Unsupervised Data Driven Machine Learning in Hyperspectral Imaging and Echocardiography Videos

by

KAZI TANZEEM SHAHID

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON
May 2021

ACKNOWLEDGEMENTS

There have been many people in my life that have blessed me to be standing here today. I am thankful to all of the experiences and influences from them, that I cherish, and shall do so in all my future endeavors.

First and foremost would be Dr. Ioannis D. Schizas. I consider myself truly lucky to have him as my supervising professor. He has acted not only as a supervisor, but as a mentor in the truest sense of the word. He taught me not only theoretical subjects, but he has also taught me how to understand and tackle the daunting task of learning new and unfamiliar concepts, the value of hard and sincere work, how to be patient in the face of failures, and to appreciate them as experiences for future. Dr. Schizas has been a brilliant blend of intelligence, respect, professionalism, and kindness, and I have often bragged to my friends and family about how great of a mentor he has been, for good reason.

I would also like to thank Dr. Michael T. Manry, Dr. Ramtin Madani, Dr. Saibun Tjuatja, and Dr. Venkat Devarajan for taking time off their busy schedules to be in my PhD defense committee.

I would also like to appreciate my former colleagues for their invaluable help and lessons, Guohua Ren, Jia Chen and last but far from the least, Akshay Malhotra. Akshay has been one of the few people without whom my PhD would have been an insurmountable challenge. He has been a crucial companion in my academic life, helping with studies in coursework as well as my doctoral research. On top of that, he has been a strong source of support in my personal life throughout my PhD, being a source of advice and wisdom in critical times. I have been very fortunate to have him as a friend.

I am also thankful for my friend Pratik Ghate, who has been a colleague during my role as a Graduate Teaching Assistant, and has gone out of his way countless times to help me understand how to perform my duties. Once we were matched in experience, he has been a consistence source of morale throughout my studies, and his friendship has been a great aid on many occasions.

Among the friends I have made through my life in UT Arlington, I shall also mention Cyndi Needels, Jerry Sandoval, Emily Goodyear Klophaus and Kara Lindsey Milton for their insightful and meaningful conversations, and their time as a sympathetic ear during the mental hurdles faced throughout the daunting challenges of a PhD. Through you, I understand truly what it means to say ”you are doing God’s work”.

The Bangladesh Student Organization at UT Arlington has been a large group of Bangladeshi students who have been a slice of home away from home, and their friendship has been nothing short of invaluable in upholding my morale, and mitigating my homesickness while pursuing my PhD in a country far from my family.

I would also like to extend my heartiest gratitude to Dr. Alan Taylor of Cardiology Partners, who very kindly showed me the theoretical background behind echocardiograms, and generously provided echocardiogram data for my applications. The staff at Cardiology Partners deserve recognition as well in this regard, particularly Renuka Shah, given how they spent time throughout their busy schedules to accommodate my request.

Lastly, I would like to extend my sincere gratitude to the family members without whom I would not be the person I am today. My parents, and brothers, through their strong and ethical professional principles, have instilled in me a desire to carve my own path in life, and their humility have taught me to appreciate the value of creating achievements for the sake of values that go beyond personal gain. My mother, Dr. Momtaz Begum has been a direct influence in my PhD on another unique aspect, given that she lent her medical expertise in expanding the theoretical background behind one of my publications. I am also fortunate to have the support of my wife, Fairuz Maliha, through her wisdom and foresight, which has been a support whose value I shall never be able to measure, though not for lack of trying.

April 30, 2021

Unsupervised Data Driven Machine Learning in Hyperspectral Imaging and Echocardiography Videos

KAZI TANZEEM SHAHID

Supervising Professor: Ioannis D. Schizas

ABSTRACT

This work discusses the problem of unsupervised classification in images. Conventional methods approached this problem with the naive assumption that the relationship among the pixels' information can be expressed sufficiently in a linear manner. However, higher accuracy was established by implementing kernel-based expressions of data to express the non-linear relationship of that data in a linear manner, when mapped in a higher dimensional space. This process allows much more effective clustering performances by increasing the informativeness of the data. Hyperspectral images, being limited in spatial resolution as a tradeoff for the significantly higher number of channels compared to traditional images, often face the challenge of having pixels that, instead of showing one material, show a mixture of multiple materials. It then becomes a challenging task of *unmixing* those materials, whose challenge is greatly exacerbated with the presence of strong noise, and/or the data being corrupted due to some damage to the sensor, causing *dead pixels* in the form of data entries containing zero values. Unlike a large body of work which focuses on a simpler approach, where it is assumed that the mixtures are obtained through a linear combination of the contributing materials, nonlinear mixtures are a more accurate representation of the mixtures obtained in real-life scenarios, and are thus tackled in our work. The unmixing problems were addressed via formulation of a constrained optimization problem which utilizes nonlinear mixing models, efficiently addressing the limitation of having a limited window of kernel parameters, tackling more complex mixture models, and reducing computational complexity by automatically reducing dimensions containing irrelevant data, with the added challenge of performing in a fully unsupervised setting. The design and implementation of a nonlinear autoencoder neural network further improves work in this respect, by fully customized designs of layers, which not only utilize spatial information via weighted averaging of the pixels based on their perceived similarities in the kernel space, but also have the added versatility in accommodation of higher degree nonlinear interactions, a technique unavailable in major current works. Going beyond hyperspectral remote sensing images, a unique approach was tackled in unsupervised heart disease diagnosis through observation of the mitral valve, and potential diseases affecting its ability to function effectively. A wide variety of datasets were used in measuring its efficacy, including data that was noisy and of lower resolutions, further increasing the difficulty of implementing a fully unsupervised heart disease diagnosis algorithm that detected the location of the mitral valve within the videos, and observed its movement to detect whether it was diseased or healthy.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

Table Page

CHAPTER 1

INTRODUCTION

When a sensor observes an environment, a particular measured data entry might give very similar quantities across multiple samples, if the data entry across those samples are affected by the same source. As an example, consider a hyperspectral remote sensing image of an agricultural landscape. If multiple pixels encompass an area containing apples, these pixels will contain similar spectral signatures across various spectral bands, and will thus be correlated. In this manner, covariances or correlations among pixels' reflectance values can be used to cluster together pixels in an image that contain the same type of material.

Hyperspectral images enable information acquisition about an area of interest in different spectral bands of light [1]. Different materials on a hyperspectral image give rise to different spectral signatures that can be utilized to cluster the hyperspectral pixels in different groups according to their information content. A considerable amount of work exists for clustering and classifying hyperspectral pixels according to the materials they observe [2–6]. The majority of existing clustering techniques are supervised and rely on training data to operate. However, when acquiring hyperspectral images a major challenge is the lack of training data that are needed to train supervised pixel classifiers [7, 8]. The aforementioned challenge has been addressed via the derivation of unsupervised techniques [9, 10] that bypass the need for training, while compromising some clustering performance. As part of the work in this thesis, I work towards implementing a Canonical Correlations analysis (CCA) framework [16] in order to extract maximally correlated components from a collection of data. However, as the correlations is usually nonlinear in hyperspectral images, the Kernel Trick [17] was applied in order to map the data to a higher dimensional space where the usually nonlinear correlations of the hyperspectral pixel information will be expressed linearly. Thus, in an unsupervised manner, the scheme will accurately cluster together pixels that contain information of the same materials.

The nature of hyperspectral images is that they contain hundreds of different spectral bands. These bands encompass the visible light spectrum, including short wave infrared, as well as long wave infrared and ultraviolet. As opposed to traditional images, the advantage of receiving spectral reflectance values across many separate bands is the encapsulation of far more additional information regarding different materials/objects that are present in the images. This extra information comes at a price, however. The extra information across the spectral domain contained in vector pixels, results in a tradeoff where the spatial information is significantly lower, causing each pixel of an image encompassing a much larger area. This results in *mixed* pixels that can contain information regarding multiple materials, and makes it essential to design a scheme that is able to separate from a collection of mixed pixels, the spectral reflectances of the original pure materials. Spectral unmixing is a well-researched topic in remote sensing, with considerable work relying on the assumption that mixtures are formed from a linear combination of pure materials' signatures [11–13]. To that end, my scheme revises the regularized CCA formulation to unmix the hyperspectral image pixels with mixed spectral signatures and isolate pure materials' spectral responses, while being fully unsupervised, which further implies no need for training data, or knowledge of the number of different materials contributing to the mixing process.

Sensor damage, speckle dust can cause the hyperspectral image sensor to have artifacts that render pixels unresponsive, causing data entries in the images to have no response [14]. These *dead* pixels cause disruption in the data, rendering many established methods inefficient in spectral clustering and unmixing. My developed scheme is shown to be robust even in the case of severe dead pixel presence.

Neural networks have been a realm of study that has gained traction with regards to hyperspectral unmixing in recent years. In unsupervised settings, the formation of autoencoders have been used to tackle both linear and nonlinear mixing models, with utilization of spatial information [81, 82], as well as without [62, 71]. The application of nonlinear mixing models has been explored to a limited degree in [71], due to the complexity of implementation of more sophisticated nonlinear mixing models into a neural network setting. To that end, my work focused on building an autoencoder neural network structure with layers that were custom-built in order to tackle the unmixing problem effectively. To that end, the advantage of RBF (Radial Basis Function) kernel transformation was exploited to measure the similarities of the various mixed pixels w.r.t. K-means cluster centers that act as initial representations of the pure materials. This similarity, which can be interpreted as estimations of the fractions of these materials, can then be fed into subsequent layers in this neural network structure, where the weights of these layers are initialized as estimations of the pure materials, and can be optimized through training of the autoencoder structure.

Although nonlinear mixing models usually calculate upto 2^{nd} degree cross-products of the materials to replicate the nonlinear interactions among them, higher degree cross-products from nonlinear interactions of more than two materials can also exist, but they are usually ignored. This is due to the fact that as the values pertaining to these materials are usually normalized to be between 0 and 1, higher degree products of such fractional values yield smaller and more insignificant magnitudes, that may not warrant the complexity to consider such terms in the unmixing algorithms. However, with the goal of having a very versatile and highly accurate unmixing method, I have also designed the neural network structure to accommodate higher degree cross-products as well.

It is common practice in hyperspectral unmixing to assume prior knowledge of the number of materials that exist in the datasets. However, it would be of significant advantage if the unmixing algorithm is able to estimate this number, further bolstering its characteristics as a fully unsupervised method. To that end, the kernel covariance matrix, which would have a block diagonal structure, having as many blocks as there are materials, is calculated, and its reconstruction errors w.r.t. various rank-equivalent matrices are observed to effectively estimate the number of materials contributing to the datasets. On top of that, the problem of dead pixels is further observed, utilizing the spatial property present in these images, to employ an RBF-based weighted averaging filter throughout the data to fill the gaps obtained from unresponsive pixels with values that accurately represent the data that would have been observed had the sensors not contained dead pixels.

In the realm of image processing, medical imaging is another field that has received a great deal of attention. Heart disease is a leading cause of death among adults worldwide. As the heart's function is to pump blood throughout the body with valves regulating blood flow, a very common cause of crippling heart disease comes from infections preventing proper function of the valves and preventing blood flow. A popular medical imaging procedure for diagnosing potential heart disease is through ultrasound (echocardiograms), as it is noninvasive, relatively cheaper and requires little preparation on the patients' side. Although some schemes exist for diagnosis through valve tracking from echocardiography videos, they are supervised, and unsupervised schemes are very difficult as the hearts vary noticeably among various patients. On top of that, this is exacerbated by involuntary movement of the heart. My scheme was able to effectively observe a wide array of echocardiography videos, and in a fully unsupervised manner, use valve tracking to determine whether a particular heart is diseased or not, with reasonable accuracy.

As part of my work, I have attempted to implement unsupervised algorithms to cluster and unmix hyperspectral remote sensing images, and observing ultrasound videos of the human heart to diagnose heart disease. Some key contributions of my work encompass:

1. A kernelized form of the regularized CCA based clustering scheme.

2. A reformulation of the regularized kernel-based CCA scheme into a constrained optimization problem that performs unsupervised unmixing of mixed signals.

3. A custom-built autoencoder network that performs nonlinear hyperspectral unmixing through implementation of the RBF kernel trick.

4. A customized autoencoder for nonlinear unmixing, that tackles dead pixels, while also estimating the number of materials contributing to the datasets.

5. Unsupervised valve tracking-based heart disease diagnosis in echocardiography videos.

The rest of the document is structured as follows. Chapter 2 introduces the linear CCA based clustering framework. Chapter 3 illustrates the kernelized variant of the CCA framework. Chapter 4 shows the kernelized CCA framework extended for spectral unmixing. Chapter 5 details the autoencoder network for nonlinear unmixing. Chapter 6 describes the aforementioned autoencoder structure with further implementation of endmember number estimation while tackling dead pixels with a novel averaging filter. Chapter 7 discusses the heart disease diagnosis framework for echoardiography videos, and Chapter 8 closes the report with concluding remarks, and directions of future work.

CHAPTER 2

LINEAR CCA FRAMEWORK

Hyperspectral images consist of vector pixels in which each entry corresponds to the intensity on different spectral bands of the electromagnetic spectrum [1]. Let F denote the number of spectral bands information that is acquired for each pixel, while set $\mathcal{F}_s := \{1, \ldots, F\}$ denote the spectral band indices. Further, let $\mathbf{X}_f$ denote a $S_x \times S_y$ matrix that contains the intensity of all the pixels of a hyperspectral image acquired at spectral band f, for $f = 1, \ldots, F$ with S_x and S_y denote the number of pixels per row and column in $\mathbf{X}_f$. The main goal of this work was to develop an unsupervised framework to cluster pixels of hyperspectral images according to the material they contain information about. In detail, our goal is to cluster a collection $\mathcal{P}$ of $p \leq S_x \cdot S_y$ pixels, that contain information about q materials, into q different groups according to their information content *without using training pixels*.

The set of pixels in $\mathcal{P}$ is splitted into two data vector sequences $\mathbf{x}_f \in \mathbb{R}^{p_x \times 1}$ and $\mathbf{y}_f \in \mathbb{R}^{p_y \times 1}$, that contain the intensity of the corresponding pixels at spectral band f such that $p_x + p_y = p$. The sequences $\mathbf{x}_f$ and $\mathbf{y}_f$ do not contain overlapping pixels. For simplicity in exposition and without loss of generality it is assumed here that $p_x = p_y$. Moreover, let $f \in \mathcal{F}_s$, where $\mathcal{F}_s$ is a subset of all available spectral bands in $\mathcal{F}$ which will be used for the clustering; and $F_s := |\mathcal{F}_s|$. It is also assumed that the pixels in both $\mathbf{x}_f$, and $\mathbf{y}_f$ sequences contain information about all the q materials of interest. This assumption can be easily satisfied when the number of pixels p is sufficiently larger than the number q of different materials. In practice, sensing units acquiring information across different spectral bands may be malfunctioning causing certain pixel intensities in $\mathbf{x}_f$, and $\mathbf{y}_f$ to be missing. Missing pixels (a.k.a. dead pixels) here will be assumed to have a zero intensity value.

To perform clustering of the p pixels of interest according to the material they sense, statistical correlation between pixels with similar information content in $\mathbf{x}_f$, and $\mathbf{y}_f$ will be exploited. The novel framework proposed here is built on the canonical correlation analysis (CCA) framework [16] that is capable of linearly extracting maximally correlated common objects/features present in two data sequences $\mathbf{x}_f$, and $\mathbf{y}_f$. In fact, it was demonstrated in [15] that information clustering via CCA can be achieved by utilizing norm-one regularization mechanisms in the standard CCA formulations [15] , i.e.,

$$\begin{aligned}(\hat{\mathbf{E}}, \hat{\mathbf{D}}) = \arg\min_{\mathbf{E},\mathbf{D}} F_s^{-1} \textstyle\sum_{f\in\mathcal{F}_s} ||\mathbf{y}_f - \mathbf{E}\mathbf{D}\mathbf{x}_f||_2^2 \\ + \textstyle\sum_{\rho=1}^{q} \lambda_\rho^{\mathbf{E}} ||\mathbf{E}_{:\rho}||_1 + \textstyle\sum_{\rho=1}^{q} \lambda_\rho^{\mathbf{D}} ||\mathbf{D}_{\rho:}||_1,\end{aligned} \tag{2.1}$$

where $\hat{\mathbf{D}} \in \mathbb{R}^{q\times p_x}$ and $\hat{\mathbf{E}} \in \mathbb{R}^{p_y\times q}$ correspond to pertinent sparse clustering matrices that are obtained such that the nonzero entries (support) of each of the q rows and columns in $\mathbf{D}$ and $\mathbf{E}$ respectively point to the entries (pixels) in $\mathbf{x}_f$, and $\mathbf{y}_f$ with similar information content [15]. The operator $\|\cdot\|_1$ denotes norm-one, while the parameters $\lambda_\rho^{\mathbf{D}}$ and $\lambda_\rho^{\mathbf{E}}$ correspond to sparsity controlling coefficients that adjust the number of nonzero entries in $\mathbf{D}$ and $\mathbf{E}$, respectively [15].

CHAPTER 3

KERNELIZED CCA FOR CLUSTERING

A main limitation of the sparse CCA formulation in (2.1) is that it relies on extracting linear data correlations, whereas the pixels in hyperspectral images are in practice nonlinearly related [7,8]. To this end, nonlinear kernel mappings will be introduced in the formulation in (2.1) to address nonlinear pixel dependencies. Further, the kernel trick [17] will be utilized to obtain an efficient minimization technique that will allow computationally efficient extraction of pertinent sparse clustering matrices $\mathbf{D}$ and $\mathbf{E}$.

3.1 Regularized Kernel-Based CCA

In order to determine the presence of nonlinear correlations present among pixel entries in $\mathbf{x}_f$ and $\mathbf{y}_f$, we introduce nonlinear mappings $\boldsymbol{\phi}_x(\mathbf{x}_f)$ and $\boldsymbol{\phi}_y(\mathbf{y}_f)$. These are representations of $\mathbf{x}_f$ and $\mathbf{y}_f$ in a higher dimensional space, obtained by row-wise implementation across the entries of $\mathbf{x}_f$ and $\mathbf{y}_f$. The main goal of these nonlinear mappings is to transform as much as possible pixel correlations to linear dependencies. In this higher dimensional space, we can exploit the potentially linear correlations using the formulation in (2.1). So we replace $\mathbf{x}_f$ and $\mathbf{y}_f$ with nonlinear mappings $\boldsymbol{\phi}_x(\mathbf{x}_f)$ and $\boldsymbol{\phi}_y(\mathbf{y}_f)$ in (2.1) to obtain the kernel regularized canonical correlations (KRCC) formulation

$$\begin{aligned}(\hat{\mathbf{E}}, \hat{\mathbf{D}}) = \arg\min_{\mathbf{E},\mathbf{D}} F_s^{-1} \textstyle\sum_{f\in\mathcal{F}_s} ||\boldsymbol{\phi}_y(\mathbf{y}_f) - \mathbf{E}\mathbf{D}\boldsymbol{\phi}_x(\mathbf{x}_f)||_2^2 \\ + \textstyle\sum_{\rho=1}^{q} \lambda_\rho^{\mathbf{E}} ||\mathbf{E}_{:\rho}||_1 + \textstyle\sum_{\rho=1}^{q} \lambda_\rho^{\mathbf{D}} ||\mathbf{D}_{\rho:}||_1. \qquad (3.1)\end{aligned}$$

Let $J^s(\mathbf{E},\mathbf{D})$ denote the first term, while $J(\mathbf{E},\mathbf{D})$ denotes the entire cost function in (3.1). The cost in (3.1) will be minimized by implementing the subgradient descent method [18]. In this way, at iteration k, the updates for the clustering matrices $\mathbf{E}$ and $\mathbf{D}$ can be found by replacing $\mathbf{D}$ and $\mathbf{E}$ with their updates obtained from the latest iteration, and utilizing the following rules to recursively update $\mathbf{E}$ and $\mathbf{D}$ until convergence

$$\hat{\mathbf{E}}_k = \hat{\mathbf{E}}_{k-1} - c\nabla^{\mathbf{E}}_{\hat{\mathbf{E}}_{k-1},\hat{\mathbf{D}}_{k-1}} J(\mathbf{E},\mathbf{D}), \tag{3.2}$$

$$\hat{\mathbf{D}}_k = \hat{\mathbf{D}}_{k-1} - c\nabla^{\mathbf{D}}_{\hat{\mathbf{E}}_{k-1},\hat{\mathbf{D}}_{k-1}} J(\mathbf{E},\mathbf{D}). \tag{3.3}$$

Here $c > 0$ represents the step-size, which is chosen on a trial and error basis to ensure convergence on eqs. (3) and (4). Also, $\nabla^{\mathbf{E}}_{\hat{\mathbf{E}}_{k-1},\hat{\mathbf{D}}_{k-1}} J(\mathbf{E},\mathbf{D})$ and $\nabla^{\mathbf{D}}_{\hat{\mathbf{E}}_{k-1},\hat{\mathbf{D}}_{k-1}} J(\mathbf{E},\mathbf{D})$ are the partial subgradients of $J(\mathbf{E},\mathbf{D})$ with respect to $\mathbf{E}$ and $\mathbf{D}$ respectively, with $\hat{\mathbf{E}}_{k-1}$ and $\hat{\mathbf{D}}_{k-1}$ taken as input parameters in the cost $J(\mathbf{E},\mathbf{D})$.

Writing the subgradient terms in (3.2) and (4) in more detail we obtain the following detailed expressions

$$\nabla^{\mathbf{E}} J(\mathbf{E},\mathbf{D}) = \frac{\delta J^s(\mathbf{E},\mathbf{D})}{\delta \mathbf{E}} + \text{sgn}(\mathbf{E})\text{diag}(\boldsymbol{\lambda}^{\mathbf{E}}) \tag{3.4}$$

$$\nabla^{\mathbf{D}} J(\mathbf{E},\mathbf{D}) = \frac{\delta J^s(\mathbf{E},\mathbf{D})}{\delta \mathbf{D}} + \text{diag}(\boldsymbol{\lambda}^{\mathbf{D}})\text{sgn}(\mathbf{D}). \tag{3.5}$$

The second term in the rhs (right hand side) of eqs. (5) and (6) corresponds to a subgradient of the norm-one terms in (3.1), with respect to $\mathbf{E}$ and $\mathbf{D}$ respectively. Observe that diag($\boldsymbol{\lambda}^{\mathbf{D}}$) corresponds to a diagonal matrix, where the diagonal entries contain $[\lambda_1^{\mathbf{D}}, \ldots, \lambda_q^{\mathbf{D}}]$, while sgn($\mathbf{M}$) is the matrix we obtain after applying the sign operator across the entries of matrix $\mathbf{M}$.

Application of the kernel trick, after properly selecting the nonlinear mapping $\boldsymbol{\phi}(\cdot)$ as detailed in [17], enables to rewrite the term $J^s(\mathbf{E},\mathbf{D})$ in (3.1) as follows

$$J^s(\mathbf{E},\mathbf{D}) = \text{tr}(\hat{\mathbf{K}}_{\mathbf{y}} - 2{\cdot}\mathbf{E}{\cdot}\mathbf{D}{\cdot}\hat{\mathbf{K}}_{\mathbf{xy}} + \mathbf{E}{\cdot}\mathbf{D}{\cdot}\hat{\mathbf{K}}_{\mathbf{x}}{\cdot}\mathbf{D}^T{\cdot}\mathbf{E}^T). \tag{3.6}$$

Here the matrices $\hat{\mathbf{K}}_x, \hat{\mathbf{K}}_y, \hat{\mathbf{K}}_{xy}$ denote the weighted-average (cross)-covariance matrices of the transformed data as follows

$$\hat{\mathbf{K}}_{\mathbf{x}} := \sum_{f\in\mathcal{F}_s} w_f^x \hat{\mathbf{K}}_x(f), \tag{3.7}$$

$$\hat{\mathbf{K}}_{\mathbf{y}} := \sum_{f\in\mathcal{F}_s} w_f^y \hat{\mathbf{K}}_y(f), \tag{3.8}$$

$$\hat{\mathbf{K}}_{\mathbf{xy}} := \sum_{f\in\mathcal{F}_s} w_f^{xy} \hat{\mathbf{K}}_{xy}(f). \tag{3.9}$$

Here, w_f^x, w_f^y and w_f^{xy} represent the weights placed upon each spectral band f. The aforementioned weights will be set proportionally to the variability of each of the kernel covariance matrices entries generated by $\hat{\mathbf{K}}_x(f) := \boldsymbol{\phi}_x(\mathbf{x}_f)\boldsymbol{\phi}_x^T(\mathbf{x}_f)$, $\hat{\mathbf{K}}_y(f) := \boldsymbol{\phi}_y(\mathbf{y}_f)\boldsymbol{\phi}_y^T(\mathbf{y}_f)$, and $\hat{\mathbf{K}}_{xy}(f) := \boldsymbol{\phi}_x(\mathbf{x}_f)\boldsymbol{\phi}_y^T(\mathbf{y}_f)$ corresponding to spectral band f. The larger the variance of the covariance matrix entries is at a specific spectral band, the larger the corresponding weight will be set and vice versa. Details on how the weighting scheme works in (3.7)-(3.9) are given in the section 3.3 and specific details on how to obtain the kernel matrices $\hat{\mathbf{K}}_x(f)$ and center them are presented in section 3.2 and section 3.5, respectively.

Further, the first term of the rhs in Eqs. 3.4 and 3.5 denote the partial derivatives of the term $J^s(\mathbf{E}, \mathbf{D})$ with respect to $\mathbf{E}$ and $\mathbf{D}$ respectively, that may be obtained after differentiating (3.6) with respect to $\mathbf{E}$ and $\mathbf{D}$ as follows

$$\begin{aligned}
\frac{\delta J^s(\mathbf{E}, \mathbf{D})}{\delta \mathbf{D}} &= -2 \cdot \hat{\mathbf{K}}_{\mathbf{xy}}^T \cdot \mathbf{D}^T + \mathbf{E} \cdot \mathbf{D} \cdot \hat{\mathbf{K}}_{\mathbf{x}}^T \cdot \mathbf{D}^T + \mathbf{E} \cdot \mathbf{D} \cdot \hat{\mathbf{K}}_{\mathbf{x}} \cdot \mathbf{D}^T, \\
\frac{\delta J^s(\mathbf{E}, \mathbf{D})}{\delta \mathbf{E}} &= -2 \cdot \mathbf{E}^T \cdot \hat{\mathbf{K}}_{\mathbf{xy}}^T + \mathbf{E}^T \cdot \mathbf{E} \cdot \mathbf{D} \cdot \hat{\mathbf{K}}_{\mathbf{x}}^T + \mathbf{E}^T \cdot \mathbf{E} \cdot \mathbf{D} \cdot \hat{\mathbf{K}}_{\mathbf{x}}.
\end{aligned}$$

3.2 Kernel Variance Selection

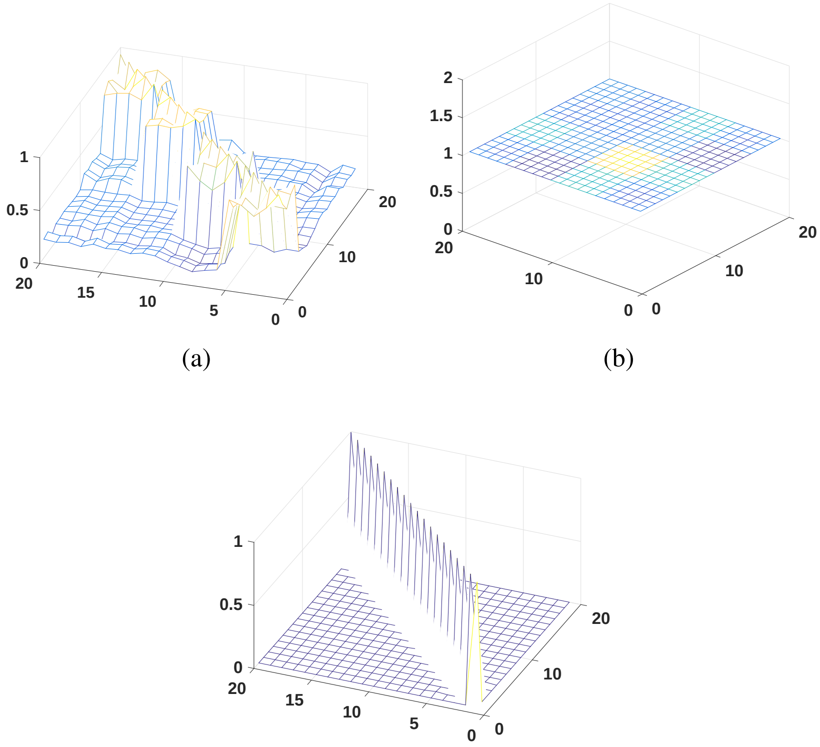
A kernel utilized in hyperspectral images successfully [7, 8] is the Gaussian radial basis function (RBF) kernels which can be expressed as follows

$$k_x(\mathbf{x}_f, \mathbf{x}_{f'}) = \exp\left(-\frac{||\mathbf{x}_f - \mathbf{x}_{f'}||^2}{2\sigma^2}\right), \tag{3.10}$$

where $\mathbf{x}_f$ and $\mathbf{x}_{f'}$ correspond to the pixel vectors at different spectral bands. The kernel variance, σ^2, represents the allowable within class variance and it is a crucial parameter because it controls the data in the infinite dimensional space, and thus affects the shape of the kernel covariance matrix and inherently, the overall clustering accuracy. An incorrect choice of the σ^2 value can thus result a non-informative kernel covariance matrix which may be insufficient in providing acceptable clustering performance.

To demonstrate the importance of σ^2 in pixel clustering, consider an example with 20 hyperspectral image pixels, from the Indian Pines dataset [51], representing 4 different materials/classes. The pixels are arranged such that each set of 5 sequential pixels represent one of the 4 different materials. Using a mapping derived by employing a Gaussian kernel of the most suitable variance, we obtain a covariance matrix like the one shown in Fig. 3.1(a). The four visible diagonal blocks correspond to the highly correlated pixels observing the same material, while the low magnitude in the remaining entries of the covariance matrix shows the uncorrelatedness between pixels of different materials.

However, if the value of σ^2 is set too high, as per (6.5), irrespective of the value of $||\mathbf{x}_f - \mathbf{x}_{f'}||^2$, all the entries of the covariance matrix will be close to one. Such a covariance matrix provides no meaningful information towards the clustering task considered here. The kernel covariance matrix for a variance $\sigma^2 = 10^{10}$ is shown in Fig. 3.1(b). On the other hand, if we choose a variance σ^2 that is too small, then the diagonal covariance entries for which $\mathbf{x}_f$ would be equal to $\mathbf{x}_{f'}$, are equal to one, while the majority of the nondiagonal entries will be close to zero as demonstrated in Fig. 3.1(c). This diagonal covariance matrix



(a)

(b)

(c)

Figure 3.1: Effect of selecting different kernel variance σ^2 in the kernel covariance structure: (a) $\sigma^2 = 10^{3.5}$, (b) $\sigma^2 = 10^{10}$, (c) $\sigma^2 = 10^{-5}$

is not pertinent for the clustering task considered here. Through extensive numerical tests, it has been observed that as the kernel variance σ^2 increases within the range 10^{-5} to 10^{10} the approximately diagonal kernel covariance matrix depicted in Fig.3.1(c) transforms into a block diagonal matrix given in Fig. 3.1(a) before transforming into the matrix depicted in Fig. 3.1(b) whose entries are all equal. This reflects that the matrix elements corresponding to the covariance between pixels from the same material increase in magnitude much earlier than the elements corresponding to covariance across different materials as the variance σ^2 is increased. The block diagonal structure of the covariance matrix is highly desirable as diagonal blocks point to the hyperspectral pixels that need to be clustered together. Clearly, from Fig. 3.1(a), there are 4 block diagonals, each of the size 5×5, that have a large magnitude. This basically indicates that amongst the pixels considered as input, the first 5 are derived from the same source or represent the same material and so on for the other three sets of 5 pixels.

Let the variability ν_f of the kernel covariance matrix entries for spectral band f be defined as

$$\nu_f = \sum_{i=1}^{p/2}\sum_{j=1}^{p/2}(\hat{\mathbf{K}}_{\mathbf{x}}^f(i,j) - \bar{m}_{\hat{\mathbf{K}}_{\mathbf{x}}^f})^2, \tag{3.11}$$

where $\hat{\mathbf{K}}_{\mathbf{x}}^f(i,j)$ corresponds to the (i,j)th entry of $\boldsymbol{\phi}_x(\mathbf{x}_f)\boldsymbol{\phi}_x^T(\mathbf{x}_f)$ at spectral band f while

$$\bar{m}_{\hat{\mathbf{K}}_{\mathbf{x}}^f} = \frac{1}{(p/2)^2}\sum_{i=1}^{p/2}\sum_{j=1}^{p/2}\hat{\mathbf{K}}_{\mathbf{x}}^f(i,j) \tag{3.12}$$

corresponds to the average of the kernel covariance entries.

Interestingly, numerical experiments show that ν_f increases as the value of σ^2 increases until it reaches a maximum value during which covariance $\hat{\mathbf{K}}_{\mathbf{x}}^f$ becomes block diagonal in structure. After this point ν_f decreases as σ^2 keeps increasing. Therefore, ν_f can be used as a metric to choose a pertinent value for the variance σ^2 value by observing when ν_f reaches its maximum value.

As discussed earlier, as the σ^2 increases, the block diagonal elements of the kernel covariance matrix increase in magnitude. Simultaneously, the variability ν_f also increases since the magnitude of only a few elements (the block diagonals) has increased and all the other elements are still constant. But post a transition point in the range of σ^2, all the other elements of the covariance matrix increase in magnitude too and as a result ν_f decreases beyond this transition point. At the transition point, the block diagonal structure is most well formed. Intuitively, the variability, namely ν_f is an indicator of the informativeness of the kernel covariance matrix, and thus a large value for ν_f is desirable.

For each spectral band $f \in \mathcal{F}_s$, a grid of n possible values of the variance σ^2 is formed. For each of these n values and spectral bands the corresponding kernel covariance matrices $\hat{\mathbf{K}}_{\mathbf{x}}^{f}$ are evaluated by employing the kernel trick. Using (3.11), n different variabilities ν_f^i, $i = 1, \ldots, n$ are calculated for each spectral band f. For each spectral band f, the variance resulting the maximum variability $\nu_f^{\max} := \max_{i=1,\ldots,n}(\nu_f^i)$ is then used as the kernel variance parameter to evaluate the corresponding kernel covariance $\hat{\mathbf{K}}_{\mathbf{x}}^{f}$ (a similar process is done also for $\hat{\mathbf{K}}_{\mathbf{y}}^{f}$ and $\hat{\mathbf{K}}_{\mathbf{xy}}^{f}$).

3.3 Covariance Weighting Scheme

Among the F spectral bands present in a hyperspectral image, not all spectral bands exhibit the same amount of useful information that can be exploited via kernel-based correlation analysis to perform clustering. In fact, in many datasets, some spectral bands are removed as a preprocessing measure, because atmospheric water absorption causes little to no information from the objects to reach the image sensor at all. The information present in a particular spectral band will be dependent on the nature of the object present in the image, as well as its particular spectral reflectance. Thus, the most informative spectral bands correspond to the ones in which the objects/materials of interest exhibit the strongest spectral responses.

To this end, we utilize an unsupervised scheme similar to the one discussed for kernel variance selection in Section 3.2 to sort the spectral bands according to their informativeness. Specifically, the spectral bands are allocated weights in proportion to the information they contain. Thus, the kernel covariance matrices in (3.7)-(3.9) utilized by the KRCC framework will be formed as a weighted average giving more emphasis to the informative spectral bands.

To determine the weights in (3.7)-(3.9), the variability measure ν_f^{max} for each of the $|\mathcal{F}_s|$ covariance matrices is utilized. As mentioned in Section 3.2, a high value for ν_f^{max} implies more information. Thus, the highest ν_f^{max} will point to the spectral band which would provide the most useful information pertaining to the materials/objects represented by the pixels. The weights w_f^x, w_f^y and w_f^{xy} are calculated as a proportion of the variability ν_f^{max} obtained from $\hat{\mathbf{K}}_x^f, \hat{\mathbf{K}}_y^f$ and $\hat{\mathbf{K}}_{xy}^f$ respectively, where $f \in \mathcal{F}_s$. The weights are then normalized such that their sum equals one. Thus, for the weights w_f^x, we have

$$w_f^x = \frac{\nu_f^{max}}{\sum_{f=1}^{|\mathcal{F}_s|} \nu_f^{max}}, \tag{3.13}$$

while w_f^y and w_f^{xy} can be found similarly.

3.4 Stitching Algorithm

Even though having much greater information along the third dimension corresponding to spectral reflectance in various spectral bands causes a tradeoff in having a lower spatial resolution compared to RGB images, the resulting spatial resolution in hyperspectral images is still substantial causing high complexity issues when attempting to cluster all pixels on the image. To address complexity we devise the following three-stages scheme:

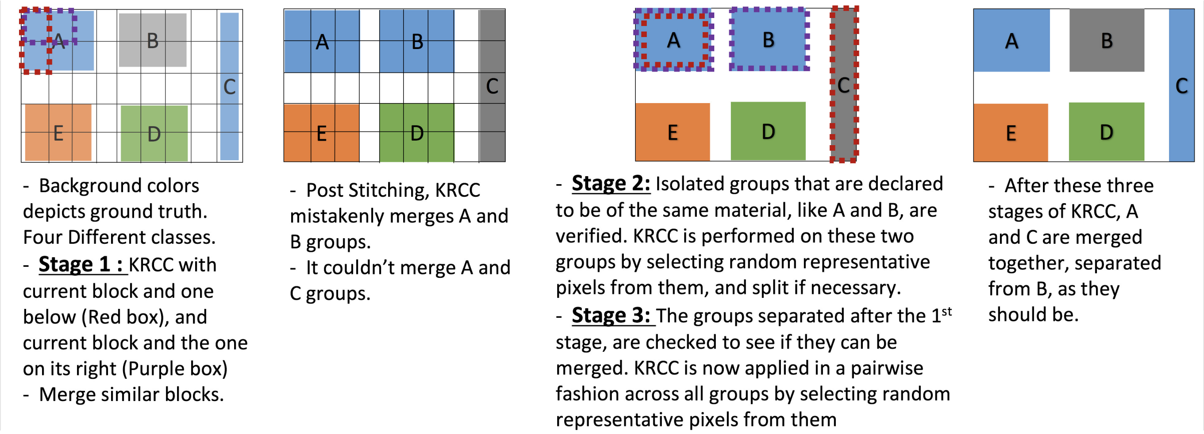
3.4.1 Stage 1

We split the image into small 5×5 patches of pixels, and assume all labeled pixels in each patch to represent the same material. Beginning from the upper leftmost portion, we take each portion and perform the KRCC framework with adjacent portions, and recursively perform this operation until the opposite corner of the image is reached, and the entire image has been covered.

