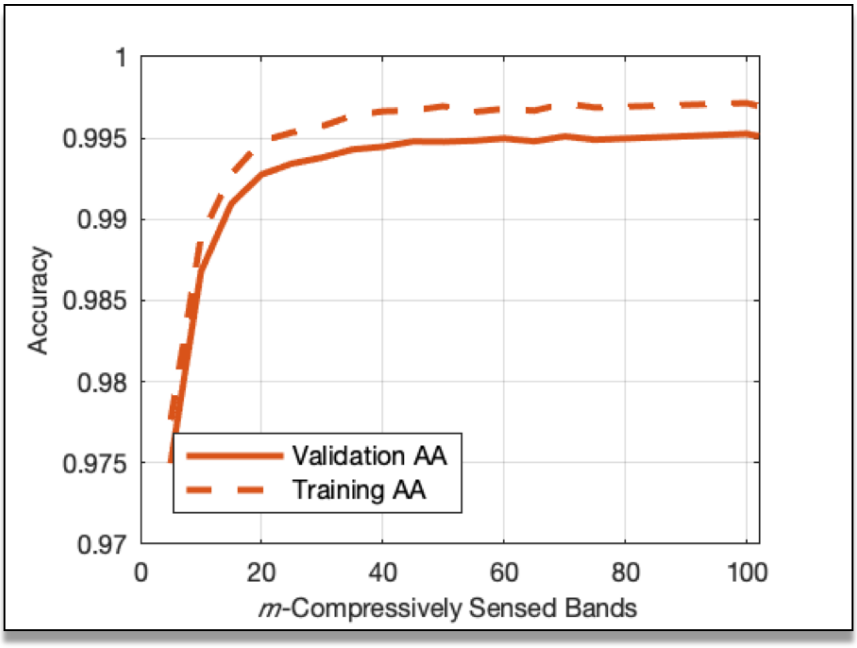

Figure 43: Comparison of training and test P_{AA} for Pavia Centre.

The correlation between the training and test performance can be easily quantified by using a measure such as the Pearson correlation coefficient, which is defined as $\frac{cov(a,b)}{std(a)*std(b)}$, where a and b are arbitrary vectors of equal length, cov

represents the covariance and std represents the standard deviation. The Pearson correlation coefficient between the training and test performance is equal to 0.9980, 0.9811, 0.9996, and 0.9996 for Indian Pines, Salinas, Pavia University, and Pavia Centre, respectively. It is quite clear for these images that the relationship between the training and test data are highly correlated, as a function of the number of CSBs, and that the training data can be used reliably. This an important realization because it allows for m_{opt} to be selected based solely on the training data, and ensures that it will be relevant for unseen data.

Class Specific Bounds and Adaptive Compression for the SVM Classifier

For a final comparison, a single $m_{opt}(i)$ was estimated for each class, from all of the images. The individual bound was then combined with the classifier results from Chapter 3, to determine what the resulting CSER and algorithm efficacy would be. For the filter approach, the value of m_{opt} corresponding to the expected value of the estimated distribution was selected. For the wrapper approach, the value m determined by the change-point detection algorithm, with a standard deviation summary statistic, was selected. For each of the experimental cases, the results have been summarized into a collection of tables. The efficacy of the individual class accuracy is shown for the all three filtering methods, as well as the SVM wrapper method. Additionally, the average accuracy, overall accuracy and the compressively-sensed element ratio are shown at the bottom of the table.

The results for both algorithms run on the Indian Pines image are summarized in Table 10. As should be expected, the wrapper method is able to select the value of *m* that produces the highest efficacy, for most cases, since it is specifically tuned to the

SVM classifier. In general, the filter methods do a very good job of estimating reasonable lower bounds for many of the individual classes. In this case, 11 of the 16 classes show equivalent performance between the filter and the wrapper methods. Amongst the filter methods, the NSED and SAM filters provide similar results, with the SAM filter showing the best agreement, for the SVM classifier. The SID filter results in the lowest bound estimates and can considered to be a more aggressive estimate. In regards to the CSER, the filter methods result in ratios that are 1.5 to 3 times smaller than the wrapper method.

Table 10: Indian Pines CSER and Efficacy Results -SVM

	NSED Filter	**SAM Filter**	**SID Filter**	**Wrapper**
	$P_{CA_{Eff}}$			
Alfalfa	0.99	**1.00**	0.99	**1.00**
Corn-notil	0.87	0.93	0.80	**0.95**
Corn-mintil	0.84	0.93	0.82	**0.95**
Corn	0.99	**1.00**	**1.00**	**1.00**
Grass-pasture	**0.99**	**0.99**	**0.99**	**0.99**
Grass-trees	**1.00**	**1.00**	**1.00**	**1.00**
Grass-mowed	0.98	**0.99**	0.96	0.98
Hay-windrowed	**1.00**	**1.00**	**1.00**	**1.00**
Oats	0.93	**0.99**	0.87	0.94
Soy-notil	0.94	0.96	0.93	**0.97**
Soy-mintil	0.91	0.93	0.89	**0.98**
Soy-clean	0.94	**0.98**	0.89	**0.98**
Wheat	**1.00**	**1.00**	**1.00**	**1.00**
Woods	**1.01**	**1.01**	**1.01**	**1.00**
Buildings	0.95	0.97	0.91	**0.98**
SS Towers	**1.00**	**1.00**	**1.00**	**1.00**
AA_{Eff}	0.96	**0.98**	0.94	**0.98**
OA_{Eff}	0.94	0.96	0.91	**0.98**
CSER	0.09	0.14	0.07	0.24

The results for the bound estimation algorithms run on the Salinas image are summarized in Table 11. In this case, the filter and wrapper methods show better

agreement, with the NSED achieving identical efficacy for all classes. The wrapper method results in full efficacy for all classes except for one. The CSER, however, produced from the wrapper method is again1.5 to 3 times higher than the filter methods.

Table 11: Salinas CSER and Efficacy Results - SVM

	NSED Filter	SAM Filter	SID Filter	Wrapper
	$P_{CA_{Eff}}$			
Broccoli 1	**1.00**	**1.00**	**1.00**	**1.00**
Broccoli 2	**1.00**	**1.00**	**1.00**	**1.00**
Fallow	**1.00**	**1.00**	**1.00**	**1.00**
Fallow Rough Plow	**1.00**	**1.00**	**1.00**	**1.00**
Fallow Smooth	**1.00**	**1.00**	**1.00**	**1.00**
Stubble	**1.00**	**1.00**	**1.00**	**1.00**
Celery	**1.00**	**1.00**	**1.00**	**1.00**
Grapes Untrained	**0.99**	0.98	0.98	**0.99**
Soil Vineyard	**1.00**	**1.00**	**1.00**	**1.00**
Corn	**1.00**	**1.00**	0.99	**1.00**
Lettuce 4 Week	**1.00**	**1.00**	**1.00**	**1.00**
Lettuce 5 Week	**1.00**	**1.00**	**1.00**	**1.00**
Lettuce 6 Week	**1.00**	**1.00**	**1.00**	**1.00**
Lettuce 7 Week	**1.00**	**1.00**	**1.00**	**1.00**
Vineyard Untrained	**1.00**	**1.00**	**1.01**	**1.00**
Vineyard VT	**1.00**	**1.00**	**1.00**	**1.00**
AA	**1.00**	**1.00**	**1.00**	**1.00**
OA	**1.00**	**1.00**	**1.00**	**1.00**
CSER	0.10	0.12	0.07	0.31

The results for the bound estimation algorithms run on the ROSIS images are summarized in Table 12, and Table 13. Similar to the AVIRIS images, both approaches showed good agreement, with the wrapper method being more conservative. The NSED and SAM filters resulted in very similar predictions and the SID filter again resulted in the most aggressive estimates.

Table 12: Pavia University CSER and Efficacy Results - SVM

	NSED Filter	SAM Filter	SID Filter	Wrapper
	$P_{CA_{Eff}}$			
Asphalt	0.98	0.98	0.97	**1.00**
Meadows	0.99	0.99	0.98	**1.00**
Gravel	**0.99**	**0.99**	**0.99**	**0.99**
Trees	**1.00**	**1.00**	**1.00**	**1.00**
Painted Metal Sheets	**1.00**	**1.00**	**1.00**	**1.00**
Bare Soil	**1.00**	**1.00**	**1.00**	**1.00**
Bitumen	**1.00**	**1.00**	**1.00**	**1.00**
Bricks	**1.00**	**1.00**	0.90	**1.00**
Shadows	**1.00**	**1.00**	**1.00**	**1.00**
AA	**1.00**	**1.00**	0.99	**1.00**
OA	0.99	0.99	0.99	**1.00**
CSER	0.18	0.21	0.13	0.27

Table 13: Pavia Centre CSER and Efficacy Results - SVM

	NSED Filter	SAM Filter	SID Filter	Wrapper
	$P_{CA_{Eff}}$			
Water	**1.00**	**1.00**	**1.00**	**1.00**
Trees	**1.00**	**1.00**	0.98	**1.00**
Asphalt	0.99	0.99	0.99	**1.00**
Self-blocking Brick	**1.00**	**1.00**	**1.00**	**1.00**
Bitumen	**1.00**	**1.00**	**1.00**	**1.00**
Tiles	**1.00**	**1.00**	**1.00**	**1.00**
Shadows	**1.00**	**1.00**	0.99	0.99
Meadows	**1.00**	**1.00**	**1.00**	**1.00**
Bare Soil	**1.00**	**1.00**	**1.00**	**1.00**
AA	**1.00**	**1.00**	0.99	**1.00**
OA	**1.00**	**1.00**	**1.00**	**1.00**
CSER	0.20	0.21	0.14	0.47

Conclusion

Two supervised approaches, based on feature selection, were presented for estimating an optimal number of CSBs. The first approach was a filter method based on observing the behavior of the average class similarity measure, as a function of the number of

CSBs. The filter approach resulted in a probability distribution of possible values of m_{opt}. A hard estimate could then be derived by selecting a characteristic of the distribution such as the maximum probability or the expected value. The second approach was a wrapper method based on observing the behavior of classifier performance on the training data. A hard bound was directly estimated by choosing the value of m were performance improvements saturated.

Both of the proposed algorithms were automated using a change-point detection algorithm called optimal partitioning. A simple experiment was performed to determine the appropriate summary statistic for each algorithm. The results showed that the mean statistic was more appropriate for the filter method and the standard deviation statistic was more appropriate for the wrapper method. The fully automated version of the algorithms were tested using the classification results presented in Chapter 3. An individual estimate of m_{opt} was made for each class and then used to perform class specific compression. Both algorithms were capable of successfully estimating adequate bounds for the number of CSBs. The wrapper method more consistently estimated bounds with higher efficacy; however, this was at the cost of a CSER of 1.5. to 3 times larger than the filter method. Among the filter methods, the NSED and SAM filters produced similar results, and the SID filter was slightly less conservative and typically resulted in the lowest CSER and efficacy.

Chapter 5: Compressed Progressive Band Hyperspectral Classification

Introduction

Progressive band processing (PBP) is a hyperspectral processing technique based on iteratively processing an image with full spatial information, but reduced spectral bands (Chang 2012). PBP algorithms offer the benefit of providing immediate feedback while the full image cube is being acquired and iteratively updating the algorithm output as more spectral information is received. Progressive band versions of many different algorithms have been proposed; such as, dimensionality reduction (Chang, Wang, et al. 2011), target detection (Wang, et al. 2013), anomaly detection (Chang, Li, et al. 2015), spectral unmixing (Chang and Liu 2014), and endmember extraction (Schultz, Hobbs and Chang 2014). These progressive band algorithms make it possible to transmit portions of hyperspectral images in smaller, more efficient packets, while still generating near real-time feedback that improves as more spectral bands are introduced.

In this Chapter, a compressed progressive band hyperspectral classification framework is proposed. The proposed approach is based on a general compressive sensing system, where CSBs are either collected or transmitted serially. In the first case, a snapshot hyperspectral imager (Hagen and Kudenov 2013) could be used to generate instantaneous images at full spatial and spectral resolution, but that are never digitally sampled. Rather, the digital sampling is limited to compressive band mixtures that are collected serially. This approach would greatly benefit remote or unmanned platforms

that require feedback from classification predictions to inform decisions or autonomy. In the second case, the compressed band hyperspectral images can be fully acquired on the sensor platform, and the progressive band classification can be performed off-sensor as the CSBs are received. This approach could be adapted to work with many different types of hyperspectral sensors and is analogous to the original PBP algorithms.

A progressive band framework is ideally suited for the compressively-sensed band images that have been presented in this work. Specifically, a compressed progressive band classifier (CPBC) is introduced that offers two advantages over the more traditional approach. First, the progressive nature provides immediate feedback without the need to collect the full number of CSBs. As it was shown in Chapter 3, classification of some classes converged with a very small number of classes, which means that even some of the earliest progressive iterations will provide extremely informative results. Second CPBP provides an unsupervised approach to determining the required number of CSBs directly from *in-situ* measurements, that can work with any compressed classifier. By observing the behavior of the classifier in between progressions, a stopping criterion can be developed to adaptively determine when a sufficient number of CSBs have been acquired.

There are three key aspects to the compressed progressive band classifier: the progressive classifier, the progression metrics, and the stopping criteria. The progressive classifier must be capable of iteratively processing a hyperspectral image for varying amounts of CSBs. The progression metrics refer to measures that are able to quantify the perceived change in performance between successive iterations. Finally,

the stopping criteria are definitive rules, based on the progression metrics, that determine when a sufficient number of CSBs have been processed. Each of this aspects are discussed in full detail, in the following sub-sections.

Progressive Classifier

To enable a progressive band approach, the classifier must be capable of processing images with a varying number of CSBs. To accomplish this, a straightforward and general approach is developed based on employing a set of classifiers, $F = \{f_m(\boldsymbol{r}), f_{m+\delta}(\boldsymbol{r}), f_{m+2\delta}(\boldsymbol{r}), \dots, f_{m_{max}}(\boldsymbol{r})\}$, that are functions of only the pixel vector, rather than a single classifier, $f(\boldsymbol{r}, m)$, that is a function of both m and the pixel vector. In this notation, $f_m(\boldsymbol{r})$ corresponds to a classifier that has been individually trained with m CSBS, where m is any integer on the interval $[1, m_{max}]$, and δ_m is an arbitrary step size. The full parameter space is captured when $\delta_m = 1$ and $m_{max} = L$; however, in practice a larger step size and a lower maximum number CSBs will likely be sufficient, for most cases. A short analysis is performed on the effects of δ_m and m in the experiment section.

To effectively train the set of classifiers, the compressive sampling matrix must be known *a priori*, and specifically included in the training process. Furthermore, the order of the columns in the sampling matrix must be preserved during both algorithm training and deployment. Both of these requirements can be easily satisfied by generating a random sampling matrix of maximum size, $\boldsymbol{\Phi} \in \Re^{Lxm_{max}}$, and then

selecting contiguous sub-matrices for each classifier[4]. The training data is then projected into the CSBD and the normal classifier training procedure is used. This process is repeated until the desired set of classifiers have been trained. The complete training procedure is summarized in Table 14. Note, that for simplicity, the training procedure has been described with a uniform step size; however, it in practice it is trivial to alter the training procedure to allow for a non-uniformly spaced number of CSBs.

Table 14: Progressive band classifier training procedure

Progressive Band Classifier Training Procedure
Input: A hyperspectral training dataset $\boldsymbol{R}_N \in \Re^{N \times L}$ and sampling matrix $\boldsymbol{\Phi} \in \Re^{Lxm_{max}}$ 1. Choose an initial number of CSBs, $m = m_{min}$ and a CSB step size δ. 2. Form a sub-matrix, $\grave{\boldsymbol{\Phi}} \in \Re^{L \times m}$, by selecting columns 1 to m from $\boldsymbol{\Phi}$ and re-normalizing along the columns. 3. Project all of the training pixels into the CSBD, $\boldsymbol{R}_{CSBD} = \boldsymbol{R}\grave{\boldsymbol{\Phi}}$. 4. Train the classifier $f_m(\boldsymbol{r})$. 5. Increment the number of compressed bands, $m = m + \delta$. 6. Repeat steps 2 through 5 until $m = m_{max}$. **Output**: A set of trained classifiers $F = \{f_m(\boldsymbol{r}), f_{m+\delta}(\boldsymbol{r}), f_{m+2\delta}(\boldsymbol{r}), \ldots, f_{m_{max}}(\boldsymbol{r})\}$

[4] It is important to stress that on a physical system, $\boldsymbol{\Phi}$ is implemented directly in the sampling hardware and must be coordinated with the classifier training procedure.

The presented approach has two desirable traits. First, this procedure does not require the classifier to be altered in any way. This allows for any classifier to be implemented without modification and for all of the standard classifier training procedures to be leveraged. Second, since the compressed classifier is not directly tied to the progression metrics or stopping criteria, the variations of the CPBC algorithm can be easily created by simply including a different set of classifiers. Moreover, it is even possible to train multiple sets of classifiers and employ classifier fusion techniques.

The main shortcoming to this approach is that for very complicated classifiers, such as deep neural networks, the required training time may become intractable. However, the free parameters δ_m and m_{max}, provide a mechanism for trading between training time and progressive band resolution. Additionally, it is possible to develop adaptive training procedures that use the classifier performance of the training data to update δ and to inform what m_{max} should be.

Progression Metrics

Progression metrics are required to measure the change in classifier performance between progressive iterations. Such metrics are essential for determining if classifier performance has converged or if additional CSBs should be collected. As it was shown in Chapter 3, a compressed classifier performance tends to asymptote at a particular number of CSBs, based on the individual scene complexity. A progression metric is desired that will correlate well with classification performance, since classification performance cannot be measured *in-situ*. In this section, the notion of a progressive band confusion matrix is introduced and a number of metrics are derived from it. A

connection is made between the derived metrics and the well-known Tanimoto Coefficient (TC). Finally, the progression metrics are cast into a probabilistic framework to provide additional insight into the subtleties of the proposed metrics.

Confusion Matrix Approach

The proposed approach leverages the specific nature of the classification problem to develop progression metrics that can be used for determining stopping criteria. For each progression of the progressive band algorithm, an estimate of class membership is produced for each of the current pixels. Let $\widehat{\boldsymbol{C}}(m)$ represent the class predictions for m -CSBs and $\widehat{\boldsymbol{C}}(m+\delta_m)$ represent the class predictions for $(m+\delta_m)$-CSBs, where δ_m is any positive integer and $m+\delta_m < L$. The relationship between the predicted class membership at subsequent iterations can be exploited to provide several measures of progression. The notion of a progressive band confusion matrix is introduced to aid in the exploration of the relationship between these class predictions.

Refer to the progressive band confusion matrix shown in Figure 44. The progressive band confusion matrix is similar to the standard classification confusion matrix shown in Figure 5; however, the rows and the columns both represent class predictions. Specifically, n_{ij} represents the number of pixels to belong to the $i'th$ class with m -CSBs and to the $j'th$ class with $(m+\delta_m)$-CSBs. The presented convention treats the predictions for m-CSBs as the confusion "predictions" and the predictions for $(m+\delta_m)$-CSBs as the confusion "truth"[5]. This convention is motivated by the fact that

[5] It is important to note that the terms "prediction" and "truth" are used here to be consistent with the traditional classification confusion matrix; however, class membership ground truth is not available *in-situ* and this approach is indeed completely unsupervised.

predictions made using more CSBs will, with high probability, provide more accurate estimates, thus they should be assumed to be more reliable.

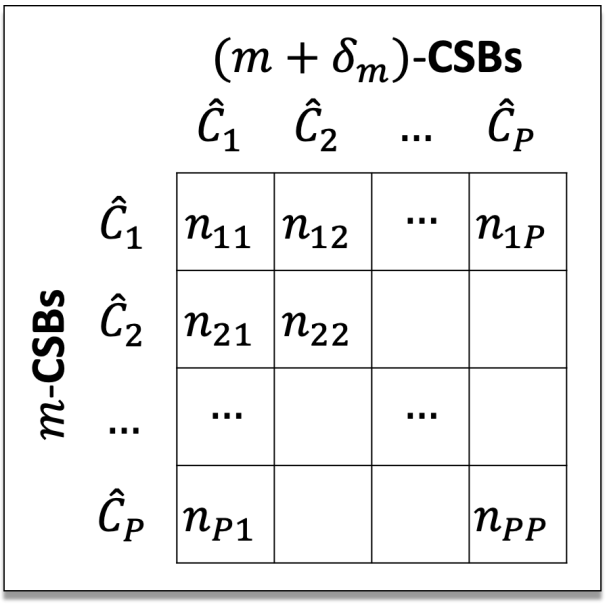

Figure 44: Progressive band confusion matrix.

Using the progressive band confusion matrix and following an analogous approach to classification performance, it is possible to define several progression metrics. Based on the class accuracy shown in equation (1), the progression accuracy is defined in equation (37). The progression accuracy, for the $p'th$ class, is the number of pixels predicted, in both iterations, to belong to class C_p divided by the number of pixels whose prediction change to C_p from a different class. Based on the classification precision shown in equation (2), the progression precision is defined in equation (38). The progression precision, for the $p'th$ class, is the number of pixels predicted, in both iterations, to belong to class C_p divided by the number of pixels that were previously predicted as C_p and are now predicted to belong to a different class. Finally, based on

the classification overall accuracy shown in equation (3), the overall progression accuracy is defined in equation (39). The overall progression accuracy is the number of pixels that share class predictions in both iterations divided by the total number of pixels.

$$P_{PA}(p) = \frac{n_{pp}}{\sum_{j=1}^{P} n_{jp}} \tag{37}$$

$$P_{PP}(p) = \frac{n_{pp}}{\sum_{j=1}^{P} n_{pj}} \tag{38}$$

$$P_{OPA} = \frac{\sum_{j=1}^{P} n_{jj}}{N} \tag{39}$$

Similar to the measure of classification performance, it is also convenient to consider the average progression accuracy, $P_{APA} = \frac{1}{P}\sum_{1}^{P} P_{PA}(p)$, and the average progression precision, $P_{APP} = \frac{1}{P}\sum_{1}^{P} P_{PP}(p)$.

Relationship to the Tanimoto Coefficient

An interesting connection can be made between the proposed progression metrics and a binary similarity metric referred to as the Tanimoto coefficient[6] (Rogers and Tanimoto 1960). In the context of a progression metric, the Tanimoto coefficient (TC), for class C_p, is defined in equation (40) as the ratio of the intersection of class

[6] This is also referred to as the Jaccard Index (Jaccard 1901).

predictions to the union of the class predictions, where $|\cdot|$ denotes the number of elements in the set. In this context, $\widehat{\boldsymbol{C}}_p \in \mathfrak{N}^{N_x \times N_y}$ is a Boolean matrix where a true value indicates that a particular pixel belongs to class C_p.

$$P_{TC}(p) = \frac{\left|\widehat{\boldsymbol{C}}_p(m) \cap \widehat{\boldsymbol{C}}_p(m+\delta)\right|}{\left|\widehat{\boldsymbol{C}}_p(m) \cup \widehat{\boldsymbol{C}}_p(m+\delta)\right|} \tag{40}$$

The Tanimoto coefficient, for a single class, can be written in terms of the progression confusion matrix, as shown in equation (41).

$$P_{TC}(p) = \frac{n_{pp}}{\sum_{j=1}^{P} n_{jp} + \sum_{j=1}^{P} n_{pj} - n_{pp}} \tag{41}$$

In this form, it is clear to see that P_{TC} is very similar to P_{PA} and P_{PP}, but with a normalization term in the denominator that incorporates all predictions from both iterations, simultaneously. Another very important difference is that P_{TC} is a symmetric measure, while P_{PA} and P_{PP} are not. More specifically, P_{TC} does not depend on the convention that is adopted to define the confusion matrix "truth", whereas, both P_{PA} and P_{PP} depend on it.