For each pixel patch, KRCC is performed with the neighboring portion on its right, then with the neighboring one on its bottom, using a small number (8 were used in our experiments) of $representative\ pixels$ randomly selected from the patches to form the KRCC input vectors $\mathbf{x}_f$ and $\mathbf{y}_f$. For a relatively large patch of pixels representing the same material, this would imply that by implementing this broom-like scheme, such patches will be initially split into multiple separate portions, and then eventually be merged together. If at some point, two adjacent 5×5 portions are merged, then we would not take representative pixels from only one of these merged 5×5 portions, we would take them from *all* pixels that were merged together from previous steps. This would ensure a more accurate pattern that would be exhibited by an object represented by these merged pixels, since the same object might exhibit pronounced variability across certain spectral bands depending on surrounding conditions (For example, the same crop would exhibit spectral reflectance values if growing on healthy soil, which might be profoundly different from the same crop growing in unhealthy soil). If, however, KRCC deems that two adjacent portions do not belong together, then they will not be merged. Note that choosing a small selection of representative pixels is crucial to avoid the immense computational complexity arising from generating kernel matrices of large magnitudes. Since the time in the generation of kernel matrices increases exponentially with the size increase of the matrices.

3.4.2 Stage 2

After the initial stitching algorithm, KRCC may cluster in the same group patches of data separated by unlabeled pixels. However, if such adjacent patches are close together, they will exhibit some similarity in spectral reflectance that will be caused by its adjacency in the spatial domain. This will cause patches surrounded by unlabeled pixels which are from different sources, to be merged together by KRCC. To alleviate this, after completing the stitching algorithm, isolated patches of data that KRCC groups together will undergo another KRCC step, where the number of rows in clustering matrices $\mathbf{D}$ and $\mathbf{E}^T$ will be the same as the number of isolated patches present. This will utilize a larger number of representative pixels because we now have patches that are much greater than 5x5 portions,



- Background colors depicts ground truth. Four Different classes.
- **Stage 1 :** KRCC with current block and one below (Red box), and current block and the one on its right (Purple box)
- Merge similar blocks.

- Post Stitching, KRCC mistakenly merges A and B groups.
- It couldn't merge A and C groups.

- **Stage 2:** Isolated groups that are declared to be of the same material, like A and B, are verified. KRCC is performed on these two groups by selecting random representative pixels from them, and split if necessary.
- **Stage 3:** The groups separated after the 1st stage, are checked to see if they can be merged. KRCC is now applied in a pairwise fashion across all groups by selecting random representative pixels from them

- After these three stages of KRCC, A and C are merged together, separated from B, as they should be.

Figure 3.2: Summary of the proposed algorithm showing the three stages of the algorithm using a hyperspectral example having 4 classes spreading across 5 patches.

and thus more pixels will make it easier for KRCC to find a correlation that will effectively cluster together patches containing the same object.

3.4.3 Stage 3

Alternatively, we might come across patches in the image from the same object, but distant from each other from a spatial standpoint. Since the stitching algorithm will consider only adjacent patches, such cases (depending on the distribution of the objects in the image) would cause these patches to be misclassified as separate objects. To this end, after the KRCC stage mentioned above, we implement another $pairwise$ KRCC, where this time it is implemented across separated patches, as opposed to merged patches. This will ensure that we have considered all possible combinations of the data existing in the hyperspectral image. The stiching algorithm is summarized in Fig. 3.3

3.5 Algorithmic Details

When generating the kernel matrices according to (3.7)-(3.9) the transformed data $\phi_x(\mathbf{x}_f)$ and $\phi_y(\mathbf{y}_f)$ need to be centered by subtracting the 'average' quantities from them

$\bar{\boldsymbol{\phi}}_x(\mathbf{x}) := F_s^{-1}\sum_{f\in\mathcal{F}_s}\boldsymbol{\phi}_x(\mathbf{x}_f)$ and $\bar{\boldsymbol{\phi}}_y(\mathbf{y}) := F_s^{-1}\sum_{f\in\mathcal{F}_s}\boldsymbol{\phi}_y(\mathbf{y}_f)$, respectively [42]. This eventually leads to the centering operation applied in the kernel matrices as shown below. The centered kernel matrices can be formed using the following formula:

$$\hat{\mathbf{K}}_x = F_s^{-1}\textstyle\sum_{f\in\mathcal{F}_s}[\boldsymbol{\phi}_x(\mathbf{x}_f) - \bar{\boldsymbol{\phi}}_x(\mathbf{x})][\boldsymbol{\phi}_x(\mathbf{x}_f) - \bar{\boldsymbol{\phi}}_x(\mathbf{x})]^T$$

where the (i,j)th entry of $\hat{\mathbf{K}}_x$, namely $[\hat{\mathbf{K}}_x]_{i,j}$ can be written as $[\hat{\mathbf{K}}_x]_{i,j} = F_s^{-1}\sum_{f\in\mathcal{F}_s}[\hat{\mathbf{K}}_x(f)]_{i,j}$ with

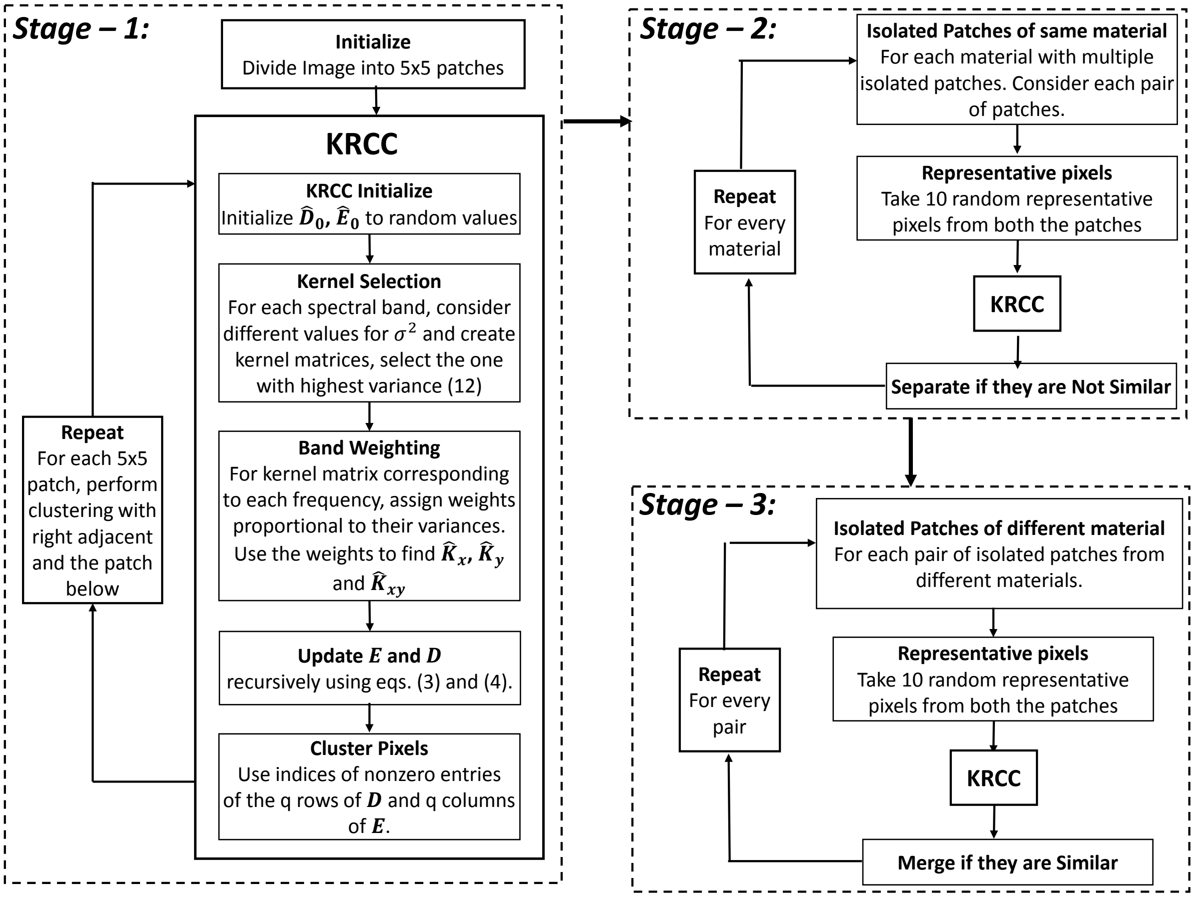
$$\begin{aligned}[\hat{\mathbf{K}}_x(f)]_{i,j} :&= [[\boldsymbol{\phi}_x(\mathbf{x}_f) - \bar{\boldsymbol{\phi}}_x(\mathbf{x})]_i]\cdot[[\boldsymbol{\phi}_x(\mathbf{x}_f) - \bar{\boldsymbol{\phi}}_x(\mathbf{x})]_j]^T \\ &= \mathbf{k}_x^{i,j}(f,f) - F_s^{-1}\textstyle\sum_{f'\in\mathcal{F}_s}\mathbf{k}_x^{i,j}(f,f') \\ &- F_s^{-1}\textstyle\sum_{f'\in\mathcal{F}_s}\mathbf{k}_x^{j,i}(f,f') + F_s^{-2}\textstyle\sum_{f',f''\in\mathcal{F}_s}\mathbf{k}_x^{i,j}(f',f''),\end{aligned} \tag{3.14}$$

while $\mathbf{k}_x^{i,j}(f,f') := [\boldsymbol{\phi}_x(\mathbf{x}_f)]_i\cdot\boldsymbol{\phi}_x(\mathbf{x}_{f'})]_j$ while $[\boldsymbol{\phi}_x(\mathbf{x}_f)]_i$ represents the ith row of $\boldsymbol{\phi}_x(\mathbf{x}_f)$. This is a row vector of infinite dimensionality, and it can be determined from the RBF kernel in (6.5) (see kernel trick in [17]).

As discussed in Section 3.3, data from some of the frequency components is more informative than the others and thus we introduce the idea of weighing each component in proportion to its informativeness. To incorporate this weighting, the centering scheme has to be altered too. Therefore we appropriately scale the $\boldsymbol{\phi}_x(\mathbf{x}_f)$ with the weights to get $\sqrt{w_f^x}\boldsymbol{\phi}_x(\mathbf{x}_f)$. The average is then given by $\bar{\boldsymbol{\phi}}_x(\mathbf{x}) := F_s^{-1}\sum_{f\in\mathcal{F}_s}\boldsymbol{\phi}_x(\mathbf{x}_f)\sqrt{w_f^x}$ and the centered kernel with the weighting factor is given as:

$$\begin{aligned}w_f^x[\hat{\mathbf{K}}_x(f)]_{i,j} :&= [[\boldsymbol{\phi}_x(\mathbf{x}_f)\sqrt{w_f^x} - \bar{\boldsymbol{\phi}}_x(\mathbf{x})]_i]\cdot[[\boldsymbol{\phi}_x(\mathbf{x}_f)\sqrt{w_f^x} - \bar{\boldsymbol{\phi}}_x(\mathbf{x})]_j]^T \\ &= w_f^x\mathbf{k}_x^{i,j}(f,f) - F_s^{-1}\big(\textstyle\sum_{f'\in\mathcal{F}_s}\mathbf{k}_x^{i,j}(f,f') + \sum_{f'\in\mathcal{F}_s}\mathbf{k}_x^{j,i}(f,f')\big)\sqrt{w_f^x w_{f'}^x} \\ &+ F_s^{-2}\sum_{f',f''\in\mathcal{F}_s}\mathbf{k}_x^{i,j}(f',f'')\sqrt{w_{f'}^x w_{f''}^x},\end{aligned} \tag{3.15}$$

to be used in equations (3.7)-(3.9).



Stage – 1:
Initialize
Divide Image into 5x5 patches
KRCC
KRCC Initialize
Initialize $\widehat{D}_0$, $\widehat{E}_0$ to random values
Kernel Selection
For each spectral band, consider different values for σ^2 and create kernel matrices, select the one with highest variance (12)
Band Weighting
For kernel matrix corresponding to each frequency, assign weights proportional to their variances. Use the weights to find $\widehat{K}_x$, $\widehat{K}_y$ and $\widehat{K}_{xy}$
Update E and D
recursively using eqs. (3) and (4).
Cluster Pixels
Use indices of nonzero entries of the q rows of D and q columns of E.
Repeat
For each 5x5 patch, perform clustering with right adjacent and the patch below
Stage – 2:
Isolated Patches of same material
For each material with multiple isolated patches. Consider each pair of patches.
Representative pixels
Take 10 random representative pixels from both the patches
KRCC
Separate if they are Not Similar
Repeat
For every material
Stage – 3:
Isolated Patches of different material
For each pair of isolated patches from different materials.
Representative pixels
Take 10 random representative pixels from both the patches
KRCC
Merge if they are Similar
Repeat
For every pair

Figure 3.3: Summary of the proposed algorithm showing the three stages in a block diagram

We initialize the entries of $\mathbf{D}$ and $\mathbf{E}$ with random values at the first iteration, where $k = 0$ in (3.2). The novel clustering algorithm is summarized in the block diagram as shown in Fig 3.3.

3.6 Numerical Results

The Salinas dataset contains 54129 labeled pixels, making this the most time-consuming dataset among the ones implemented. The resolution is $512 \times 217 \times 224$, with 224 being the number of spectral bands. Traditionally, 204 spectral bands are used after removing 20 bands containing atmospheric water absorption, but in the KRCC scheme, these contain the lowest variances in the corresponding kernel covariance matrix, thus their weights will be extremely low, implying that identifying and isolating such spectral bands will be unnecessary. Furthermore, we found with further testing, that including these spectral bands can improve the clustering performance, implying that even such spectral bands will have some information useful to our novel clustering scheme.

For the initial stitching algorithm, we take 8 representative pixels from each 5x5 portion, while the values of the λ parameters were chosen to be 0.5. This finally resulted in approximately 45 classes, where 34 of them contained less than 150 pixels, and were subsequently ignored as negligible errors. What remained were 11 classes, where there are actually 16 in total, implying that several classes were misclassified as belonging together. Since KRCC took isolated patches surrounded by unlabeled pixels to belong to the same object, we can conclude that each of these patches contain one object, as opposed to containing multiple objects. Following with this conclusion, we take those combination of patches that the stitching algorithm concluded to belong to the same object, and run KRCC on them again. This time, however, we take 10 representative pixels from each isolated patch, and use a smaller value of λ parameters, equal to 0.1.

After this step, we have 16 classes in total, but we need to take into consideration that we might obtain multiple patches that actually belong to the same object, but are distant enough from each other so that the stitching algorithm will fail to merge them. To this end, we take 10 representative pixels from each class as declared by the first two stages of KRCC, and run a final stage of KRCC among each other in a pairwise manner, meaning that KRCC will be implemented on two classes at a time. Comparisons were made with KSVM, Kmeans and KSEM [10], where Table 4.5 shows the accuracy of clustering for each class, as well as average and overall accuracy of different methods. Note that for the obtained results, the values pertaining to KSEM were quoted directly from [10]. We can observe that KRCC not only measures several objects with complete accuracy, but overall maintains

Class names	KSVM (1% train)	Kmeans	Biophy- -sical method	KSEM	KRCC
Brocoli Green Weeds 1	96.72	98.25	62.71	93.34	**100**
Brocoli Green Weeds 2	95.82	43.86	25.06	**99.67**	90
Fallow	95.30	63.78	30.06	75.64	**100**
Fallow Rough Plow	95.01	95.48	64.06	**99.19**	56.25
Fallow Smooth	**94.77**	58.75	52.83	91.37	40.96
Stubble	94.63	61.19	32.83	**99.73**	92.28
Celery	**97.14**	69.46	27.66	77.51	59.31
Grapes Untrained	69.88	49.87	18.15	**99.57**	98.70
Soil Vinyard Develop	96.68	77.01	46.10	95.65	**100**
Corn Senesced Green Weeds	80.54	35.07	27.18	63.26	**100**
Lettuce Romaine 4wk	88.96	7.45	39.79	47.07	**100**
Lettuce Romaine 5wk	98.02	4.58	40.84	100	**100**
Lettuce Romaine 6wk	94.93	40.27	55.45	88.92	**100**
Lettuce Romaine 7wk	79.51	36.60	31.30	88.94	**100**
Vinyard Untrained	**62.08**	44.07	18.91	0	20
Vinyard Vertical Trellis	97.00	32.69	15.44	99.22	**100**
Overall Accuracy	**84.65**	58.96	31.55	79.17	81.14
Average Accuracy	**89.81**	51.15	36.77	82.44	84.84

Table 3.1: Clustering Performance in Salinas for 0% dead pixels.

a performance that is of a competitive degree with a supervised method, despite being unsupervised. Objects such as Fallow-Smooth and Fallow-Stubble are different forms of the same object, and thus would exhibit spectral patterns that may be too similar for KRCC to accurately separate, thus some challenge in this regard is expected.

Next, we introduce dead pixels within the data, and repeat the whole process for this corrupted data. Fig. 3.4 illustrates the classification accuracies of different methods based on varying percentages of dead pixels within the given data. It can be clearly observed, that KRCC is not only on par with KSVM with no dead pixels, it is capable of consistent performance even when missing pixels are implemented. The performance degradation of KSVM is even further than for Kmeans after going beyond 1 % of dead pixels. This is

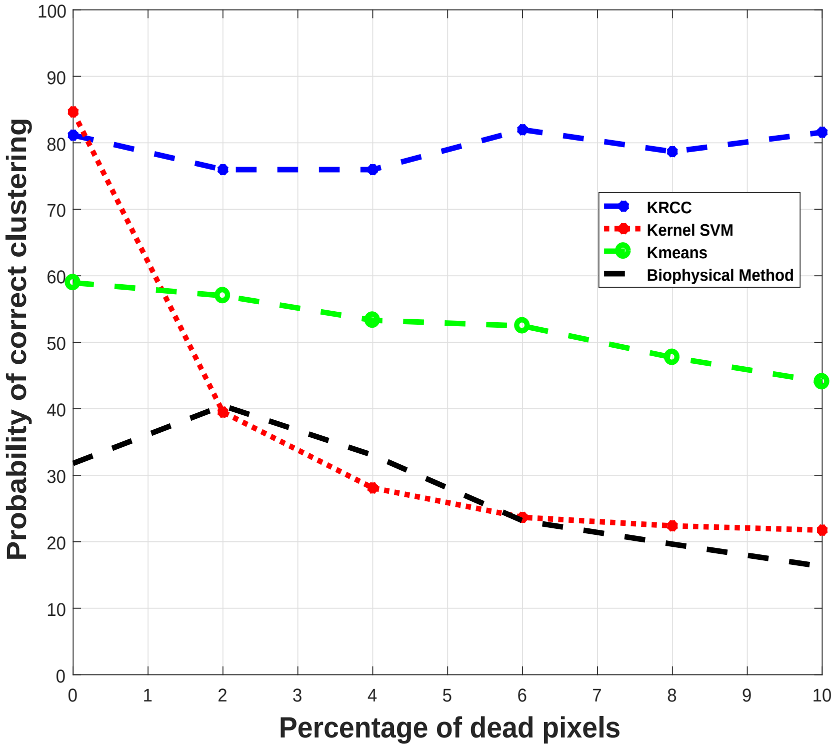


Figure 3.4: Probability of correct clustering versus percentage of dead pixels in Salinas

because even though both methods will suffer with missing entries, since any change of value along even one dimension will alter the position of the data points when we consider its mapping along a Euclidean space, thus making clustering difficult for such points, but this is even more disadvantageous for KSVM, since if such corrupted pixels are randomly selected to be training samples, then the classifier will attempt to cluster data with a corrupted training scheme. However, since we are taking a weighted average of the kernel matrices for KRCC, such entries containing missing data will have little impact when they are among other entries that will be averaged out, thus explaining the consistency of performance in KRCC. A side-by-side comparison of the estimated map in comparison with the reference map has been illustrated in Fig. 3.5.

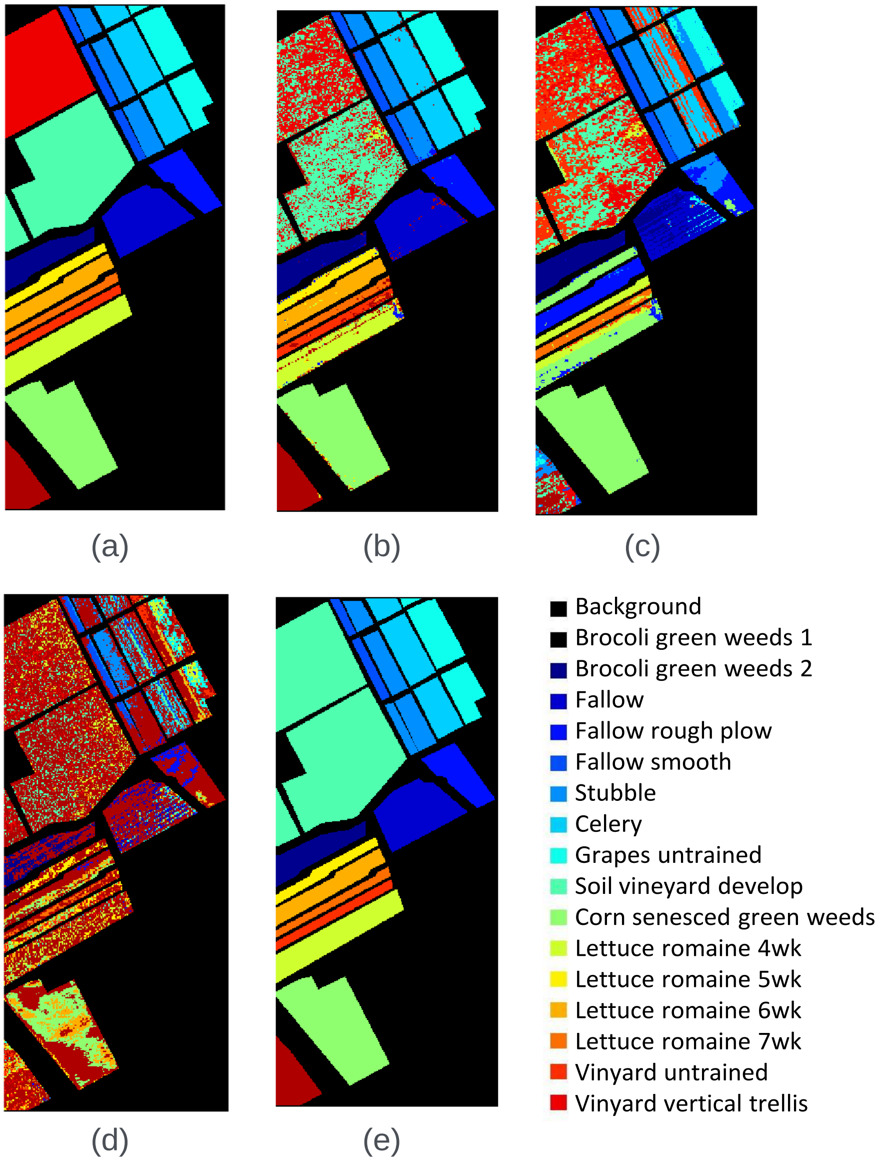


Figure 3.5: Maps of Salinas generated by various methods compared to the reference map, (a) Ground Truth, (b) KSVM, (c) Kmeans, (d) Biophysical method, (e) KRCC

CHAPTER 4

KERNELIZED CCA FOR UNMIXING

As mentioned previously, the hyperspectral image pixels have higher spectral resolution at the sacrifice of a lower spatial resolution. If a hyperspectral remote sensing image is acquired across a large area, then each pixel will encompass an area that can contain multiple materials. This will result in a pixel containing spectral reflectances that are a mixture of those materials, as opposed to containing only one material. To this end, it is necessary to *unmix* the spectral reflectances of the contributing materials, a nontrivial problem when tackled from an unsupervised perspective.

4.1 Unsupervised Hyperspectral unmixing

Consider a set of hyperspectral pixels that contain information across S spectral bands, while let set $\mathcal{S} := \{1, \ldots, S\}$ contain the indices of the spectral bands across which information is gathered. Further, let $\mathbf{M} \in \mathbb{R}^{Y \times S}$ contain the intensities of Y pixels of a hyperspectral image across the spectral bands in $\mathcal{S}$. The linear mixing model (LMM) has been used extensively to emulate the mixing effect [11]. In this model, the reflectance values from a mixed pixel is assumed to be a linear combination of multiple objects. Such materials are ‘pure’ (no mixing exists in them), and they are known as *endmembers*. For the remainder of this chapter, the terms *endmembers* and *pure pixels* will be used interchangeably.

The mixed hyperspectral pixels in matrix $\mathbf{M}$ can be modeled via the linear mixing of R endmembers which gives

$$\mathbf{M} = \mathbf{P} \cdot \mathbf{U} + \boldsymbol{\epsilon}, \tag{4.1}$$

where $\mathbf{P} \in \mathbb{R}^{Y \times R}$ represents the matrix of abundances, where each entry quantifies the contribution of each endmember, i.e., $\mathbf{P}_{i,j} \in [0,1]$, while the rows of $\mathbf{U} \in \mathbb{R}^{R \times S}$ contain the unknown endmembers, and $\boldsymbol{\epsilon}$ represents white Gaussian noise. Since the abundances represent the fractional contributions of endmembers, by definition the entries of $\mathbf{P}$ would have nonnegative entries, while the sum of each row of $\mathbf{P}$ equals to 1. Further, as the entries of $\mathbf{U}$ represent reflectance values, they will be nonnegative as well. Furthermore, the data provided is assumed to be normalized to unit energy, thus $\mathbf{U}_{i,j} \in [0,1]$.

Mixtures in hyperspectral data can be more accurately represented by a nonlinear mixing model as opposed to a linear one [19,20,55]. A better representation of the mixing effect can be achieved by utilizing a polynomial nonlinear mixing model, commonly known as the Fan model [21], which for the ith row of $\mathbf{M}$ gives

$$\mathbf{M}_{i,:} = \sum_{\text{J}=1}^{R} \mathbf{P}_{i,j}\mathbf{U}_{j,:} + b \sum_{j=1,l=j+1}^{R-1,R} (\mathbf{P}_{i,j}\mathbf{P}_{i,l})\mathbf{U}_{j,:} \odot \mathbf{U}_{l,:} + \boldsymbol{\epsilon}, \tag{4.2}$$

for $i = 1, 2, \ldots, Y$, while $\odot$ represents the Hadamard product (elementwise product), while b is a coefficient that controls the strength of the nonlinear polynomial part. Note that $b = 0$ reverts (4.2) back to the LMM in (6.1).

The set of pixels in $\mathbf{M}$ is split evenly into two vectors $\mathbf{x}_s \in \mathbb{R}^{p_x \times 1}$ and $\mathbf{y}_s \in \mathbb{R}^{p_y \times 1}$, that contain the reflectance values of the pixels at spectral band s, while $p_x + p_y = p$, and the pixels are non-overlapping among the two sets. We also assume that the pixels in both the $\{\mathbf{x}_s\}$ and $\{\mathbf{y}_s\}$ sets contain information regarding all R materials present.

The CCA formulation can initially be considered to be similar to the formulation described in Eq. 2.1 in Chapter 2, but as shown below, the $\mathbf{D}$ and $\mathbf{E}$ matrices have added constraints.

$$(\hat{\mathbf{E}}, \hat{\mathbf{D}}) = \arg\min_{\mathbf{E},\mathbf{D}} S^{-1} \textstyle\sum_{s\in\mathcal{S}} ||\mathbf{y}_s - \mathbf{E}\mathbf{D}\mathbf{x}_s||_2^2 + \sum_{\gamma=1}^{R} \lambda_\gamma^{\mathbf{E}} ||\mathbf{E}_{:\gamma}||_1 + \sum_{\gamma=1}^{R} \lambda_\gamma^{\mathbf{D}} ||\mathbf{D}_{\gamma:}||_1, \tag{4.3}$$

$$\text{s. to } \mathbf{D}(\gamma, j) \geq 0, \mathbf{E}(i, \gamma) \geq 0,\ \mathbf{E}(i, :)\mathbf{1} = 1,$$

where $i, j = 1, \ldots p,\ \gamma = 1, \ldots, R$, and the operator $\|\cdot\|_1$ denotes norm-one, with $\mathbf{1}$ indicating a vector of ones. The matrix $\hat{\mathbf{E}} \in \mathbb{R}^{p_y \times R}$ can be viewed as an estimate of the abundances of the endmembers that form the pixel intensities in $\mathbf{y}_s$ for spectral band s. The nonnegativity constraints for $\mathbf{D}$ and $\mathbf{E}$ correspond to the nonnegativity property of the entries in matrices $\mathbf{P}$ and $\mathbf{U}$. Proper selection of the nonnegative parameters $\lambda_\gamma^{\mathbf{E}}$ and $\lambda_\gamma^{\mathbf{D}}$, controls the population of zeros, known as sparsity, in $\mathbf{E}$ and $\mathbf{D}$ corresponding to the rationale that only a handful of endmembers contribute to a mixed pixel.

Similar to (4.3), we can formulate a similar equation for the abundances pertaining to the mixed pixels in the set $\mathbf{x}_s$

$$(\hat{\mathbf{E}'}, \hat{\mathbf{D}'}) = \arg\min_{\mathbf{E}',\mathbf{D}'} S^{-1} \textstyle\sum_{s\in\mathcal{S}} ||\mathbf{x}_s - \mathbf{D}'\mathbf{E}'\mathbf{y}_s||_2^2 + \sum_{\gamma=1}^{R} \lambda_\gamma^{\mathbf{D}'} ||\mathbf{D}'_{:\gamma}||_1 + \sum_{\gamma=1}^{R} \lambda_\gamma^{\mathbf{E}'} ||\mathbf{E}'_{\gamma:}||_1, \tag{4.4}$$

$$\text{s. to } \mathbf{D}'(i, \gamma) \geq 0, \mathbf{E}'(\gamma, j) \geq 0,\ \mathbf{D}'(i, :)\mathbf{1} = 1.$$

The matrices $\hat{\mathbf{D}'} \in \mathbb{R}^{p_x \times R}$ and $\hat{\mathbf{E}'} \in \mathbb{R}^{R \times p_y}$ have similar definitions as the $\mathbf{E}$ and $\mathbf{D}$ matrices given earlier, except here, they correspond to the pixels in $\mathbf{x}_s$, and as such the nonnegativity constraints similarly apply, while the unit summation constraints will apply to the rows of $\mathbf{D}'$.

4.2 Regularized Kernel-Based Correlation Analysis

Akin to the previous chapter, the kernelized version of the CCA formulation in 4.3 can be written as follows to create the kernel regularized canonical correlation unmixing (KRCCU) formulation

$$\arg\min_{\mathbf{E},\mathbf{D}} \text{tr}(\hat{\mathbf{K}}_{\mathbf{y}} - 2\cdot\mathbf{E}\cdot\mathbf{D}\cdot\hat{\mathbf{K}}_{\mathbf{xy}} + \mathbf{E}\cdot\mathbf{D}\cdot\hat{\mathbf{K}}_{\mathbf{x}}\cdot\mathbf{D}^T\cdot\mathbf{E}^T) + \textstyle\sum_{\gamma=1}^{R} \lambda_{\gamma}^{\mathbf{E}} \mathbf{1}^T \mathbf{E}_{:\gamma} + \sum_{\gamma=1}^{R} \lambda_{\gamma}^{\mathbf{D}} \mathbf{D}_{\gamma:}\mathbf{1}, \quad (4.5)$$

$$\text{s. to } \mathbf{D}(\gamma, j) \geq 0, \mathbf{E}(i, \gamma) \geq 0,\ \mathbf{E}(i, :)\mathbf{1} = 1.$$

Different from the formulation in Chapter 3, the KRCCU framework in (4.5) involves nonnegativity constraints for $\mathbf{D}$ and $\mathbf{E}$, and unit summation constraints for the rows of $\mathbf{E}$ that facilitate the task of unmixing. Due to the nonnegativity constraints on $\mathbf{E}$ and $\mathbf{D}$ the corresponding norm-one terms in (4.3) boil down to a sum. The constrained kernelized nature of (4.5) calls for a completely novel approach for tackling this unmixing framework not addressed in Chapter 3.

We can similarly obtain the transformation of the cost in (4.4) for the $\mathbf{y}_s$ data to obtain the KRCCU formula as follows

$$\begin{aligned} J(\mathbf{E}', \mathbf{D}') = \text{tr}(&\hat{\mathbf{K}}_{\mathbf{x}} - 2\mathbf{D}'\cdot\mathbf{E}'\cdot\hat{\mathbf{K}}_{\mathbf{yx}} + \mathbf{D}'\cdot\mathbf{E}'\cdot\hat{\mathbf{K}}_{\mathbf{y}}\cdot\mathbf{E}'^T\cdot\mathbf{D}'^T) \\ &+ \textstyle\sum_{\gamma=1}^{R} \lambda_{\gamma}^{\mathbf{D}'} \mathbf{1}^T \mathbf{D}'_{:\gamma} + \sum_{\gamma=1}^{R} \lambda_{\gamma}^{\mathbf{E}'} \mathbf{E}'_{\gamma:}\mathbf{1} \end{aligned} \quad (4.6)$$

It should be noted that in this case, though the nonnegativity constraints hold for $\mathbf{E}'$ and $\mathbf{D}'$ akin to before, here the unit summation constraints will be considered for each row of $\mathbf{D}'$, as opposed to each row of $\mathbf{E}$ previously.

4.3 Kernel Selection and Weighting

While the same problem with kernel variance selection from Sec. 3.5 exists here, a novel approach was taken to tackle this problem, from some preliminary work in [22].

We first consider V different values for the kernel variance on a discrete 1-D grid. Let $\hat{\mathbf{K}}_x^j(s)$ correspond to the kernel covariance matrix obtained when the jth kernel variance value is used. Having R endmembers forming the mixed pixels, and also assuming that for each pixel, we have one endmember whose abundance is higher than the rest, allows the grouping of the pixels in R groups, where each group has one dominant endmember different from the other groups. When the kernel variance is selected properly such that it linearizes inter-pixel correlations data, then the kernel-transformed pixel covariance matrices, namely $\hat{\mathbf{K}}_x(s)$ and $\hat{\mathbf{K}}_y(s)$, are expected to contain R row- and column-permuted diagonal block submatrices of rank one (when sensing noise is weak). This stems from the presence of R dominant and uncorrelated endmembers that result in R uncorrelated groups of correlated pixels.

To this end, the objective is to select a value of the variance whose resultant $\hat{\mathbf{K}}_x(s)$ has a rank close to R. In detail, a proper variance value is chosen such that it results in a matrix $\hat{\mathbf{K}}_x(s)$ with the strongest R^{th} eigenvalue, which leads to a covariance rank close to R (see also [22]). For this purpose, we obtain the eigenvalues and their corresponding eigenvectors for all V possible kernel matrices $\{\hat{\mathbf{K}}_x^j(s)\}_{j=1}^V$ for variance values $\{\sigma_j^2\}_{j=1}^V$. For the jth kernel matrix, consider the eigenvectors to be $\{\mathbf{v}_1^j, \ldots, \mathbf{v}_i^j, \ldots, \mathbf{v}_{p_x}^j\}$, and the corresponding eigenvalues to be $\{\lambda_1^j, \ldots, \lambda_i^j, \ldots, \lambda_{p_x}^j\}$, where the λ_i^j's are sorted such that $\lambda_1^j \geq \ldots \geq \lambda_{p_x}^j$. The desired kernel covariance matrix $\hat{\mathbf{K}}_x^{j\star}(s)$ is chosen by finding variance σ_j^{*2} maximizing

$$\max_{\{\sigma_j^2\}_{j=1}^V} ||\hat{\mathbf{K}}_x^j(s) - \sum_{i=1}^{R-1} \lambda_i^j \mathbf{v}_i^j \mathbf{v}_i^{j^T}||_F^2 - ||\hat{\mathbf{K}}_x^j(s) - \sum_{i=1}^{R} \lambda_i^j \mathbf{v}_i^j \mathbf{v}_i^{j^T}||_F^2. \tag{4.7}$$

The variance value maximizing (4.7) gives a kernel covariance $\hat{\mathbf{K}}_x^{j\star}(s)$ with the strongest R^{th} eigenvalue for each spectral band s. This further implies that the resulting kernel covariance matrix $\hat{\mathbf{K}}_x^{j\star}(s)$ is expected to have a rank close to R.

When calculating the $\hat{\mathbf{K}}_x$ to be used in (4.5), one way would be to take the average of all S kernel matrices $\hat{\mathbf{K}}_x^{j\star}(s) = \boldsymbol{\phi}_x(\mathbf{x}_s)\boldsymbol{\phi}_x(\mathbf{x}_s)^T$ obtained earlier. However, since not all spectral bands yield equally useful information for unmixing, it would be advantageous to reward informative spectral bands, and consequently put less emphasis, or weights, on the less informative bands. To this end, we choose weights that are proportional to the largest difference between the R^{th} and $(R-1)^{th}$ eigenvalue for each of the S kernel matrices after (4.7) is applied. In other words, $w_s^x = \frac{\rho_R(\hat{\mathbf{K}}_x^{j\star}(s))}{\sum_{s\in\mathcal{S}}\rho_R(\hat{\mathbf{K}}_x^{j\star}(s))}$, where $\rho_R(\cdot)$ refers to the largest difference between the R^{th} and $(R-1)^{th}$ eigenvalue of $\hat{\mathbf{K}}_x^{j\star}(s)$. Thus, we obtain the final kernel matrices as $\hat{\mathbf{K}}_{\mathbf{x}} := \sum_{s\in\mathcal{S}} w_s^x \hat{\mathbf{K}}_x(s), \hat{\mathbf{K}}_{\mathbf{y}} := \sum_{s\in\mathcal{S}} w_s^y \hat{\mathbf{K}}_y(s)$ and $\hat{\mathbf{K}}_{\mathbf{xy}} := \sum_{s\in\mathcal{S}} w_s^{xy} \hat{\mathbf{K}}_{xy}(s)$.