These differences can be further explored by posing the problem in a probabilistic framework. Substituting in the basic definitions for a conditional and a joint probability, $P_{TC}(p)$, can be re-written in terms of probabilities as shown in equation (42).

$$P_{TC}(p) = \frac{P\left(\widehat{\boldsymbol{C}}_p(m) \cap \widehat{\boldsymbol{C}}_p(m+\delta)\right)}{P\left(\widehat{\boldsymbol{C}}_p(m) \cup \widehat{\boldsymbol{C}}_p(m+\delta)\right)}$$

$$= \frac{p(\widehat{\boldsymbol{C}}_p(m), \widehat{\boldsymbol{C}}_p(m+\delta))}{P\left(\widehat{\boldsymbol{C}}_p(m)\right) + P\left(\widehat{\boldsymbol{C}}_p(m+\delta)\right) - P\left(\widehat{\boldsymbol{C}}_p(m), \widehat{\boldsymbol{C}}_p(m+\delta)\right)} \tag{42}$$

Similarly, the proposed progression accuracy and progression precision can also be re-written in terms of probabilities, as shown in equations (43) and (44).

$$P_{PA}(p) = \frac{P\left(\widehat{\boldsymbol{C}}_p(m), \widehat{\boldsymbol{C}}_p(m+\delta)\right)}{P\left(\widehat{\boldsymbol{C}}_p(m)\right)}$$

$$= \frac{P\left(\widehat{\boldsymbol{C}}_p(m+\delta) | \widehat{\boldsymbol{C}}_p(m)\right) P\left(\widehat{\boldsymbol{C}}_p(m)\right)}{P\left(\widehat{\boldsymbol{C}}_p(m)\right)} = \tag{43}$$

$$P\left(\widehat{\boldsymbol{C}}_p(m+\delta) | \widehat{\boldsymbol{C}}_p(m)\right)$$

$$P_{PP}(p) = \frac{P\left(\widehat{\boldsymbol{C}}_p(m), \widehat{\boldsymbol{C}}_p(m+\delta)\right)}{P\left(\widehat{\boldsymbol{C}}_p(m)\right)}$$

$$= \frac{P\left(\widehat{\boldsymbol{C}}_p(m) | \widehat{\boldsymbol{C}}_p(m+\delta)\right) P\left(\widehat{\boldsymbol{C}}_p(m+\delta)\right)}{P\left(\widehat{\boldsymbol{C}}_p(m+\delta)\right)} \tag{44}$$

$$= P\left(\widehat{\boldsymbol{C}}_p(m) | \widehat{\boldsymbol{C}}_p(m+\delta)\right)$$

Interestingly, for the adopted convention, the progression accuracy is the *a priori* conditional probability and the progression precision is the *a posteriori* conditional probability. That is, $P_{PA}(p)$ is conditioned on the previous iteration, while $P_{PP}(p)$ is conditioned on the most recent iteration. Comparatively, $P_{TC}(p)$ is the joint probability

conditioned by the union of the marginal probabilities. It is again obvious from this form that $P_{TC}(p)$ is indeed symmetric and will not depend on the particular convention that is adopted.

The Overall Tanimoto Coefficient

Finally, an overall Tanimoto coefficient (OTC) can be defined by considering all classes simultaneously, as shown in equation (45).

$$P_{OTC} = \frac{\sum_{p=1}^{P} \left| \widehat{\boldsymbol{C}}_p(m) \cap \widehat{\boldsymbol{C}}_p(m+\delta) \right|}{\sum_{p=1}^{P} \left| \widehat{\boldsymbol{C}}_p(m) \cup \widehat{\boldsymbol{C}}_p(m+\delta) \right|} \tag{45}$$

By recognizing that the denominator of this expression is equal to the total number of pixels, N, and that the intersection is the sum of the diagonal elements of the progression confusion matrix, $\sum_{j=1}^{P} n_{jj}$, it can then be shown that the overall progression accuracy and the overall TI are indeed equivalent.

$$P_{OTC} = \frac{\sum_{j=1}^{P} n_{jj}}{N} = P_{OPA} \tag{46}$$

Each of these metrics are explored in the experiment section and an analysis is performed to select the most appropriate measure for use with a stopping rule.

Stopping Criteria

Stopping criteria must be defined, based on the progression metrics, to indicate when enough CSBs have been collected. At first glance, this seems similar to the changepoint detection problem posed in Chapter 4, for determining the appropriate number of CSBs

a-priori. However, a key difference is that for progressive band processing, the stopping criterion must be causal, which was not the case for the changepoint detection algorithm. There are, however, a few specific considerations that can be made for the progressive band convergence. First, all of the progression metrics defined in the previous section are normalized between 0 and 1, which allows for target values to be directly included in a stopping rule. Second, in a noise free system, the progression metrics should all be monotonically increasing, which can further reduce some of the complexity in the stopping rules.

To identify convergence in progression, a set of rules based on the absolute progression metric value and the magnitude of the difference in successive iterations are proposed. The stopping rule is based on two user inputs: a progression metric threshold, τ_{PM}, and a difference threshold, τ_{δ}. The first threshold is simply the minimum progression metric that must be reached before convergence can be declared. This threshold is useful for preventing the algorithm converging at local minimum due to noise in the system. The second threshold represents the maximum change that is tolerated for a converged sequence. Values that are greater than this threshold are considered to have not yet converged. To make this threshold more intuitive, it is applied as a relative percentage of the current progression metric value. For a set of arbitrary progression metrics, P_{PM}, with m and $(m + \delta_m)$ CSBS, the stopping rule can be defined as shown in the following pseudo-code.

```
If (P_PM(m + δ_m) − P_PM(m)) > 0
    If P_PM(m) > τ_PM and (P_PM(m+δ_m) − P_PM(m)) / P_PM(m) < τ_δ
        Declare convergence
    Else
```

```
            Continue
Else
      Ignore negative values
```

Experiments

Setup

For these experiments, the experimental procedure presented in Chapter 3 was repeated to calculated the progression metrics. However, to properly simulate the *in-situ* nature of the progressive band processing, the results from a single trial are analyzed rather than the average of a collection of experiments. Similarly, the same number of training/testing samples, and classifier hyperparameters were maintained. In the interest of brevity, the performance metrics presented within this section are limited to the classification accuracy metrics, P_{OA} and P_{AA}; however, the results of the precision metrics were also in agreement. The experiments were again conducted on all four of the real-world hyperspectral images introduced in Chapter 1.

Training Step Size and Maximum Bound

One potential concern is that for classifiers with a computationally burdensome inference process, using small step sizes may become intractable. This raises the question of how fine the step size must be to adequately capture the performance progression. To answer this question, a simple experiment was conducted to explore the effects of step size on the classification progression. The spectral-spatial classifier was trained on Indian Pines image with a step size of 1 additional CSB at each iteration. Multiple subsets of results were then created by sub-sampling with step-sizes of 2, 5, 7, and 10, to simulate the process of using larger step sizes. Finally, the root-mean-

square (RMS) error of the overall accuracy was calculated between the step size of 1 and all of the larger step sizes. A simple linear interpolation was used for the sub-sampled results to align them with the finely sampled results. It should be noted that, for simplicity, only a single experiment was conducted and the P_{OA} achieved on the training data has been reported.

The step-size experiment results, for the Indian Pines image, are shown in Figure 45, and the RMS errors are tabulated and displayed as annotations. It is immediately obvious that the step size will have minimal impact on the progression metric. The larger step sizes adequately track the performance progression and result in RMS errors that less than 1% P_{OA}. This result is re-assuring, as it suggests that the step size can be chosen completely based on the desired fidelity and the available computational resources.

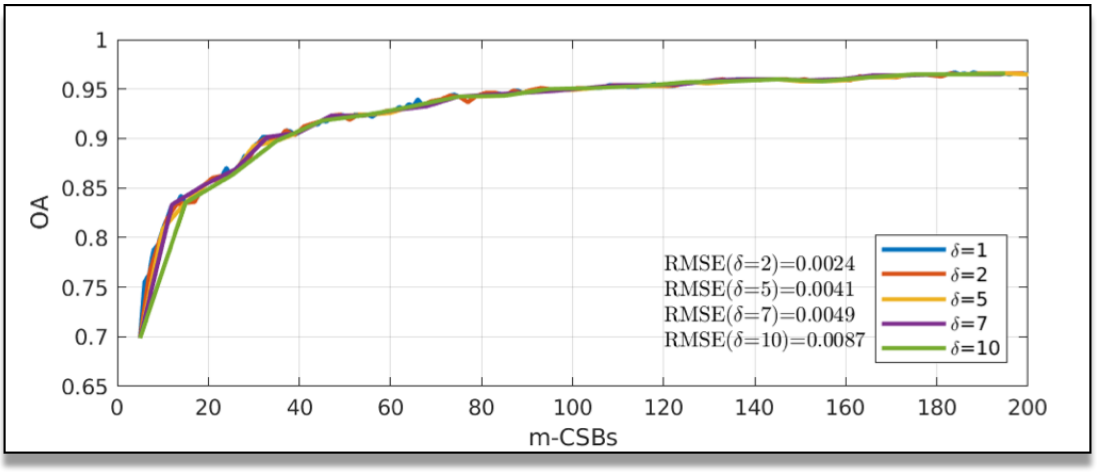

Figure 45: Step-size experiment for Indian Pines.

Progressive Band Classification

To fully illustrate the progressive classification sequence, a collection of classification maps and difference maps have been generate for each of the images. The classification maps display the current class predictions for each pixel, and the difference maps highlight the pixels that have changed between updates. The corresponding number of CSBs for each of the maps is annotated directly on the figure. For reference, the ground truth and the full band predictions are shown in the top left corner of each plot.

The progressive band classification sequence for the Indian Pines image is shown in Figure 46. The progressions from $m = 5$ to $m = 30$, $m = 35$ to $m = 60$, and $m = 65$ to $m = 90$ are grouped together in the top two rows, middle two rows, and bottom two rows, respectively. It is readily apparent from the difference maps that the class predictions rapidly change in the early iterations, with large clusters of pixels changing within the first 20 CSBs. While the majority of the pixels eventually converge, there are still small number of pixels that continue to change through 90 CSBs. Given that the Indian Pines is the more difficult of the test images and that the classification accuracy is limited to the lower 90% range, it is not unreasonable to expect for some pixels to continually change, even for the larger number of CSBs. Nevertheless, the classification maps, with as few as 20 CSBs, provide a useful view of the scene and could be used to inform real-time decision making for many potential applications.

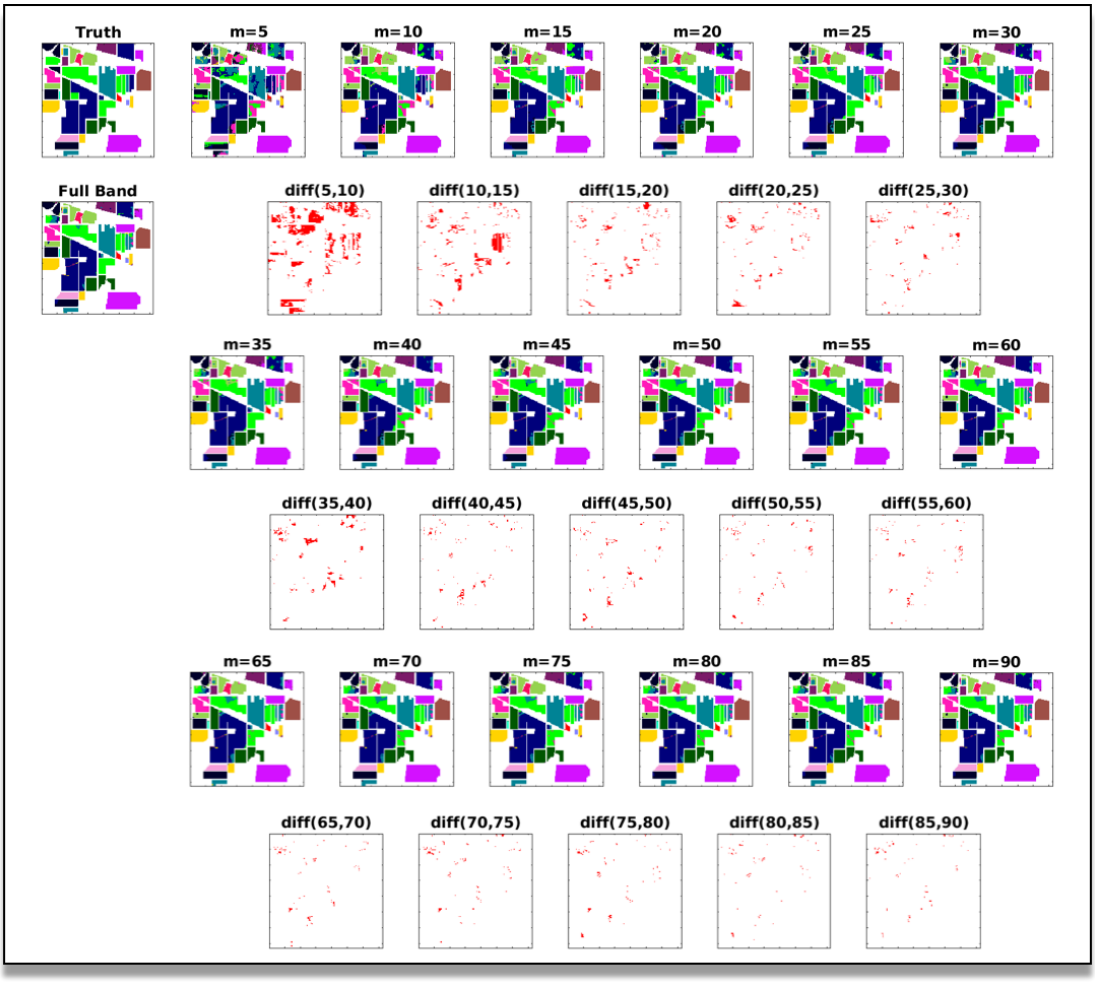

Figure 46: Indian Pines progressive classification sequence

The progressive band classification sequence for the Salinas image is shown in Figure 47. Even with only 10 CSBs, the class predictions provide an excellent approximation to the ground truth, and could potentially inform many real-time decisions. The largest change in class predictions occur between 5 and 10 CSBs, and the changes are predominantly limited to two specific classes. Similar to the Indian Pines image, the overall accuracy of the image is limited to ~96%, thus it is plausible that 4% of the pixels may be subject to fluctuations if they are close to the class boundaries.

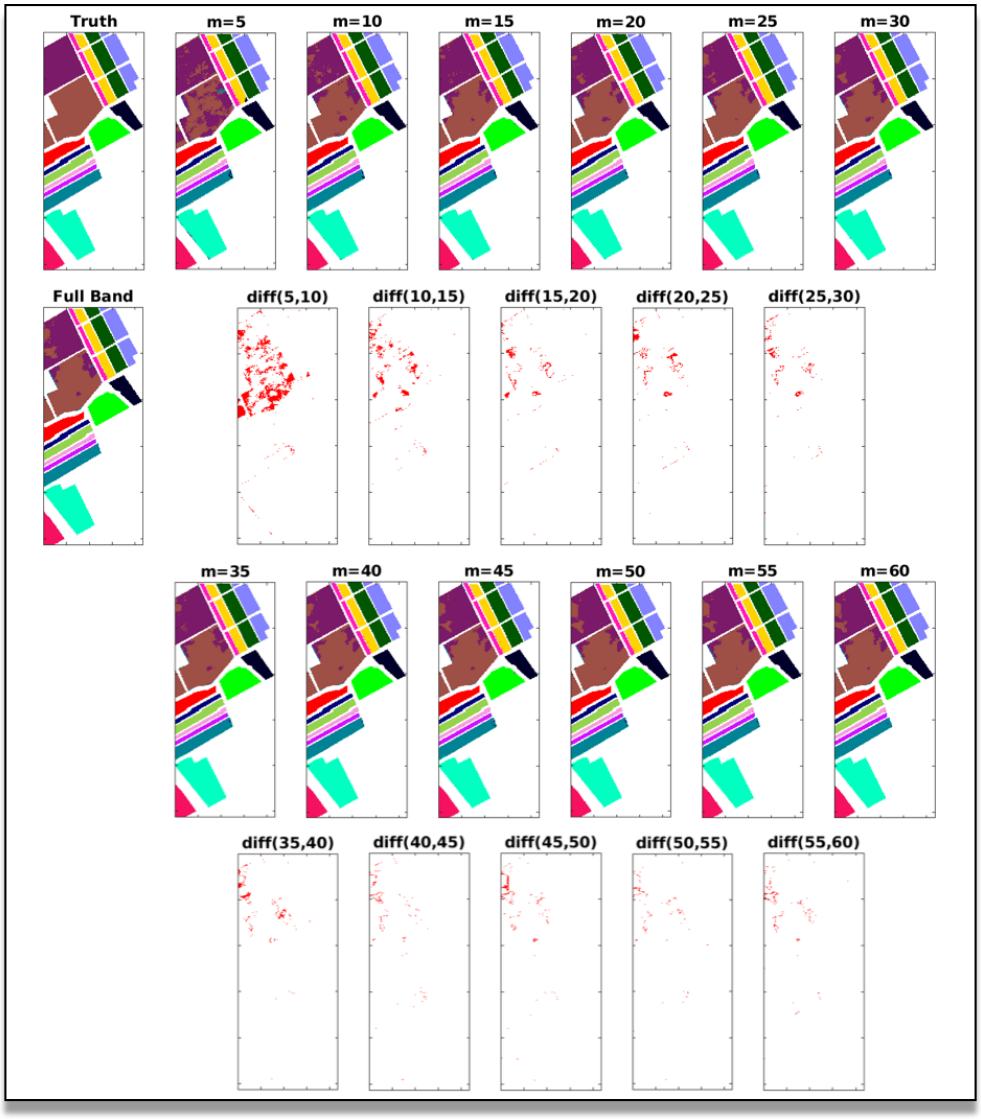

Figure 47: Salinas progressive classification sequence

The progressive band classification sequence for Pavia University is shown in Figure 48. Similar to Indian Pines, there multiple large clusters of pixels that change within the first 20 CSBs; however, the class predictions begin to converge around 30 CSBs and show minimal changes beyond that point.

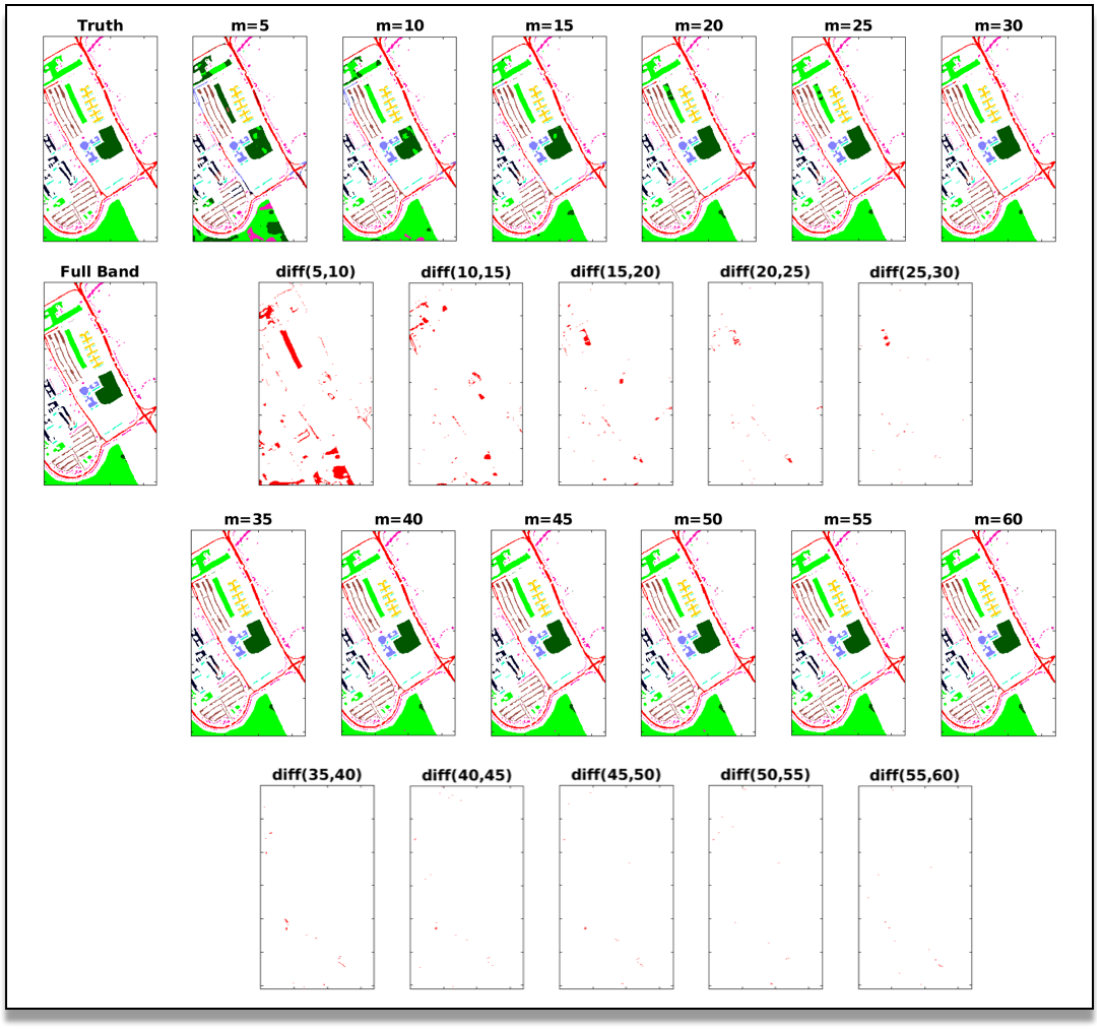

Figure 48: Pavia University progressive classification sequence

The progressive band classification sequence for Pavia Centre is shown in Figure 49. This image converges faster than all of the other three images. Even with the minimum of 5 CSBs the classification map looks nearly identical to the ground truth. The difference maps show minimal changes as the number of CSBs is increased.