4.4 Reducing complexity

Now, the V kernel variances used to implement this method necessitate generating S kernel matrices for each variance. For a large V, we would obtain a large range of kernel variances, which would surely include the ideal value for any provided data, but it would be time-consuming. On the other hand, using a small V would be much faster, but the best variance may be missed since the range is smaller. To achieve the best of both situations, we propose below a novel technique, that starts with a large V, and uses a small portion of the data to quickly estimate a smaller range where the ideal kernel variances will be located.

To account for the sensitivity of the kernel variance to data, we start with a wide range of kernel variance values, but only the average value of the entries of the first row for each of these kernel covariance matrices is calculated and used to shrink the range of variance values of interest. Calculating the row average value is much faster than calculating matrix eigenvalues, but it still suffices in finding a smaller proper range of variance values. After finding the shrunk range, the process in Sec. 4.3 is applied to refine the variance selection.

When the kernel variance is too small, then the resulting diagonal matrix will have a value 1 in each diagonal entry, while the rest of the entries will be close to zero. Thus, the average of each row of entries will have a small magnitude. Increasing the kernel variance would have little effect, until we reach a value close to the one maximizing the Rth eigenvalue. At that point the entries in every row, corresponding to pixels containing the same materials, will start increasing, and thus the row average along with it. However, after the ‘ideal’ variance is reached, and increased even more, the entries corresponding to pixels containing uncorrelated materials will increase as well. However, due to the kernel centering operation (see details later), the row average actually decreases when the variance is increasing more than the ‘ideal’ range. Fig. 4.1 (top) depicts the first row average value versus different variance values selected from a grid in $[10^{-19}, 10^{21}]$, where it can be seen that the ‘ideal’ variance value is around 10^5. Thus, the range of variance values can be reduced to the set $\{10^0, \ldots, 10^7\}$.

It is of interest to determine how small the largest variance value of the grid can be without missing the pertinent variance value. Fig. 4.1 (bottom) depicts the difference of the first row average values between kernel covariance matrices constructed using neighboring variance values on the grid. As the row average eventually dips by continuously increasing the kernel variance, the sharpest dip takes place between kernel variances 10^6 and 10^7. For this reason, any kernel matrix from this variance and beyond is not of interest. Thus, the ideal smaller range of kernels would be V kernel variances prior to this value. For example, considering $V = 8$, the range here would be the discrete grid of values $\{10^0, \cdots, 10^7\}$. Extensive testing has proven that this approach consistently and effectively chooses a small but informative range from a much wider range of variance values.

4.5 Regularized Kernelized Correlations Based Unmixing

To minimize the kernelized formulation in (5), we will employ the Lagrange multipliers method, see e.g., [23, Chp. 4], combined with coordinate and gradient descent iterations. Coordinate descent will be utilized to minimize the nonconvex (5) wrt $\mathbf{E}$ while fixing $\mathbf{D}$ and vice versa. The task of minimizing (5) either wrt $\mathbf{E}$ or $\mathbf{D}$ will be done via the

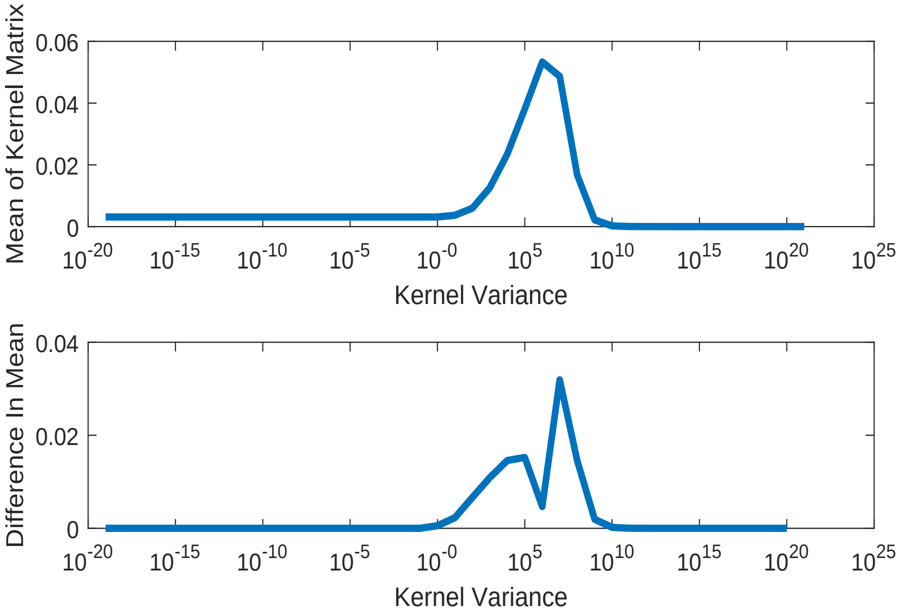


Figure 4.1: The average of the first row entries of kernel matrices versus variance (top); Difference of first-row average values between covariance matrices formed by neighboring variance grid values versus variance (bottom).

method of multipliers. We first determine the augmented Lagrangian of the cost in (4.5) and constraints in (3)

$$\begin{aligned}J_L(\mathbf{E},\mathbf{D},\mathbf{Q},\mathbf{M},\mathbf{N}) &= \mathrm{tr}(\hat{\mathbf{K}}_{\mathbf{y}} - 2\mathbf{E}{\cdot}\mathbf{D}{\cdot}\hat{\mathbf{K}}_{\mathbf{xy}} + \mathbf{E}{\cdot}\mathbf{D}{\cdot}\hat{\mathbf{K}}_{\mathbf{x}}{\cdot}\mathbf{D}^T{\cdot}\mathbf{E}^T) + \textstyle\sum_{\gamma=1}^{R} \lambda_\gamma^{\mathbf{E}} \mathbf{1}^T \mathbf{E}_{:\gamma} + \sum_{\gamma=1}^{R} \lambda_\gamma^{\mathbf{D}} \mathbf{D}_{\gamma:}\mathbf{1} \\ &+ \textstyle\sum_{i=1}^{p_y} \left[Q_i{\cdot}(\mathbf{E}_{i:}\mathbf{1} - 1) + \frac{c_1}{2}{\cdot}||\mathbf{E}_{i:}\mathbf{1} - 1||_F^2\right] + \frac{1}{2c_2}{\cdot} \sum_{i=1}^{p_y} \sum_{j=1}^{R} \{(\max\{0, M_{ij} - c_2 E_{ij}\})^2 - M_{ij}^2\} \\ &+ \frac{1}{2c_2}{\cdot} \textstyle\sum_{i=1}^{R} \sum_{j=1}^{p_x} \{(\max\{0, N_{ij} - c_2 D_{ij}\})^2 - N_{ij}^2\}, \qquad (4.8)\end{aligned}$$

where Q_i corresponds to the Lagrange multiplier associated with the unit summation constraint for the ith row of $\mathbf{E}$, while $\mathbf{Q} := [Q_1 \ldots Q_{p_y}]$, and let $\mathbf{M} \in \mathbb{R}^{p_y \times R}$ and $\mathbf{N} \in \mathbb{R}^{R \times p_x}$ represent the Lagrange multipliers corresponding to the nonnegativity constraints for $\mathbf{E}$ and $\mathbf{D}$ respectively Also, c_1 and c_2 are fixed step-size constants to be used during the update of the Lagrange multipliers corresponding to the equality and inequality constraints for $\mathbf{E}$ and $\mathbf{D}$, respectively.

Let $\mathbf{E}^{\tau}$ and $\mathbf{D}^{\tau}$ denote updates for matrices $\mathbf{E}$ and $\mathbf{D}$ at block coordinate iteration τ. During block coordinate iteration τ the update $\mathbf{E}^{\tau}$ is first formed, along with the corresponding Lagrange multiplier updates $\mathbf{Q}^{\tau+1}$ and $\mathbf{M}^{\tau+1}$, while fixing $\mathbf{D}^{\tau-1}$ and multipliers $\mathbf{N}^{\tau-1}$, according to the Lagrange multipliers method [23, Chp. 4] involving the following steps:

S.1) Determine

$$\mathbf{E}^{\tau,\ell+1} = \underset{E}{\text{argmin}}\; J_L(\mathbf{E}, \mathbf{D}^{\tau-1}, \mathbf{Q}^{\tau,\ell}, \mathbf{M}^{\tau,\ell}, \mathbf{N}^{\tau-1}), \tag{4.9}$$

S.2) Update the multipliers $\mathbf{Q}$, $\mathbf{M}$ as follows:

$$\mathbf{Q}^{\tau,\ell+1} = \mathbf{Q}^{\tau,\ell} + c_1 \cdot (\mathbf{E}^{\tau,\ell+1}\mathbf{1} - \mathbf{1}) \tag{4.10}$$

$$\mathbf{M}^{\tau,\ell+1} = \max(0, \mathbf{M}^{\tau,\ell} - c_2 \cdot \mathbf{E}^{\tau,\ell+1}). \tag{4.11}$$

Note that $\ell = 0, 1, 2...$ refers to the multipliers method iteration index, and it is running faster than block coordinate index τ (since it is nested). S.1 and S.2 are repeatedly applied until convergence, which is guaranteed from [23, Chp. 4] since the minimization problem in S.1 is convex for fixed $\mathbf{D}^{\tau-1}$ and $\mathbf{N}^{\tau-1}$, while $\mathbf{E}^{\tau} = \lim_{\ell\to\infty} \mathbf{E}^{\tau,\ell}$, $\mathbf{Q}^{\tau} = \lim_{\ell\to\infty} \mathbf{Q}^{\tau,\ell}$ and $\mathbf{M}^{\tau} = \lim_{\ell\to\infty} \mathbf{M}^{\tau,\ell}$. Note that the multipliers can be initialized using warm starts, i.e., $\mathbf{Q}^{\tau,0} = \mathbf{Q}^{\tau-1}$, and $\mathbf{M}^{\tau,0} = \mathbf{M}^{\tau-1}$.

After $\mathbf{E}^{\tau}$, $\mathbf{Q}^{\tau}$ and $\mathbf{M}^{\tau}$ have been updated, they are fixed in the augmented Lagrange function J_L, while $\mathbf{D}^{\tau}$ and its corresponding multiplier $\mathbf{N}^{\tau}$ are calculated using similar reasoning as in S.1 and S.2. Specifically:

S.3) $\mathbf{D}^{\tau,\ell+1} = \arg\min_D J_L(\mathbf{E}^{\tau}, \mathbf{D}, \mathbf{Q}^{\tau}, \mathbf{M}^{\tau}, \mathbf{N}^{\tau,\ell})$,

S.4) Update $\mathbf{N}^{\tau,\ell+1} = \max(0, \mathbf{N}^{\tau,\ell} - c_2 \cdot \mathbf{D}^{\tau,\ell+1})$.

Again S.3 and S.4 are applied recursively until convergence which is ensured since the cost involved in S.3 is convex for fixed $\mathbf{E}^{\tau}, \mathbf{Q}^{\tau}$ and $\mathbf{M}^{\tau}$. Again $\mathbf{D}^{\tau} = \lim_{\ell\to\infty} \mathbf{D}^{\tau,\ell}$, $\mathbf{N}^{\tau} = \lim_{\ell\to\infty} \mathbf{N}^{\tau,\ell}$ and the multiplier can be initialized in a warm start fashion, i.e., $\mathbf{N}^{\tau,0} = \mathbf{N}^{\tau-1}$.

Towards evaluating the minimizer updates $\mathbf{E}^{\tau,\ell+1}$ and $\mathbf{D}^{\tau,\ell+1}$ in steps S.1 and S.3 we will employ nested gradient descent iterations with updating index k which runs faster than ℓ and τ (thus this is the third nested layer). The needed derivatives of the augmented Lagrangian J_L with respect to $\mathbf{D}$ and $\mathbf{E}$ can be found as follows [for notational simplicity we dropped the dependence of J_L on $\mathbf{Q}, \mathbf{M}$ and $\mathbf{N}$ below]

$$\begin{aligned}\frac{\delta J_L(\mathbf{E},\mathbf{D})}{\delta \mathbf{D}} &= -2 \cdot \mathbf{E}^T \cdot \hat{\mathbf{K}}_{\mathbf{xy}}^T + \mathbf{E}^T \cdot \mathbf{E} \cdot \mathbf{D} \cdot \hat{\mathbf{K}}_{\mathbf{x}} + \mathbf{E}^T \cdot \mathbf{E} \cdot \mathbf{D} \cdot \hat{\mathbf{K}}_{\mathbf{x}} \\ &\quad + \boldsymbol{\lambda}^{\mathbf{D}} \mathbf{1}_{p_x}^T - max(0, \mathbf{N} - c_2 \cdot \mathbf{D}) \\ \frac{\delta J_L(\mathbf{E},\mathbf{D})}{\delta \mathbf{E}} &= -2 \cdot \hat{\mathbf{K}}_{\mathbf{xy}}^T \cdot \mathbf{D}^T + \mathbf{E} \cdot \mathbf{D} \cdot \hat{\mathbf{K}}_{\mathbf{x}}^T \cdot \mathbf{D}^T + \mathbf{E} \cdot \mathbf{D} \cdot \hat{\mathbf{K}}_{\mathbf{x}} \cdot \mathbf{D}^T \\ &\quad + \mathbf{1}_{p_y} (\boldsymbol{\lambda}^{\mathbf{E}})^T + \mathbf{Q} \cdot \mathbf{1}_R^T + c_1 \cdot (\mathbf{E} \cdot \mathbf{1}_R \cdot \mathbf{1}_R^T \\ &\quad - \mathbf{1}_{p_y} \cdot \mathbf{1}_R^T) - max(0, \mathbf{M} - c_2 \cdot \mathbf{E}), \end{aligned} \tag{4.12}$$

where $\boldsymbol{\lambda}^{\mathbf{E}} := [\lambda_1^{\mathbf{E}} \ldots \lambda_R^{\mathbf{E}}]^T$ and $\boldsymbol{\lambda}^{\mathbf{D}} := [\lambda_1^{\mathbf{D}} \ldots \lambda_R^{\mathbf{D}}]^T$. Then, the gradient descent updates for $\mathbf{D}$ and $\mathbf{E}$ for carrying out the minimization tasks in S.1 and S.3 are implemented as

$$\mathbf{E}_{k+1}^{\tau,\ell+1} = \mathbf{E}_k^{\tau,\ell+1} - c \frac{\delta J_L(\mathbf{E}_k^{\tau,\ell+1}, \mathbf{D}^{\tau-1}, \mathbf{Q}^{\tau,\ell}, \mathbf{M}^{\tau,\ell}, \mathbf{N}^{\tau-1})}{\delta \mathbf{E}}, \tag{4.13}$$

$$\mathbf{D}_{k+1}^{\tau,\ell+1} = \mathbf{D}_k^{\tau,\ell+1} - c \frac{\delta J_L(\mathbf{E}^{\tau}, \mathbf{D}_k^{\tau,\ell+1}, \mathbf{Q}^{\tau}, \mathbf{M}^{\tau}, \mathbf{N}^{\tau,\ell})}{\delta \mathbf{D}}, \tag{4.14}$$

where $\mathbf{E}_{k+1}^{\tau,\ell+1}$ refers to the gradient descent update during gradient descent iteration k towards obtaining the minimizer $\mathbf{E}^{\tau,\ell+1}$ (similarly for $\mathbf{D}_{k+1}^{\tau,\ell+1}$), while c refers to the step-size parameter used for gradient descent, and it is different than the step-sizes c_1, c_2 used to update the Lagrange multipliers. Gradient descent iterations are run for a sufficiently larger number of iterations until convergence of $\mathbf{E}_{k+1}^{\tau,\ell+1}$. Note that for proper selection of c, $\lim_{k\to\infty} \mathbf{E}_{k+1}^{\tau,\ell+1} = \mathbf{E}^{\tau,\ell+1}$, similarly $\lim_{k\to\infty} \mathbf{D}_{k+1}^{\tau,\ell+1} = \mathbf{D}^{\tau,\ell+1}$. The algorithm for determining the matrices $\mathbf{E}$ and $\mathbf{D}$ is summarized in the following table.

Note that steps 4-12 in Algorithm 1 involve the combination of multipliers method with gradient descent to find an update $\mathbf{E}^{\tau}$ for fixed $\mathbf{D}^{\tau-1}$. These steps, as explained earlier, converge due to the convexity of the associated cost when fixing $\mathbf{D}^{\tau-1}$ and corresponding

Algorithm 1 Recursive Estimation of the Abundance Matrices.

```
Initialize matrices $\mathbf{D}^0$, $\mathbf{E}^0$ randomly, and set matrices $\mathbf{Q}^0, \mathbf{M}^0, \mathbf{N}^0$ to zero.
for $\tau = 1, 2, \ldots$ do
    Set $\mathbf{Q}^{\tau,0} = \mathbf{Q}^{\tau-1}$ and $\mathbf{M}^{\tau,0} = \mathbf{M}^{\tau-1}$.
    for $\ell = 0, 1, 2 \ldots$ do
        Set $\mathbf{E}_0^{\tau,\ell+1} = \mathbf{E}^{\tau,\ell}$ obtained from recursion $\ell$.
        for $k = 0, 1, \ldots,$ do
            Update $\mathbf{E}_{k+1}^{\tau,\ell+1}$ using (14).
            If $\|\mathbf{E}_{k+1}^{\tau,\ell+1} - \mathbf{E}_k^{\tau,\ell+1}\|_2 < \epsilon$ then break gradient descent loop; return $\mathbf{E}^{\tau,\ell+1}$.
        end for
        Update $\mathbf{Q}^{\tau,\ell+1}$ and $\mathbf{M}^{\tau,\ell+1}$ via (11) and (12), respectively.
        If $\|\mathbf{E}^{\tau,\ell+1} - \mathbf{E}^{\tau,\ell}\|_2 < \epsilon$ then break multipliers loop; return $\mathbf{E}^{\tau}$.
    end for
    Set $\mathbf{N}^{\tau,0} = \mathbf{N}^{\tau-1}$.
    for $\ell = 0, 1, 2 \ldots$ do
        Set $\mathbf{D}_0^{\tau,\ell+1} = \mathbf{D}^{\tau,\ell}$ obtained from recursion $\ell$.
        for $k = 0, 1, \ldots,$ do
            Update $\mathbf{D}_{k+1}^{\tau,\ell+1}$ using (15).
            If $\|\mathbf{D}_{k+1}^{\tau,\ell+1} - \mathbf{D}_k^{\tau,\ell+1}\|_2 < \epsilon$ then break gradient descent loop; return $\mathbf{D}^{\tau,\ell+1}$.
        end for
        Update $\mathbf{N}^{\tau,\ell+1}$ via S.4.
        If $\|\mathbf{D}^{\tau,\ell+1} - \mathbf{D}^{\tau,\ell}\|_2 < \epsilon$ then break multipliers loop; return $\mathbf{D}^{\tau}$.
    end for
    If $\|\mathbf{E}^{\tau} - \mathbf{E}^{\tau-1}\|_2 + \|\mathbf{D}^{\tau} - \mathbf{D}^{\tau-1}\|_2 < \epsilon$ then stop updating.
end for
```

multiplier $\mathbf{N}^{\tau-1}$. Similarly, steps 14-22 of Alg. 1 involve a combination of multipliers method with gradient descent to find the update $\mathbf{D}^{\tau}$ for fixed $\mathbf{E}^{\tau}$. Thus, at every coordinate iteration τ the update $\mathbf{E}^{\tau}$ corresponds to the optimal solution of (5) for fixed $\mathbf{D}^{\tau-1}$ which is convex, while $\mathbf{D}^{\tau}$ corresponds to the optimal solution of (5) for fixed $\mathbf{E}^{\tau}$, thus given the coordinate descent convergence results in [24], Alg. 1 is guaranteed to converge at least to a stationary point of (5).

Starting from Alg. 1, a simpler approach was devised to carry out the numerical tests. Specifically, the two gradient loops with iteration index k for updating $\mathbf{D}$ and $\mathbf{E}$ are grouped in one. After $\mathbf{D}$ and $\mathbf{E}$ are updated, the multipliers $\mathbf{Q}$, $\mathbf{M}$ and $\mathbf{N}$ are updated together, running the ℓ multipliers loop for one iteration. Thus, the algorithm was completed with only two level nesting instead of three. This was done to reduce running time and does not guarantee convergence from a theoretical perspective, nonetheless extensive numerical testing showed that convergence was consistently reached.

A similar process is implemented for the values of $\mathbf{D}'$ and $\mathbf{E}'$ in (6). The augmented Lagrangian is similar to (4.8), except for the unit summation constraint being applied to $\mathbf{D}'$, whereas previously it was applied for $\mathbf{E}$, while using $\boldsymbol{\lambda}^{\mathbf{E}'} := [\lambda_1^{\mathbf{E}'} \ldots \lambda_R^{\mathbf{E}'}]^T$ and $\boldsymbol{\lambda}^{\mathbf{D}'} := [\lambda_1^{\mathbf{D}'} \ldots \lambda_R^{\mathbf{D}'}]^T$. Then, the gradient descent updates for $\mathbf{D}'$ and $\mathbf{E}'$ are carried out in a similar fashion as before, while the updated multipliers in this case $\mathbf{Q}' \in \mathbb{R}^R$ correspond to the unit summation constraints for $\mathbf{D}'$, while $\mathbf{N}' \in \mathbb{R}^{p_x \times R}$ and $\mathbf{M}' \in \mathbb{R}^{R \times p_y}$ represent the Lagrange multipliers corresponding to the nonnegativity constraints for $\mathbf{D}'$ and $\mathbf{E}'$, respectively. These quantities are updated in a manner similar to the one summarized in eq. (10)-(15). Alg. 1 can still be applied here after replacing $\mathbf{E}$ and $\mathbf{D}$ with $\mathbf{D}'$ and $\mathbf{E}'$ respectively, while replacing $\mathbf{Q}$, $\mathbf{M}$, and $\mathbf{N}$ with $\mathbf{Q}'$, $\mathbf{N}'$, and $\mathbf{M}'$, respectively.

Note that the mapped data $\boldsymbol{\phi}_x(\mathbf{x}_b)$ and $\boldsymbol{\phi}_y(\mathbf{y}_b)$ needed in (4.5) and (4.6) also need to be centered by subtracting from them the kernel 'mean' values $\bar{\boldsymbol{\phi}}_x(\mathbf{x}) := S^{-1} \sum_{s \in \mathcal{S}} \boldsymbol{\phi}_x(\mathbf{x}_s)$ and the similarly defined $\bar{\boldsymbol{\phi}}_y(\mathbf{y})$, see details in [42]. This is also mentioned in Sec. (4.4), where this subtraction of the ‘mean’ will cause the kernel row average to reduce when considering higher values of kernel variance. Note that during centering for lower kernel variance values, approximately only one entry of each kernel covariance row will have a significant value, resulting in subtracting the mean to have a negligible effect.

Remark: It should be emphasized that the nonlinear kernels applied are due to the fact that the relationship among the spectral responses from the materials in the hyperspectral images are usually nonlinear in nature. The nonlinear mixing model applied here in no way necessitates application of the nonlinear kernels. In other words, if the materials were linearly related to each other (for example, if the materials were simply linearly scaled versions of each other), then linear CCA based unmixing would have sufficed, even when the mixing model utilized has a nonlinear part.

4.6 Endmember and Nonlinear Coefficient Estimation

We utilize the estimated abundances for $\mathbf{x_s}$ and $\mathbf{y_s}$ contained in the $\mathbf{D}'$ and $\mathbf{E}$, to find the unknown endmembers and nonlinear coefficient that are used in the available hyperspectral pixels. The $\mathbf{P}$ and b in the mixing models (1) or (2) can be obtained by stacking the rows of $\mathbf{D}'^T$ and $\mathbf{E}$ alternatively.

Previous work on unmixing uses an endmember extraction algorithm such as VCA [11] to extract the endmembers and then use the values to get $\mathbf{U}$ in (4.2), but as our approach estimates $\mathbf{P}$ first, estimating $\mathbf{U}$ and b becomes more challenging due to the fact that the endmembers are present in multiple pixels. To this end, endmember extraction is relying on the following least-squares formulation

$$\begin{aligned}&\underset{\mathbf{U},b}{\text{argmin}}||\mathbf{M} - \textstyle\sum_{j=1}^{R}\mathbf{P}_{:,j}\cdot\mathbf{U}_{j,:}\\&- b\textstyle\sum_{j=1}^{R-1}\sum_{l=j+1}^{R}\mathbf{P}_{:,j}\cdot\mathbf{P}_{:,l}\cdot\mathbf{U}_{j,:}\odot\mathbf{U}_{l,:}||_F^2.\end{aligned} \tag{4.15}$$

The cost in (4.15) is tackled via an alternating gradient descent approach, where the endmember matrix $\mathbf{U}$ is determined recursively as

$$\mathbf{U}^{\kappa+1} = \mathbf{U}^{\kappa} - c\nabla J_{LS}(\mathbf{U}_k^{\kappa}, b^k),$$
$$b^{\kappa+1} = b^{\kappa} - c\nabla J_{LS}(\mathbf{U}_k^{\kappa+1}, b^{\kappa})$$

where $J_{LS}(\mathbf{U}, \mathbf{b})$ refers to the cost in (4.15), κ refers to the iteration steps for updating $\mathbf{U}$ and b, and c refers to a predetermined step-size. Here, $\mathbf{U}$ can be initialized by $\mathbf{U}_0 = \mathbf{M} \cdot \text{pinv}(\mathbf{P})$, where *pinv* refers to the pseudoinverse of a matrix, so essentially $\mathbf{U}$ is initialized using the LMM endmembers least-squares estimate. On the other hand, b can simply be initialized as 0. The matrix $\mathbf{U}$ obtained through this process, normalized to unit energy, will finally give the estimated endmember matrix. Note that the nonnegativity constraints for $\mathbf{U}$ in (4.15) are unnecessary since any negative value in $\mathbf{U}$ would cause the 2nd and 3rd terms in (4.15) to have a higher value, consequently increasing the overall cost.

4.7 Clipping of Low-Variance Spectral Bands

To further reduce computational complexity, we propose a scheme that discards spectral bands among which the pixels' spectral responses have low variability. Spectral bands in which different materials exhibit different spectral reflectance (low correlation), are extremely informative and useful in distinguishing among materials, and therefore help to perform efficient unmixing. Thus, it is of benefit to identify those spectral bands in which the selected kernel covariance matrices exhibit high variance across their entries. Following a similar strategy as in the clustering approach in [52], the entry-variance of the kernel covariance matrices along all spectral bands is calculated, then the spectral bands with the lowest variance are identified and discarded.

As an example, some mixed hyperspectral pixels are synthetically generated using as endmembers, pixels from the Salinas dataset [51]. We generate 40 mixed pixels, and apply

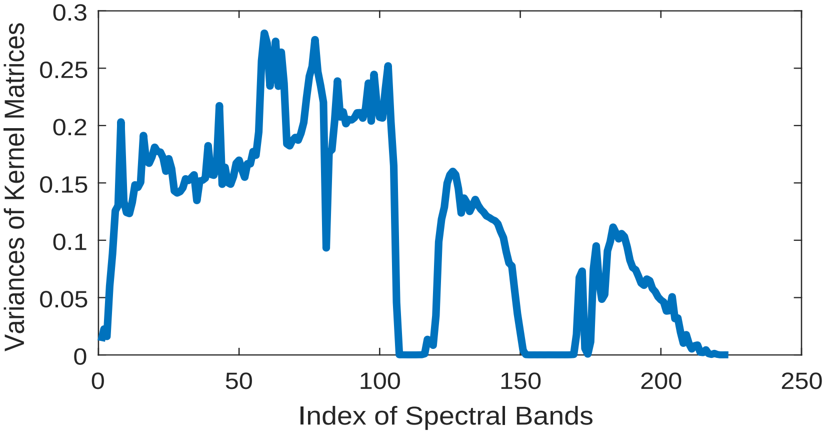


Figure 4.2: Kernel covariance entries variance versus spectral band index.

the kernel trick to obtain 224 kernel covariance matrices for the 224 spectral bands present there. Measuring the kernel matrix entry variances across each spectral band (see Fig. 4.2), we observe that little to no variance exists in some bands, implying that those bands will not be beneficial for the unmixing task. Thus, *clipping* such bands will yield more efficient computation, with no losses in terms of performance results.

4.8 Numerical Results

The performance of our novel unmixing algorithm is compared against both supervised and unsupervised methods. Specifically, KRCCU is compared with the unsupervised VCA method in [11]. For VCA, the MATLAB code provided in [34] is utilized, which generates the endmember matrix $\mathbf{U}$ in an order that is *shuffled* compared to the original order of the endmembers rows. Thus, in order to obtain the best results from VCA, the order had to be manually *reshuffled*, resulting in some supervision being involved. Two other supervised unmixing methods were implemented for comparison purposes. The first approach in [29], which is customized for the Fan model in [21], relies on VCA for endmember extraction, thus the results related to endmember estimation performance will be the same as in VCA. Further, [29] estimates the nonlinear coefficient b, where it uses a Bayesian approach along with subgradient optimization to estimate abundances. This method is abbreviated as PPNMM (Polynomial Post-Nonlinear Mixing Model). Another supervised method tested here is the one in [30], which calculates distances among the pixels projected onto a feature space to find points which would best represent the simplex that contains all the available pixels. This will be referred to as DMaxD. The distance used here was the PPNM distance shown in [30]. Since for the LMM, the nonlinear coefficient $b = 0$, the distance function used in that particular case for DMaxD is the Euclidean distance. Furthermore, another unsupervised kernel-based unmixing method was implemented as a comparative metric, the FKAA method [26]. In order to utilize the entire information from the kernel matrix, the FKAA method without the low-rank approximation of the kernel matrix was also implemented, noted as the KAA [27], where the kernel variance selection was adopted from [28].

Noise was added such that the sensing SNR was set to be 30dB. Since VCA is based on the LMM, abudance estimations were also done based on that model. For estimating abundance in VCA with nonlinear mixing models, a fully constrained least-squares (FCLS) approach is usually utilized, which was also employed here. The formulation is similar to the one shown in (4.15), only here the variable $\mathbf{U}$ is replaced with the abundances in $\mathbf{P}_{i:}$.

Numerical tests have been conducted using the two supervised (PPNMM and DMaxD), and two unsupervised methods, which correspond to the linear and nonlinear variations of VCA. For VCA with FLCS, the value of b was estimated with the algorithm provided in Sec. 4.6. The true value of b was assumed to be known in DMaxD, while PPNMM has its own algorithm for estimating b.

Four different performance metrics are considered during comparisons. The quality of endmember estimation is quantified via the Spectral Angle Measurements (SAM), taken in degrees, and calculated as $\frac{1}{R}\sum_{i=1}^{R}\cos^{-1}(\frac{\mathbf{e}_i\cdot\hat{\mathbf{e}}_i^T}{||\mathbf{e}_i||_2\cdot||\hat{\mathbf{e}}_i||_2})$, where $\mathbf{e}$ and $\hat{\mathbf{e}}$ are the actual and estimated endmembers, respectively. Another measure for endmember estimation accuracy is the Normalized Euclidean Distances (NED), calculated by $\frac{1}{R}\sum_{i=1}^{R}\frac{||\mathbf{e}_i-\hat{\mathbf{e}}_i||_2}{||\mathbf{e}_i||_2}$. For evaluating the quality of abundance estimation, Abundance Angle Measurements(AAM) are taken in degrees, with the formula $\frac{1}{Y}\sum_{i=1}^{Y}\cos^{-1}(\frac{\mathbf{a}_i\cdot\hat{\mathbf{a}}_i^T}{||\mathbf{a}_i||_2\cdot||\hat{\mathbf{a}}_i||_2})$, where $\mathbf{a}$ and $\hat{\mathbf{a}}$ are the actual and estimated abundances respectively, and Y is the number of mixed pixels. Another metric used is the Abundance Error (AE) given by $\frac{1}{Y}\sum_{i=1}^{Y}||\mathbf{a}_i-\hat{\mathbf{a}}_i||_2$.

4.8.1 Synthetic Data

For the simulated data, a 40×40 pixel field was synthetically generated consisting of four types of crops, created through combinations of four endmembers taken from the hyperspectral dataset in [51]. The simulated field is essentially divided into four parts, where each 40×10 pixel part has a major portion containing $70\% - 90\%$ purity of one endmember, with the fringes containing $55\% - 70\%$ purity of the same endmember. The two parts on the middle, would have 60% of the pixels in the center to contain the highest purity, with 20% on either side containing lower purity. For the two parts on either end of the field, 80% of the pixels on the *outer* side would contain higher purity, with 20% on the inner side containing lower purity. This configuration simulates a real-world scenario, where a crop has lower density the further we exit the crop and enter an adjoining one. Fig. 4.3 is an example of the field, where the four endmembers are obtained from four pixels from the *Pavia University* dataset. Note that the majority of each crop may have uniform color, but the color mixes with the color of other crops when the fringes are approached. Besides considering *lightly mixed* data where the major portion has $70\% - 90\%$ purity and the fringes have $55\% - 70\%$ purity, we also generated another synthetic field with *heavily mixed* pixels in which the major portions have $65\% - 75\%$ purity, while the fringes have $55\% - 65\%$ purity.

For KRCCU, as the data has to be *kernelized*, increasing the computational time combinatorially with the size of the data, it is not wise to perform the operation on the entire data at once. A more efficient approach is to perform the algorithm on a smaller part of the data, as long the separate parts are independent of each other, and obtain results for the smaller parts separately, then finally combine them together. To this end, KRCCU was implemented across each 40-pixel row of the synthetic field separately. The *Pavia University* dataset [51] was chosen since it is a very diverse hyperspectral imageset, as it not only contains various materials pertaining to life forms, but also other materials such as bricks, soil etc. To preserve this diversity in the data, the four materials chosen for the synthetic dataset were *Meadows*, *Bitumen*, *Self-Blocking Bricks*, and *Shadows*.

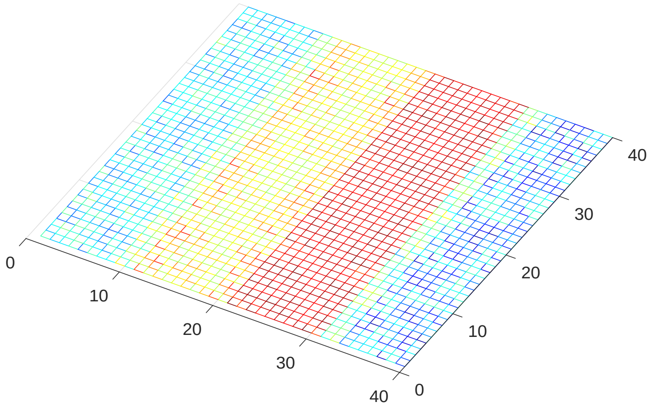


Figure 4.3: Reflectance on the 1^{st} spectral band of a synthetically generated field.

4.8.1.1 Lightly Mixed Data

Table 4.1 shows the performance results of various unmixing methods. For ease of reading, we placed a notation **(A)** for the two accuracy results for abundances, and a notation **(E)** for the two accuracy results for endmembers. Across each row, the most accurate results are highlighted in bold. Note that as the two versions of VCA differ only on the abundance estimation algorithm, the results pertaining to endmember estimation will be the same. These numbers will be the same also for PPNMM, as it relies on VCA

Measures	(b)	**% of Dead Pixels**	**K**RCCU	**D**MaxD	**V**CA (LMM)	**V**CA + FCLS	**P**PNMM	**F**KAA	**K**AA
AAM (A)	0	0	19.89	**4.83**	5.13	6.08	5.22	13.48	13.74
		5	**19.13**	53.98	80.33	63.63	63.28	60.75	63.18
	1	0	18.97	13.27	6.45	6.00	**5.78**	13.56	13.79
		5	**18.54**	53.57	80.47	63.64	63.07	62.81	63.23
	5	0	22.03	22.37	12.01	**9.19**	12.51	14.27	15.25
		5	**21.92**	50.37	78.10	62.91	62.22	63.34	63.39
AE (A)	0	0	0.28	**0.10**	0.11	0.11	**0.10**	0.24	0.24
		5	**0.27**	0.70	2.07	0.76	0.77	0.83	0.82
	1	0	0.27	0.19	0.14	**0.11**	**0.11**	0.24	0.24
		5	**0.26**	0.68	2.09	0.76	0.77	0.85	0.82
	5	0	0.30	0.30	0.29	**0.14**	0.18	0.24	0.25
		5	**0.30**	0.64	1.83	0.77	0.76	0.85	0.82
SAM (E)	0	0	1.51	**1.34**	1.60	1.60	1.60	2.51	2.56
		5	**1.31**	18.32	19.07	19.07	19.07	8.14	15.00
	1	0	**1.39**	1.52	1.40	1.40	1.41	2.57	2.61
		5	**1.40**	18.22	19.11	19.11	19.12	13.52	14.98
	5	0	1.75	**1.64**	2.28	2.28	2.28	2.89	2.91
		5	**1.89**	16.42	19.47	19.47	19.47	14.07	14.92
NED (E)	0	0	**0.02**	**0.02**	0.03	0.03	0.03	0.04	0.04
		5	**0.02**	0.31	0.34	0.33	0.34	0.14	0.26
	1	0	**0.02**	**0.02**	0.03	**0.02**	0.03	0.04	0.05
		5	**0.02**	0.31	0.33	0.33	0.33	0.24	0.26
	5	0	**0.03**	**0.03**	0.09	0.04	0.09	0.05	0.05
		5	**0.03**	0.28	0.34	0.38	0.34	0.24	0.26

Table 4.1: Comparison of accuracies in endmembers and abundances for various forms of VCA, with KRCCU, DMaxD, PPNMM and FKAA for Lightly Mixed Data, with and without dead pixelsand varying values of nonlinear coefficients.

for endmember estimation, while the supervised endmember shuffling for VCA was also applied to the PPNMM algorithm. As we can observe, during endmember estimation, KRCCU has either accuracy measures close to the most accurate (and supervised) method (DMaxD), or the best among all the methods. As for the abundance estimation, although KRCCU can trail comparative methods with no dead pixels, introducing only 5% dead pixels significantly impacts all methods except for KRCCU. Note that among the other methods, the method least impacted is FKAA, due to this method also relying on kernels. It should also be stressed, that DMaxD is the only method here where the value of the nonlinear coefficient b is known *a priori*. One reason for KRCCU being superior to FKAA, is that FKAA uses a low-rank approximation of the kernel matrix instead of generating the kernel matrix directly, which is done in KRCCU.