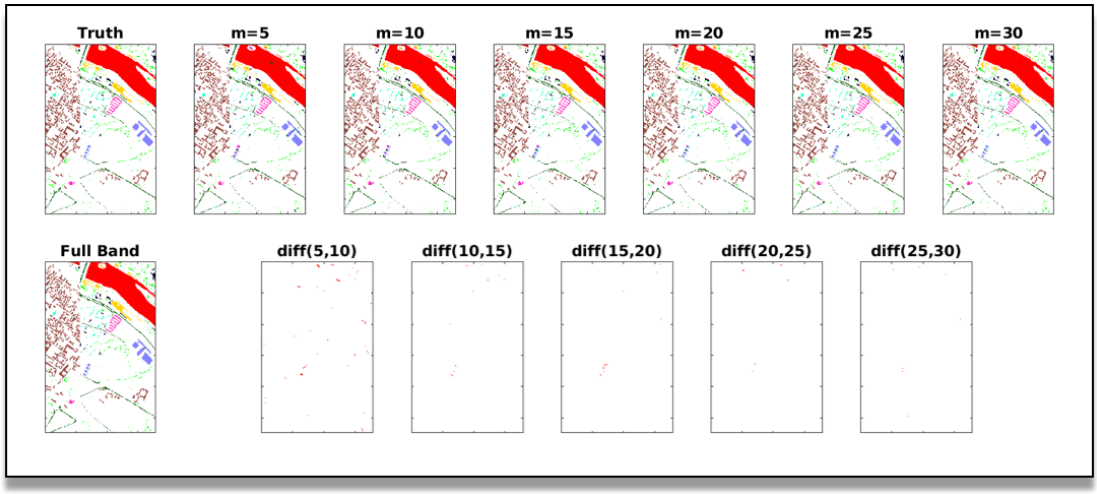

Figure 49: Pavia Centre progressive classification sequence

For each of the test images, the progressive classification sequence shows that there is large amount of useful information available, even with as few as 20 CSBs. The current class predictions and knowledge of which pixels are changing between iterations can be coupled to inform many different real-time decisions. The difference maps could also be used *in-situ* to isolate a sub-set of pixels for further analysis or for transmission in a remote system.

Progression Metric Experiment

One of the most important tasks of these experiments was to quantify the effectiveness of the proposed progression metrics as indicators of performance convergence. More simply put, how reliably could the progression metrics be used to determine when enough CSBs have been collected *in-situ*. In this section, the classification accuracy metrics, P_{OA} and P_{AA}, are compared with the progression metrics. The Pearson correlation coefficient introduced in Chapter 4, was again used to quantify the utility of the progression metrics. For each of the plots, the solid blue and red lines represent

the P_{OA} and P_{AA}, respectively. The dashed lines represent the various progression metrics: overall progression accuracy (yellow), average progression accuracy (purple), average progression precision (green), and overall Tanimoto coefficient (cyan). Furthermore, the Pearson coefficient is tabulated, for each combination of accuracy and progression metric, and is annotated directly on the plots.

There are a few notes worth mentioning regarding the background pixels. First, excluding the background pixels in the progression metrics will result in a more optimistic estimate of performance, with a higher correlation between the classifier performance. This can be understood intuitively by recalling the effect that the background pixels had on the precision metrics. Since many of the background pixels tend to be mixtures of the other classes, this can create some instability in the progression metric as the assigned label will tend to jump around more for the background pixels. Second, it is important to recognize that some additional training techniques could potentially be employed to improve the classifiers robustness to the background pixels. This line of effort is adjacent to the central thesis of this work and has been left for future research.

Figure 50 shows a comparison between the classification accuracy and the progression metrics, for the Indian Pines image. For both the cases with and without background pixels, all of the progression metrics correlate very well with the classifier performance. There is a notable turning point in all of the curves around 25 CSBs and then a more gradual increase until about 100 CSBS were the curves converge. The, the progression metrics result in a correlation coefficient greater than or equal to 0.94 and a maximum correlation of 0.99.

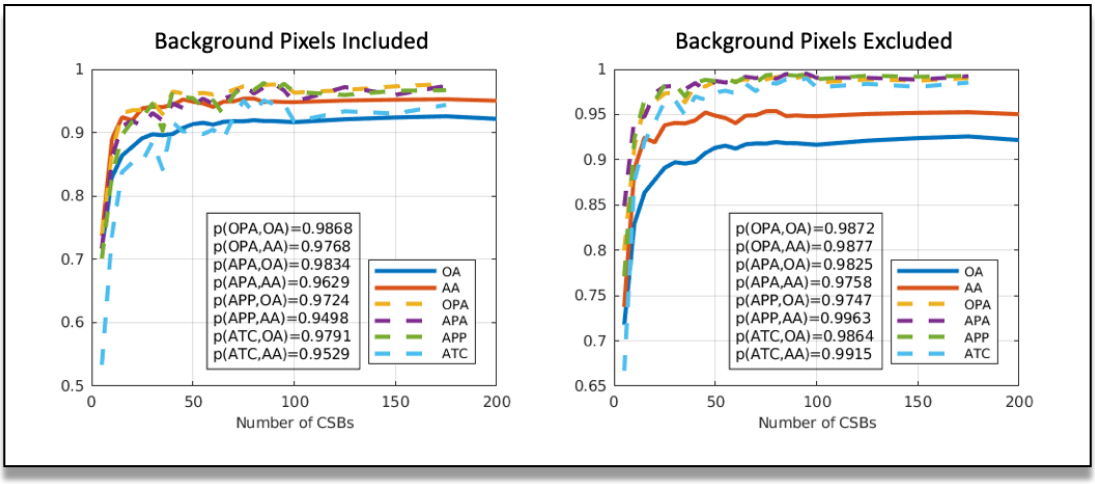

Figure 50: Indian Pines progression metrics.

Figure 51 shows a comparison between the classification accuracy and the progression metrics, for the Salinas image. In this case, there is a stark difference between including and excluding the background pixels. When the background pixels are excluded, the progression metrics show relatively high correlation with performance. However, when background pixels are considered, the correlation drops from 0.9 to 0.65. Fortunately, there is still an obvious changepoint around 40 CSBs, where the progression metrics asymptote. While a progressive band algorithm based on this particular classifier, will likely overestimate the total number of CSBs, it will still converge at a value less than 25% of the total number of bands.

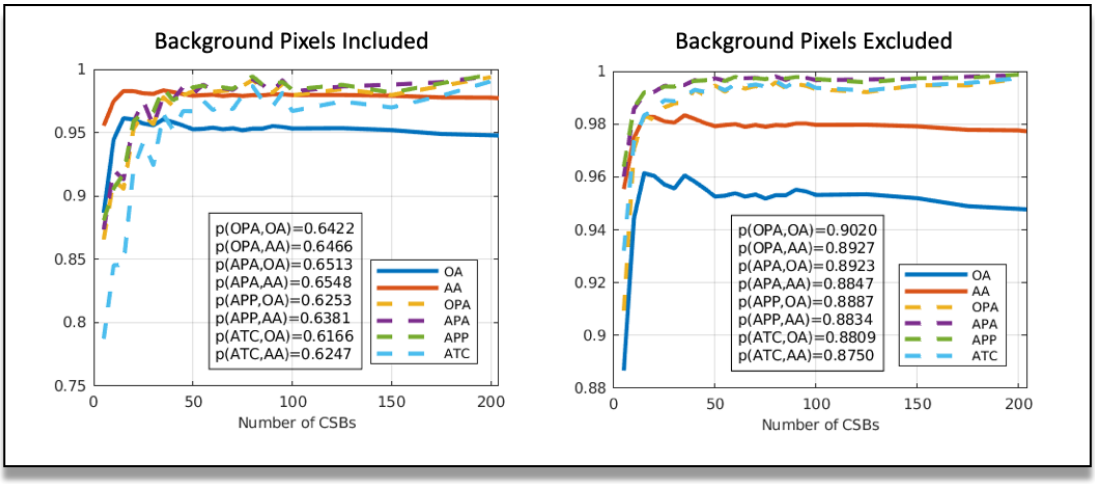

Figure 51: Salinas progression metrics.

Figure 52 shows a comparison between the classification accuracy and the progression metrics, for the Pavia University image. This is another example of where background pixels negatively impact the performance the progression metrics. Interestingly, this also the image that shows the largest discrepancies between the different progression metrics. When the background pixels are included, there is a difference of 0.22 between the lowest and the highest correlation coefficients. In general, correlation is better with the average accuracy for all metrics.

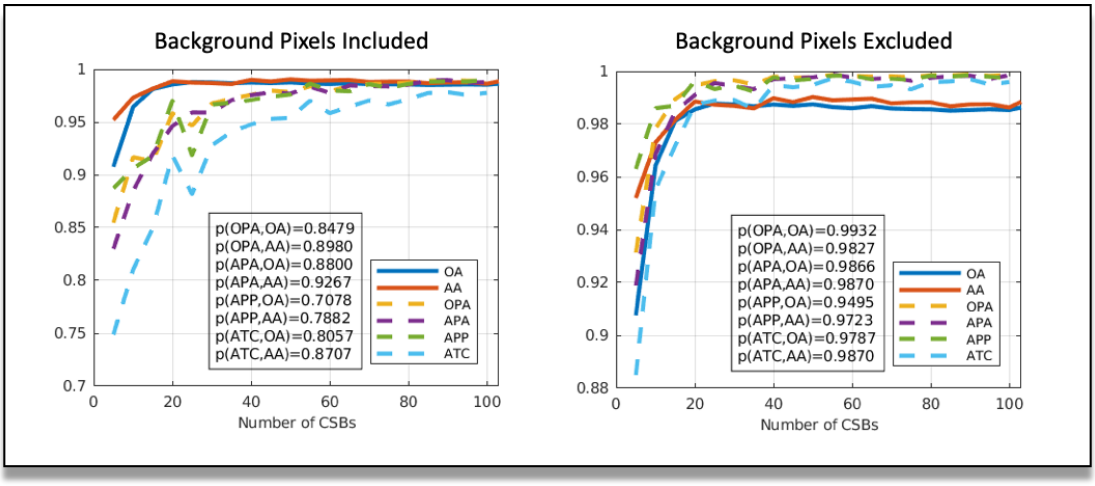

Figure 52: Pavia University progression metrics

Finally, Figure 53 shows a comparison between the classification accuracy and the progression metrics, for the Pavia Centre image. In this case, the reported correlation coefficients are reasonably high (>0.9) even if background pixels are included. However, it is still apparent that the turning point with background pixels is close to about 20 CSBs higher than the turning point in the performance metrics.

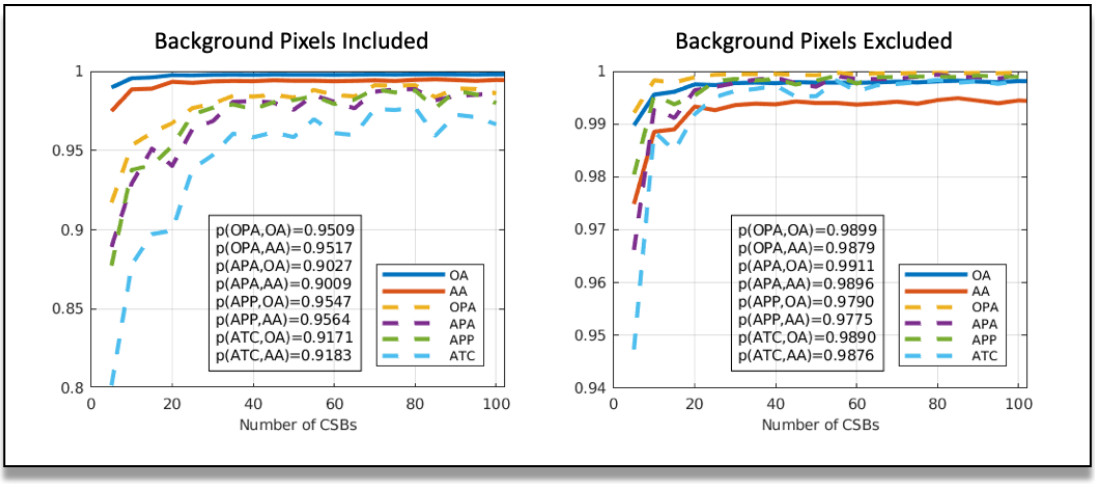

Figure 53: Pavia Centre progression metrics

Stopping Criteria Experiment

As a final test, the proposed stopping rule was applied to the progression metrics calculated in the previous section. The difference threshold was held constant at 0.01 and the progression metric threshold was varied from 0.90 to 0.99. The resulting CSBR, $P_{OA_{Eff}}$, and $P_{AA_{Eff}}$ are reported for each of the thresholds, and displayed as bar plots. For the cases where the algorithm failed to converge (i.e. stopping criteria were never satisfied), values were simply excluded from the plot. The experiment was conducted both with and without the inclusion of background pixels.