4.8.1.2 Heavily Mixed Data

The dataset was generated using the same approach as before, except that the purity had the values given in Sec. 4.8.1. Table 4.2 illustrates the performance comparisons for the various methods. For fairness, the same pixel entries were made zero when introducing dead pixels.

Table 4.2 shows that when faced with heavily mixed data, KRCCU outperforms every other method in most scenarios. The supervised method DMaxD is better at estimation of abundances, owing partly to its prior knowledge of the value of b, but KRCCU is close to, or better than the unsupervised methods, depending on the value of b. However, even with just 5% of pixels becoming unresponsive, KRCCU is mostly unaffected by this change, while it severely affects all other methods to varying degrees.

Measures	(*b*)	**% of Dead Pixels**	**K**RCCU	**D**MaxD	**V**CA (LMM)	**V**CA + FCLS	**P**PNMM	**F**KAA	**K**AA
AAM (A)	0	0	22.32	**13.56**	18.45	15.96	15.58	18.54	18.43
		5	**22.05**	52.26	75.56	57.30	55.80	61.41	55.83
	1	0	21.65	**15.62**	21.48	17.91	16.90	18.75	18.58
		5	**21.76**	52.29	76.03	58.30	59.19	62.72	55.91
	5	0	23.07	**16.13**	26.65	24.48	20.70	19.58	19.41
		5	**23.81**	49.22	74.36	51.01	51.29	62.75	55.98
AE (A)	0	0	0.31	**0.23**	0.37	0.27	0.26	0.33	0.33
		5	**0.30**	0.65	2.22	0.64	0.62	0.84	0.73
	1	0	0.30	**0.26**	0.42	0.30	0.29	0.33	0.33
		5	**0.30**	0.66	1.57	0.66	0.67	0.85	0.74
	5	0	0.32	**0.24**	0.53	0.35	0.32	0.33	0.33
		5	**0.32**	0.59	2.39	0.58	0.58	0.85	0.74
SAM (E)	0	0	**2.62**	3.19	3.83	3.83	3.83	3.69	3.78
		5	**2.40**	16.07	18.35	18.35	18.35	7.90	8.25
	1	0	**2.68**	3.47	4.93	4.93	4.93	3.75	3.83
		5	**2.59**	16.18	18.71	18.71	18.71	13.52	8.48
	5	0	**2.92**	3.49	5.84	5.84	5.84	4.27	4.35
		5	**3.23**	17.30	16.99	16.99	16.99	14.53	9.17
NED (E)	0	0	**0.04**	0.06	0.07	0.07	0.07	0.06	0.07
		5	**0.04**	0.27	0.32	0.32	0.32	0.14	0.14
	1	0	**0.04**	0.06	0.09	0.09	0.09	0.07	0.07
		5	**0.04**	0.28	0.32	0.32	0.32	0.23	0.15
	5	0	**0.05**	0.06	0.17	0.17	0.17	0.07	0.08
		5	**0.05**	0.30	0.30	0.30	0.30	0.25	0.16

Table 4.2: Comparison of accuracies in endmembers and abundances for VCA, with KRCCU, DMaxD, PPNMM and FKAA for Heavily Mixed Data, with and without dead pixels and varying values of nonlinear coefficients.

4.8.1.3 Nonlinear Coefficient Estimation

Among the methods discussed here, three had some approach towards estimation of the nonlinear coefficient b. In order to compare the performance of b estimation across these three methods, the Mean Square Error (MSE) of the estimated bs was measured. It is worth noting that, in VCA+FCLS and PPNMM, as endmembers were first estimated, abundances can be calculated separately for each mixed pixel as these abundances do no affect each other. Thus, for N mixed pixels, we can obtain N estimations of abundances, and consequently, N estimations of b. On the other hand, KRCCU estimates the abundances first, and estimates b alongside the estimation of the endmembers, which contribute to every mixed pixel. Therefore, only one estimation of endmembers, and consequently, only one estimation of b is obtained. The best accuracies are in bold.

Also, when tackling such a large number of samples, PPNMM's somewhat inconsistent estimations of b cause some to reach extremely high numbers, which result in a very high MSE. To give a fairer idea of the accuracy of most of the estimations of PPNMM, we also adjusted the estimations to remove a number of the ones that caused the most error. The performance of PPNMM with, and without these adjustments are shown in Table 4.3. Note that in some cases of PPNMM and VCA+FCLS, some improvement was noted with the introduction of dead pixels. This is presumed to be coincidental, as upon our observation, introduction of dead pixels introduced a higher minimal cost in the optimization problem used for the estimation.

4.8.1.4 PBRT Data

Another form of synthetic data generation is with implementation of the PBRT (Physically Based Ray Tracing) Model [31]. We have measured unmixing accuracy of the various methods over a synthetic dataset generated using this model [32]. The data is of a forest

Mixture Type	Nonlinear Coefficient (b)	% of Dead Pixels	PPNMM (adjusted	PPNMM (no adjustment)	VCA + FCLS	KRCCU
Lightly Mixed	0	0	0.03	4.02x10^{7}	1x10^{-5}	**0**
		5	19.44	1.86x10^{5}	0.03	**0**
	1	0	0.34	1.31x10^{3}	2.51	**0.05**
		5	12.30	4.09x10^{3}	0.30	**0.04**
	5	0	12.52	5.31x10^{6}	7.19	**0.80**
		5	**5.26**	5.28x10^{4}	9.90	8.18
Heavily Mixed	0	0	1.71	5.05x10^{25}	4x10^{-5}	**0**
		5	10.90	11.98	0.02	**0**
	1	0	1.2x10^{7}	6.75x10^{26}	2.72	**0.05**
		5	4.12	5.43	**0.73**	1
	5	0	3.77x10^{8}	3.67x10^{28}	13.07	**0.01**
		5	4.85	4.88	5.08	**4.33**

Table 4.3: MSE of estimated values of nonlinear coefficients for VCA, with PPNMM, DMaxD and KRCCU in Heavily Mixed Data, with and without dead pixels and varying values of nonlinear coefficients.

with two types of trees, called 'Beech' and 'Poplar', composed of almost 80% of the former and 20% of the latter. The remaining small contributions to the mixture come from the soil, which is negligible as the trees are expansive enough to cover the area. As we can observe from Table 4.4, KRCCU outperforms all other methods. DMaxD was implemented with $b = 1$ as that gave it the best accuracy, and FKAA faced difficulty with making the estimated abundances have unit summation, so that was manually implemented afterwards to give it an advantage. The best accuracies are shown in bold.

4.8.2 Real Data

For real hyperspectral data, we utilized two types of datasets. For the first one, the reference data available is a class label for *one material* only in each pixel. Since there are no ground truth abundances available, a more prudent approach would be to apply

Methods	**AAM (deg) (A)**	**AE (A)**	**SAM (deg) (E)**	**NED (E)**
KRCCU	**13.73**	**0.17**	**3.70**	**0.06**
DMaxD	42.61	0.57	5.21	0.09
VCA(LMM)	26.14	0.35	3.99	0.07
VCA + FCLS	26.37	0.35	3.99	0.07
PPNMM	26.41	0.35	3.99	0.07
FKAA	35.63	0.50	3.99	0.07
KAA	37.28	0.52	4.59	0.08

Table 4.4: Comparison of endmember and abundance accuracy measures for VCA, with PPNMM, DMaxD and KRCCU in PBRT Dataset.

unmixing, and consider the material with the highest abundance to be the sole material in that corresponding pixel. Thus, unmixing is employed to perform clustering on real hyperspectral datasets. For the second dataset, the actual abundance values are unavailable, and only the spectral signatures of the pure materials are known. Thus, we measure only the accuracy of the estimated endmembers.

Instead of applying the unmixing scheme on the entire dataset, a 'divide and conquer' approach is implemented as a workaround, where the data are segmented into smaller non-overlapping square patches, and the algorithm is performed separately on the patches, introducing further computational savings (for details see clustering approach in [52]).

4.8.2.1 Salinas

The Salinas dataset contains 54129 labeled pixels. The resolution is $512 \times 217 \times 224$, where 224 is the number of spectral bands. Usually, 204 spectral bands are left after removing 20 bands due to atmospheric water absorption, but in the proposed KRCCU scheme, this will be unnecessary for two reasons. Firstly, because there is little variation present between the pixels in such spectral bands, the clipping algorithm proposed in Sec.

4.7 will discard the data in these bands. Secondly, even if one such band is not clipped, having little useful information means that this will have a very low weight assigned to it during the weighting process in Sec. 4.3.

Regardless, the clipping algorithm removed around 10 spectral bands that were considered to have no relevant information for the unmixing task at hand. Table 4.5 gives in detail, the accuracy averaged across multiple iterations for each of the 16 individual classes. Averaging was performed by running each method for 50 independent trials. The mean of these accuracies yield the Average Accuracy(AA), while the accuracy of all the pixels considered together yield the Overall Accuracy (OA) estimate. For each row, the highest accuracy is given in bold.

As we can observe in Table 4.5, KRCCU significantly outperforms all other methods, both supervised and unsupervised. Materials that contributed to some of the inaccuracy come from two versions of *Fallow*, where as the spectral responses come from the same type of crop, they can cause high correlation to come from each other, causing KRCCU to face difficulty in separating the two classes apart.

Fig. 4.4 depicts the Overall Accuracy across a varying percentage of dead pixels. KRCCU is noticeably more accurate than existing methods, while it is also the most robust even with ten percent dead pixel entries. On the other hand, note that the two versions of VCA, and PPNMM, have a similar pattern, since all three of them are using the same endmember estimation method. Also, the presence of dead (zero) pixels entries causes a number of pixel points to *shift* location, heavily impacting methods, such as DMaxD, that rely on the pixel position on a vector subspace (see Fig. 4.4). FKAA is also impacted severely, even more so with further dead pixels. It is, however, interesting to note FKAA's performance compared to KAA even when using a lower rank approximation of the kernel matrix.

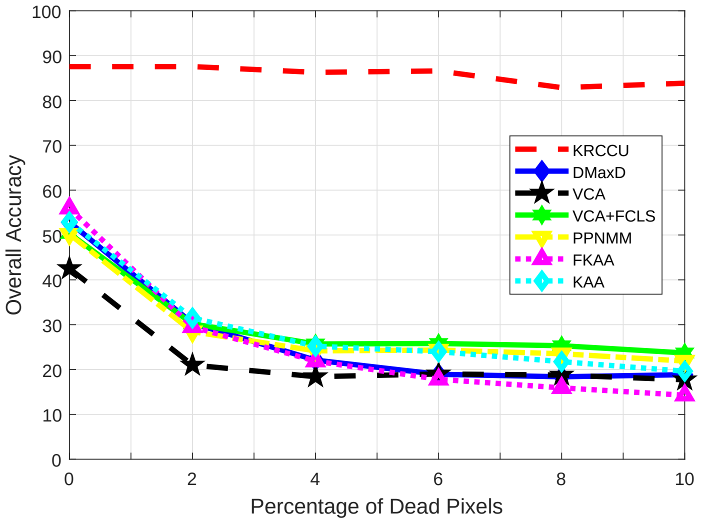


Figure 4.4: Unmixing accuracy versus percentage of dead pixels in Salinas dataset.

Fig. 4.5 shows a comparison of estimated class labels for the *Salinas* map, using the estimated abundances from all the unmixing methods under comparison. It is worth noting that VCA+FLCS and PPNMM have nearly the same estimated map. Notice that KRCCU clusters correctly the majority of hyperspectral pixels. This can be seen by the same color pixels in Fig. 5 (b) arranged in almost identical shapes with the ground truth clusters in Fig. 5 (a). On the contrary, there is noticeable erroneous clustering in competing alternatives as indicated by the non-uniform coloring of pixels that belong to the same ground truth group.

4.8.2.2 Cuprite

The Cuprite dataset [33] consists of 224 channels encompassing wavelengths from 370nm to 2480nm, of which 36 are removed due to atmospheric water absorption. Table 4.6 illustrates the accuracy measures in endmember estimations for different methods. Note that since the endmember extraction is identical in VCA, VCA+FCLS, and PPNMM, only VCA's accuracy was shown here among these three. We can observe that KRCCU yields the least error among all the comparable methods, with FKAA coming second.

Class names	**KRCCU**	**DMaxD**	**VCA**	**VCA + FCLS**	**PPNMM**	**FKAA**	**KAA**
Brocoli Green Weeds 1	**100**	99.85	81.41	99.01	98.98	98.75	99.67
Brocoli Green Weeds 2	**100**	37.60	69.71	9.45	9.41	70.24	57.97
Fallow	**100**	99.79	41.72	76.59	74.67	56.02	54.56
Fallow Rough Plow	41.96	**99.85**	34.95	56.20	56.85	99.06	96.51
Fallow Smooth	40.96	0.03	22.31	39.39	35.74	49.47	**59.63**
Stubble	47.94	**99.87**	64.08	85.19	86.57	99.54	97.55
Celery	56.46	**99.18**	33.90	36.56	35.59	85.12	77.00
Grapes Untrained	**93.52**	91.51	48.17	69.53	68.97	55.15	55.95
Soil Vinyard Develop	**100**	0	42.15	14.81	11.09	49.35	35.01
Corn Senesced Green Weeds	**100**	29.10	23.90	14.89	14.56	50.48	47.31
Lettuce Romaine 4wk	**100**	0	3.33	0.17	0.20	4.30	2.13
Lettuce Romaine 5wk	**100**	0	0.97	0.20	0.21	0	11.38
Lettuce Romaine 6wk	**100**	0	4.51	4.28	4.17	98.14	78.79
Lettuce Romaine 7wk	**100**	41.12	3.55	0.73	0.90	86.72	70.37
Vinyard Untrained	**100**	0.04	2.30	1.41	1.31	33.10	17.28
Vinyard Vertical Trellis	**100**	30.38	25.60	41.05	40.24	39.01	45.76
OA	**87.55**	52.84	42.51	50.33	50.19	57.86	52.89
AA	**86.30**	45.52	31.41	34.34	33.72	60.91	56.68

Table 4.5: Clustering Performance in the Salinas dataset.

	KRCCU	**DMaxD**	**VCA**	**FKAA**	**KAA**
SAM	**3.35**	5.97	6.39	4.62	4.40
NED	**0.06**	0.10	0.10	0.08	0.08

Table 4.6: Endmember estimation performance in the Cuprite dataset.

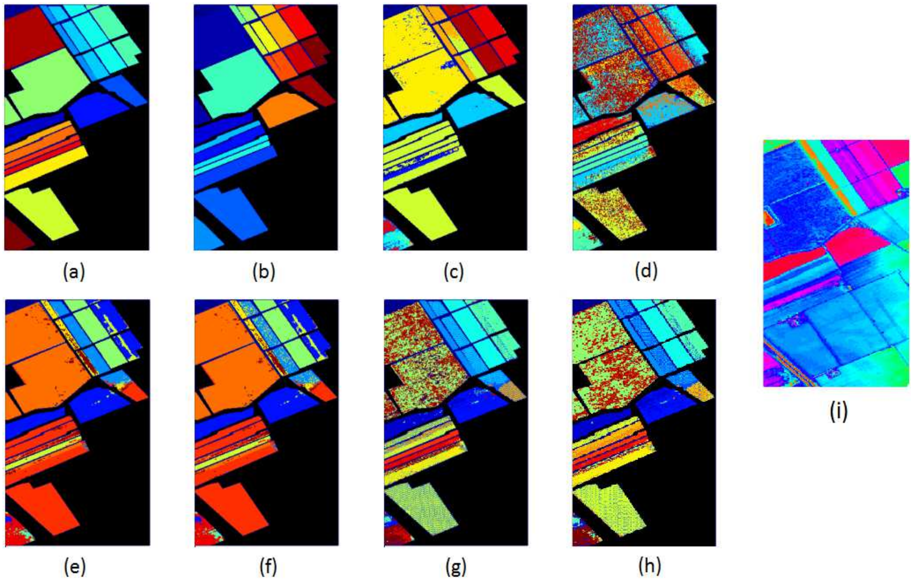


Figure 4.5: Clustering maps in the Salinas dataset: (a) Ground Truth, (b) KRCCU, (c) DMaxD, (d) VCA, (e) VCA + FCLS, (f) PPNMM (g) FKAA (h) KAA (i) False-color RGB image.

The performance of the algorithms in endmember estimation was also measured in the presence of corrupted data, simulated by dead pixels. Fig 4.6 shows the SAM values across varying percentages of dead pixels. As we can clearly observe, not only is KRCCU outperforming all other methods, its performance with increasing introduction of dead pixels has the highest consistency. Understandably, since FKAA also works with the kernel trick in averaging kernel matrices, its performance is noteworthy. DMaxD, however, has considerable difficulty in tackling this corruption, even more so than VCA, presumably because of the reliance on the distance between the now *shifted* points in the feature space. Also, KAA's advantage in using the entire kernel matrix brought some improvement compared to FKAA, though as it was not performing as well in Salinas, this improvement is clearly inconsistent.

We also consider observing the estimated endmembers across 3 materials, in comparison with the actual endmembers (after normalizing all of them with unit energy) in Fig

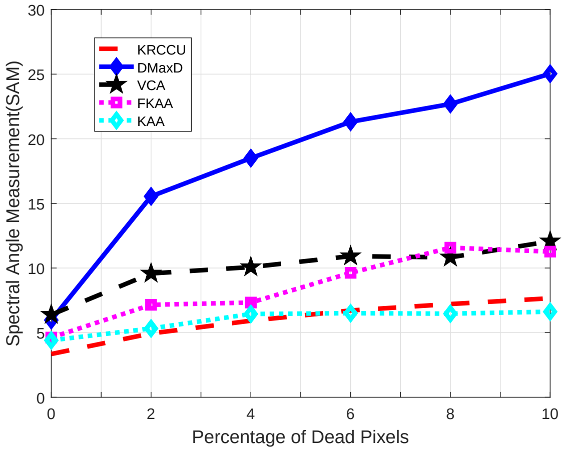


Figure 4.6: Unmixing accuracy versus percentage of dead pixels in Cuprite dataset.

4.7. The upper, middle and lower plot represents the materials with the closest, median, and most challenging estimation respectively. As we can see, except for minor instances like spectral bands $100 \sim 108$ in Chalcedony, the estimation of KRCCU (red) is usually closest to the actual endmember's spectral reflectance (blue), further illustrating KRCCU's superior performance in endmember estimation.

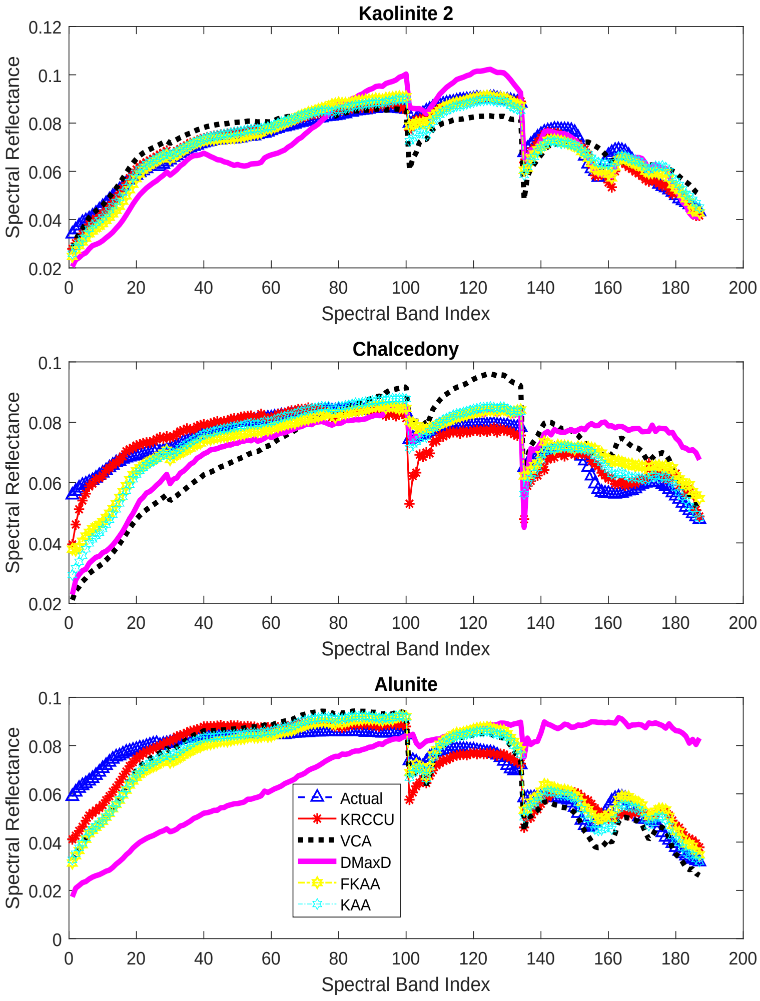


Figure 4.7: Endmember estimation for various methods across three endmembers in Cuprite

CHAPTER 5

UNSUPERVISED HYPERSPECTRAL UNMIXING VIA NONLINEAR AUTOENCODERS

5.1 Problem Statement and Preliminaries

Hyperspectral images contain information in hundreds of bands, thus providing a finer view of different materials being observed. Let *B* be the number of spectral bands across which information is obtained per pixel, while the set $\mathcal{B} := \{1, \ldots, B\}$ denotes the indices of the spectral bands. Let $\mathbf{M} \in \mathbb{R}^{N \times B}$ contain the reflectances of N pixels. One major issue in hyperspectral images is that of mixing where pixels are formed as (non-)linear combinations of reflectances of pure materials; the *endmembers*.

A popular mixing model where the reflectance values of mixed pixels $\mathbf{M}$ are expressed as a linear combination of P endmembers is the linear mixing model (LMM), i.e.,

$$\mathbf{M} = \mathbf{A} \cdot \mathbf{E} + \epsilon, \tag{5.1}$$

where $\mathbf{A} \in \mathbf{R}^{N \times P}$ represents the matrix of fractional contributions of the endmembers, also known as *abundances*. Matrix $\mathbf{E} \in \mathbf{P}^{P \times B}$ contains the endmember reflectances, and ϵ represents white additive noise. From Eq. (6.1) the mixed pixels can be expressed individually as

$$\mathbf{M}_{i,:} = \textstyle\sum_{j=1}^{P} \mathbf{A}_{i,j} \cdot \mathbf{E}_{j,:} + \epsilon, \forall i \in 1, 2, \ldots, N. \tag{5.2}$$

Since the abundances represent the fractional contributions of endmembers, matrix $\mathbf{A}$ has nonnegative entries. Further, the sum of the entries in each row of $\mathbf{A}$ will sum up to one. Similarly, the entries of matrix $\mathbf{E}$ are nonnegative being the reflectance values of endmembers across the sampled spectral bands. Thus, $\mathbf{A}_{i,j} \geq 0, \mathbf{E}_{j,:} \geq 0, \forall i \in 1, 2, \ldots, N$ and $\forall j \in 1, 2, \ldots, P$, and also $\sum_{j=1}^{P} \mathbf{A}_{i,j} = 1, \forall i \in 1, 2, \ldots, N$.

There are three main nonlinear mixing models often used. These are the Fan model, the bilinear model [72], and the PPNM (Post Polynomial Nonlinear Mixing) model [75].

$$\begin{aligned} \mathbf{M}_{i,:} = & \sum_{j=1}^{P} \mathbf{A}_{i,j} \cdot \mathbf{E}_{j,:} + \gamma_i \sum_{j=1}^{P-1} \sum_{l=j+1}^{P} \mathbf{A}_{i,j} \cdot \mathbf{A}_{i,l} \cdot \mathbf{E}_{j,:} \odot \mathbf{E}_{l,:} \\ & + \epsilon, \forall i \in 1, 2, \ldots, N. \end{aligned} \tag{5.3}$$

First, the Fan model in Eq. (6.3) takes cross-products across different endmembers to model the nonlinear mixing part with $\odot$ representing the Hadamard product, or elementwise multiplication of two arrays. The values γ_i are scaling factors that control the strength of the nonlinear components with respect to (w.r.t.) the overall mixture. Typically, these hyperparameters are defined by the user. However, our framework will treat them as unknown. Second, the bilinear model is given as

$$\begin{aligned} \mathbf{M}_{i,:} = & \sum_{j=1}^{P} \mathbf{A}_{i,j} \cdot \mathbf{E}_{j,:} + \gamma_i \sum_{j=1}^{P} \sum_{l=1}^{P} \mathbf{A}_{i,j} \cdot \mathbf{A}_{i,l} \cdot \mathbf{E}_{j,:} \odot \mathbf{E}_{l,:} \\ & + \epsilon, \forall i \in 1, 2, \ldots, N. \end{aligned} \tag{5.4}$$

Note the similarity to the Fan model, with the one difference being that it will also include self-interacting terms in the nonlinear components, as that has been shown to be a factor to also be considered [73, 74].

Third, the PPNM [75] model will simply take the LMM mixture, and sum it with the square of itself.

$$\begin{aligned} \mathbf{M}_{i,:} = & \sum_{j=1}^{P} \mathbf{A}_{i,j} \cdot \mathbf{E}_{j,:} + \mathbf{b}_i (\sum_{j=1}^{P} \mathbf{A}_{i,j} \cdot \mathbf{E}_{j,:}) \odot (\sum_{j=1}^{P} \mathbf{A}_{i,j} \cdot \mathbf{E}_{j,:}) \\ & + \epsilon, \forall i \in 1, 2, \ldots, N. \end{aligned} \tag{5.5}$$

Here, the unknown vector $\mathbf{b}$ with length N, has values that determine the strength of the nonlinear components of the mixtures, thus it is similar to the vector γ_i in Eqs. (6.3)-(5.4).

Hyperspectral images are found to be *unmixed* with much greater ease if the unmixing process takes place in properly selected kernel spaces of higher dimensionality than the original data space; see details in [52, 61, 68]. Kernels often used in practice include the linear kernel, polynomial kernel and Gaussian kernel [17] etc. The Gaussian (RBF) kernel has been shown to be effective with hyperspectral images, and will be employed here. In detail, the mixed pixels are represented as points in the Gaussian kernel space, and the endmembers will be treated as cluster centers in that kernel space. Thus, for a certain mixed pixel, the lower its distance is from a particular center, the closer the mixed pixel is in resemblance to that corresponding endmember, and thus the higher the abundance will be for that endmember. The RBF distance of mixed pixel $\mathbf{x}$ from endmember center $\mathbf{c}_i$ is written as

$$[\hat{\mathbf{a}}_x]_i = \exp(-\beta \cdot ||\mathbf{x} - \mathbf{c}_i||^2), \forall i \in 1, 2, \dots, P. \tag{5.6}$$

Here, β is a parameter that controls the spread of the Gaussian center. Since our framework is unsupervised, there will be no prior knowledge of the endmembers, and so they will be estimated using K-means [76] on the mixed pixels, thus the Gaussian centers are initially estimated to be the cluster centers. In order to effectively visualize the concept, an example is shown in Figure 5.1 where a mixed pixel is depicted in the kernel space, with three endmembers as the centers of three circular areas. As the mixed pixel is closer to endmember 1, it will have a higher abundance corresponding to that endmember. Similarly, for that mixed pixel, the abundance of endmember 2 will be lower, and the abundance for endmember 3 will be even lower. It should be noted that, the areas in Fig. 5.1 are circular, in order to depict neighborhoods using Euclidean distance. This better illustrates the concept of the nearest neighbors concept being implemented in the kernel space.

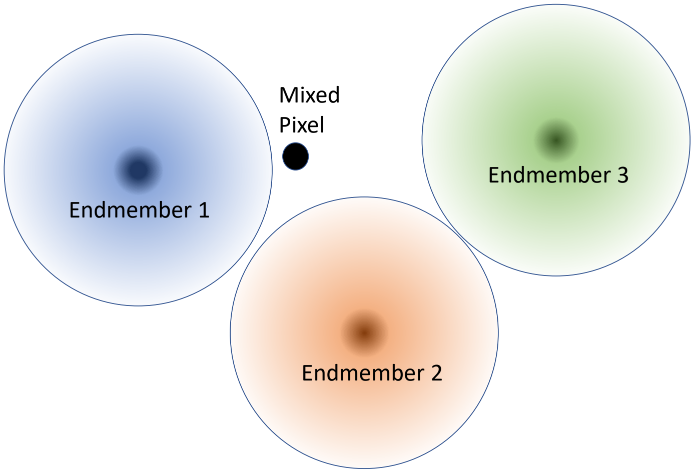


Figure 5.1: Kernelized mixed pixel and endmembers' geometry.

Note that since these distances are a measure of the abundances, the notation $[\hat{\mathbf{a}}_x]_i$ has been used, with $[\hat{\mathbf{a}}_x]$ denoting a row of the estimated abundance matrix $\hat{\mathbf{A}}$. After obtaining the P distances, normalizing them to unit summation will yield the abundance vector for this mixed pixel. It should also be noted, that since these values are a measure of distance, they cannot be negative and thus, the nonnegativity constraint of the abundances will be automatically met.

The reason behind using K-means in conjunction with the RBF kernel trick is the similarity in their concepts. As the K-means is being performed in the feature space, it can reasonably be assumed that mixed pixels with a higher abundance towards the same endmember, will encompass the same area in the feature space. Thus, the center of that area would be where the corresponding endmember will likely reside, and so K-means would be likely able to yield a point in the feature space that would be a close approximation of that endmember. This idea is similar to the area in the kernel space around which data points would reside, where they have a higher abundance of one endmember. The center of that area, which would be circular defined by the Gaussian kernel being used, would thus likely be an accurate representation of the corresponding endmember. This similarity between Gaussian centers in the RBF space, and the K-means cluster centers in the feature space, make the K-means a logical choice to implement in obtaining an estimation of the endmembers.

5.2 Unsupervised Hyperspectral Unmixing Autoencoders

5.2.1 Autoencoders for Nonlinear Unmixing

5.2.1.1 Fan & Bilinear Models

A novel autoencoder neural network structure is outlined next for the task of nonlinear unmixing. Fig. 6.2 depicts the network structure for the three different models, where it is assumed (for simplicity and without loss of generality) to have 4 endmembers. Each mixed pixel vector of the hyperspectral image is fed at the input layer and the RBF distance from the 4 K-means estimated endmember centers is calculated. After normalizing the RBF distances to unit summation, they are treated as the abundance estimates forming the RBF layer. To account for the bilinear and Fan nonlinear mixing models, a generator is introduced where crossproduct terms involving the abundances are formed. The vector of abundances, along with their crossproducts are then fed through the final layer, where the weights of the layer will be multiplied with the expanded abundance vector which includes the higher order crossproducts, to reconstruct the input mixed pixel. As the calculation of the product closely resembles the mixing models in Eqs. (6.3) and (5.4), it can be observed that these weights can be interpreted as the endmembers' reflectance values and their crossproducts, and thus these weights can be used to estimate the endmembers.

Although Fig. 6.2 (top right) depicts the crosproduct terms corresponding to the Fan model, modifications to accommodate the bilinear model are straightforward by incorporating the extra self-interacting terms present in the bilinear case and adjusting the final layer weights to account for the extra terms in the crossproduct generator. It should also be noted that, since the RBF centers, and the weights in the final layer are both learnable parameters initialized with the K-means centers, both can actually act as estimates of the endmembers.

The autoencoder weights $\mathbf{E}$ in the final layer are trained to minimize the mean square error of some input mixed pixel $\mathbf{x}$

$$\min_{\mathbf{E},\gamma_x} B^{-1}||\mathbf{x} - \hat{\mathbf{x}}(\mathbf{E}, \gamma_x)||_2^2 \tag{5.7}$$

Here, $\hat{\mathbf{x}}(\mathbf{E}, \gamma_x)$ represents the estimated reconstruction of the input pixel at the final output layer, and γ_x refers to the value of γ pertaining to pixel $\mathbf{x}$. Thus, after obtaining the estimated abundance vector $[\hat{\mathbf{a}}_x]$ from Eq. (6.5), depending on whether the Fan or bilinear model is used, $\hat{\mathbf{x}}(\mathbf{E}, \gamma_x)$ can take two forms. If the Fan model is used then

$$\hat{\mathbf{x}}(\mathbf{E}, \gamma) = \sum_{j=1}^{P}[\hat{\mathbf{a}}_x]_j \cdot \mathbf{E}_{j,:} + \gamma_x \sum_{j=1}^{P-1} \sum_{l=j+1}^{P} [\hat{\mathbf{a}}_x]_j \cdot [\hat{\mathbf{a}}_x]_l \cdot \mathbf{E}_{j,:} \odot \mathbf{E}_{l,:}$$

whereas if the bilinear model is used then

$$\hat{\mathbf{x}}(\mathbf{E}, \gamma) = \sum_{j=1}^{P}[\hat{\mathbf{a}}_x]_j \cdot \mathbf{E}_{j,:} + \gamma_x \sum_{j=1}^{P} \sum_{l=1}^{P} [\hat{\mathbf{a}}_x]_j \cdot [\hat{\mathbf{a}}_x]_l \cdot \mathbf{E}_{j,:} \odot \mathbf{E}_{l,:}.$$

5.2.1.2 PPNM Model

The network architecture when PPNM mixing model is utilized is depicted in Fig. 6.2 (bottom right). The PPNM layer has the nonlinear component formed as the Hadamard product of the linearly mixed pixel summation with itself, which is added to the linearly mixed pixel.

The learnable parameters here would be nearly the same for the RBF layer. The exception is that for the PPNM layer, the $\mathbf{b}$ values are the learnable parameters. The cost function to minimize here would be very similar to Eq. (6.18), with the difference of having the nonlinear coefficient γ_x being replaced by b_x. The training cost takes the form

$$\min_{\mathbf{E},b_x} B^{-1}||\mathbf{x} - \hat{\mathbf{x}}(\mathbf{E}, b_x)||_2^2. \tag{5.8}$$

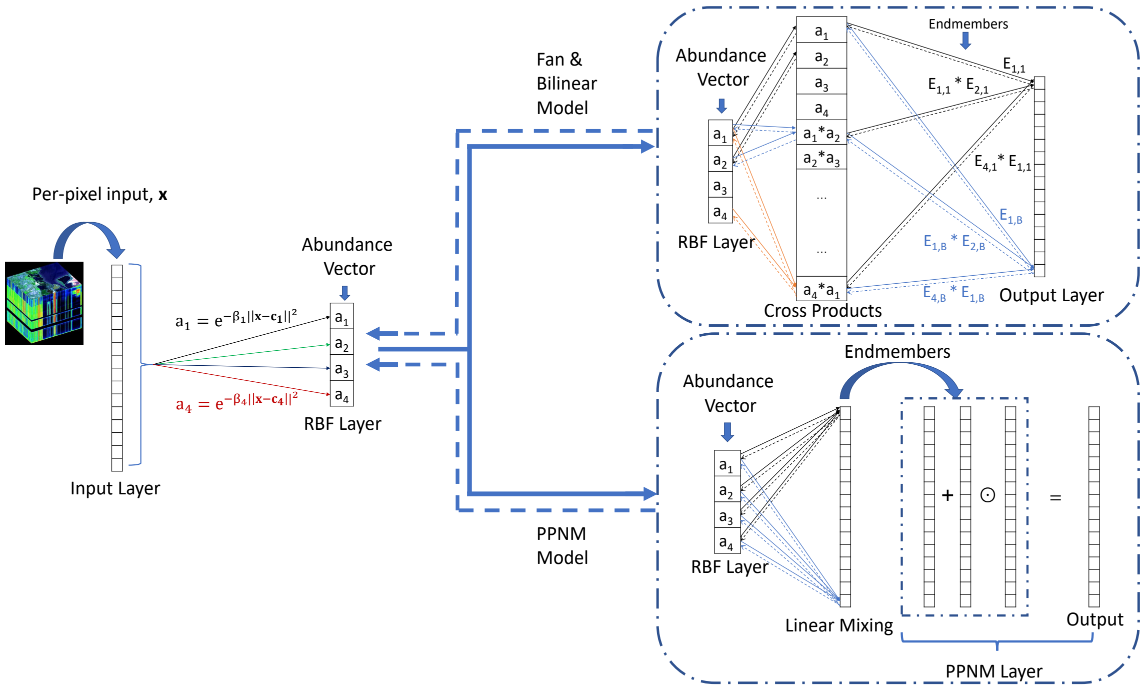


Figure 5.2: Autoencoder setup for the Fan, bilinear and PPNM mixing models(solid and dashed lines represent forward and backward propagation respectively).

Similarly, the reconstruction $\hat{\mathbf{x}}$ can be expressed as

$$\hat{\mathbf{x}}(\mathbf{E}, \gamma) = \sum_{j=1}^{P} [\hat{\mathbf{a}}_x]_j \mathbf{E}_{j,:} + b_x (\sum_{j=1}^{P} [\hat{\mathbf{a}}_x]_j \mathbf{E}_{j,:}) \odot (\sum_{j=1}^{P} [\hat{\mathbf{a}}_x]_j \mathbf{E}_{j,:}).$$

5.2.2 RBF Layer

For each hyperspectral pixel feeding the autoencoder neural network in Fig. 6.2 as input, the RBF distances from the kernelized endmember centers are calculated. As explained in Sec. 6.1, the RBF distances quantify the contribution of each endmember in the input mixed pixel, thus providing an estimate for the unknown abundances. After the P RBF distances are obtained, they are simply normalized to unit summation, and the estimated abundances are obtained.

In the RBF layer, the learnable parameters would be the RBF centers, as well as the β value in Eq. (6.5). The β value basically controls the radius of the multidimensional Gaussian bell footprint area (which depicts the area in which approximately $\sim$99% of the probability mass is). As this radius is dependent upon the distance of the mixed pixels from the centers they belong to, β is taken as the inverse of the squared average distance of the mixed pixels from the cluster centers that they belong to; with the centers found via K-means. This will ensure that approximately 99% of the time the mixed pixels will belong to the right center area when β is selected as described. Then, for a mixed pixel $\mathbf{M}_{i,:}$, let the label for one of the contributing classes be $L_{M_{i,:}}$, and the corresponding cluster center be $\mathbf{c}_{L_{M_{i,:}}}$. Thus, β is calculated as

$$\beta = (N^{-1} \cdot \textstyle\sum_{\imath=1}^{N} ||\mathbf{M}_{i,:} - \mathbf{c}_{L_{M_{i,:}}}||_2)^{-2}. \tag{5.9}$$

Since the weights $\mathbf{E}$ in the autoencoder configuration represent the endmember entries, they are initialized with the K-means cluster centers of all the mixed pixels in the dataset. Even though VCA [11] is more popular as an initialization option, and can be argued as a better alternative, there is one issue. With a dataset having high SNR, VCA performs endmember estimation with slightly higher accuracy, however, with low SNR, K-means is found to be more accurate. This is because, VCA will choose P mixed pixels and consider them to be the best estimations of the endmembers. Not only would this be less accurate with a dataset having pixels where pure pixels do not exist, but also when dealing with data having high noise, this will compromise the endmember estimation. Additionally, if severe noise distorts the endmember, there is a possibility of VCA finding multiple endmembers from one material, while disregarding other materials. However, with K-means, since the centers are pixel averages within some cluster, the noise will be reduced leading to more accurate endmember estimation.

Recall that the RBF distances are nonnegative, therefore the abundances estimated will satisfy the nonnegativity constraints. The RBF layer's output values would need to be normalized to conform to the unit summation constraint, i.e.,

$$[\hat{\mathbf{a}}_x]_i = \frac{[\hat{\mathbf{a}}_x]_i}{\sum_{j=1}^{P}[\hat{\mathbf{a}}_x]_j}, \forall i \in 1, 2, \ldots, P. \tag{5.10}$$

Then, the vector of abundances needs to be mapped to a higher dimensional space to include the crossproducts vector, the length of which is dependent upon whether the Fan or bilinear model is chosen. The expansion is a straightforward method, which can be achieved without defining a new layer. This will result in the vector labeled 'crossproducts' in Fig. 6.2.

5.2.3 Backpropagation

Simplifying Eq. (6.18) the error can be rewritten as $e := ||\mathbf{x} - \hat{\mathbf{x}}||_2^2$. Thus, in the RBF layer, after taking the K-means cluster centers as an initial estimate of the endmembers, let the first cluster center vector be denoted $\mathbf{c}_1$. The dimensions of β is $P \times 1$, where P represents the number of endmembers. Although these values are all initialized by a single scalar following the value obtained by Eq. (6.16), as they are weights, they can update to different values. The cluster centers have dimensions $P \times B$, where B represents the number of spectral bands. Let f_1 denote the activation function giving the first estimated abundance in the RBF layer, i.e.,

$$f_1 = \exp(-\beta_1 \cdot ||\mathbf{x} - \mathbf{c}_1||_2^2). \tag{5.11}$$

Here, β_1 will be the value of β corresponding to the first cluster center (or estimated endmember) $\mathbf{c}_1$. Performing backpropagation, the derivative of the function f_1 w.r.t. $\mathbf{c}_1$ is

$$\frac{\delta f_1}{\delta \mathbf{c}_1} = -2\beta_1 \exp(-\beta_1 \cdot ||\mathbf{x} - \mathbf{c}_1||_2^2)(\mathbf{x} - \mathbf{c}_1).$$

The derivative of the function f_1 w.r.t. β_1 is

$$\frac{\delta f_1}{\delta \beta_1} = -||\mathbf{x} - \mathbf{c}_1||_2^2 \exp(-\beta_1 \cdot ||\mathbf{x} - \mathbf{c}_1||_2^2).$$

For properly selected step size α, the weights during iteration *i+1* are updated using gradient descent as

$$\mathbf{c}_1^{i+1} = \mathbf{c}_1^i - \alpha \frac{\delta e}{\delta \mathbf{c}_1^i}, \beta_1^{i+1} = \beta_1^i - \alpha \frac{\delta e}{\delta \beta_1^i}. \tag{5.12}$$

In the final layer, let the first value in the estimated pixel vector be $\hat{\mathbf{x}}_1$. Thus, following the Fan model from Eq. (6.3)

$$\hat{\mathbf{x}}_1 = \sum_{i=1}^{P} [\hat{\mathbf{a}}_x]_i \cdot \mathbf{E}_{i,1} + \gamma \sum_{i=1}^{P-1} \sum_{j=i+1}^{P} [\hat{\mathbf{a}}_x]_i \cdot [\hat{\mathbf{a}}_x]_j \cdot \mathbf{E}_{i,1} \cdot \mathbf{E}_{j,1},$$

Note that γ has no subscript here as it is assumed to be a scalar for one mixed pixel. The derivatives of output $\hat{\mathbf{x}}_1$ w.r.t. $\mathbf{E}_{1,1}$ and γ would be

$$\frac{\delta \hat{\mathbf{x}}_1}{\delta \mathbf{E}_{1,1}} = [\hat{\mathbf{a}}_x]_1 + \gamma \sum_{j=2}^{P} [\hat{\mathbf{a}}_x]_j \cdot \mathbf{E}_{j,1},$$
$$\frac{\delta \hat{\mathbf{x}}_1}{\delta \gamma} = \sum_{i=1}^{P-1} \sum_{j=i+1}^{P} [\hat{\mathbf{a}}_x]_i \cdot [\hat{\mathbf{a}}_x]_j \cdot \mathbf{E}_{i,1} \cdot \mathbf{E}_{j,1}, \tag{5.13}$$

Similarly to Eq. (5.12), the weights during iteration *i+1* are updated using gradient descent as

$$\mathbf{E}_{1,1}^{i+1} = \max(\mathbf{E}_{1,1}^i - \alpha \frac{\delta e}{\delta \mathbf{E}_{1,1}^i}, 0), \gamma^{i+1} = \gamma^i - \alpha \frac{\delta e}{\delta \gamma^i}. \tag{5.14}$$

where the projected gradient step imposes the nonnegativity constraints for the weights $\mathbf{E}_{i,j}$. A similar procedure can be followed for the rest of the weights and other nonlinear models considered. Notice also from (6.21) that each weight $\mathbf{E}_{i,j}$ will be updated while fixing the other ones to their most recent update. Since the Adam optimizer will be used here, there is no need to explicitly define such equations for backpropagation during implementation, rather the optimizer will account for this automatically. Nevertheless, these equations have been provided for further clarity.

5.2.4 Higher degree cross-products

Earlier a crossproduct generator was introduced to calculate the nonlinear components of the mixture via second degree products of the abundances and endmembers, thus emulating the interactions between light from the materials as they propagate towards the sensor. This is the reason why the mixing models in Eqs. (6.3)-(6.4) addressed the nonlinear components via second degree crossproducts only. It is possible that light from more than two materials can interact with each other, which justifies the additional calculation of higher degree products of corresponding endmembers and abundances.

In certain cases, higher degree terms can be significant enough to warrant their calculations [59]. To that end, our novel autoencoder setup in Fig. 6.2 for the Fan and bilinear models facilitates the incorporation of higher degree crossproducts. This can be done readily by stacking higher-order crossproduct terms right below the second-order terms, while introducing additional endmember weights in the output layer. For example, when considering a nonlinear mixing model accounting for fourth-order crossproduct terms, the reconstructed mixed pixel at the output of the autoencoder structure can be expressed as

$$\begin{aligned}\hat{x}(E,\gamma) &= \sum_{j=1}^{P}[\hat{\mathbf{a}}_x]_j \cdot \mathbf{E}_{j,:} + \gamma_x(\sum_{j=1}^{P-1}\sum_{l=j+1}^{P}[\hat{\mathbf{a}}_x]_j \cdot [\hat{\mathbf{a}}_x]_l \cdot \mathbf{E}_{j,:}\odot\mathbf{E}_{l,:}\\ &+\sum_{j=1}^{P-2}\sum_{l=j+1}^{P-1}\sum_{m=l+1}^{P}[\hat{\mathbf{a}}_x]_j \cdot [\hat{\mathbf{a}}_x]_l \cdot [\hat{\mathbf{a}}_x]_m \cdot \mathbf{E}_{j,:}\odot\mathbf{E}_{l,:}\odot\mathbf{E}_{m,:}\\ &+\sum_{j=1}^{P-3}\sum_{l=j+1}^{P-2}\sum_{m=l+1}^{P-1}\sum_{n=m+1}^{P}[\hat{\mathbf{a}}_x]_j \cdot [\hat{\mathbf{a}}_x]_l \cdot [\hat{\mathbf{a}}_x]_m \cdot [\hat{\mathbf{a}}_x]_n\\ &\cdot \mathbf{E}_{j,:}\odot\mathbf{E}_{l,:}\odot\mathbf{E}_{m,:}\odot\mathbf{E}_{n,:}).\end{aligned} \tag{5.15}$$

The introduction of higher degree terms can be computationally taxing. Nevertheless, the inclusion of this feature significantly adds to the proposed scheme's versatility, and results will be shown of the proposed framework's performance advantage when working with higher degree terms. Note that existing works are usually limited to bilinear terms only.

Also, note that due to the unique dynamic structure of the output layer due to this property, although the layer is similar in ways to a fully connected layer, this causes the number of nodes in the input to be variable. The PPNM model will be an exception to this fact, as that model cannot accommodate this feature. As an example, if the chosen mixing model has P endmembers and is considering upto the n^{th} degree nonlinear terms,then the Fan model will have $\sum_{i=2}^{n} {}^{P}\mathbf{C}_i$ and the Bilinear model will have $\sum_{i=2}^{n} i^P$ nodes in the input, where ${}^{P}\mathbf{C}_i$ denotes denotes 'P choose i'.

5.3 Numerical Tests

The proposed novel unmixing scheme is compared with four other methods: 1) the linear autoencoder in [62], 2) the nonlinear autoencoder in [71], 3) the MVCNMF (minimum volume constrained non-negative matrix factorization) [54] method, and 4) the VCA [11] with FCLS (fully constrained least squares) for abundance calculation, since VCA is used only for endmember extraction. These methods encompass all possible combinations of hyperspectral unmixing schemes that are either neural network-based or not, and focused either on the linear or a nonlinear mixing model.

[62] is an autoencoder network that uses mainly fully connected layers in order to take the input layer and compress it into a latent vector that represents the abundances. For the activation functions in the layers of this network, the leaky rectified linear unit (ReLU) was used. This method will be referred to as the LNN (linear neural network). [71] is an autoencoder network that performs unsupervised nonlinear unmixing, and it utilizes the PPNM mixing model. Here, the ReLU activation was used for the layers. This method will be referred to as the NLNN (non-linear neural network). [54] is an unsupervised scheme that aims to minimize the volume spanned by the endmembers, and uses that as a constraint to perform nonnegative matrix factorization for unmixing.

The tests were conducted using both semi-synthetic and real datasets. For the semi-synthetic datasets, three source datasets were used, namely the Pavia Center, Botswana and Pavia University datasets [51]. Four classes from each dataset were chosen, and for every 1000 pixels, one class had a majority abundance of between 80% and 90%, which was randomly chosen. The remaining abundances were randomly split between the other three classes. Doing this for every class, each dataset had a total of 4000 pixels. During creation of the data, for the Fan and bilinear models, the values for γ_i were set to 1, while for the PPNM model, the values of $\mathbf{b}_i$ were following a uniform distribution within the interval [-0.3,0.3]. For the VCA+FCLS method specifically, for the Fan and bilinear models, the values for γ_i were presumed to be known, while for the PPNM model, the values for $\mathbf{b}_i$ were assigned by a different set of randomly generated values using the same uniform distribution, within the same interval [-0.3,0.3]. This implies that the VCA+FCLS method has an advantage over the other methods in the PPNM model.

As mentioned before, the K-means initialization of the endmember weights is a better choice compared to the well-used VCA when dealing with significant noise. To further challenge the methods used, the data were contaminated with white additive noise, such that the signal-to-noise ratio (SNR) values for the data were ranging from 0dB to 20dB. For all datasets, all SNR values, and all mixing models, each accuracy metric was obtained by averaging across 50 independent trials.

The autoencoder used the Adam optimizer [78] implementing the updating recursions in Eqs. (5.12) and (5.14) to learn weights that would yield an output that minimizes the RMSE (root mean square error) between the output and the input. The number of epochs was set to 50, and the learning rate was set to 10^{-4}. For the Fan and bilinear models, the learnable weights for γ were initialized to be equal to 1, while the learnable weights for values of the vector $\mathbf{b}$ in the PPNM model, were initialized to be equal to 0. The weights for the RBF centers in the RBF layer, as well as the weights in the final layer for the Fan and bilinear models, were initialized to be equal to the Kmeans cluster centers of the dataset. The tests were conducted on a PC with a core i5 6400 processor with an Nvidia RTX 2060 GPU in Tensorflow 2.1.0. Since the centers in the RBF layer, as well as the weights in the final layer can both be used as endmember estimations, either can be used to measure endmember estimation accuracy. However, the weights in the final layer form more accurate endmember estimates as they interact directly with the nonlinear terms of the abundances. The novel framework code is available at [79].

To measure endmember estimation accuracy, the spectral angle distance (SAD) metric was used. For some actual endmember $\mathbf{e}$ and its estimation $\hat{\mathbf{e}}$, the SAD is

$$SAD = P^{-1} \sum_{i=1}^{P} cos^{-1}(\frac{\mathbf{e}_i{}^T \cdot \hat{\mathbf{e}}_i}{||\mathbf{e}_i||_2 \cdot ||\hat{\mathbf{e}}_i||_2}). \tag{5.16}$$

For abundance estimation, the metric used was the RMSE found as

$$RMSE = (P \cdot N)^{-1} \sum_{i=1}^{N} (||\mathbf{a}_i - \hat{\mathbf{a}}_i||_2), \tag{5.17}$$

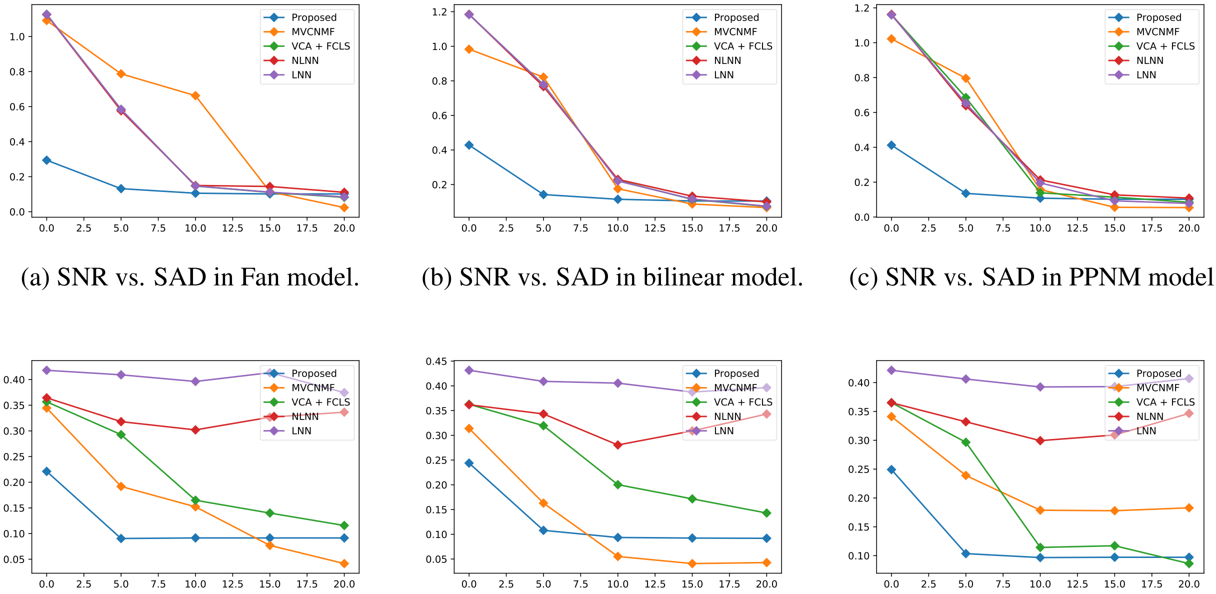
where the actual abundance for some mixed pixel is $\mathbf{a}$, and its estimation is $\hat{\mathbf{a}}$.

5.3.1 Semi-Synthetic Data

For the generation of semi-synthetic data, the AVIRIS airbone sensor acquiring hyperspectral images over Pavia University and Pavia Center, of Pavia, Italy, and of a region over the country of Botswana were used [51]. The batch size was set to 4000. This is convenient in the sense that it allows for faster operations, while giving a smoother reduction of the cost function over the epochs. The reason is that the gradient descent calculates the gradients for all the samples in a batch, and takes the average of those gradients when updating the weights. If the batch size is small, then gradient descent moves towards a direction that optimizes taking into consideration only samples in the batch. Since samples in subsequent batches can cause the gradient to descend in a different direction, the resultant gradient descent updates have a higher likelihood to fluctuate and have an erratic pattern. Contrary, larger batch sizes will allow for a smoother and more stable descent over the entire dataset. It should be noted that for simplicity, the results obtained here were with assumption of prior knowledge of the correct mixing model during the unmixing process, since extensive testing showed that choosing a different unmixing layer produced a negligible difference. Although this means that the network could have been designed with only one mixing model, the three major mixing models have nevertheless been incorporated in order have a highly versatile network architecture.

5.3.1.1 Pavia University

Fig. 5.3 shows the plots of the SAD and RMSE values for the various methods in the Pavia University dataset. It can be seen that for low SNR, VCA will be susceptible to the high noise, and its accuracy will be quite low compared to the proposed method. For higher SNR, given the high effectiveness of VCA, it will have a slight edge over K-means as far as endmember estimation is concerned. However, the key advantage in the proposed method lies in its ability to achieve higher separability in the kernel space, given that some hyperspectral materials might have properties that are too similar in the feature space for VCA to find correct endmembers. Due to this, VCA may erroneously ignore materials, and instead take multiple endmembers from the same material instead. Although VCA might still be better than K-means for endmember estimation in high SNR values, VCA can perform worse during abundance estimation, since the higher abundances might be attributed to the wrong endmember. This is observed in the RMSE, as the LNN and NLNN methods face considerable difficulty. VCA+FCLS and the MVCNMF have a slight edge over the proposed method in higher SNR values, but depending on the mixing model, this edge is inconsistent.



(a) SNR vs. SAD in Fan model.

(b) SNR vs. SAD in bilinear model.

(c) SNR vs. SAD in PPNM model.

(d) SNR vs. RMSE in Fan model.

(e) SNR vs. RMSE in bilinear model.

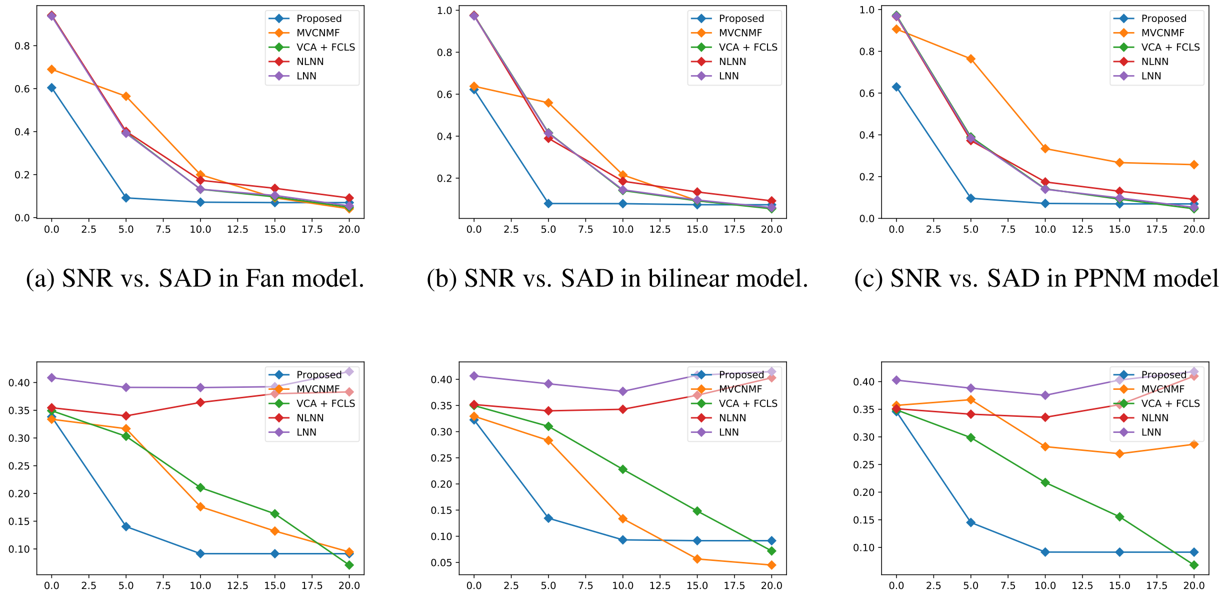
(f) SNR vs. RMSE in PPNM model.

Figure 5.3: SNR (dB) vs. SAD and SNR (dB) vs. RMSE in all methods for the Pavia University dataset.

Furthermore, it is likely that the LNN and NLNN are facing further difficulties due to the structure of using fully connected layers to obtain a latent vector that is assumed to contain the features of the endmember materials in a highly compressed fashion. Although this can be effective, it can be highly sensitive to the nature of the materials involved, and will be vulnerable if the spectral reflectances of different materials are too similar. Evidently, the kernelized pixels will be facilitating better the separation of features of even challenging data such as this, and will yield much better accuracies with greater consistency than MVCNMF and VCA+FCLS.

5.3.1.2 Botswana

Fig. 5.4 shows the accuracy measures of the various methods vs. SNR values. The proposed method will either have extremely better values, or noticeably better values until



(a) SNR vs. SAD in Fan model.

(b) SNR vs. SAD in bilinear model.

(c) SNR vs. SAD in PPNM model.

(d) SNR vs. RMSE in Fan model.

(e) SNR vs. RMSE in bilinear model.

(f) SNR vs. RMSE in PPNM model.

Figure 5.4: SNR (dB) vs. SAD and SNR (dB) vs. RMSE in all methods for the Botswana dataset.

15dB SNR, then the performance of our method will be slightly behind the MVCNMF and the VCA+FLCS methods. The RMSE is a different matter however. Not only are the LNN and NLNN methods considerably inaccurate, MVCNMF faces significant difficulty for the PPNM model, while VCA+FCLS fares a little better for higher SNR. This dataset is particularly challenging for 0dB SNR, as shown by the fact that the proposed method's SAD and RMSE are almost as high as the other methods. However, for 5dB SNR, the proposed method is far better at accurately unmixing the pixels. In fact, above 5dB SNR, the SAD values plateau, showing that it is performing at its peak even in severe noise. The RMSE does not trail far behind as well, reaching its peak accuracy at 5dB.

5.3.1.3 Pavia Center

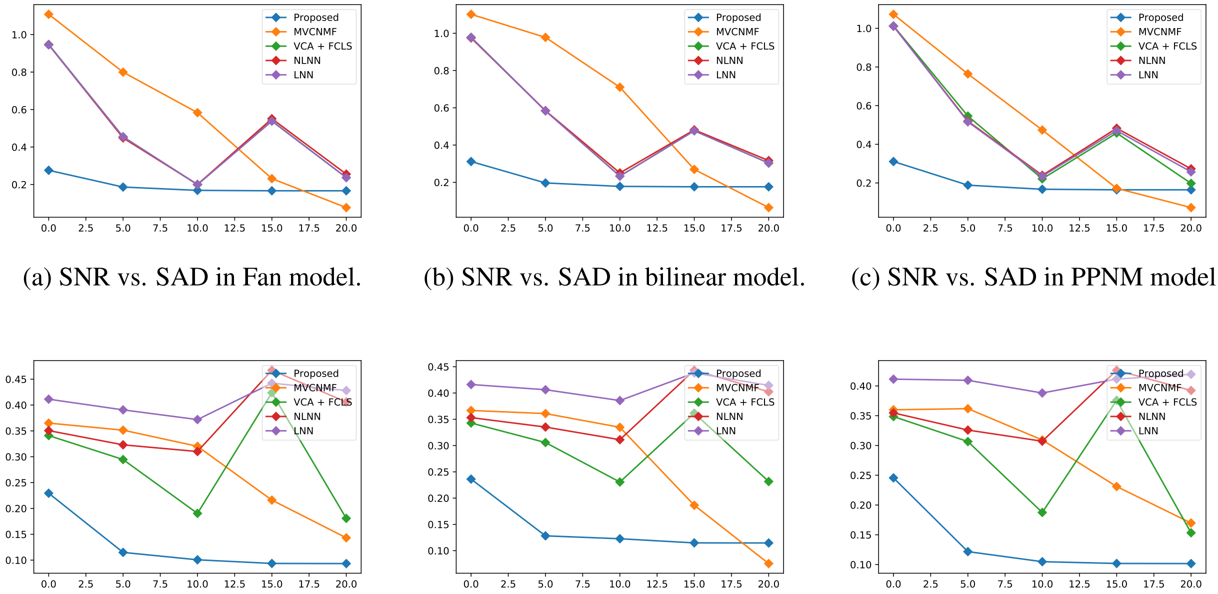
The third semi-synthetic dataset to test is the Pavia Center. This was a particularly interesting dataset, because the materials in this dataset exhibited similarities that were too high for VCA to discern well. Fig. 5.5 depicts that this caused high inconsistencies even in higher SNR values, especially in the 15dB value. In fact, across the 50 iterations being run, nearly all of them caused especially high SAD and RMSE values, as can be observed in the plots. For SAD values, only MVCNMF had slightly better accuracy than the proposed method for 20dB. However, for RMSE values, except only for the bilinear model at 20dB compared to MVCNMF, the proposed method not only achieved the best accuracies for all values of SNR, it began to plateau at 5dB, further showing its extremely high effectiveness in the presence of high noise. Note that the SAD of the proposed method is nearing the plateau at 0dB SNR, showing its robustness to severe noise.

5.3.1.4 Higher Degree Cross-Products

As mentioned previously, the proposed framework is versatile for easily allowing high-degree nonlinear terms. To this end, as there were four classes making up the various mixed pixels, the Fan model was expanded upon, and allowed for up to third degree, and also up to fourth degree cross-products [cf. Eq. (5.15).].

The plots in Fig. 5.6 show that there is little to no difference in the RMSE values. These is, however, a small difference in SAD values for the Pavia University and Pavia Center datasets.

Since the higher order crossproducts have little effect in the unmixing performance results, the experiments were re-run with a higher magnitude of γ, raising its value from 1 to 10, thus making the nonlinear component much more significant in the mixture. Despite not changing the initial value of the weights pertaining to γ in the final layer of our



(a) SNR vs. SAD in Fan model. (b) SNR vs. SAD in bilinear model. (c) SNR vs. SAD in PPNM model.

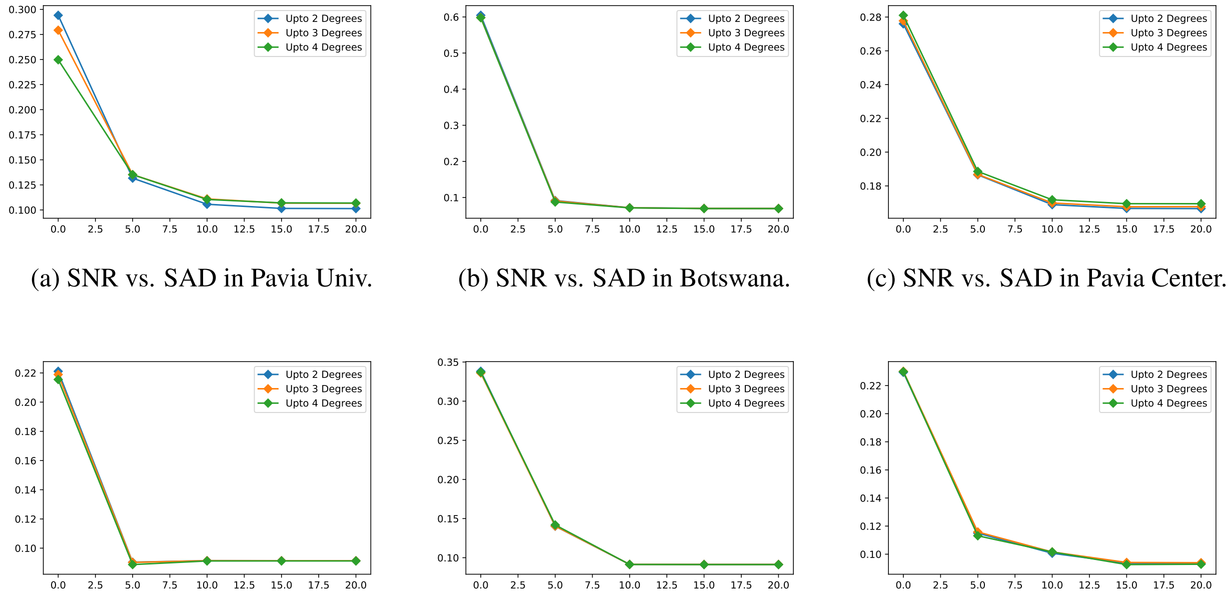
(d) SNR vs. RMSE in Fan model. (e) SNR vs. RMSE in bilinear model. (f) SNR vs. RMSE in PPNM model.

Figure 5.5: SNR (dB) vs. SAD and SNR (dB) vs. RMSE in all methods for the Pavia Center dataset.

neural network, it was clearly the most robust with respect to the new mixtures. Fig. 5.7 clearly indicates that the other methods, especially the MVCNMF method was impacted with the now-stronger nonlinear component in the mixture. However, the performance of our proposed method is largely unaffected.

5.3.1.5 K-means vs. VCA Initialization

In all existing neural network-based unmixing schemes where weights are being used for endmember estimation, VCA will usually be chosen for initializing. Differently, here K-means was utilized. The primary reason is for dealing with severe data noise. As seen in the semi-synthetic results, the competing methods using VCA for endmember initialization had higher SAD values than the proposed method, only in the highest SNR values, and even then those superior scores were not consistent among all the datasets. In order to further



(a) SNR vs. SAD in Pavia Univ.

(b) SNR vs. SAD in Botswana.

(c) SNR vs. SAD in Pavia Center.

(d) SNR vs. RMSE in Pavia Univ.

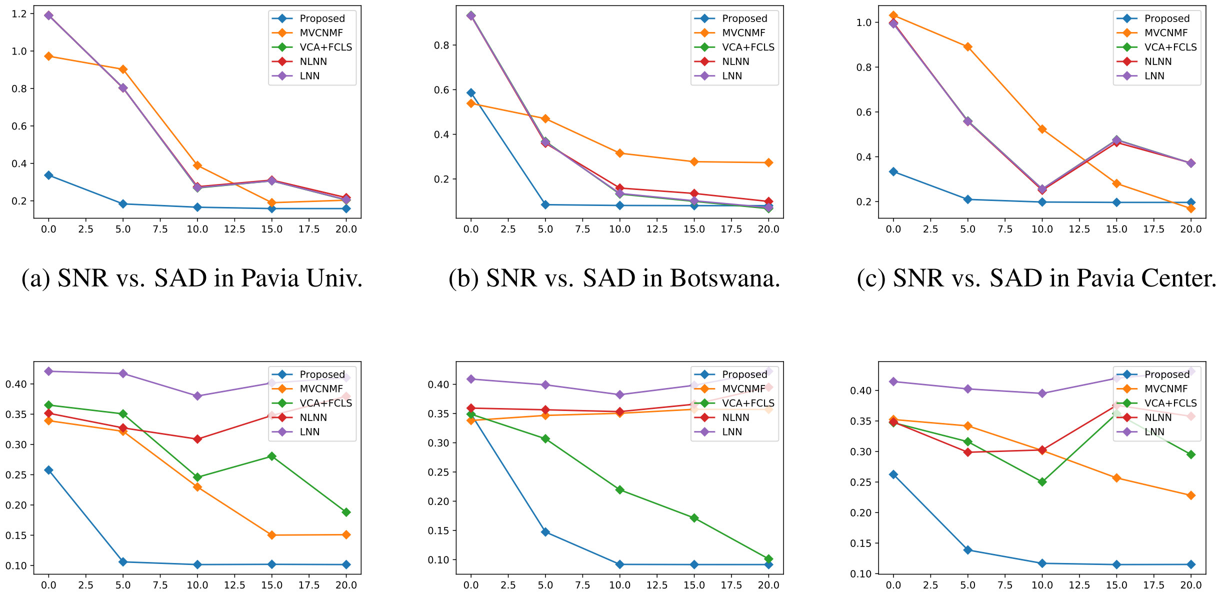
(e) SNR vs. RMSE in Botswana.

(f) SNR vs. RMSE in Pavia Center.

Figure 5.6: SNR (dB) vs. SAD and SNR (dB) vs. RMSE of the proposed method with varying maximum crossproduct degrees.

illustrate the advantage of K-means initialization in our scheme over VCA, K-means was replaced with VCA instead, then all the previously conducted tests were re-run for 50 trials each. The plots in Fig. 5.8 show the advantage of having K-means as the choice of initialization over VCA for the Pavia Center dataset.

It should be mentioned, that this advantange of K-means over VCA is not effective solely with the proposed method. To show how robust K-means is over VCA for severe noise, the NLNN method was re-run such that the initialization of the endmember weights were done with K-means instead of VCA for the Pavia Center dataset. Fig. 5.9 shows that when initializing the NLNN method's endmember weights with K-means instead of VCA, the SAD values have an advantage over VCA in most cases for SNR values of 5dB, and every case for 0dB. Furthermore, in the case of RMSE values, the novel framework achieves better accuracy over nearly all SNR values and over all mixing models.



(a) SNR vs. SAD in Pavia Univ.

(b) SNR vs. SAD in Botswana.

(c) SNR vs. SAD in Pavia Center.

(d) SNR vs. RMSE in Pavia Univ.

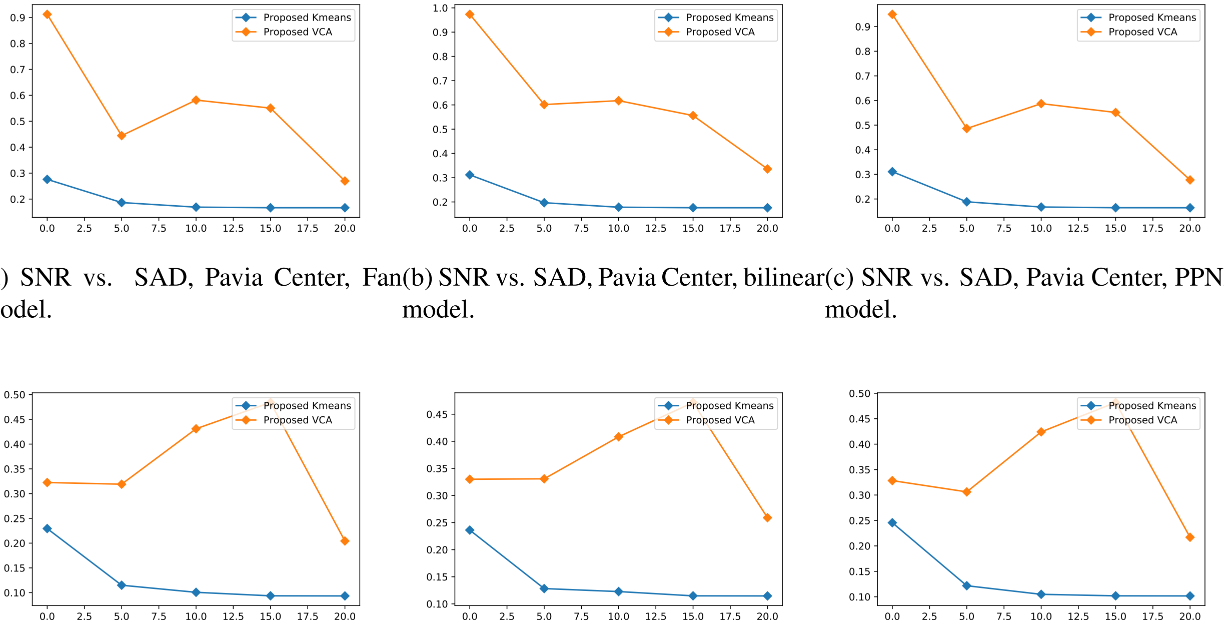
(e) SNR vs. RMSE in Botswana.

(f) SNR vs. RMSE in Pavia Center.

Figure 5.7: SNR (dB) vs. SAD and SNR (dB) vs. RMSE for all methods with up to 3^{rd} degree crossproducts with $\gamma = 10$.

5.3.1.6 Choice of Unmixing Layer

In the performance results provided above, the unmixing layer was always chosen to be the same as the mixing model for generating the semi-synthetic datasets. However, through extensive testing it was found that even if the unmixing layer is different from the mixing model, the difference in performance was negligible. Indeed, this was the case when the nonlinear terms during the mixing are noticeably weak compared to the linear terms. On the other hand, the stronger the nonlinear terms are, the worse the performance will be when the unmixing model is chosen incorrectly. As an example, consider the Botswana dataset where, the mixed pixels are generated with $\gamma = 50$. In Fig. 5.10 we can see that if the nonlinear components are extremely strong, the correct choice of the unmixing layer becomes significant. Thus, in the case of the datasets used previously, even though the



(a) SNR vs. SAD, Pavia Center, Fan model.

(b) SNR vs. SAD, Pavia Center, bilinear model.

(c) SNR vs. SAD, Pavia Center, PPNM model.

(d) SNR vs. RMSE, Pavia C., Fan model.