The results for each of the images, without including background pixels, are shown in Figure 54, Figure 55, Figure 56, and Figure 57. In general, the stopping criterion performs exceptionally well for all of the proposed summary progression metrics. For three of the four images, an efficacy of nearly exactly 1 is obtained, while maintaining a CSBR less than 0.25. For the fourth image, an efficacy of 0.95 is achieved with the same CSBR.

The algorithm also appears to be fairly robust to the specific choice of the progression metric threshold. Increasing the threshold does not result in an overestimate of the number of CSBs. There was a single case, for the Indian Pines image, where the algorithm failed to converge with the P_{ATC} and $\tau_{PM} = 0.99$. This was due to the fact that P_{ATC} never reached a value of 0.99.

Regarding the choice of a particular progression metric, each of the proposed progression metrics provide fairly similar results. There were no consistent trends across all of the images that would suggest one metric to be better than another.

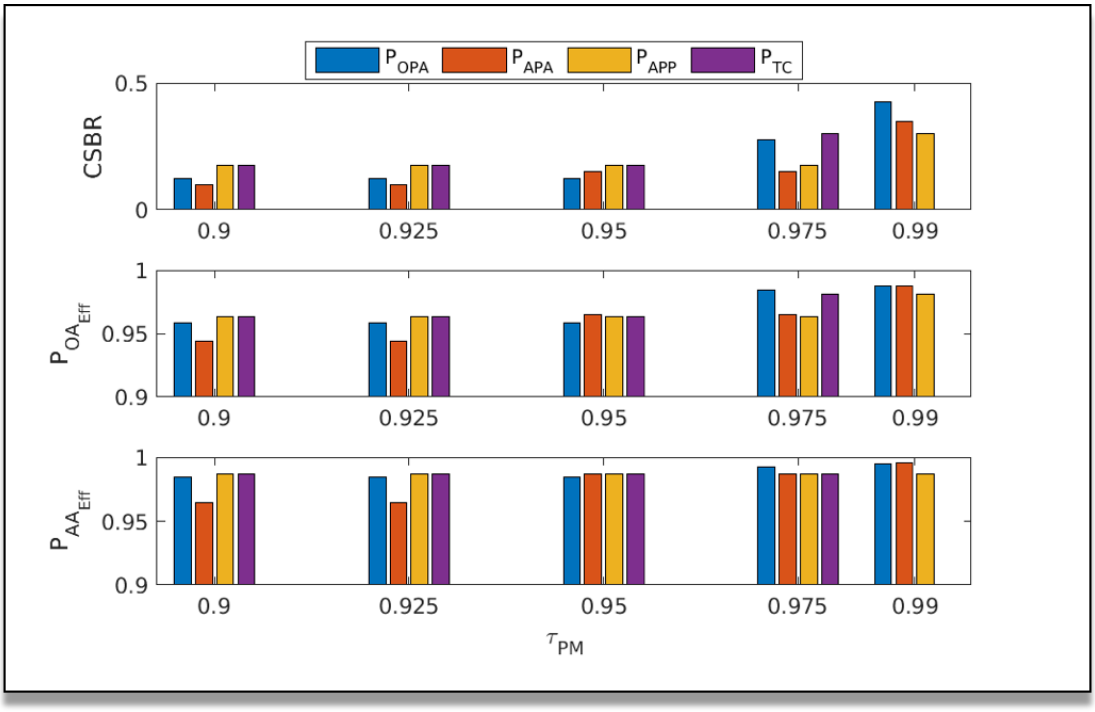

Figure 54: Indian Pines progressive band statistics without BKG pixels

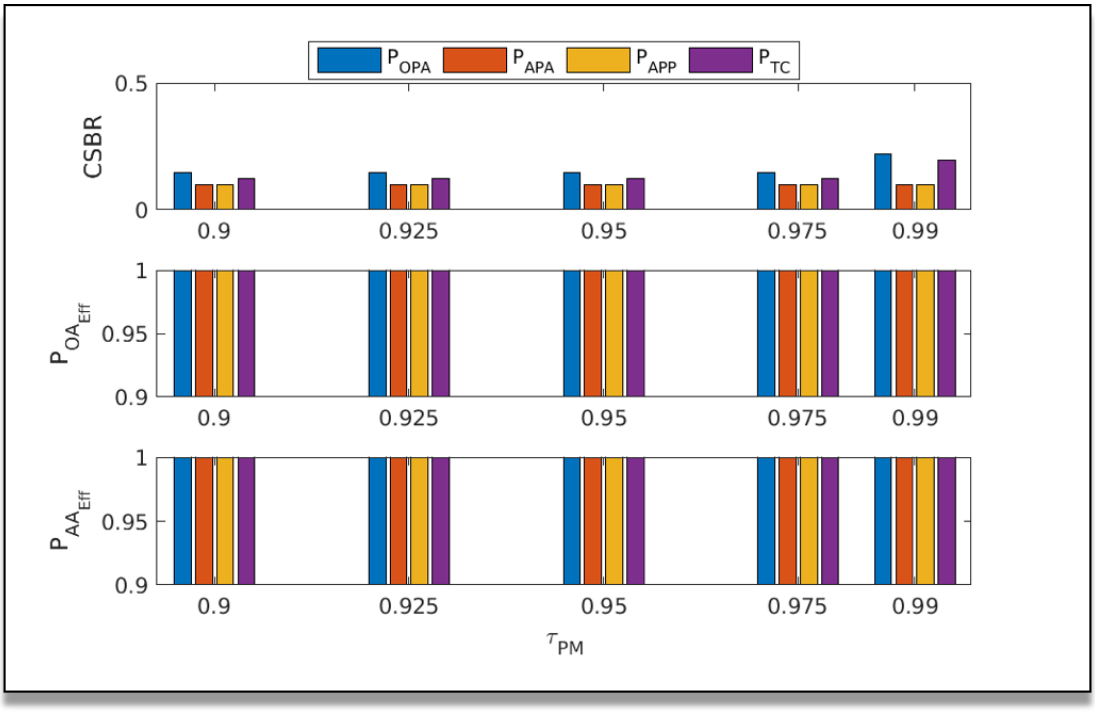

Figure 55: Salinas progressive band statistics without BKG pixels

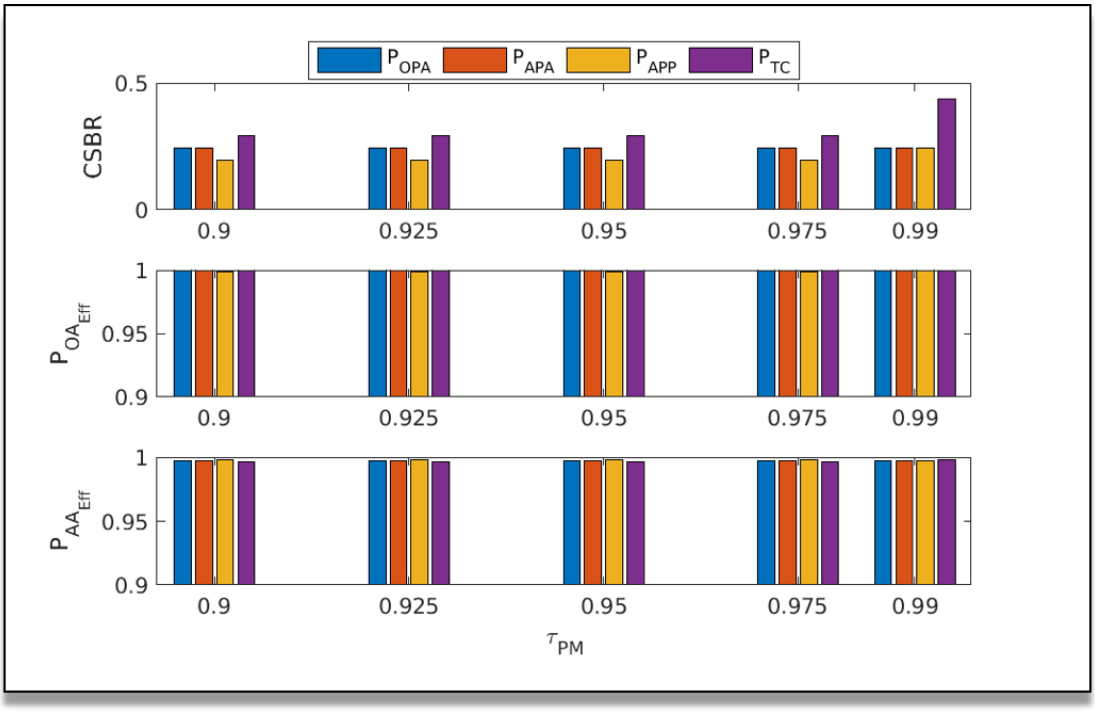

Figure 56: Pavia University progressive band statistics without BKG pixels

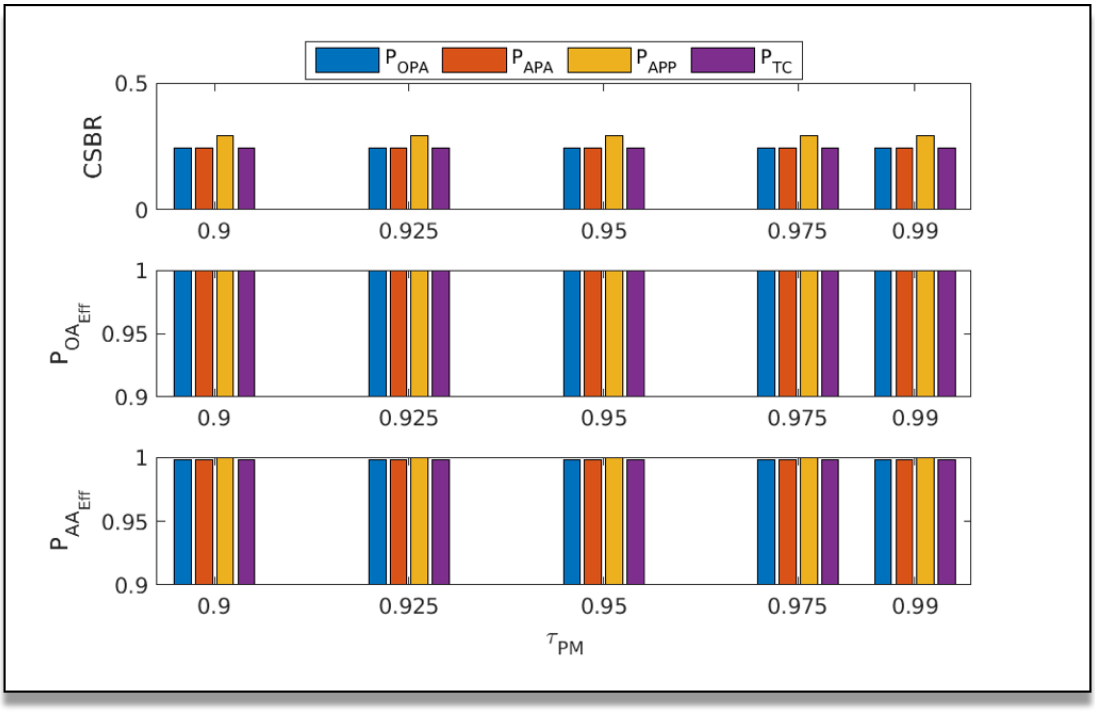

Figure 57: Pavia Centre progressive band statistics without BKG pixels

When considering the case were background pixels are included, there are two primary effects on the performance of the algorithm. First, the absolute value of the progression metrics tend to decrease. This reduction will directly affect the valid range of values that can be chosen for τ_{PM}. To illustrate the effects on the threshold, refer to the Indian Pines results in Figure 58. Note that values of τ_{PM} are now displayed from 0.85 to 0.99. In this case, the lower threshold values result in CSBR and performance efficacies that are similar to those reported when excluding the background pixels. For the higher thresholds, less than half of the cases converged, because the progression metrics never exceeded the threshold. Interestingly, P_{OPA} and P_{APA} converged in all cases but one. Conversely, P_{APP} converged less than half of the time and P_{ATC} only converged once. This observation was limited to the Indian Pines image. For the other three images, the algorithm converged almost every time.

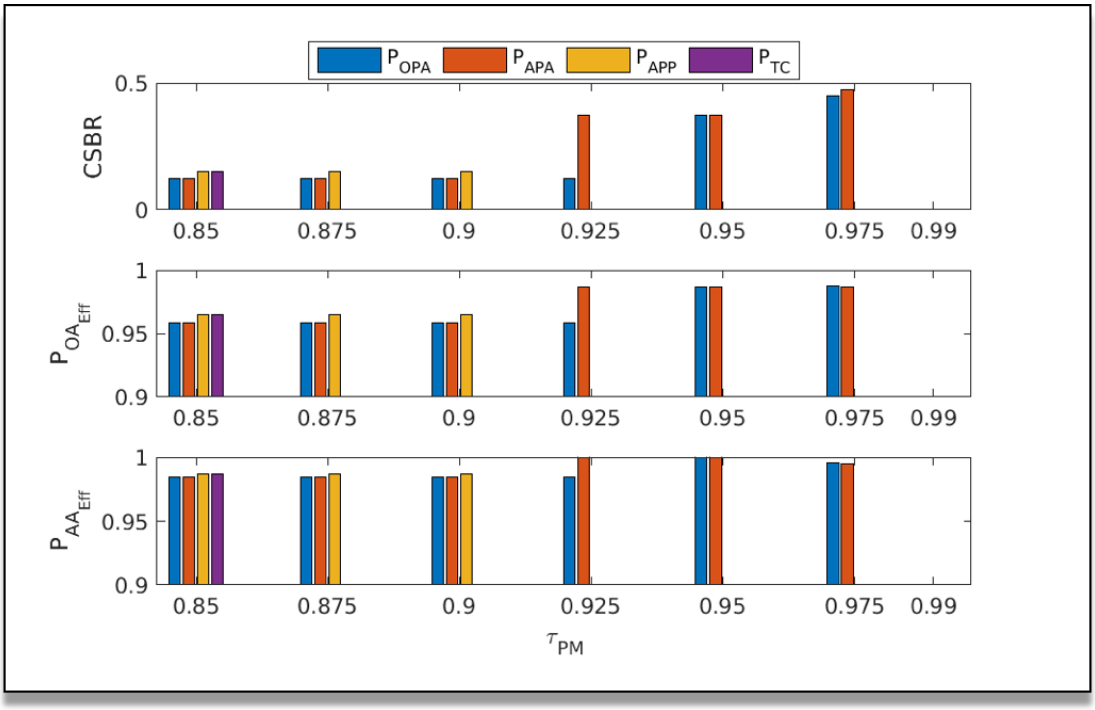

Figure 58: Indian Pines progressive band statistics with BKG pixels

Second, as it was shown in the previous section, the correlation between the progression metrics and classifier performance decreases. For two of the images, this introduced an obvious lag between the classifier performance improvements and the progression metric. This lag will manifest as an overestimate of the required number of CSBs, and consequently a higher CSBR. This effect was most evident in the Pavia University image, shown in Figure 59. For the case where background pixels were excluded, CSBR around 0.25 with full efficacy was achieved; however, with the inclusion of background pixels, the CSBR has nearly doubled.

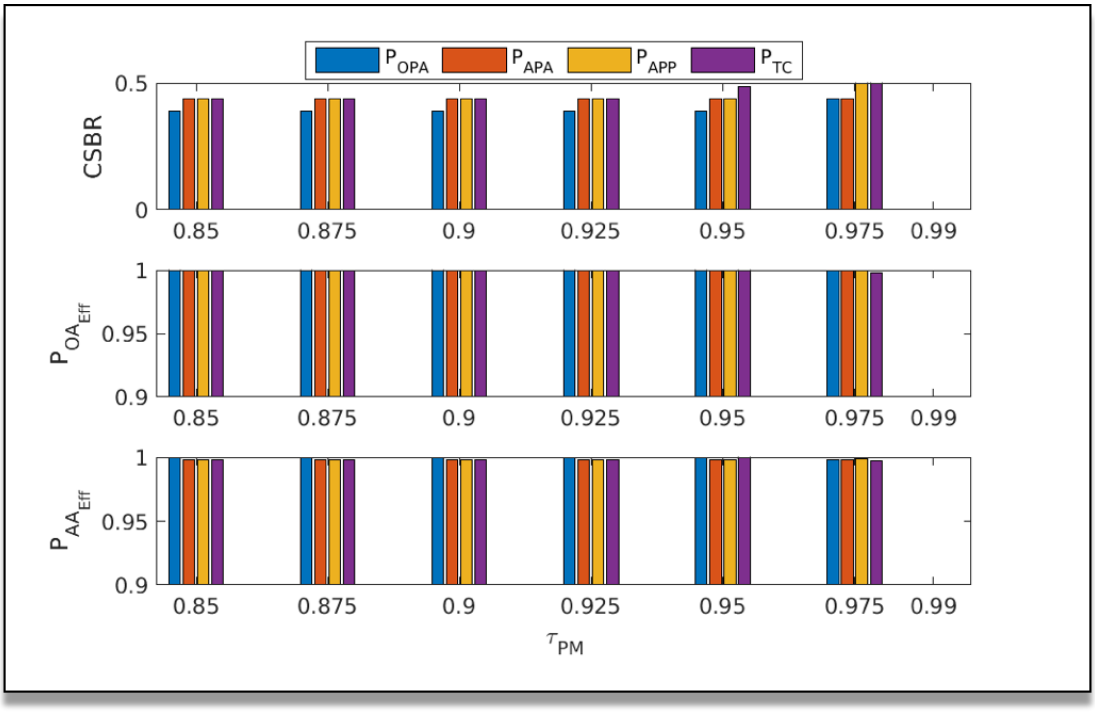

Figure 59: Pavia University progressive band statistics with BKG pixels

Conclusion

This chapter presented an *in-situ* approach to estimating the appropriate number of CSBs, based on a compressed progressive band classifier. This approach coupled band sequential processing with iterative compressive sensing. The resulting algorithm provides intermediate class predictions that could be used for decision making, as well as adaptively determines when enough CSBs have been collected. The progressive band algorithm was defined in three parts: progressive classifier, progression metrics, and stopping criteria. Each of these parts was discussed in detail, and a corresponding experiment was performed to analyze the proposed approach.

Several important conclusions were drawn from the experiments. First, the algorithm is not sensitive to the step-size between iterations. This allows for the step-size to be chosen based on practical constraints; such as computational resources and system latency. Second, all of the progression metrics showed very high correlation with classifier performance. The effects of background pixels were specifically considered, and were shown, for two of the images, to drastically reduce correlation. The proposed explanation for this decorrelation is that background pixels that fall near class boundaries, will tend to jump back and forth. However, potential mitigation strategies were proposed to lessen the effects of background pixels. Finally, the proposed stopping criteria was demonstrated to be well-suited. In general, the user thresholds were fairly robust across all of the images. For the case without the inclusion of background pixels, the algorithm was able to achieved nearly full efficacy for all images, with a CSBR less than 0.25. The case with inclusion of background pixels was also considered and some isolated effects were noted.

Chapter 6: Summary and Conclusions

The contributions of this dissertation provide the fundamental foundation for compressed classification of hyperspectral images. A new approach for hyperspectral classification in the compressively-sensed band domain (CSBD) was presented. Compressive sensing (CS) was proposed as an enabling technology to reduce the high spectral band count, through the creation of compressively-sensed bands (CSBs). A CS model based on the universality of random sensing was proposed for the analysis of hyperspectral classification in the compressed domain. It was shown that the universal model satisfied the restricted isometry property (RIP) and guaranteed optimal performance could be achieved, without the need of identifying the sparse representation. An experiment was performed to demonstrate the proposed model and universality property that it possesses.

A spectral-spatial classifier based on the support vector machine and guided filters was analyzed in the CSBD. An error analysis, based on RIP, was derived and used to show that compressed classifier will indeed asymptote to full band performance, as the number of CSBs are increased. These findings were empirically validated through a set of simulated experiments, based on four common hyperspectral images. The experiments demonstrated that full classification performance could be achieved with as few as 10% of the total bands for some of the images. The analysis also concluded that the scene complexity was a major driver in the required number of CSBs to achieve full performance, and that additional methods are required to intelligently determine the correct number of CSBs.

Two supervised algorithms based on a feature selection framework were proposed for estimating the minimum lower bound on the required number of CSBs. The first algorithm was based on feature filtering techniques and the second algorithm is based on classifier wrapping. Three variants of the filtering approach were implemented, based on Euclidean distance, spectral angle mapper, and spectral information divergence. All of the filter algorithms showed excellent agreement and were robust across all images. The wrapper method also showed excellent agreement with the employed classifier, demonstrating that the relationship between the number of CSBs and performance is maintained between training and test data.

Finally, a compressed progressive band classification algorithm was developed that is able to adaptively determine the required number of CSBs *in-situ*. Several new progression metrics were developed and showed to correlate well with classifier accuracy performance. A fully automated algorithm was developed based on a difference threshold and a progression metric threshold. The experimental results showed that the algorithm performed excellently and that performance was fairly robust to the choice of threshold. In addition to the ability to adaptively determine a sufficient number of compressed bands, the progressive classifier is also capable of providing intermediate class predictions at each iteration. The iterative class predictions can be combined with logic to develop new types of autonomy.

While this work provided a fairly complete analysis of compressed hyperspectral classification, there are still many open research questions. First, this research has focused on the signal processing and machine learning aspects of the compressed system, and there a number of hardware considerations that need to be

researched further before a practical system can be deployed. Second, the compressive sensing models explored so far have been limited to Gaussian sampling, which is likely not the most practical. Additional random sampling distributions and deterministic sampling approaches should be considered. Third, this research was limited to a single type of classifier. A more comprehensive analysis of various types of classifiers, such as modern neural networks, should be examined in the CSBD. Finally, specific applications should be considered and the appropriate logic and autonomy should be developed to fully exploit the new capabilities presented within this work.

Bibliography

Amhed, N, T Natarajan, and K R Rao. 1974. "Discrete Cosine Transform." *IEEE transactions on Computers* (IEEE) 100 (1): 90-93.

Antonini, M, M Barlaud, P Mathieu, and I Daubechies. 1992. "Image coding using wavelet transform." *IEEE Transactions on image processing* (IEEE) 1 (2): 205-220.

Baraniuk, R, M Davenport, R De Vore, and M B Wakin. 2008. "A Simple Proof of the Restricted Isometry Property for Random Matrices." *Constructive Approximation* 28 (3): 253-263.

Basseville, Michèle, and Igor V. Nikiforov. 1993. *Detection of abrupt changes: theory and application.* Vol. 104. Englewood Cliffs: Prentice Hall.

Baumgardner, M F, L L Biehl, and D A Landgrebe. 2015. "220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pines Test Site 3." *Purdue University Research Repository.*

Bioucas-Dias, J M, and M Figueiredo. 2010. "Alternating direction algorithms for constrained sparse regression: Application to hyperspectral unmixing." *Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*. IEEE. 1-4.

Bishop, C M. 1995. *Neural Networks for Pattern Recognition.* New York: Oxford Univ. Press.