(e) SNR vs. RMSE, Pavia C., Bilin. model.

(f) SNR vs. RMSE, Pavia C., PPNM model.

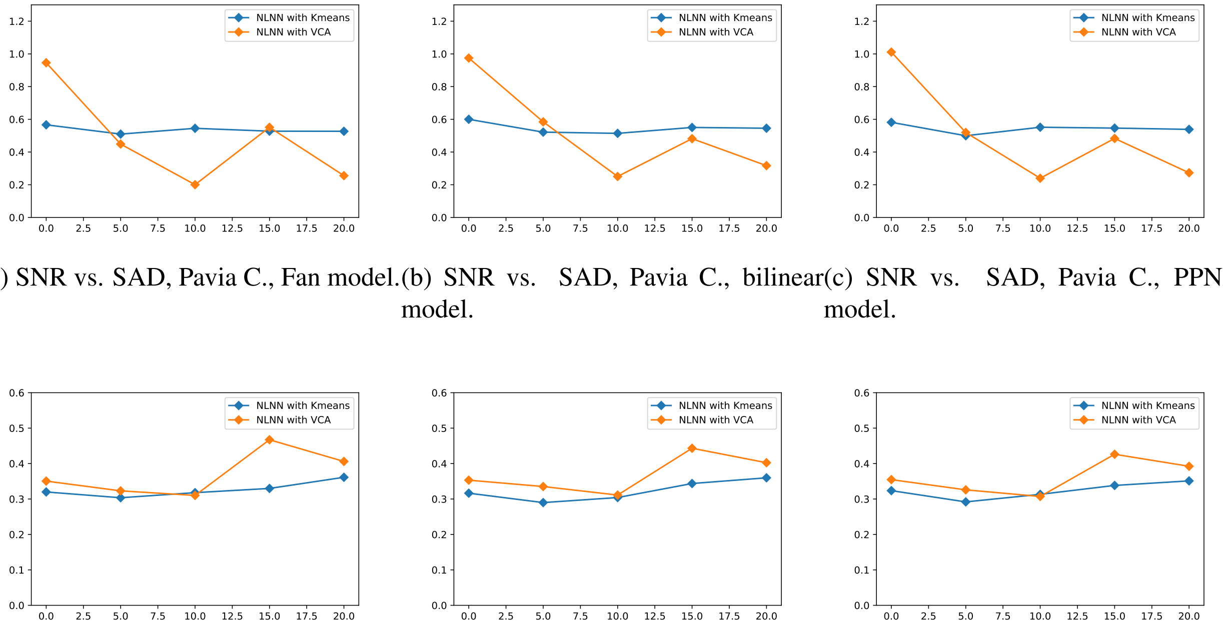
Figure 5.8: SNR (dB) vs. RMSE and SAD in the novel method for Pavia Center with varying endmember weight initializations.

proposed method will be highly accurate regardless of the unmixing layer used, it will still be extremely effective as long as the nonlinear terms are not significantly amplified with respect to the linear terms. Under such a setting in Fig. 5.7 (where $\gamma = 10$), existing alternatives behave much worse than our method. Nonetheless, when nonlinear terms are strong knowledge of the mixing model is required.

5.3.2 Real-World Data

5.3.2.1 Samson

For emulating real-world scenarios, the well-used Samson dataset [77] was used for testing. This is a 95×95 pixel image of a waterside area. Fig. 6.9 shows the heatmaps for the '*Soil*', '*Tree*', and '*Water*' classes. The NLNN method faces considerable difficulty with



(a) SNR vs. SAD, Pavia C., Fan model.

(b) SNR vs. SAD, Pavia C., bilinear model.

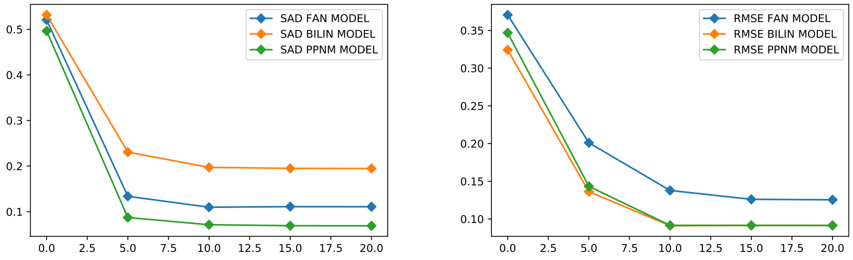
(c) SNR vs. SAD, Pavia C., PPNM model.

(d) SNR vs. RMSE, Pavia C., Fan model.

(e) SNR vs. RMSE, Pavia C., bilinear model.

(f) SNR vs. RMSE, Pavia C., PPNM model.

Figure 5.9: SNR (dB) vs. SAD and SNR (dB) vs. RMSE in the NLNN [71] method for Pavia Center with varying endmember weight initializations.



(a) SNR vs. SAD

(b) SNR vs. RMSE

Figure 5.10: SNR (dB) vs. SAD and SNR (dB) vs. RMSE in the proposed method for Botswana, with only PPNM unmixing layer against various mixing models having $\gamma = 50$.

the '*Soil*' class of the data, and the LNN method found it challenging to tackle the same class. In the '*Tree*' class, the outer fringes of the corresponding areas provided a challenge for the proposed novel method, but it showed the best balance among all the three classes. In the lower left portion of the image, a small area proved difficult to correctly classify as part of the '*Tree*' class, instead it was misclassified as belonging mostly to the '*Water*' class.

For VCA+FCLS, the PPNM mixing model was used, while for the proposed method, the Fan model was employed.

5.3.2.2 Jasper Ridge

Another dataset to observe is the Jasper Ridge dataset [77]. This is a 100×100 pixel image of an area with a body of water through the middle. Fig. 5.12 shows the heatmaps for the '*Tree*', '*Water*', '*Dirt*' and '*Road*' classes. Although all the classes fared well with the 'Water' class, the MVCNMF method found higher abundances for water in the rest of the image. The proposed method was significantly accurate with the 'Dirt' class, and was also the most accurate with the 'Road' class, with MVCNMF identifying the road as well, but also considered the body of water to partly contain the same material as well. In the 'Tree' class, some areas had significantly high abundances in the ground truth, characterized by the highly bright areas. Although the MVCNMF did relatively well with this class, the proposed method fared the best particularly in classifying these brighter areas. Overall, the proposed method clearly performed the best.

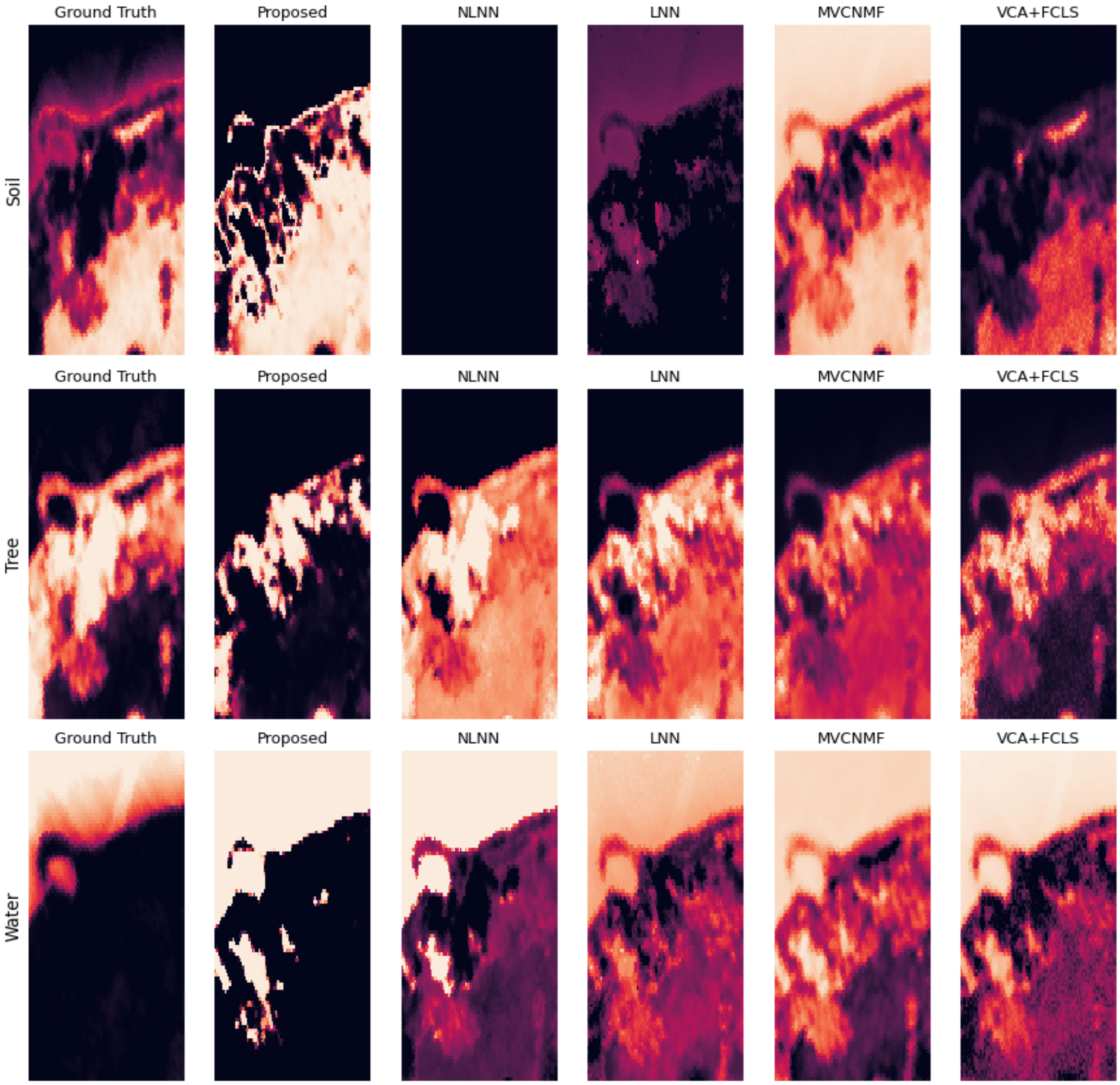


Figure 5.11: Heatmaps of the Samson dataset for different unmixing methods.

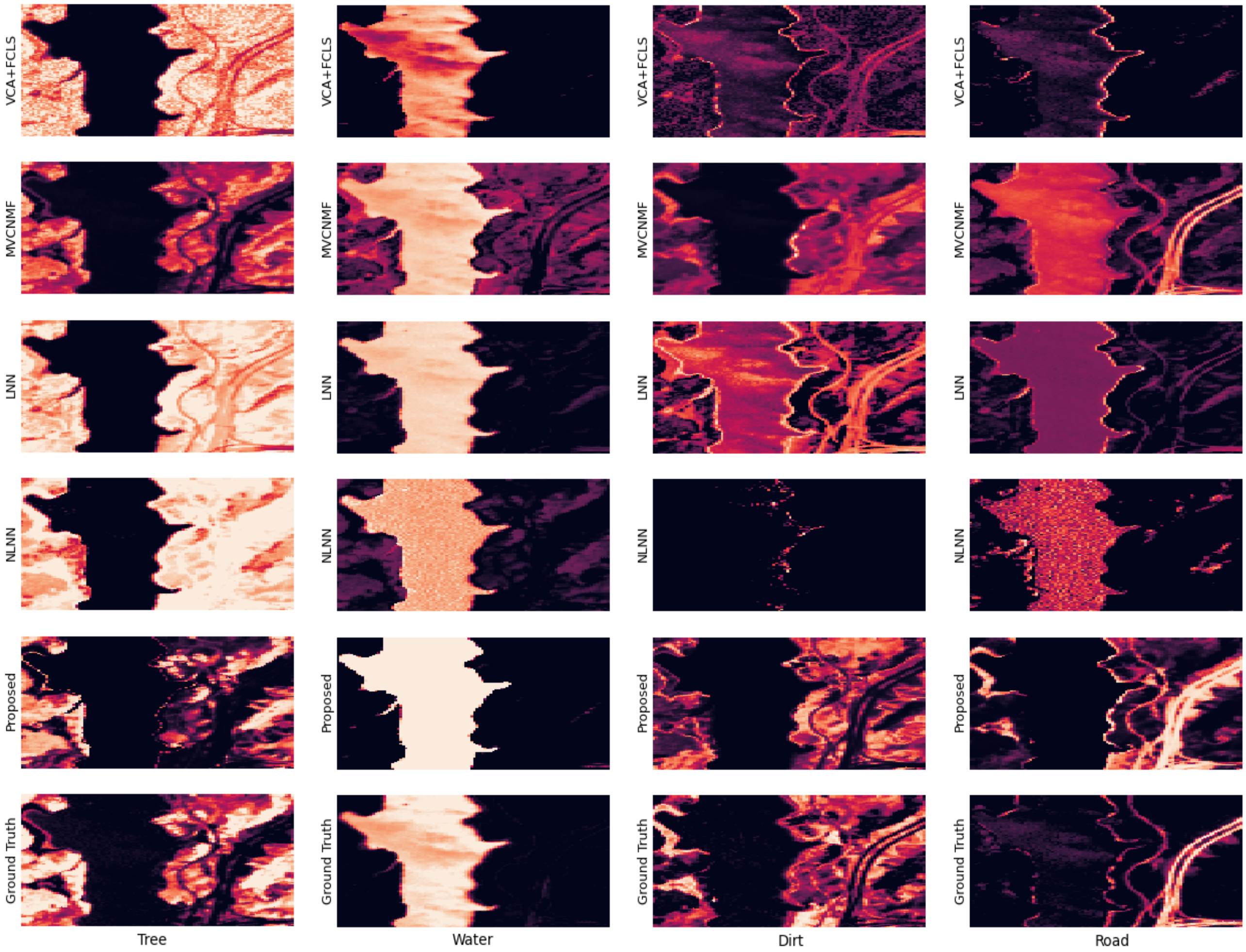


Figure 5.12: Heatmaps of the Jasper Ridge dataset for different unmixing methods.

CHAPTER 6

SPATIAL AWARE HYPERSPECTRAL NONLINEAR UNMIXING AUTOENCODER WITH ENDMEMBER POPULATION ESTIMATION

6.1 Problem Statement and Preliminaries

Hyperspectral images contain information in hundreds of bands, and so record much finer information pertaining to the sensed materials. Assume S to be the number of spectral bands across which the hyperspectral image sensor gathers information. The set $\mathcal{S} := \{1, \ldots, S\}$ contains the indices of the respective spectral bands. Let the information across the aforementioned spectral bands stored among N pixels, be represented by matrix $\mathbf{L} \in \mathbb{R}^{N\times S}$. A key obstacle with hyperspectral images due to lower spatial resolution is that, as opposed to each pixel containing one material, the pixels are affected by a mixture of materials known as *endmembers*.

A well-known model that emulates these mixtures, encapsulated by $\mathbf{L}$, coming from linear combinations of R endmembers, is the linear mixing model (LMM), i.e.,

$$\mathbf{L} = \mathbf{F} \cdot \mathbf{P} + \epsilon. \tag{6.1}$$

Here, $\mathbf{F} \in \mathbf{R}^{N\times R}$ is a matrix that quantifies the fractional contributions of the endmembers in the mixed pixels. These contributions are known as *abundances*. As mentioned previously, unlike a majority of existing methods where R is assumed to be known, in our framework is an unknown to be estimated. The endmembers are represented by $\mathbf{P} \in \mathbf{R}^{R\times S}$, and ϵ denotes sensing noise. Following this formula, each mixed pixel can be modeled individually as

$$\mathbf{L}_{i,:} = \textstyle\sum_{j=1}^{R} \mathbf{F}_{i,j} \cdot \mathbf{P}_{j,:} + \epsilon_{i,:}, \forall i \in 1, 2, \ldots, N. \tag{6.2}$$

Because $\mathbf{F}$ represents the abundances, or fractional contributions of the endmembers, its values are non-negative, and will sum to one for every mixed pixel (across each row). Being spectral reflectance values, the entries of $\mathbf{P}$ will also be nonnegative, and these conditions can thus be written as $\mathbf{F}_{i,j} \geq 0, \mathbf{P}_{j,:} \geq 0, \forall i \in 1, 2, \ldots, N$ and $\forall j \in \{1, 2, \ldots, R\}$, and also $\sum_{j=1}^{R} \mathbf{F}_{i,j} = 1, \forall i \in \{1, 2, \ldots, N\}$.

The LMM however, is limited in that it cannot account for light from the same, or differing endmembers from scattering with each other. Such interactions are instead accounted for, mainly in three nonlinear mixing models used in current literature. These are known as the Fan model, the Bilinear model [72], and the PPNM (Post Polynomial Nonlinear Mixing) model [75]. The Fan model

$$\begin{aligned} \mathbf{L}_{i,:} = & \textstyle\sum_{j=1}^{R} \mathbf{F}_{i,j} \cdot \mathbf{P}_{j,:} + \gamma_i \sum_{j=1}^{R-1} \sum_{l=j+1}^{R} \mathbf{F}_{i,j} \cdot \mathbf{F}_{i,l} \\ & \cdot \mathbf{P}_{j,:} \odot \mathbf{P}_{l,:} + \epsilon, \forall i \in 1, 2, \ldots, N. \end{aligned} \tag{6.3}$$

incorporates interactions across different endmembers through their Hadamard products (elementwise products). The γ_i values represent scaling factors to control the strength of the nonlinear mixtures in relation to the overall mixture. Since it has been established that nonlinear interactions can also occur among the same endmember [73, 74], they are included in the Bilinear model, which is the same as Eq. (6.3), with the exception that both the summation limits in the nonlinear components, range from 1 to R.

Finally, the PPNM [75] model is simpler, in that it will take the LMM mixture, and add it to the square of itself.

$$\begin{aligned} \mathbf{L}_{i,:} = & \textstyle\sum_{j=1}^{R} \mathbf{F}_{i,j} \cdot \mathbf{P}_{j,:} + \mathbf{b}_i (\sum_{j=1}^{R} \mathbf{F}_{i,j} \cdot \mathbf{P}_{j,:}) \\ & \odot (\textstyle\sum_{j=1}^{R} \mathbf{F}_{i,j} \cdot \mathbf{P}_{j,:}) + \epsilon, \forall i \in 1, 2, \ldots, N. \end{aligned} \tag{6.4}$$

The vector $\mathbf{b}$ of length N resembles γ in Eq. (6.3), as well as for the Bilinear model, as it controls the strength of the nonlinear part of the model.

6.1.1 Kernel Data Mappings

Mixed pixels can be represented as points in the feature space, while estimation of the endmembers is obtained through calculating the K-means [76] cluster centers, and the distances from the mixed pixels w.r.t. the cluster centers gauge the similarities between the endmembers and the mixed pixels. This in turn points to how much the endmembers may be contributing to the mixed pixels to result in these similarities, thus providing estimations of the abundances. However, in the feature space, similarities between different endmembers can cause unmixing to be highly challenging. Rather, this can be achieved with greater ease and accuracy, as has been shown in [52, 61, 68], if the data is mapped from the feature space to a higher dimensional space through implementation of the kernel trick [17]. In particular, the Gaussian Radial Basis Function (RBF) kernel has proven to be particularly effective with hyperspectral images, and will thus be used here. As the RBF distance is a measure of similarity, the higher the RBF distance is between two points, the greater their similarity. Using this logic, with the mixed pixels and K-means cluster centers obtained in the feature space, the higher the RBF distance is of a mixed pixel from a cluster center that represents an estimated endmember, the higher the similarity between them can be inferred. This, in turn, implies that this higher similarity exists due to that respective endmember contributing more towards the mixed pixel, which consequently translates to that endmember having a higher abundance in that mixed pixel. Thus, the RBF distance of some mixed pixel $\mathbf{x}$ from an estimated endmember/cluster center $\mathbf{c}_i$ can be expressed as

$$[\hat{\mathbf{f}}_x]_i = \exp(-\beta \cdot ||\mathbf{x} - \mathbf{c}_i||^2), \forall i \in 1, 2, \ldots, R, \tag{6.5}$$

where β is a parameter which controls the spread of the cluster center in the kernel space. Since our framework is unsupervised, there will be no prior knowledge of the endmembers, and so they will be estimated using K-means on the mixed pixels, thus the Gaussian centers are initially estimated to be the cluster centers. See more details in [84]. What remains then is to calculate these abundances, normalize them to unit summation, and they will represent vector $\hat{\mathbf{f}}_x$ for a given mixed pixel $\mathbf{x}$, which would be a row of the abundance matrix $\hat{\mathbf{F}}$. One key point to observe is that these values, being distances, cannot be negative, which automatically helps to satisfy the nonnegativity constraint without added effort.

Another challenge that can hamper the classification results considerably, is the presence of noise in the images, which can emanate from dust particles in the air, adverse climate and weather conditions, and simply the quality of the sensor. Further, any damage or malfunction in the image sensor's equipment can cause parts of the sensor to become unresponsive, creating entries in the image arrays to be equal to zero, since no information is being gathered. To tackle this issue, spatial information will be utilized in that the pixels in the images are recalculated to be a weighted averaged version of itself and its surrounding neighborhood. The weights to be determined, are calculated as a proportion of the similarity between the center pixel and the pixels in the neighborhood, and the RBF distance is an effective similarity metric. The averaging calculation done here, in conjunction with the averaging performed with Kmeans as part of the endmember estimation algorithm, as well as the weighted averaging filters implemented in the neural network, are extremely effective in mitigating noise and dead pixels.

6.2 Unsupervised Hyperspectral Unmixing Autoencoders

6.2.1 Spatial Averaging Filter

During hyperspectral image acquisition, faulty equipment can cause some pixels in the images to become unresponsive. Such 'dead' pixels in the image data, carry no information and are set at a zero value. With the simple presumption that adjacent pixels are likely to contain information from the same material, the effect of dead pixels, as well as cases where the data contains high noise, are both alleviated by designing a proper spatial moving average filter. Before the image is fed through the network, an averaging filter is applied to every pixel. Let the size of this filter be $n \times n$, where the value of n is odd. For any given pixel, it will be the center of an $n \times n$ neighborhood. The weights in the averaging will be calculated based on the RBF distance of the center pixel w.r.t the surrounding pixels in its neighborhood. Denote the pixel of interest as **x**, and let $\mathcal{A}$ represent the set of pixels in the neighborhood surrounding **x**, so $\mathcal{A} = \{\mathbf{y}_i\}, \forall i = 1, \cdots, n^2$, note that $\mathcal{A}$ is inclusive of **x**. The weights are calculated as

$$w_i = \exp(-1 \cdot \beta_a \cdot ||\mathbf{x} - \mathbf{y}_i||_2^2), \forall i = 1, \cdots, n^2, \tag{6.6}$$

where $\beta_a = (n^{-2} \sum_{i=1}^{n^2} ||\mathbf{x} - \mathbf{y}_i||_2^2)^{-2}, \forall \mathbf{y}_i \in \mathcal{A}$. Thus, β_a is calculated by taking the inverse squared average of the Euclidean distances between **x** and $\mathbf{y}_i$, where the subscript a signifies that this is the β specifically for the averaging filter. This formula for β_a is chosen because, the Euclidean distances among the surrounding pixels w.r.t. the center pixel gives a good idea of the span of the pixels' positions in the kernel space, thus ensuring that the RBF distances are scaled appropriately. Once all the w_i values are calculated, normalizing them to unit summation will yield the weights used to calculate the average for each pixel across all the spectral bands as

$$w_i = \frac{w_i}{\sum_{j=1}^{n^2} w_j}, \forall i = 1, \cdots, n^2, \quad \text{with } \bar{\mathbf{x}} = \sum_{i=1}^{n^2} w_i * \mathbf{y}_i \tag{6.7}$$

The averaged pixel $\bar{\mathbf{x}}$ thus replaces the original pixel **x** for all subsequent steps in the algorithm, and the same applies to all other pixels in the dataset. This averaging filter also works very effectively even with introduction of dead pixels. The reason can be found in Eq. (6.6). If one or more entries in **x** and $\mathbf{y}_i$ are zero, since the norms of their differences are calculated, the impact from the zero entries is far less significant.

6.2.2 Unsupervised Endmember Number Estimation

Different from the majority of existing unmixing approaches, the number of endmembers is not available. After the moving averaging filter is applied, a novel scheme is utilized to determine the number of endmembers in the data. As the kernel covariance matrix of the data would ideally have a block diagonal form, where the diagonal blocks are present due to the existence of the various endmembers, a rank equivalent matrix - where the rank number is equal to the number of endmembers - would ideally be a close approximation of the original kernel covariance matrix with minimal mean-square error. This block diagonal structure arises since the endmembers being sufficiently dissimilar can be viewed as uncorrelated in the kernel space (negligible RBF distances). Thus, the number of unknown endmembers will be found by estimating the rank of the kernel covariance matrix corresponding to the mixed pixels. To that end, the RBF kernel covariance matrix of the data is first calculated. For a hyperspectral image containing N pixels, the pairwise pixel distance matrix $\mathbf{C} \in \mathbb{R}^{(N \times N)}$ is defined by

$$\mathbf{C}_{i,j} = ||x_i - x_j||^2. \tag{6.8}$$

Since the number of endmembers R is unknown, it is assumed to be within $[2, Z]$ where Z is an adequately high number. The BRF kernel covariance matrix $\mathbf{K}$

$$\mathbf{K}_{i,j} = \text{exp}(-\beta_c * \mathbf{C}_{i,j}), \forall i, j = 1, \cdots, N \tag{6.9}$$

where the calculation of β_c will be explored later on. Once $\mathbf{K}$ is calculated, the task remains to find which lowest rank equivalent matrix of $\mathbf{K}$ will provide minimal reconstruction error, which would point towards the most likely number of endmembers. Thus, we calculate the reconstruction error for different rank approximations of $\mathbf{K}$, with the, z^{th} rank reconstruction error being found as

$$\mathbf{e}_z = ||\mathbf{K} - \sum_{i=1}^{z} \lambda_i \mathbf{v}_i \mathbf{v}_i^T||_2, \forall z = 1, \cdots, Z \tag{6.10}$$

Here, $\lambda_i, \mathbf{v}_i$ represent the i^{th} most significant eigenvalue and the corresponding eigenvector, respectively. Once the Z reconstruction errors are calculated, we have to observe where the error stabilizes. For this purpose, we calculate the differences of the errors between every two consecutive ranks Thus, we obtain the vector of differences $\tilde{\mathbf{e}}$ as

$$\tilde{\mathbf{e}}_i = \mathbf{e}_i - \mathbf{e}_{i+1}, \forall i = 1, \cdots, Z-1. \tag{6.11}$$

6.2.2.1 Calculation and scaling of β_c

During RBF kernel mapping, since the values inside the exponential are scaled by a value commonly denoted in a generalized way as β, obtaining the kernel covariance matrix $\mathbf{K}$ also concerns the choosing of a proper value for β value, known as β_c, where the subscript c signifies that this is related to the matrix $\mathbf{C}$. This is calculated by averaging the row-wise max values of the upper triangular of $\mathbf{C}$, which we shall denote as $\mathbf{C}_U$. Thus,

$$\beta_c = (N)^{-1} \sum_{i=1}^{N} max(\mathbf{C}_U[i, i:]). \tag{6.12}$$

Now, Eqs. (6.9)-(6.11) can be implemented with the strict assumption that the value of β_c is ideal for the kernel covariance matrix. However, given the diversity of the data in hyperspectral images, this might not be the case. Because of this, the value of β_c should be appropriately scaled. As we are dealing with an unsupervised model and thus have no prior information regarding how to appropriately scale β_c, the best way to approach this problem is to measure for a range of values to scale β_c with. Let us assume the vector of scaling values to be $\mathbf{s}_c$, whose length is Q and chosen manually, so Eq. 6.9 will be rewritten to look like

$$[\mathbf{K}_{i,j}]_k = \exp(-\mathbf{s}_{c_k} \cdot \beta_c \cdot [\mathbf{C}_{i,j}]_k), \tag{6.13}$$

$\forall i, j = 1, \cdots, N, k = 1, \cdots, Q$. For Q different kernel matrices, we would be repeating Eqs. (6.10)-(6.11) and obtain Q different $\tilde{\mathbf{e}}$ vectors, which we can group into a matrix $\tilde{\mathbf{E}}$ of size $Q \times (Z-1)$, where $\tilde{\mathbf{E}}$ = $[\tilde{\mathbf{e}}^1, \cdots, \tilde{\mathbf{e}}^Q]^T$. Ideal values in $\mathbf{s}_c$ will yield corresponding $\tilde{\mathbf{e}}$'s (each $\tilde{\mathbf{e}}$ being a row in $\tilde{\mathbf{E}}$), where values in the corresponding $\tilde{\mathbf{e}}_1, \cdots, \tilde{\mathbf{e}}_{R-1}$'s will be high and subsequent values will be low, thus causing a steep drop between $\tilde{\mathbf{e}}_{R-1}$ and $\tilde{\mathbf{e}}_R$. However, non-ideal values in $\mathbf{s}_c$ will cause lower values in $\tilde{\mathbf{e}}_1, \cdots, \tilde{\mathbf{e}}_{R-1}$ as well, due to the resultant kernel covariance matrix not being the best representation of the data. Thus, for all rows in $\tilde{\mathbf{E}}$, the values across the first R columns will vary significantly, while values between $R, \cdots, Z-1$ will be consistently low, thus varying very little. Logically, measuring the variance across the columns of $\tilde{\mathbf{E}}$, and seeing where the variance drops will lead us towards the number of endmembers in the data.

To this end, we calculate the variance of the columns of $\tilde{\mathbf{E}}$, using the formula

$$\sigma_z^2 = Q^{-1} \textstyle\sum_{i=1}^{Q} (\tilde{\mathbf{E}}_{i,z} - \sum_{j=1}^{Q} \tilde{\mathbf{E}}_{j,z}/Q)^2, \tag{6.14}$$

$\forall z \in \{1, \cdots, Z-1\}$. Then, we observe where the variance drops to a low value. We have set this to be below a predetermined threshold, which is a fraction of the peak variance. The drop below this threshold, as discussed previously, will point us towards the correct number of endmembers.

The drop in the variance is calculated by measuring the differences of variances across adjacent columns of $\tilde{\mathbf{E}}$. While this works well, there is one risk with this approach. Among the first variances prior to reaching R, if two adjacent variances are high, but coincidentally very similar in value, their difference might be so low that it goes below the threshold, and the algorithm mistakenly chooses a value of $\hat{R} < R$. To tackle such fringe cases, the variances themselves are also observed alongside their differences. To illustrate how this works, consider a dataset with 4 endmembers, and observe the plots in variances for the corresponding $\tilde{\mathbf{E}}$ matrix in Fig. 6.1. Note that when reaching index 4 (corresponding to 4 endmembers), because the variance drops from index 3 to 4, this can be used to conclude that there are indeed 4 endmembers in the data. Also, note that in the differences of variances across $\tilde{\mathbf{E}}$'s columns (orange line), the value drops significantly between indices 1 and 2. This might have caused $\hat{R}$ to be erroneously estimated as 2, but if validated with the blue line, this can be avoided.

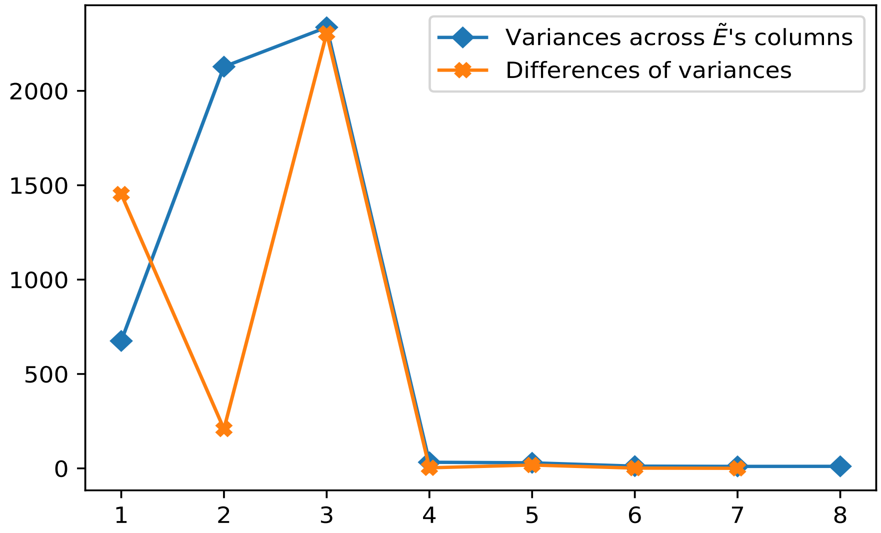


Figure 6.1: Plots of variances for $\tilde{\mathbf{E}}$ matrix's columns.

6.2.3 Autoencoders for Nonlinear Unmixing

The autoencoder unmixing neural network structure is shown in Fig. 6.2 with R=4 endmembers for illustration purposes and without loss of generality.. This network can accommodate the three different mixing models, where we can assume unmixing of a dataset containing 4 endmembers. After the mixed pixel is fed into the proposed neural network structure along with its neighborhood, it is averaged with the RBF weights, and then the

RBF distance of this averaged pixel is calculated w.r.t. the estimated endmember centers obtained through K-means, where the number of centers is the same as the estimated number of endmembers $\hat{R}$ from Sec. 6.2.2. When these RBF distances are normalized to unit summation, they will represent the estimation of abundances. After this step, unmixing is performed according to the user's choice of model (see details in [84]). Note that [84] is limited in the sense that the true number of endmembers is assumed to be known while spatial information is not exploited to address dead pixels and noise. The generalized objective of the autoencoder would be to take input pixel **x** and minimize its mean square error from the reconstructed estimation of the input, $\hat{\mathbf{x}}$ summarized in

$$\min S^{-1}||\mathbf{x} - \hat{\mathbf{x}}||_2^2. \tag{6.15}$$

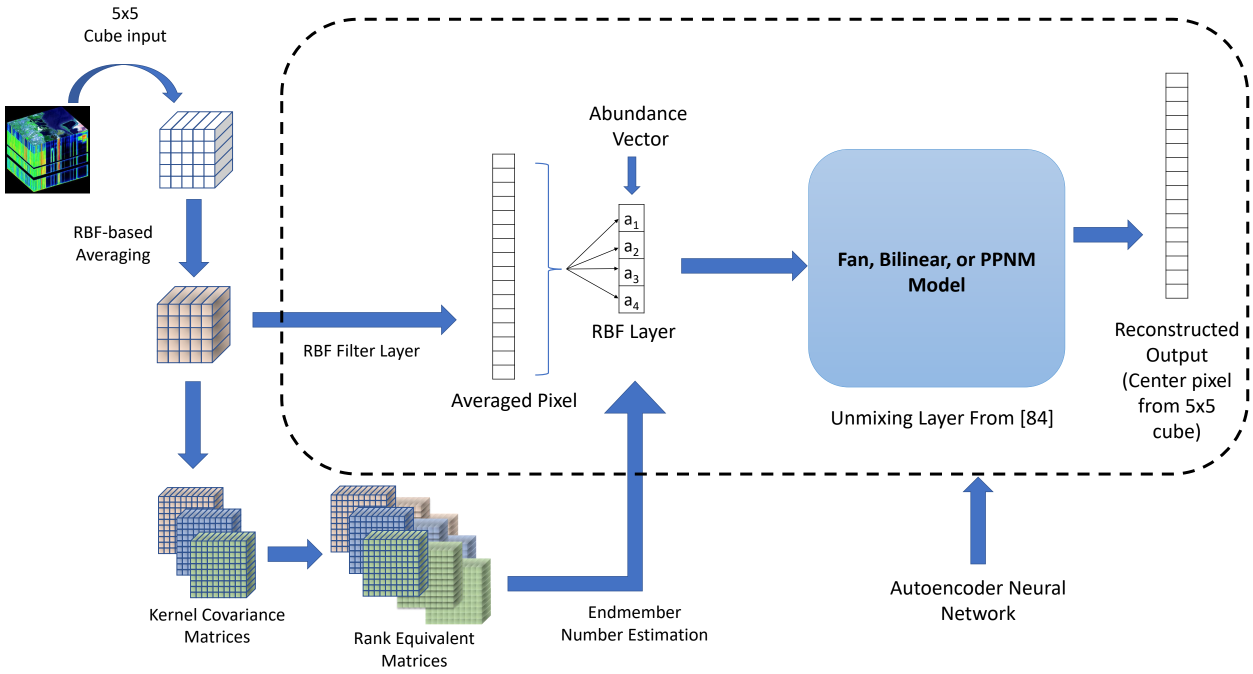


Figure 6.2: Autoencoder setup for the Fan, Bilinear and PPNM mixing models.

The novel neural network structure consists of:

6.2.3.1 RBF Filter Layer

With the number of endmembers estimated, the task remains to feed the pixels into the autoencoder neural network in order to perform unmixing. The first step in the proposed neural network model, is the formation of a weighted average of the input pixel that takes into account its neighboring spatial information. A weighted average is taken of the pixel with its neighborhood, that resembles the RBF-based filter in Sec. 6.2.1, so that the pixel is averaged with higher weights given to adjacent pixels that come from the same material.

It should be noted that the spatial filter weights [similar to the w_i's in Eq. (6.7)] will be learnable weights in the network, and will be initialized according to Eqs. (6.6),(6.7), with the update rules given in Sec. 6.2.4. Thus, the dimensions of these weights will be $N_b \times n \times n$, where N_b represents the number of samples in one batch. The averaged pixel finally becomes the output. Note that the weights would be repeated across the spectral bands.

6.2.3.2 RBF Layer

Once the averaged pixel is calculated, it is then fed into the next layer, where its RBF distance is measured from the $\hat{R}$ K-means centers that are representing the estimated endmembers of the mixed image data. As the layer is described in detail in [84], this will be discussed briefly here. The learnable parameters here would be the β value, and the cluster centers $\mathbf{c}_j$. While the cluster centers are initialized via K-means, the β parameter is initialized with the formula given below. If one pixel is $\mathbf{L}_{i,:}$, and its nearest cluster center is $\mathbf{c}_j$, β is initialized by

$$\beta = (N^{-1} \cdot \textstyle\sum_{i=1}^{N} ||\mathbf{L}_{i,:} - \mathbf{c}_j||_2)^{-2}, \forall j = 1, \cdots, \hat{R}. \tag{6.16}$$

The autoencoder network initializes the weights representing the estimated endmembers $\mathbf{P}$ through K-means. Since K-means measures cluster centers via averaging of the pixels, represented by points in the feature space, dead pixels can cause a major concern. This is because since dead pixels would cause certain pixel entries to be zero, the points along those corresponding dimensions in the feature space will be shifted towards the origin. This shifting causes clustering techniques such as K-means to face considerable difficulty. However, thanks to the spatial averaging filter introduced in Sec. 6.2.1, this concern is alleviated and K-means performs effectively.