Boureau, Y-Ian, and Yann l Cun. 2008. "Sparse feature learning for deep belief networks." *In Advances in neural information processing systems* 1185-1192.

Calderbank, R, S Jafarpour, and R Schapire. 2009. "Compressed learning: universal sparse dimensionality reduction and learning in the measurement domain." *Preprint.*

Camps-Valls, Gustavo, and Lorenzo Bruzzone. 2005. "Kernel-based methods for hyperspectral image classification." *IEEE Transactions on Geoscience and Remote Sensing.*

Camps-Valls, Gustavo, Devis Tuia, Lorenzo Bruzzone, and Jon Atli Benediktsson. 2014. "Advances in hyperspectral image classification: Earth monitoring with statistical learning methods." *IEEE Signal Processing Magazine.*

Camps-Valls, Gustavo, Luiz Gomez-Chova, Jordi Muñoz-Marí, Joan Vila-Francés, and Javier Calpe-Maravilla. 2006. "Composite kernels for hyperspectral image classification." *IEEE Geoscience and Remote Sensing Letters.*

Candes, E J, and M B Wakin. 2008. "An Introduction to Compressive Sampling." *IEEE Signal Processing Magazine* 25 (2): 21-30.

Candes, E. J., and T. Tao. 2005. "Decoding by Linear Programming." (IEEE Trans. Inform. Theory,) 51 (12): 4203-4215.

Chang, Chein-I. 2012. "Progressive hyperspectral imaging." *International Society for Optics and Photonics.* SPIE. 853907.

—. 1999. "Spectral information divergence for hyperspectral image analysis." *In IEEE 1999 International Geoscience and Remote Sensing Symposium (Cat. No. 99CH36293).* IEEE. 509-511.

Chang, Chein-I, and Keng-Hao Liu. 2014. "Progressive band selection of spectral unmixing for hyperspectral imagery." *IEEE Transactions on Geoscience and Remote Sensing* 52 (4): 2002-2017.

Chang, Chein-I, Su Wang, Keng-Hao Liu, Mann-Li Chang, and Chinsu Lin. 2011. "Progressive band dimensionality expansion and reduction via band prioritization for hyperspectral imagery." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 4 (3): 591-614.

Chang, Chein-I, Yao Li, Marissa C Hobbs, Robert Schultz, and Wei-Min Liu. 2015. "Progressive band processing of anomaly detection in hyperspectral imagery." *EEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 8 (7): 3558-3571.

Chen, Chen, Wei Li, Hongjun Su, and Kui Liu. 2014. "Spectral-spatial classification of hyperspectral image based on kernel extreme learning machine." *Remote Sensing* 6 (6): 5795-5814.

Chen, S. S., D. L. Donoho, and M. A. Saunders. 2001. "Atomic Decomposition by Basis Pursuit." *SIAM Review* 43 (1): 129-159.

Chen, Y, M Nasrabadi, and T D Tran. 2011. "Hyperspectral image classification using dictionary-bases sparse representation." *IEEE Transactions on Geoscience and Remote Sensing* (IEEE) 49 (10): 3973-3985.

Chen, Y, M Nasrabadi, and T D Tran. 2013. "Hyperspectral image classification via kernel sparse representation." *IEEE Transactions on Geoscience and Remote Sensing* 51 (1): 217-231.

Chen, Yushi, Xing Zhao, and Xiuping Jia. 2015. "Spectral–spatial classification of hyperspectral data based on deep belief network." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 8 (6): 2381-2392.

Chen, Yushi, Zhouhan Lin, Zing Zhao, Gang Wang, and Yanfeng Gu. 2014. "Deep learning-based classification of hyperspectral data." *EEE Journal of Selected topics in applied earth observations and remote sensing* 7 (6): 2094-2107.

Cortes, Corinna, and Vladimir Vapnik. 1995. "Support-Vector Machines." *Machine Learning* 20: 273-297.

Davenport, M A, M F Duarte, M B Wakin, J N Laska, D Takhar, K F Kelly, and R G Baraniuk. 2007. "The smashed filter for compressive classification and target recognition." *Computational Imaging V* (SPIE) 6498: 64980H.

De Carvalho, O. Abilio, and Paulo Roberto Meneses. 2000. "Spectral correlation mapper (SCM): an improvement on the spectral angle mapper (SAM)." *In Summaries of the 9th JPL Airborne Earth Science Workshop 00-18*. Pasadena: JPL Publication.

Du, Q, J M Bioucas-Dias, and A Plaza. 2012. "Hyperspectral band selection using a collaborative sparse model." *Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE. 3054-3057.

Duarte, Marco F, Mark A Davenport, Dharmpal Takhar, Jason N Lask, Ting Sun, Kevin F Kelly, and Richard G Baraniuk. 2008. "Single-pixel imaging via compressive sampling." *IEEE signal processing magazine* (IEEE) 25 (2): 83-91.

Fauvel, Mathieu, Jon Atli Benediktsson, Jocelyn Chanussot, and Johannes R Sveinsson. 2008. "Spectral and spatial classification of hyperspectral data using SVMs and morphological profiles." *IEEE Transactions on Geoscience and Remote Sensing.*

Fauvel, Mathieu, Yuliya Tarabalka, Jon Atli Benediktsson, Jocelyn Chanussot, and James C Tilton. 2013. "Advances in spectral-spatial classification of hyperspectral images." *Proceedings of the IEEE.*

Fornasier, M., and S. Peter. n.d. "An Overview on Algorithms for Sparse Recovery." (Sparse Reconstruction and Compressive Sensing in Remote Sensing).

Green, Robert O, Micheal L Eastwood, Charles M Sarture, Thomas G Chrien, Mikael Aronsson, Bruce J Chippendale, Jessica A Faust, et al. 1998. "Imaging Spectroscopy and the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)." *Remote sensing of environment* 65 (3): 227-248.

Haar, Alfred. 1910. "Zur Theorie der orthogonalen Funktionensysteme." *Mathematische Annalen* 69 (3): 331-371.

Hadoon, David R, Sandor Szedmak, and John Shawe-Taylor. 2004. "Canonical correlation analysis: An overview with application to learning methods." *Neural computation* 16 (12): 2639-2664.

Hagen, Nathan A, and Michael W Kudenov. 2013. "Review of snapshot spectral imaging technologies." *Optical Engineering* 52 (9): 090901.

Hahm, J, S Rosenkranz, and A M Zoubir. 2014. "Adaptive compressed classification for hyperspectral imagery." *Acoustics, Speed and Signal Processing (ICASSP)* 1020-1024.

Holzwarth, S, A Muller, M Habermayer, R Richter, A Hausold, S Thiemann, and P Strobl. 2003. "HySens-DAIS 7915/ROSIS imaging spectrometers at DLR." *In Proceedings of the 3rd EARSeL Workshop on Imaging Spectroscopy.* 3-14.

Huang, Guang-Bin, Qin-Yu Zhu, and Chee-Kheong Siew. 2006. "Extreme learning machine: theory and applications." *Neurocomputing* 70 (1-3): 489-501.

Jaccard, Paul. 1901. "Étude comparative de la distribution florale dans une portion des Alpes et des Jura." *Bulletin de la Société Vaudoise des Sciences Naturelles* 37: 547–579.

Jordache, M, J M Bioucas-Dias, and A Plaza. 2011. "Sparse unmixing of hyperspectral data." *IEEE Transactions on Geoscience and Remote Sensing* 49 (6): 2014-2039.

Kang, Xudong, Shutao Li, and Jon Atli Benediktsson. 2014. "Spectral-spatial hyperspectral image classification with edge-preserving filtering." *IEEE Transactions on Geoscience and Remote Sensing.*

Krizhevsky, Alex, IIya Sutskever, and Geoffrey E Hinton. 2012. "Imagenet classification with deep convolutional neural networks." *In Advances in neural information processing systems* 1097-1105.

Kruse, Fred A., A. B. Lefkoff, J. W. Boardman, K. B. Heidebrecht, A. T. Shapiro, P. J. Barloon, and A. F. Goetz. 1993. "The spectral image processing system (SIPS)—interactive visualization and analysis of imaging spectrometer data." *Remote sensing of environment* 44 (2-3): 145-163.

Lavielle, Marc. 2005. "Using penalized contrasts for the change-point problem." *Signal processing* 85 (8): 1501-1510.

Li, C, L Ma, Q Wang, Y Zhou, and N Wang. 2013. "Construction of Sparse Basis By Dictionary Training for Compressive Sensing Hyperspectral Imaging." *IEEE International Geosicence and Remote Sensing Symposium (IGARSS).* 1442-1445.

Li, Tong, Junping Zhang, and Ye Zhang. 2014. "Classification of hyperspectral image based on deep belief networks." *IEEE Image Processing (ICIP).*

Lv, Qi, Xin Niu, Yong Dou, Jiaqing Xu, and Yuanwu Lei. 2016. "Classification of hyperspectral remote sensing image using hierarchical local-receptive-field-based extreme learning machine." *IEEE Geoscience and Remote Sensing Letters* 13 (3): 434-438.

Rogers, David J, and Taffee T Tanimoto. 1960. "A Computer Program for Classifying Plants." *Science* 132 (3434): 1115-1118.

Samat, Alim, Peijun Du, Sicong Liu, Jun Li, and Liang Cheng. 2014. "E2LMs: Ensemble Extreme Learning Machines for Hyperspectral Image Classification." *Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (IEEE) 7 (4): 1060-1069.

Schölkopf, Bernhard, Ralf Herbrich, and Alex J Smola. 2001. "A generalized representer theorem." *International Conference on Computational Learning Theory.* Berlin.

Scholkopf, N, and A.J. Smola. 2002. *Learning with Kernels, Support Vector Learning.* Cambridge, MA: MIT Press.

Schultz, Robert C, Marissa Hobbs, and Chein-I Chang. 2014. "Progressive band processing of simplex growing algorithm for finding endmembers in

hyperspectral imagery." *Satellite Data Compression, Communications, and Processing X.* International Society for Optics and Photonics. 91240L.

Shannon, Claude E. 1949. "Communication in the presence of noise." *Proceedings of the Institute of Radio Engineers.* 37 (1): 10-21.

Tang, Jiliang, Salem Alelyani, and Huan Liu. 2014. "Feature Selection for Classification: A Review." *Data Classifications: Algorithms and Applications* 37.

Tropp, J. A., and A. C. Gilbert. 2007. "Signal Recovery from Random Measurements Via Orthogonal Matching Pursuit." *IEEE Transactions on Information Theory* 53 (12): 4655-4666.

Tuia, D, R Flamary, and N Courty. 2015. "Multiclass feature learning for hyperspectral image classification: Sparse and hiearchical solutions." *ISPRS Journal of Photogrammetry and Remote Sensing* (105): 272-285.

n.d. *Universidad del Pais Vasco Grupo de Inteligencia Computacional.* Accessed June 2018. http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes#Salinas.

Vincent, Pascal, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. "Extracting and composing robust features with denoising autoencoders." *n Proceedings of the 25th international conference on Machine learning.* ACM. 1096-1103.

Wang, Yulei, Robert Schultz, Shih-Yu Chen, Chunhong Liu, and Chein-I Chang. 2013. "Progressive constrained energy minimization for subpixel detection."

Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XIX. International Society for Optics and Photonics. 874321.

Wold, Savante, Kim Esbensen, and Paul Geladi. 1987. "Principal Component Analysis." *Chemometrics and Intelligent Laboratory Systems* 2 (1-3): 37-52.

Zhong, P, and R Wang. 2008. "Learning sparse CRFs for feature selection and classification of hyperspectral imagery." *IEEE Transactions on Geoscience and Remote Sensing* (IEEE) 46 (12): 4186-4197.

Zhou, Yicong, Jiangtao Peng, and Chen Chen. 2015. "Extreme learning machine with composite kernels for hyperspectral image classification." *Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (IEEE) 8 (6): 2351-2360.