Another major advantage is that since RBF distances are nonnegative by nature, the estimated abundances will automatically satisfy its nonnegativity constraints. However, in order to satisfy the unit summation constraint, the output from the RBF layer would need to be normalized to unit summation,

$$[\hat{\mathbf{f}}_x]_i = \frac{[\hat{\mathbf{f}}_x]_i}{\sum_{j=1}^{\hat{R}} [\hat{\mathbf{f}}_x]_j}, \forall i \in 1, 2, \ldots, \hat{R}. \tag{6.17}$$

After this step, if the Fan or Bilinear model is chosen for unmixing, the abundance vector is expanded to include its cross-products. This is a straightforward method that does not necessitate the utilization of any layer.

6.2.3.3 Fan, Bilinear & PPNM Models

In [84], the process of obtaining the vector containing the cross-product terms of the abundances is described in detail. The vector in question can be expanded to accommodate both the Fan and Bilinear models, in which case the cost in Eq. (6.15) becomes

$$\min_{\mathbf{P}, \gamma_{\mathbf{x}}} S^{-1} ||\mathbf{x} - \hat{\mathbf{x}}(\mathbf{P}, \gamma_{\mathbf{x}})||_2^2, \tag{6.18}$$

where $\gamma_{\mathbf{x}}$ refers to the value of γ related to mixed pixel **x**. Once the abundance $[\hat{\mathbf{f}}_{\mathbf{x}}]$ is obtained from Eq. (6.17), $\hat{\mathrm{x}}(\mathbf{P}, \gamma_{\mathbf{x}})$ will be obtained from the mixing model used, e.g., for the Fan model,

$$\hat{\mathrm{x}}(\mathbf{P}, \gamma) = \sum_{j=1}^{\hat{R}} [\hat{\mathbf{f}}_{\mathbf{x}}]_j \cdot \mathbf{P}_{j,:} + \gamma_{\mathbf{x}} \sum_{j=1}^{\hat{R}-1} \sum_{l=j+1}^{\hat{R}} [\hat{\mathbf{f}}_{\mathbf{x}}]_j \cdot [\hat{\mathbf{f}}_{\mathbf{x}}]_l \cdot \mathbf{P}_{j,:} \odot \mathbf{P}_{l,:},$$

The Bilinear model would be the similar, only with both the limits in the summations of the nonlinear components, being between 1 and $\hat{R}$. On the other hand, if the PPNM model is used then $\hat{\mathrm{x}}(\mathbf{P}, b_{\mathbf{x}})$ in the cost in Eq. (6.18) would resemble the form in Eq. (6.4) with $\gamma_{\mathbf{x}}$ being replaced by $b_{\mathbf{x}}$.

6.2.4 Backpropagation

From Eq. (6.18), we can define the error as $e := ||\mathbf{x} - \hat{\mathrm{x}}||_2^2$. In the RBF layer, let the first estimated endmember obtained through K-means clustering be $\mathbf{c}_1$. Further, first estimated abundance g_1 is formulated as

$$g_1 = \exp(-\beta_1 \cdot ||\mathbf{x} - \mathbf{c}_1||_2^2). \tag{6.19}$$

Here, β_1 will be the value of β relevant to the first estimated endmember $\mathbf{c}_1$. Essential in backprpagation will be the derivative of the function g_1 w.r.t. $\mathbf{c}_1$ is

$$\frac{\delta g_1}{\delta \mathbf{c}_1} = -2\beta_1 \exp(-\beta_1 \cdot ||\mathbf{x} - \mathbf{c}_1||_2^2)(\mathbf{x} - \mathbf{c}_1).$$

The derivative of the function g_1 w.r.t. β_1 is

$$\frac{\delta g_1}{\delta \beta_1} = -||\mathbf{x} - \mathbf{c}_1||_2^2 \exp(-\beta_1 \cdot ||\mathbf{x} - \mathbf{c}_1||_2^2).$$

Since **x** is obtained from the RBF filter layer, it is obtained through filter weights w_j, multiplied by the pixels $\mathbf{y}_j$ in the neighborhood $\mathcal{A}_{\mathbf{x}}$, so $\mathbf{x} = \sum_{j=1}^{n^2} w_j \mathbf{y}_j$, $\forall \mathbf{y}_j \in \mathcal{A}_{\mathbf{x}}$. Since in the RBF filter layer, the learnable weights are w_j, the derivative of function **x** w.r.t. w_1 is

$$\frac{\delta \mathbf{x}}{\delta w_j} = \frac{\delta(\sum_i^{n^2} w_i \mathbf{y}_i)}{\delta w_j} = \mathbf{y}_j, \forall j = 1, \cdots, n^2.$$

For an appropriately chosen step size α, the weights during iteration *i+1* are updated implementing gradient descent

$$\mathbf{c}_1^{i+1} = \mathbf{c}_1^i - \alpha\frac{\delta e}{\delta \mathbf{c}_1^i}, \beta_1^{i+1} = \beta_1^i - \alpha\frac{\delta e}{\delta \beta_1^i}, w_1^{i+1} = w_1^i - \alpha\frac{\delta e}{\delta w_1}. \tag{6.20}$$

In the final layer, let us assume the first value in the reconstructed pixel to be $\hat{\mathbf{x}}_1$. Using the Fan model in Eq. (6.3)

$$\hat{\mathbf{x}}_1 = \sum_{i=1}^{\hat{R}}[\hat{\mathbf{f}}_x]_i \cdot \mathbf{P}_{i,1} + \gamma\sum_{i=1}^{\hat{R}-1}\sum_{j=i+1}^{\hat{R}}[\hat{\mathbf{f}}_x]_i \cdot [\hat{\mathbf{f}}_x]_j \cdot \mathbf{P}_{i,1} \odot \mathbf{P}_{j,1}.$$

Note that γ has no subscript here. This is because it represents a scalar value for one mixed pixel. The derivatives of $\hat{\mathbf{x}}_1$ w.r.t. $\mathbf{P}_{1,1}$, and also γ would be

$$\begin{aligned}\frac{\delta \hat{\mathbf{x}}_1}{\delta \mathbf{P}_{1,1}} &= [\hat{\mathbf{f}}_x]_1 + \gamma\sum_{j=2}^{\hat{R}}[\hat{\mathbf{f}}_x]_j \cdot \mathbf{P}_{j,1}, \\ \frac{\delta \hat{\mathbf{x}}_1}{\delta \gamma} &= \sum_{i=1}^{\hat{R}-1}\sum_{j=i+1}^{\hat{R}}[\hat{\mathbf{f}}_x]_i \cdot [\hat{\mathbf{f}}_x]_j \cdot \mathbf{P}_{i,1} \odot \mathbf{P}_{j,1}.\end{aligned} \tag{6.21}$$

Similar to Eq. (6.20), for iteration *i+1*, the weights will be updated through gradient descent as

$$\mathbf{P}_{1,1}^{i+1} = \mathbf{P}_{1,1}^i - \alpha\frac{\delta e}{\delta \mathbf{P}_{1,1}^i}, \gamma^{i+1} = \gamma^i - \alpha\frac{\delta e}{\delta \gamma^i}. \tag{6.22}$$

A similar process can be considered for the remaining weights in this model, and also the bilinear and PPNM models. Also, note that in (6.21), every weight $\mathbf{P}_{i,j}$ will be updated while other ones remain fixed to their most recent update. Originating from [84], the proposed neural network structure also has the unique property of accommodating higher degree cross-products, unlike traditional unmixing algorithms that only allow up to second degree cross-products. As this property is not the main focus of the work formulated here, this property was not utilized, but it nevertheless exists, and is implemented extensively in [84].

6.3 Numerical Tests

The proposed novel unmixing scheme is compared with three other methods (available at the time the paper was finalized) that also utilize spatial information. These methods are: 1) the 3D CNN autoencoder in [81], 2) the 2D CNN autoencoder in [82], and 3) the VCA [11] with FCLS (fully constrained least squares) for abundance calculation, since VCA is used only for endmember extraction.

[81] is an autoencoder network that utilizes 3D convolutional layers (CNN3D) in conjunction with fully connected layers for hyperspectral unmixing. [82] on the other hand, is a 2D CNN method (CNN2D) that estimates both the endmembers and abundances, and will thus be measured in both SAD and RMSE values.

The tests were conducted using both semi-synthetic and real datasets. For the semi-synthetic datasets, two source datasets were used, namely the Pavia Center and Pavia University datasets [51]. The data was created to resemble a field, having 20×20 pixels of crops from each material, adjacent to each other. For each dataset, when considering 4 endmembers, there were thus 4 20×20 crops side by side, creating a field of 20×80 pixels. For every 20×20 crop, one class had a majority abundance of between 80% and 90%, which was randomly chosen. The remaining abundances were randomly split between the other classes. While generating the data, for the Fan and Bilinear models, the values for $\gamma_{\mathbf{x}}$ were set to 1, and for the PPNM model, the values of $b_{\mathbf{x}}$ were according to a uniform distribution within the interval [-0.3,0.3]. Also, during the VCA+FCLS method, for the Fan and Bilinear models specifically, the values for γ_i were presumed to be known *a priori*, whereas for the PPNM model, the values for $b_{\mathbf{x}}$ were assigned by creating a different set of randomly generated values, which had the same uniform distribution, and the same interval [-0.3,0.3]. Because of this, in the PPNM model, the VCA+FCLS method has a distinct advantage over the other methods. In addition, white additive noise was added in the data where signal-to-noise ratio (SNR) values were ranging from 0dB to 20dB. Also, to simulate the presence of dead pixels, a percentage of the data entries were set to zero. These were done for 0 and 20 percent of the data. Accuracy values were calculated as an average of 20 Monte Carlo trials.

The autoencoder used the Adam optimizer [78] implementing the updating recursions in Eqs. (6.20) and (6.22) to learn weights that would yield an output that minimizes the RMSE (root mean square error) between the output and the input. The number of epochs was 100, and the learning rate was 10^{-4}. For the Fan and Bilinear models, the learnable weights for γ were initialized to 1, whereas the learnable weight vector $\mathbf{b}$ in the PPNM model, was initialized with zeros. The weights for the centers in the RBF layer, and the endmember weights in the final layer were initialized to be equal to the K-means cluster centers of the dataset. The tests were conducted on a PC with a Core i5 6400 processor with an Nvidia RTX 2060 GPU in Tensorflow 2.1.0. One interesting point to note is that due to similar initializations, both the RBF centers and the weights in the final layer can be used as endmember estimations. However, as the weights in the final layer interact with the nonlinear cross-products directly, that was more preferable. The Python code of the novel framework is available at [80].

To measure endmember estimation accuracy, the spectral angle distance (SAD) metric was used. For some actual endmember $\mathbf{p}_i$ and its estimation $\hat{\mathbf{p}}_i$, the SAD is

$$SAD = R^{-1} \sum_{i=1}^{R} cos^{-1}\left(\frac{\mathbf{p}_i{}^T \cdot \hat{\mathbf{p}}_i}{||\mathbf{p}_i||_2 \cdot ||\hat{\mathbf{p}}_i||_2}\right). \tag{6.23}$$

It should be noted that for CNN3D, the SAD values are not calculated as they assume knowledge of the original endmembers. For abundance estimation, the metric used was the RMSE found as

$$RMSE = (R \cdot N)^{-1} \sum_{i=1}^{N} (||\mathbf{f}_i - \hat{\mathbf{f}}_i||_2), \tag{6.24}$$

using the actual abundance $\mathbf{f}_i$ and its estimate is $\hat{\mathbf{f}}_i$.

However, this is the process to measure SAD and RMSE if the number of estimated endmembers is correct ($\hat{R} = R$). If this is not the case, then each actual endmember is chosen one at a time, to find the estimated endmember that is the best match. If $\hat{R} > R$, then the R best matches are taken and the rest discarded. If, on the other hand, $\hat{R} < R$, then the matching is done R times, while ensuring that each estimated endmember is chosen at least once.

6.3.1 Semi-Synthetic Data

For the generation of semi-synthetic data, the AVIRIS airbone sensor acquiring hyperspectral images over Pavia University and Pavia Center were used [51]. Unlike the proposed method, all other methods assume prior knowledge of the number of endmembers. In order to give some idea as to the difference in performance due to the proposed method estimating the number of endmembers, the tests will also be re-run under the assumption that the number of endmembers are known. This will be referred to as ”Proposed++”.

The batch size was set to 1600, thus the entire dataset was put into one batch. The reason behind this is twofold. Firstly, this allows for faster operations since only one batch is dealt with instead of multiple batches. Secondly, when calculating gradient descent, all the samples in a batch are considered in calculating gradients via averaging. This process enables calculation of gradient descent directions that are optimal for a larger portion of the dataset at a time, thus making them less likely to fluctuate due to noise.

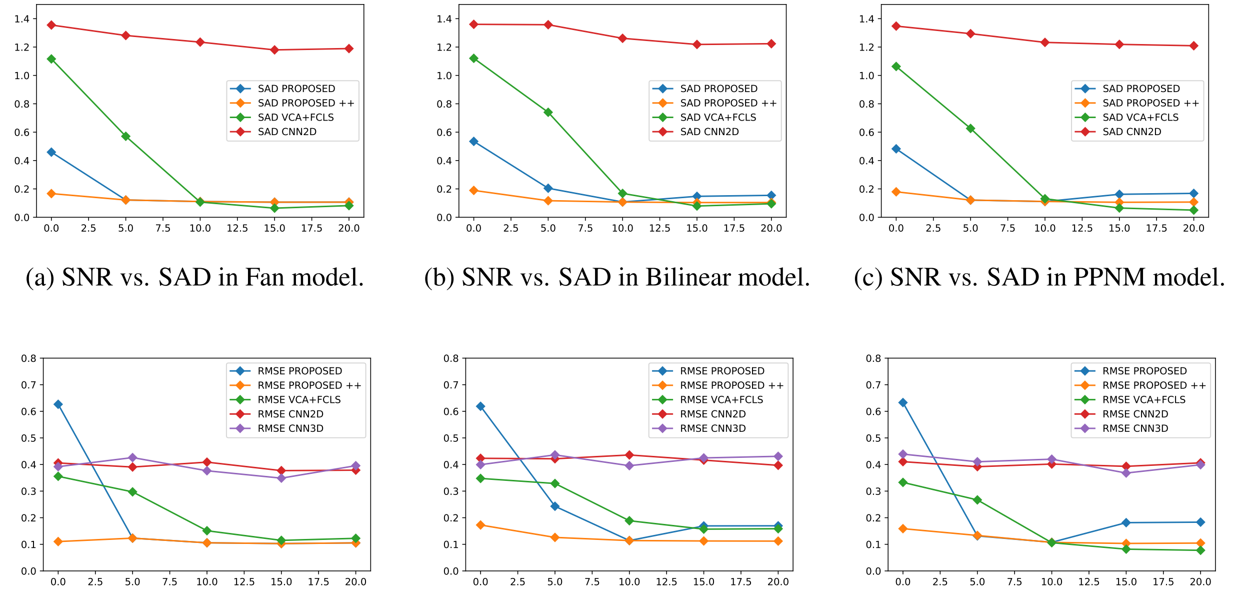
6.3.1.1 Pavia University

The first dataset to observe is the Pavia University dataset. In Fig. 6.3, it can be observed that VCA+FCLS begins to catch up with the proposed method at relatively higher SNR values, which is understandable because the dataset has no dead pixels, allowing the method to perform effectively. CNN2D has significantly worse SAD performances, likely owing to its difficulty in differentiating between the endmembers without initialization with endmember estimation algorithms such as VCA. However, it can also be observed that the proposed method's SAD score is nearly a flat line from 5dB SNR and onwards. The reason behind higher values in 0dB SNR, is due to its difficulty in correct estimation of endmember numbers with such considerable noise. This fact is attested by Proposed++ achieving excellent values even at 0dB SNR. This shows the very high effectiveness of the proposed method's neural network structure in tackling extreme noise. In the RMSE values, CNN3D and CNN2D both face considerable difficulty throughout the whole range of SNR values, due to the challenging nature of the dataset having some similarity among the endmembers.

The superior performance of the proposed method remains consistent with the introduction of dead pixels. As shown in Fig. 6.4 unlike the other methods, the proposed method remains largely unaffected. Again, except 0dB SNR, the performance is highly consistent, even though the other methods are affected with 20% of the pixels being dead, as depicted in Fig. 6.4.

6.3.1.2 Pavia Center

The other semi-synthetic dataset to test is the Pavia Center. This dataset is particularly challenging because of the higher similarities present in the spectral reflectances among the endmembers, and is effective in highlighting the proposed method's superiority in differentiating between them, owing to the better separability in the kernel space and the RBF-based layers' effectiveness in exploiting this property. In Fig. 6.5, even in the presence of no dead pixels, the proposed method is consistently superior to the other methods. CNN2D faces considerable difficulty, while for 0% dead pixels, VCA faces difficulty with 15dB SNR. This challenge is also considerable for CNN3D. Also, in the case of SAD values, it can be seen that in 0dB SNR, the task of the estimation of endmember numbers



(a) SNR vs. SAD in Fan model.

(b) SNR vs. SAD in Bilinear model.

(c) SNR vs. SAD in PPNM model.

(d) SNR vs. RMSE in Fan model.

(e) SNR vs. RMSE in Bilinear model.

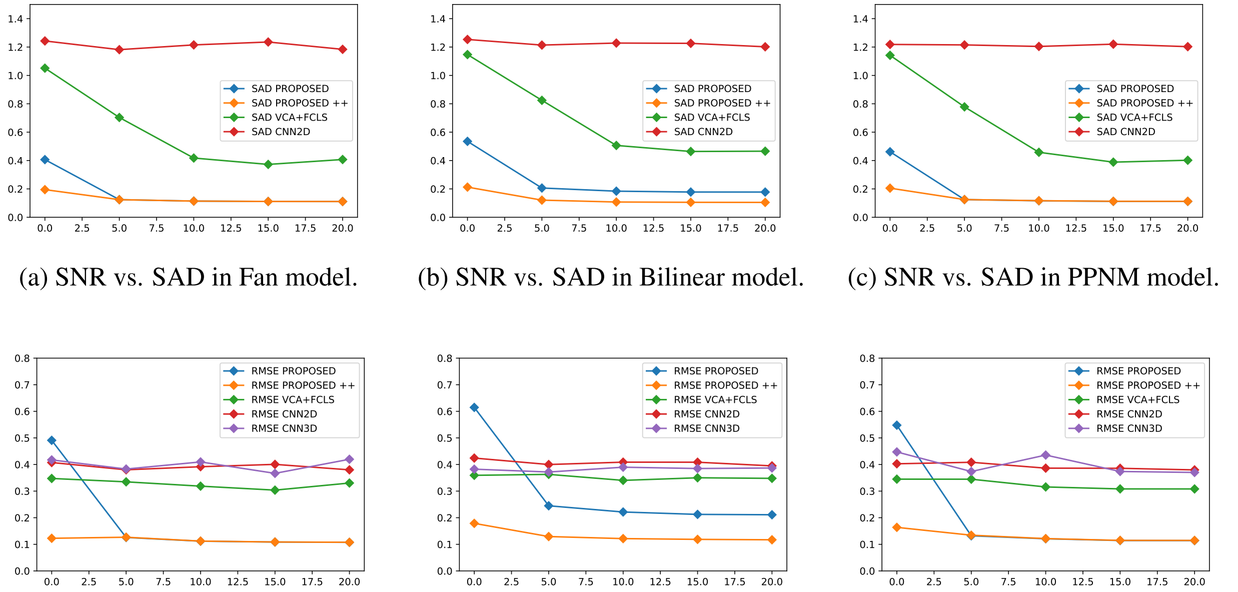
(f) SNR vs. RMSE in PPNM model.

Figure 6.3: SNR (dB) vs. SAD and SNR (dB) vs. RMSE for 0% dead pixels for 4 classes in Pavia University dataset.

pose a challenge for the proposed method. However, as Proposed++ shows, if the proposed method, similarly to the other methods, knows the correct number of endmembers, then the performance in 0dB SNR is nearly the same as any other value of SNR, creating a near-flat line. This further attests to the proposed autoencoder network's high robustness with extreme noise. In the case of added dead pixels, the superiority of the proposed method is even more significant, as shown in Fig. 6.6.

6.3.2 Accuracy in Counting Endmembers

Another accuracy measure to observe was the proposed method's ability in estimating the number of endmembers correctly. Since it has been observed that 0dB SNR creates a significant challenge in finding the correct number of endmembers, Table 6.1 measures the number of correct estimations of endmember numbers for 0dB separately from the rest.



(a) SNR vs. SAD in Fan model.

(b) SNR vs. SAD in Bilinear model.

(c) SNR vs. SAD in PPNM model.

(d) SNR vs. RMSE in Fan model.

(e) SNR vs. RMSE in Bilinear model.

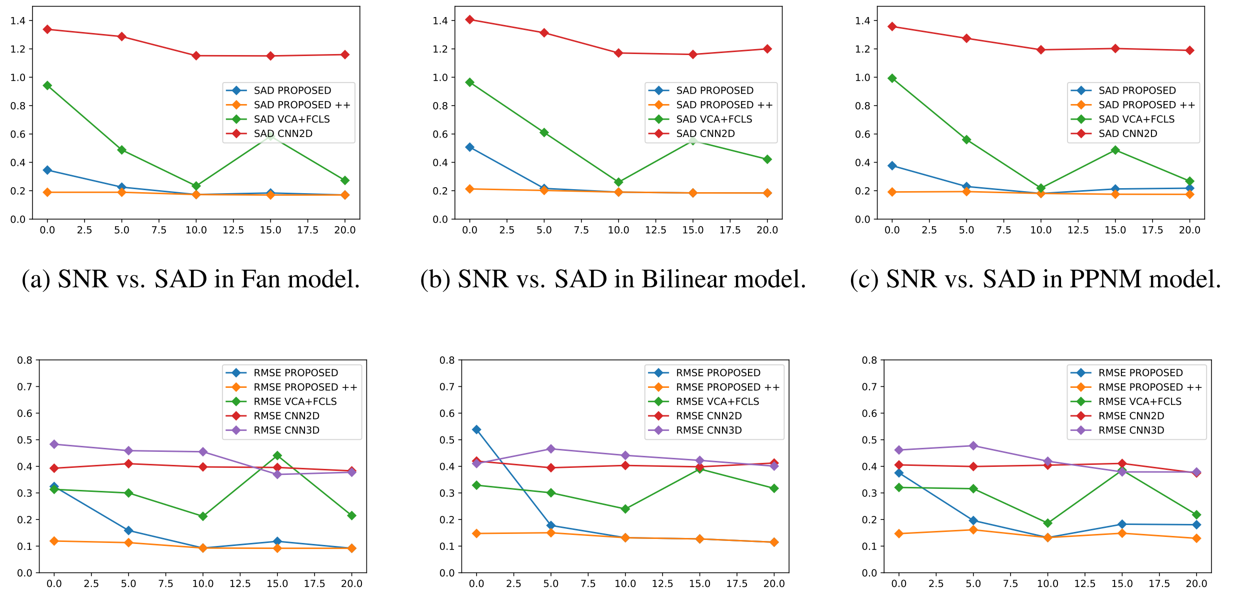
(f) SNR vs. RMSE in PPNM model.

Figure 6.4: SNR (dB) vs. SAD and SNR (dB) vs. RMSE for 20% dead pixels for 4 classes in Pavia University dataset.

As can be observed, while 0dB SNR is a difficult setting to find the correct number of endmembers, the proposed method has very high accuracy from SNR values that are even slightly higher than 0dB.

6.3.3 Comparison of Other Methods With Incorrect Endmember Numbers

In order to illustrate the impact of correctly estimating the number of endmembers, the other compared methods were re-run with an incorrect estimated number of endmembers, since in these methods, the number of endmembers are usually assumed to be known.We used the Pavia University dataset with 20% dead pixels and employing the Fan model. The tests were re-run assuming that there were 3 endmembers, and also assuming that there were 5 endmembers, instead of the correct number of 4. Note that for the CNN3D method, as the actual endmembers were known, re-running this method while as-



(a) SNR vs. SAD in Fan model.

(b) SNR vs. SAD in Bilinear model.

(c) SNR vs. SAD in PPNM model.

(d) SNR vs. RMSE in Fan model.

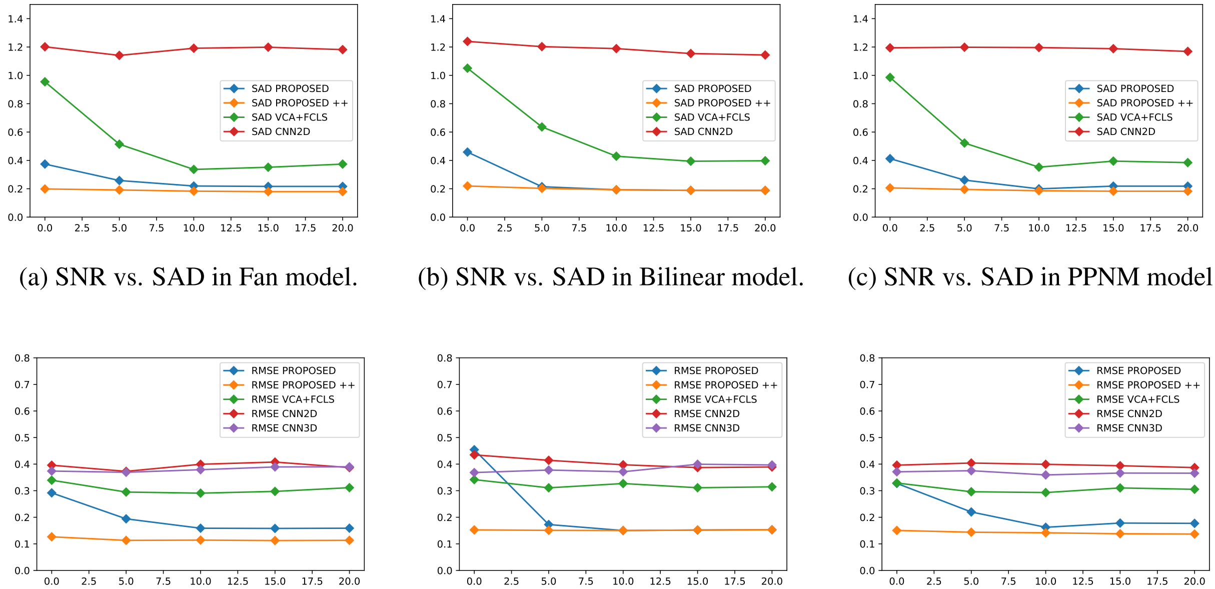
(e) SNR vs. RMSE in Bilinear model.

(f) SNR vs. RMSE in PPNM model.

Figure 6.5: SNR (dB) vs. SAD and SNR (dB) vs. RMSE for 0% dead pixels for 4 classes in Pavia Center dataset.

Dead Pixel Percentage	Dataset	Mixing Model	% Correct (0dB)	% Correct (5 - 20dB)
0%	Pavia University	Fan	5	100
		Bilinear	0	86.25
		PPNM	5	90
	Pavia Center	Fan	50	96.25
		Bilinear	5	98.75
		PPNM	40	91.25
20%	Pavia University	Fan	25	100
		Bilinear	0	80
		PPNM	15	100
	Pavia Center	Fan	55	87.50
		Bilinear	25	98.75
		PPNM	45	88.75

Table 6.1: Accuracy of endmember number estimation across various dead pixel percentages, datasets and mixing models.



(a) SNR vs. SAD in Fan model.

(b) SNR vs. SAD in Bilinear model.

(c) SNR vs. SAD in PPNM model.

(d) SNR vs. RMSE in Fan model.

(e) SNR vs. RMSE in Bilinear model.

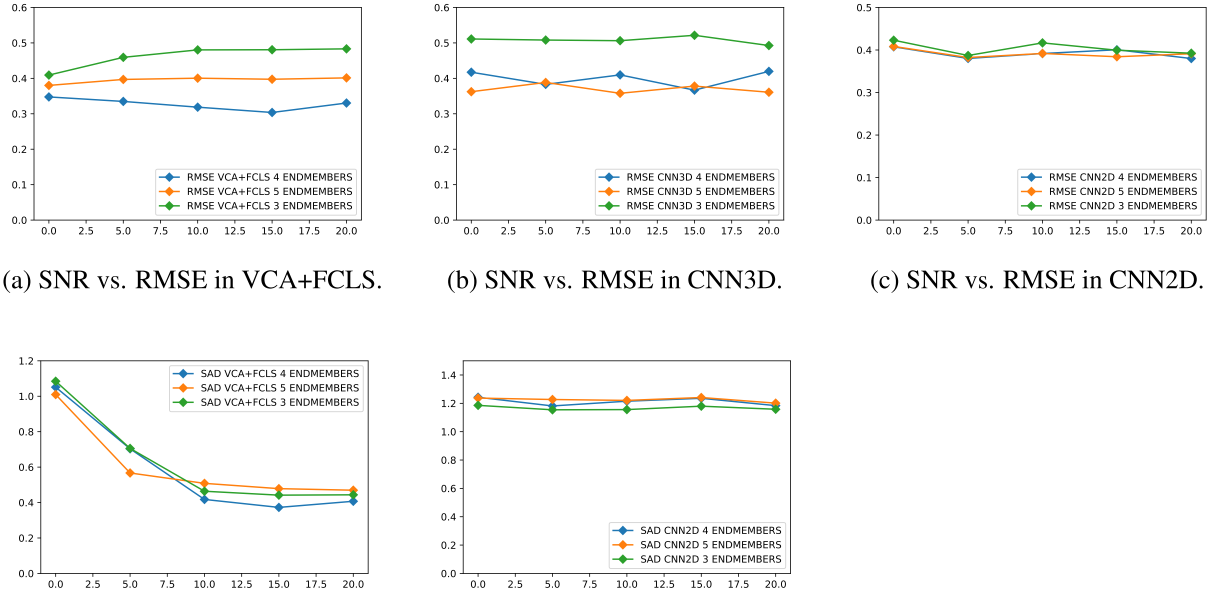
(f) SNR vs. RMSE in PPNM model.

Figure 6.6: SNR (dB) vs. SAD and SNR (dB) vs. RMSE for 20% dead pixels for 4 classes in Pavia Center dataset.

suming 3 or 5 endmembers, was achieved by removing and duplicating the 4th endmember respectively.

As we can observe in Fig. 6.7, VCA+FCLS was strongly impacted when the number of endmembers was 3 or 5. The accuracy measures in the other methods were also affected, but with less of an impact, which is understandable given the accuracy was relatively low even with the correct number of endmembers. This further attests to the proposed method's versatility having the ability to estimate the number of endmembers with high accuracy, especially with datasets as challenging as the ones used.

Another case to observe was the proposed method's accuracy w.r.t. [84], which we shall refer to as NUA (nonlinear unmixing autoencoder). Fig. 6.8 plots the NUA for the same dataset as in Fig 6.7 with 20% dead pixels, with assumed endmember numbers having a difference of ±1 from the correct value of 4. As can be seen in Fig. 6.8, the effect of dead



(a) SNR vs. RMSE in VCA+FCLS.

(b) SNR vs. RMSE in CNN3D.

(c) SNR vs. RMSE in CNN2D.

(d) SNR vs. SAD in VCA+FCLS.

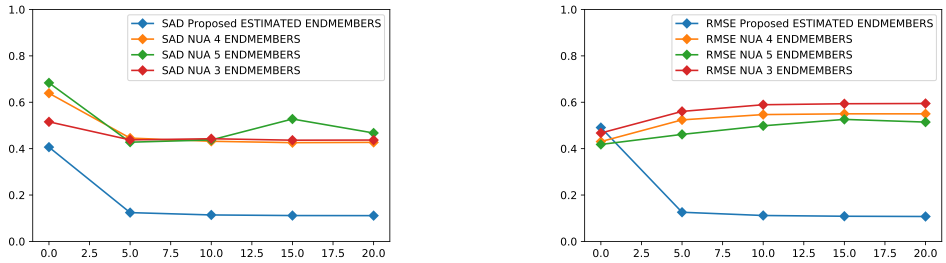
(e) SNR vs. SAD in CNN2D.

Figure 6.7: SNR (dB) vs. SAD and SNR (dB) vs. RMSE for Pavia University, with Fan model and 20% dead pixels for 4 classes, compared to estimating for ±1 number of endmembers.

pixels makes a huge impact on NUA's ability to perform unmixing even when knowing the correct number of endmembers. The plot of the proposed method shows its superiority when dead pixels are present, and when it does not have prior knowledge of the number of endmembers.

6.3.4 Real-World Data

For testing the methods in a real-world scenario, the Samson dataset [77] was used. This is a 95×95 pixel image, with three classes, namely '*Soil*', '*Tree*', and '*Water*'. The proposed method was noticeably effective at the portions of the images with higher abundances for all the materials. The VCA+FCLS method faced noticeable difficulty with the '*Water*' class, as did CNN2D. CNN2D, on the other hand, showed good performances in



(a) SNR vs. SAD in with 20% dead pixels.

(b) SNR vs. RMSE in with 20% dead pixels.

Figure 6.8: SNR (dB) vs. SAD and SNR (dB) vs. RMSE for Pavia University, with Fan model and 20% dead pixels for 4 classes, compared to estimating for ±1 classes in NUA and the proposed method.

the '*Soil*' and '*Tree*' classes, showing its effectiveness when the endmembers have significant differences among each other. In the '*Water*' class, only the proposed method correctly found zero abundance in the bottom right. For VCA+FCLS, the mixing model considered here was the PPNM model. For the proposed method, the Fan model was considered since it showed the best accuracies.

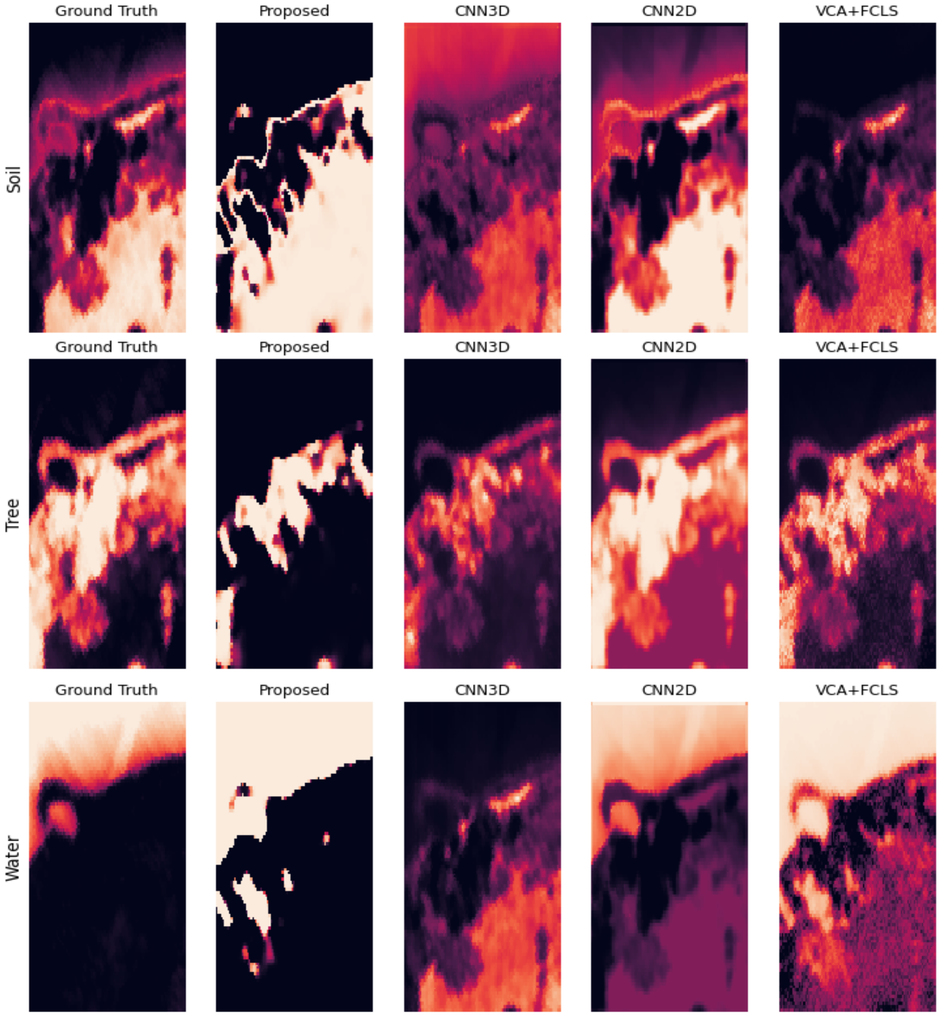


Figure 6.9: Heatmaps of the Samson dataset for various methods.

CHAPTER 7

HEART DISEASE DIAGNOSIS THROUGH MITRAL VALVE TRACKING IN ECHOARDIOGRAMS

The heart is composed primarily of four chambers. They are the left atrium, left ventricle, right atrium and right ventricle. The right atrium and ventricle pump blood only to the lungs, while the left chambers pump blood throughout the rest of the body. Because of this, the left ventricle is larger, and more often looked at during heart checkups. Oxygenated blood enters the heart through the pulmonary veins, into the left atrium. From there, the left ventricle expands, and allows blood from the left atrium to enter it. This is called the *systole* phase. As the ventricle fills up with blood, it squeezes and pumps it through the aorta, and on throughout the body. This is called the *diastole* phase [36]. As the blood flow from the left atrium to the ventricle is a crucial step of the whole process, proper maintenance of blood flow is very important. As this is controlled by the mitral valve, any disease affecting its function needs to be dealt with to ensure the body's health.

Some very common heart diseases affecting the mitral valve include mitral stenosis, where the mitral valve is thickened due to calcification, or deposits of calcium on the mitral valve leaflets, or endocarditis which causes bacteria to grow around the leaflets. These growths restrict blood flow during the systole phase, and force insufficient quantities of blood pumped through the body. Another common affliction is mitral stenosis [37], where the valve is unable to close fully during the diastole phase. This causes some part of the blood being pumped through the aorta, to instead divert back through the left atrium. This is known as *mitral regurgitation* [38]. This will ultimately also cause the body to receive insufficient blood. Other diseases will cause abnormally fast or irregular heartbeats, potentially causing exhaustion and heart failure.

7.1 Otsu's thresholding

Consider an ultrasound video with N frames, where each frame $\mathbf{F}_n$, $\forall n = 1, \ldots, N$ is an $\mathbb{R}^{w \times h}$ array, where w and h represent horizontal pixel length (width) and vertical pixel length (height) respectively. The values in this array represent pixel values, where higher values correspond to whiter pixels. Otsu's thresholding [39] is a method to choose a threshold which would turn the frame into a binary image, where 0's and 1's would correspond to pixel values below and above the threshold respectively. The threshold would be chosen in such a way, that minimizes *intra-class* variance, meaning that the threshold would ensure that pixel values below and above the threshold would have minimal variance among themselves. If we wish to binarize an image, some parts of the image will be part of the foreground, and the rest will be in the background.

Consider the pixel values in a frame $\mathbf{F}_n$ to be p_i, $\forall i = 1, \ldots, w \times h$. For some threshold t, all pixel values below it will be considered part of the background p_{b^j}, $\forall j = 1, \ldots, P_b$ and all others will be part of the foreground p_{f^k}, $\forall k = 1, \ldots, P_f$. P_b and P_f represent the number of pixels in the background and foreground respectively, meaning that $P_b + P_f = w \times h$. Thus, for a given threshold t,

$$\begin{aligned} p_i &= p_{b^j}, \text{ if } p_i < t \\ p_i &= p_{f^k}, \text{ if } p_i \geq t \end{aligned} \tag{7.1}$$

The value of the threshold t, is the one that minimizes the intra-class variance, which is a weighted sum of the variances of the foreground σ_f^2 and background σ_b^2

$$\begin{aligned}
& t = \arg\min_t [w_f(t)\sigma_f^2(t) + w_b(t)\sigma_b^2(t)] \\
\text{where, } & w_b(t) = \frac{P_b}{P_f + P_b} \\
& w_f(t) = \frac{P_f}{P_f + P_b} \\
& \sigma_b^2(t) = \frac{\sum_{j=1}^{P_b}(p_{b^j} - \mu_b)^2}{P_b} \quad \text{where, } \mu_b = \frac{1}{P_b}\sum_{j=1}^{P_b} p_{b^j} \\
& \sigma_f^2(t) = \frac{\sum_{j=1}^{P_f}(p_{f^k} - \mu_f)^2}{P_f} \quad \text{where, } \mu_f = \frac{1}{P_f}\sum_{k=1}^{P_f} p_{f^k}
\end{aligned} \tag{7.2}$$

Note that the value of t affects the values of P_f and P_b via Eq. 7.1. This effectively aids with the background noise present in the frame, while also binarizing the image and make its processing much faster.

After all the N frames are binarized, we obtain N binary arrays where each array has 1's in the pixels where tissues are likely to be present. We thus obtain N binary frames $\mathbf{B}_n$, $\forall n = 1, \ldots, N$. Note that in Eq. 7.1, the frames were vectorized as part of the generalized formula. However, it is not necessary and we did not vectorize the frames in our application as well.

Separately, we apply background subtraction to $\mathbf{F}_n$. Assume the background to be $\mathbf{S}$, where each pixel of $\mathbf{S}$ is the median value across frames 1 through N, so $\mathbf{S} = median(\mathbf{F}_1, \mathbf{F}_2, \cdots, \mathbf{F}_N)$ on an entry-by-entry basis, $\forall n = 1, \ldots, N$. We thus obtain the background subtracted images $\mathbf{F}_{\mathbf{S}^n} = \mathbf{F}_n - \mathbf{S}$. We binarize these images using the same process in Eqs. 7.1 and 7.2 and obtain the binarized versions of $\mathbf{F}_{\mathbf{S}^n}$, denoting them as $\mathbf{B}_{\mathbf{S}^n}$.

After this step, we take the coordinates of the remaining pixels containing 1's and find their centroid, and measure the Euclidean distance of the centroids across different frames. This is done so that we find the two frames that exhibit the maximum range of motion shown by the mitral valve, as the two frames with the farthest centroids will correspond to the frames where the mitral valve leaflets are fully open and closed. We assume the two frames to be $\mathbf{B}_{\mathbf{S}^x}$ and $\mathbf{B}_{\mathbf{S}^y}$. After finding these two frames, they are combined, along with the previously subtracted background. The final combined image is essentially $\mathbf{B}_x + \mathbf{B}_y$. We label this combined image as $\mathbf{C} \in \mathbb{R}^{w \times h}$. We consider the background subtracted version of this to be $\mathbf{C}_\mathbf{S}$. Figure 7.1 shows three images, where (a), (b) and (c) would be $\mathbf{B}_x$, $\mathbf{B}_y$ and $\mathbf{C}$ respectively. Note that $\mathbf{C}$ allows us to observe a closed boundary around the left atrium, while even (a) is unable to do so, since the involuntary movement of the heart makes the interatrial septum to be out of the viewing plane. To reiterate, $\mathbf{B}_x$ is the binarized version of $\mathbf{F}_x$, and $\mathbf{B}_{\mathbf{S}^x}$ is the binarized version of $\mathbf{F}_{\mathbf{S}^x}$. The presence of $\mathbf{S}$ signifies the presence of background subtraction.

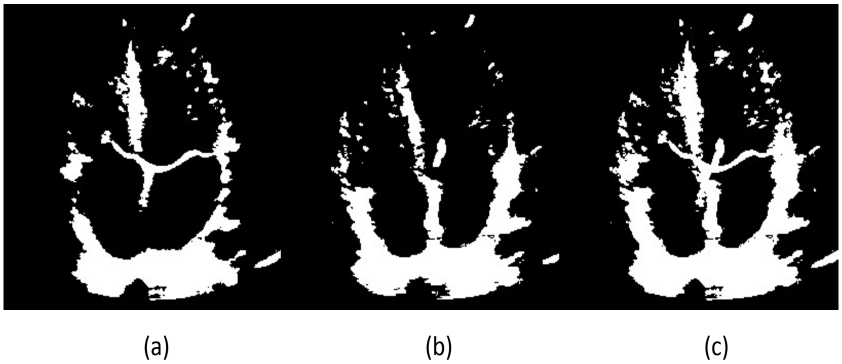


Figure 7.1: Two frames of a heart's ultrasound video after binarizing, (a) and (b) are the two frames whose centroids are farthest from each other, (c) is obtained by combining the two, i.e. $\mathbf{B}_x + \mathbf{B}_y$

7.2 Contour and Atrial Centroid Estimation

Next, we design an unsupervised approach which determines the left atrium's boundary automatically without human intervention. Due to the severe noise expected from echocardiograms, it is possible that the muscles and other tissues making up the boundary of the left atrium might not be clearly defined in every frame. This is particularly problematic with the thinness of the interatrial septum, the wall between the two atria, which can be normally as thin as $\leq$1mm. Because of this, a recursive loop is established to find a closed contour surrounding the left atrium.

We take the combined image $\mathbf{F}_x + \mathbf{F}_y$ and perform Otsu's thresholding on it, with and without background subtraction. This is because in usual cases, the mitral valve is thinner compared to the muscles surrounding the left atrium, necessitating a lower threshold for the background subtracted image, as that contains little else but the mitral valve. Thus, we would obtain two thresholds, for the image with background ($\mathbf{F}_x + \mathbf{F}_y$) and without background ($\mathbf{F}_{\mathbf{S}^x} + \mathbf{F}_{\mathbf{S}^y}$), namely l_1 and l_2 respectively. Then, with the simple rationale that the left atrium will be situated at the bottom right corner in the apical 4 chamber view, we observe how many closed boundaries exist inside the bottom right corner of the image. As no chambers exist besides the left atrium in the image's bottom right corner, any closed boundaries besides it will be formed by small noise artifacts. Thus, if a closed boundary exists that is adequately large, it would be sufficient to consider that as the left atrium. For our experiments, we consider a boundary that contains pixels more than 10% (parameter p) of the box's length and height to be adequately large. This particular value was chosen on a trial-and-error basis. In other words, b_x and b_y are the pixel lengths of the bottom right corner of the image, meaning $b_x = 0.5 \times w$ and $b_y = 0.5 \times h$.

If, however, we are unable to find a closed boundary that is large enough, it is safe to presume that the Otsu's threholds obtained were too large to create the boundary around the atrium. So, we implement a recursive loop, where the values of l_1 and l_2 are incrementally decreased by a small step size c until a closed boundary is obtained. Note that with the lowering of the threshold, more and more noise will be added to the image, but they will generally be small specks, creating closed boundaries far lower than the given 10% minimum. When a value is obtained that creates the desired boundary, the centroid is calculated from it. Algorithm 1 shows the recursive loop algorithm at work. The method used for finding closed boundaries was a combination of the Moore-Tracing algorithm, with Jacob's stopping criterion [40].

Algorithm 2 Recursive Estimation of the Closed Boundary of the Left Atrium.

```
1: Initialize values $l_1$, $l_2$ from binarizing matrices $\mathbf{F}_x+\mathbf{F}_y$ and $\mathbf{F}_{\mathbf{S}^x}+\mathbf{F}_{\mathbf{S}^y}$. Set step size $c$, percentage
   threshold $p$. Set box to be bottom right corner of image, input box length $b_x$ and height $b_y$.
   Coordinate of top left corner of box is $\{c_x,c_y\}$.
2: for $s_1 = 0, 1, \cdots$ do
3:     Set $l_1 = l_1 - c$
4:     for $s_1 = 0, 1, \cdots$ do
5:         Set $l_2 = l_2 - c$
6:         Use updated $l_1$ and $l_2$ to obtain new $\mathbf{C} := \mathbf{B}_x + \mathbf{B}_y$ and $\mathbf{C}_\mathbf{S} := \mathbf{B}_{\mathbf{S}^x} + \mathbf{B}_{\mathbf{S}^y}$.
7:         Calculate number of closed boundaries $N_B$ in $\mathbf{C} + \mathbf{C}_\mathbf{S}$.
8:         for $i = 1, \cdots, N_B$ do
9:             Set maximum and minimum coordinates along both axes to be $x_{max}$, $y_{max}$, $x_{min}$ and
               $y_{min}$ respectively.
10:            If $x_{max} - x_{min} \geq p \times b_x$, $y_{max} - y_{min} \geq p \times b_y$, $x_{min} \geq c_x$ and $y_{max} \leq c_y$ end
               update. Otherwise continue.
11:        end for
12:    end for
13:    If condition in Step 6 is still not met, reinitialize $l_2$.
14: end for
15: Calculate centroid of obtained closed boundary.
```

As we can see, the closed boundary is obtained once four conditions are met. The first two are to ensure that the closed boundary is large enough, the latter two ensure that the boundary is within the box. It should be noted also that, in our experiments, it has been observed that due to involuntary movement of the heart, the atrium has a possibility of moving beyond this box. Thus for added safety, the box is taken to be slightly larger, taking 60% of the pixels on each dimension instead of 50%. Also, we set $p = 0.1$ and $c = 0.001$ in our experiments.

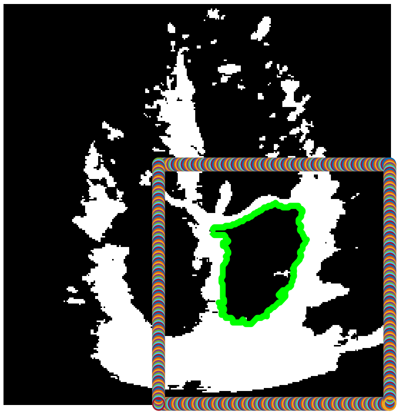


Figure 7.2: Binarized version of the combined frame, after finding a large contour (green) within the box defined in the bottom right corner of the image.

Figure 7.2 depicts the given algorithm finding the correct closed boundary around the left atrium, which is within the automatically generated box in the bottom right corner, where the left atrium always resides.

7.3 Creating Prongs

After finding the centroid of the closed boundary, in the apical 4 chamber view, the mitral valve leaflets will always be above the centroid. Now that the left atrium's location is known, we can safely presume that taking a group of prongs protruding from the centroid in the upward direction will meet the mitral valve leaflets. These prongs should form a cone within the width of the mitral valve. The length of these prongs should be long enough to ensure that even if the shrinking of the left ventricle during the diastolic phase causes the mitral valve to move farther away from the atrium's centroid, it will still be long enough to meet the valves. The advantage to keep this range as short as possible is that, longer prongs need to be created using a higher number of points, affecting processing speeds. To this effect, the length of the prongs, as well as the density of points defining these prongs will depend on the size of the estimated contour of the left atrium. We have defined the length of the prongs as a percentage of the height of the contour around the left atrium, in other words, $y_{max} - y_{min}$ once the algorithm is complete. Thus, the value $l_P = 25$, means it is to be 25$\%$ of $y_{max} - y_{min}$. The cone is set to span 40° above the centroid. Figure 7.3 shows the prongs being generated. We have chosen the number of prongs to be 5.

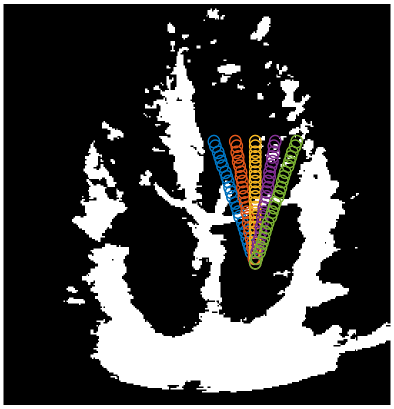


Figure 7.3: 5 prongs centered at the atrium boundary centroid and spanning 40°.

7.4 Estimating Movement of Mitral Valve

The penultimate section of the proposed method is to estimate whether the mitral valve is open or not. If the mitral valve is closed, then the prongs protruding from the centroid of the left atrium will not only touch the mitral valve leaflets, but since the leaflets will be more or less horizontal, the points where the prongs will meet the mitral valve would be close to each other. This means that their coordinates will have little standard deviation σ among each other. Alternatively, if the mitral valve is open, then the prongs would be touching the valve at points more distant from each other, or touching nothing at all. This will cause the points to have a much higher σ among them. Therefore, if a threshold is determined where the value of σ of the points in a particular frame is less than that threshold, we can label the frame's mitral valve to be 'closed', and the frame with a σ value above the threshold to be 'open'. As the threshold can depend on key factors like the size of the the left atrium, the amount of involuntary movement in the heart, the size of the opening of the mitral valve during the systole phase, the angle of the viewing plane due to placement of the probe, this threshold can vary greatly depending on the patient. Thus, we developed a method to determine the threshold in an unsupervised manner.

Depending on the echocardiogram video, no matter what the values of the σ might be, it will always be the case that closed valves will have a much smaller value compared to valves that are open. So if the values of σ across various frames are sorted in ascending order, the threshold is chosen to be the value just before the largest shift in the value of σ.

Figure 7.4 shows the obtained values of σ for the video whose frames were used in Figs. 7.1, 7.2 & 7.3. As we can see, the value of σ just before the largest shift across the frames is 15.04 in the 35^{th} frame, so anything equal to or below this threshold will signify a 'closed' valve, and anything above it will be 'open'.

7.5 Disease Estimation

Now that estimates are done in each frame of whether the mitral valve is open or closed, we use it to estimate whether the heart whose ultrasound video is being observed

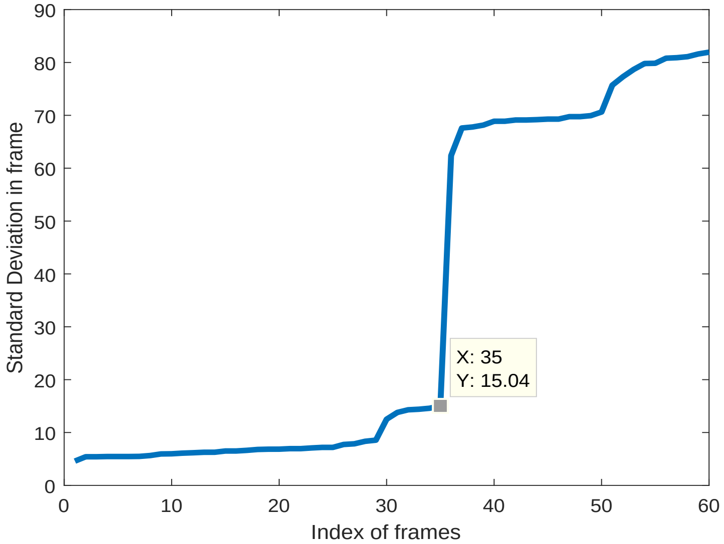


Figure 7.4: Coordinate standard deviation in a configuration involving 5 prongs within a 40 degree cone from the centroid

is healthy or diseased. We would consider a healthy heart to have an adequate number of frames with ‘open’ and ‘closed’ states, as well as an adequate number of consecutive frames having one state at a time. There are three possible scenarios being considered for the heart’s state:

1) Too few consecutive frames switch between ‘open’ and ‘closed’ states multiple times. This would imply diseases like tachycardia, arrhythmia, and any disease causing abnormal heart rates [38].

2) Too many frames in the ‘closed’ state. This would indicate diseases where the mitral valve leaflets have some form of blockage preventing the valve to open properly, such as mitral stenosis [37].

3) Too many frames in the ‘open’ state. This would point to diseases where the mitral valve leaflets lack the ability to close fully during the diastole phase, such as mitral prolapse [41].

For these effects, three thresholds are put in place. The first threshold measures how often states change twice in 3 consecutive frames. Instead of giving a fixed number for this, we set it as 5% of the number of frames of a given video, as the videos used have significantly different numbers of frames. The second and third threshold is set as 0.85, meaning that if more than 85% of the frames are either ‘open’ or ‘closed’, the heart will be classified as diseased. These thresholds are obtained on a trial-and-error basis, as will be explained in the following section.

7.6 Numerical Results

For our numerical tests, 62 samples were obtained from 59 patients. The videos were obtained online from YouTube, as well as some taken from tests conducted under a local cardiology clinic named Cardiology Partners in Mansfield, Texas. The videos had 42 healthy and 20 diseased hearts, with diseases encompassing prolapse, stenosis, endocarditis, fibrillation, tachycardia, hypertrophy, mitral calcification, to name a few. The presence of extreme noise present in ultrasound videos was exacerbated to some extent by the lossy compression technique used in storing YouTube videos, which made accurate estimations even more difficult.

In order to test the efficacy of our proposed method, it was compared with two other methods, [43] and [44]. [44] used graph cuts with a lower rank approximation of a matrix containing the pixel information of the video, to find the faster moving pixels, which would contain the mitral valve as it was presumed that all other parts of the heart would not move as fast. As in the original paper, the frames were cropped to contain only the left chambers, the same was done with our data when applying this method. To apply this method, we utilized the graph cut method in [45–48]. This method will be referred to as LRRGC (Low-Rank Representation Graph Cuts). As the scheme did not classify as to whether the valve was open or not, a manual feature was added, which would observe the span of the area tracked by the algorithm, and see how it is spread out. A valve that is horizontal is likely to be closed, while being vertical would be open.

The method in [43] does not work in an unsupervised manner, instead it relies on one supervised part. The coordinates of the initial point are hand-picked by the user, which is the point where the anterior mitral leaflet connects with the wall of the atrium. The anterior mitral leaflet is the larger of the two leaflets that make up the mitral valve. After that, estimation of the leaflet is unsupervised. This method will be referred to as the VM-Mode.

One thing to note is that the method in [43] worked with the parasternal long axis view, where the mitral leaflet's movements are horizontal. However, our proposed method works with the apical 4 chamber view, where the mitral valve's movements are vertical. To accomodate this, the apical 4 chamber videos were rotated to fit the perspective of the views that [43] is accustomed to. As a form of compensation, in comparing the performances of our proposed method with [43], the comparative method has been given a major advantage. As the initial point has to be selected carefully, we have designed a scheme where the method will not only take the given initial point, but it will consider the points in a box around the selected point. The size of this box will be 10% of the length of the frame across both dimensions, and will have 10 equidistant points on each dimension. In other words, instead of using only one point, the method will use 100 points from a box of size $0.1 * w \times 0.1 * h$ around the given point, and so 100 iterations of the method will be taken. Among these, the iteration with the highest accuracy will be chosen as the 'true' initialization point. Figure 7.5 shows the difference between taking 1 point only, and taking a box of points around the given point.

Table 7.1 shows the comparison among the four methods discussed. The method used to determine accuracy measures described in Sec. 7.5 were used on both methods to obtain performance results. The figures of merit to use were i) the number of frames cor-

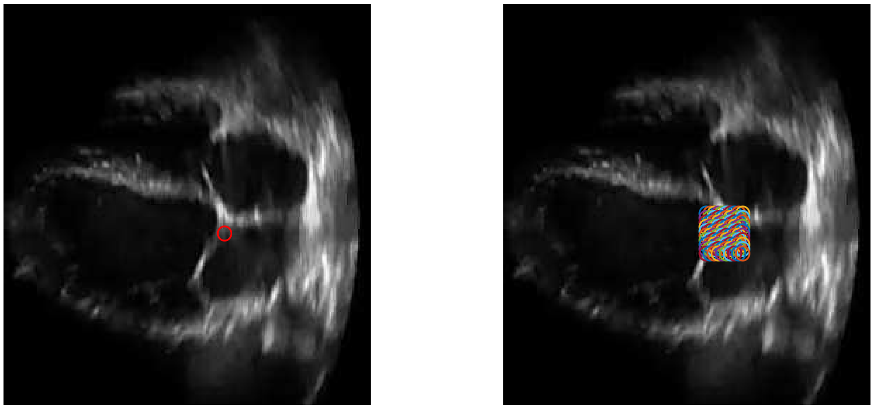


Figure 7.5: An ultrasound image with one initialization point (left) and 100 points (right)

rectly classified as open or closed, ii) the number of videos correctly classified as healthy or diseased, and iii) the time taken to obtain these results. The overall % of correct classification of frames calculates the percentage of correct frames across all the videos, while the mean % calculates the percentage of correct classification of each video separately, then takes the average of those values. The times per sample and frame, were obtained by simply taking the total time to run the algorithm for all 62 samples, and then divide by the number of samples (62), and the total number of frames across all videos respectively. The method with better results for each parameter is highlighted in bold. As we can observe, despite the advantage given to [43], our novel method was still performing better in every measure, despite being fully unsupervised as well. In order to provide an idea of the severity of the advantage given to [43], the performance of the VM-Mode without the advantage is also shown. Clearly, the improvement with the advantage is minimal. Furthermore, the number of classified diseased hearts did not improve at all.

On the other hand, for LRRGC, the issue was that due to a number of the hearts having significant involuntary movement, parts like the interatrial septum were also moving too fast to be considered as part of the background. This led to the scheme erroneously segmenting these parts as belonging to the mitral valve. Because of this, even if the valve was closed, it would be observed as open. As this caused a significant impact in the misclassification of the frames, it also caused most of the frames to be estimated as being in the 'closed' state, causing a lower accuracy in frames, and it also meant that many healthy hearts were misclassified as diseased. Consequently, it also caused a higher rate of videos that were actually diseased, to be coincidentally correctly classified. However, due to the complexity of the scheme, as well as dealing with extremely large arrays containing pixel information of an entire video, this came at great computation cost, causing much higher runtimes.

We have also added the precision, recall and f1 square values of the methods. Precision is the ratio of true positives (correctly classifying healthy hearts) and the sum of true positives with false positives (misclassifying diseased hearts as healthy). On the other hand, recall is the ratio of true positives, and the sum of true positives with false negatives(misclassifying healthy hearts as diseased). The f1 score is basically $2\frac{precision \times recall}{(precision+recall)}$. We can see that the proposed method outperforms all the other methods, except for in Recall, which is understandable as the VM-Mode was unable to classify any heart as diseased, giving the number of false negatives to be zero, and thus making the ratio equal to 1.

Test Parameters	Proposed Method	VM - Mode (with advantage)	VM - Mode (without advantage)	LRRGC
% of Correct Classification (All)	**83.87**	64.71	63.27	32.26
% of Correct Classification (Diseased)	85.00	0	0	**95**
Mean Time(s) per Sample	**2.11**	2.63	2.81	39.58
Mean Time(s) per Frame	**0.03**	0.04	0.04	0.61
Mean % of Correct Frame Classification	**62.08**	32.69	30.17	47.91
Overall % of Correct Frame Classification	**59.90**	33.62	31.22	46.89
Precision	**0.92**	0.65	0.64	0.50
Recall	0.83	**1.00**	**1.00**	0.02
F1 Score	**0.88**	0.79	0.78	0.05

Table 7.1: Performance Comparison between Proposed method, VM-Mode [43] and LR-RGC [44].

CHAPTER 8

CONCLUSION

In this work, effective kernel mapping maximized the information extracted from hyperspectral pixels to achieve highly accurate clustering performances in an unsupervised setting, even being robust under the crippling handicap of having dead pixels which severely impacted competitive methods. For mixed pixels, the unmixing performance was achieved with high accuracy that outperforms existing schemes, and also tackles significant challenges such as estimating the number of endmembers, dealing with extreme noise and unresponsive pixels. The utilization of spatial information is achieved effectively by implementing weighting averaging that measures the RBF distance, in other words the similarity in the data in the kernel space. This is effective due to exploiting the property of materials present in these hyperspectral images exhibiting high intraclass correlations, and low interclass correlations in the RBF kernel space. The use of the kernel covariance matrix in the datasets to measure reconstruction error in lower rank equivalent matrices helped to estimate the number of endmembers with high accuracy. The ability to choose between three major unmixing models, and the unique ability to incorporate higher degree cross-products in estimating the nonlinear interactions among the endmembers with the ease of only two hyperparameters aided greatly in the versatility of the novel autoencoder structure. The relationship among the data in the images being far more informative in the RBF space was further advantageous in estimating the endmember vectors and abundance values, which was evident in the numerical results when the proposed methods significantly outperformed comparative methods in very low SNR values.

The echocardiography-based heart disease estimation was highly unique, in that conventional methods went so far as to observe the movement of the mitral valve, but to extract features in order to provide a decision on the health of the heart was rarely done. The data collected was also unique in that conventional methods would gather data with the specific intent of implementation of their algorithm, which greatly lowers difficulty since it eliminates factors such as the subjectivity in the technician's angle of application of the ultrasound probe during data acquisition, and the particular conditions of the patients themselves. Our approach in gathering data from a very wide variety of sources ensured that the proposed novel method will exhibit accurate diagnosis performance in any adverse setting.

8.1 Future Directions

There are some aspects that have scope for improvement and further development. The approach for estimation of the number of endmembers in Chapter 6 is effective because it observes a range of values for the β_c variable in order to find the most informative kernel covariance matrix for that purpose, but there is a tradeoff here. If the range of values is sufficiently wide, it can perform with far better accuracy, with far more robustness w.r.t. to the diversity of the datasets being applied. However, a wider range can significantly impact the computational complexity in that the calculation of the kernel covariance matrix is an intensive process. Finding an approach to achieve a wider range without increasing the computational time would be one of the problems to tackle in future.

REFERENCES

[1] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, (2015). 220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3. Purdue University Research Repository.

[2] G. Camps-Valls, T. V. B. Marsheva, and D. Zhou, "Semi-supervised Graph-based Hyperspectral Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 10, pp. 3044–3054, 2007.

[3] A. Plaza, J. A. Benediktsson, J. W. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, A. Gualtieri, M. Marconcini, J. C. Tilton, and G. Trianni,"Recent Advances in Techniques for Hyperspectral Image Processing," *Remote Sensing of Environment*, vol. 113,pp. S110-S122, 2009.

[4] Y. Tarabalka, J. Benediktsson, and J. Chanussot, "Spectral-spatial Classification of Hyperspectral Imagery Based on Partitional Clustering Techniques," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 8,pp. 2973–2987, 2009.

[5] A. Martinez-Uso, F. Pla, J. Martinez Sotoza, P. Garcia-Sevilla, "Clustering-based Hyperspectral Band Selection using Information Measures," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 12, pp. 4158-4171, 2007.

[6] G. Camps-Valls and L. Bruzzone, "Kernel-based Methods for Hyperspectral Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 6, pp. 1351–1362, 2005.

[7] J. Li, P. R. Marpu, A. Plaza, J. M. Bioucas-Dias and J. A. Benediktsson, "Generalized Composite Kernel Framework for Hyperspectral Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 51, no. 9, pp. 4816–4829, Sep. 2013.

[8] G. C-Valls and L. Bruzzone, “Kernel-Based Methods for Hyperspectral Image Classification,” *IEEE Tans. on Geoscience and Remote Sensing*, vol. 43, no. 6, pp. 1351–1362, June 2005.

[9] C. McCann, K. S. Repasky, M. Morin, R. L. Lawrence, and S. Powell,“Novel Histogram Based Unsupervised Classification Technique to Determine Natural Classes From Biophysically Relevant Fit Parameters to Hyperspectral Data,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, IEEE Early Access Article, 2017.

[10] C. Cariou and K. Chehdi, “Unsupervised Nearest Neighbors Clustering with Application to Hyperspectral Images,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 6, pp. 1105–1116, 2015.

[11] J. M. P. Nascimento and J. M. B. Dias,“Vertex component analysis: a fast algorithm to unmix hyperspectral data,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 4, pp. 898–910, April 2005.

[12] J. Sigurdsson, M. O. Ulfarsson and J. R. Sveinsson, “Endmember constrained semi-supervised hyperspectral unmixing,” *6th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, Lausanne, pp. 1–4, 2014.

[13] M. D. Iordache, A. Plaza and J. Bioucas-Dias, “On the use of spectral libraries to perform sparse unmixing of hyperspectral data,” *2nd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing*, Reykjavik, pp. 1–4, 2010.

[14] R. Zhu, M. Dong, and J.-H. Xue, “Spectral nonlocal restoration of hyperspectral images with low-rank property,” *IEEE Journal of Sel. Topics in Applied Earth Observations and Remote Sensing,* vol. 8, no. 6, pp. 3062–3067, June 2015.

[15] J. Chen and I. D. Schizas, ”Online Distributed Sparsity-Aware Canonical Correlation Analysis,” *IEEE Transactions on Signal Processing*, vol. 64, no. 3, pp. 688–703, Feb. 2016.

[16] D. R. Brillinger, *Time Series: Data Analysis and Theory*. Expanded Edition, Holden Day, 1981.

[17] B. Schölkopf and A. J. Smola, *Learning with Kernels.* The MIT Press, Cambridge, Massachusetts, 2000.

[18] D. Bertsekas, *Nonlinear programming*. Belmont: Athena scientific, 1999.

[19] N. Keshava and J. F. Mustard, ”Spectral unmixing,” *IEEE Signal Processing Magazine*, vol. 19, no. 1, pp. 44-57, Jan 2002.

[20] T. W. Ray and B. C. Murray, “Nonlinear spectral mixing in desert vegetation,” *Remote Sens. Environ.*, vol. 55, no. 1, pp. 59–64, Jan. 1996.

[21] W. Fan, B. Hu, J. Miller, and M. Li, “Comparative study between a new nonlinear model and common linear model for analysing laboratory simulated-forest hyperspectral data,” *Remote Sens. Environ.*, vol. 30, no. 11, pp. 2951–2962, Jun. 2009.

[22] A. Malhotra, K. T. Shahid and I. D. Schizas, ”Unsupervised Kernel Learning for Correlation Based Clustering,” *Asilomar Conference on Signals, Systems and Computers (ACSSC 2018)*, Pacific Grove, CA, 2018, pp. 2007-2011.

[23] D. P. Bertsekas, ”Penalty and augmented lagrangian methods” in Nonlinear programming, 2nd ed. Belmont: Athena scientific, U.S.A., 1999, pp 397-416.

[24] P. Tseng, “Convergence of a block coordinate descent method for nondifferentiable minimization,” *J. Opt. Theory Appl.,* vol. 109, no. 3, pp.

[25] A. Malhotra, K. T. Shahid, I. D. Schizas and S. Tjuatja, ”Fault tolerant unsupervised kernel-based information clustering in hyperspectral images,” 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, 2017, pp. 2191-2194.

[26] C. Zhao, G. Zhao and X. Jia, ”Hyperspectral Image Unmixing Based on Fast Kernel Archetypal Analysis,” *in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 1, pp. 331-346, Jan. 2017.

[27] M. Mørup and L. K. Hansen (2012). “Archetypal analysis for machine learning and data mining,“ *Neurocomputing*, vol. 80, no. 15, pp. 54–63, Mar. 2012.

[28] S. Khazai, A. Safari, B. Mojaradi, and S. Homayouni, “Improving the SVDD approach to hyperspectral image classification,” IEEE Trans. Geosci. Remote Sens., vol. 9, pp. 594–598, Jul. 2012

[29] Y. Altmann, A. Halimi, N. Dobigeon and J. Y. Tourneret, ”Supervised nonlinear spectral unmixing using a postnonlinear mixing model for hyperspectral imagery,” *IEEE Transactions on Image Processing,* vol. 21, no. 6, pp. 3017-3025, June 2012.

[30] R. Heylen, P. Scheunders, A. Rangarajan and P. Gader, ”Nonlinear unmixing by using different metrics in a linear unmixing chain,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* vol. 8, no. 6, pp. 2655-2664, June 2015.

[31] M. Pharr,W. Jakob, and G. Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[32] L. Tits, B. Somers, J. Stuckens and P. Coppin, ”Validating nonlinear mixing models: Benchmark datasets from vegetated areas,” in *2014 6th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, Lausanne, 2014, pp. 1-4.

[33] Cuprite images: [Online] Available: http://lesun.weebly.com/hyperspectral-data-set.html

[34] VCA Matlab® Script: [Online] Available: http://www.lx.it.pt/ ~bioucas /code.htm

[35] W. Wang and H. Qi, “Unsupervised nonlinear unmixing of hyperspectral images using sparsity constrained probabilistic latent semantic analysis,” *2013 Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, Gainesville, FL, pp. 1–4, 2013.

[36] E.P. Widmaier, H. Raff, and K.T. Strang,”Cardiovascular Physiology”, in *Vander’s Human Physiology: The Mechanisms of Body Function*, 13th edition, McGraw Hill Education, 2014, ch. 12, pp. 378.

[37] B. A. Carabello, ”Modern Management of Mitral Stenosis.”, in *Circulation*, vol. 112, issue 3, pp. 432-437, 2005.

[38] E. D. Agabegi, and S. S. Agabegi, ”Chap. 1: Diseases of the Cardiovascular System/Section: Valvular Heart Disease”, in *Step-up to medicine, Step-up series*, Philadelphia: Lippincott Williams & Wilkins, 2008, ch. 1.

[39] N. Otsu, ”A Threshold Selection Method from Gray-Level Histograms,” in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62-66, Jan. 1979.

[40] A. Ghuneim, ”Contour Tracing”, 2015, [Online] Available: http://www.imageprocessingplace.com/downloads_V3/root_downloads/ tutorials/contour_tracing_Abeer_George_Ghuneim/moore.html

[41] E. Hayek, C. N. Gring, and B. P. Griffin, ”Mitral Valve Prolapse.”, in *The Lancet*, vol. 365, issue 9458, pp. 507-518, 2005.

[42] D. Chu, L. Liao, M. Ng, and X. Zhang, “Sparse kernel canonical correlation analysis,” *Proc. of Intl. Multiconference of Engineers and Computer Scientists*, Hong Kong, 2013.

[43] M. S. Sultan, N. Martins, E. Costa, D. Veiga, M. J. Ferreira, S. Mattos and M. T. Coimbra, ”Virtual M-Mode for Echocardiography: A New Approach for the Segmentation of the Anterior Mitral Leaflet,” in *IEEE Journal of Biomedical and Health Informatics*, vol. 23, no. 1, pp. 305-313, Jan. 2019.

[44] X. Zhou, C. Yang and W. Yu, ”Automatic Mitral Leaflet Tracking in Echocardiography by Outlier Detection in the Low-Rank Representation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, 2012, pp. 972–979.

[45] Y. Boykov, O. Veksler and R. Zabih, ”Efficient Approximate Energy Minimization via Graph Cuts”, in *IEEE transactions on PAMI*, vol. 20, no. 12, p. 1222-1239, November 2001.

[46] V. Kolmogorov and R. Zabih, ”What Energy Functions can be Minimized via Graph Cuts?”, in *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 26, no. 2, pp. 147-159, February 2004.

[47] Y. Boykov and V. Kolmogorov, ”An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision”, in *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 26, no. 9, pp. 1124-1137, September 2004.

[48] S. Bagon, ”Matlab Wrapper for Graph Cut”, December 2006, [Online] Available: https://github.com/shaibagon/GCMex.

[49] L. Yu, Y. Guo, Y. Wang, J. Yu and P. Chen, ”Segmentation of Fetal Left Ventricle in Echocardiographic Sequences Based on Dynamic Convolutional Neural Networks,” in *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 8, pp. 1886-1895, Aug. 2017.

[50] D. Bibicu and L. Moraru, ”Cardiac Cycle Phase Estimation in 2-D Echocardiographic Images Using an Artificial Neural Network,” in IEEE Transactions on Biomedical Engineering, vol. 60, no. 5, pp. 1273-1279, May 2013.

[51] Hyperspectral Images. Accessed: Jun. 5, 2020. [Online]. Available: http://www.ehu.eus/ccwintco/index.php?title= Hyperspectral_Remote_Sensing_Scenes

[52] K. T. Shahid, A. Malhotra, I. D. Schizas, and S. Tjuatja, “Unsupervised kernel correlations based hyperspectral clustering with missing pixels”, *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 6, pp. 1799–1810, Jun. 2018

[53] M. C. Torres-Madronero and M. Velez-Reyes, “Integrating spatial information in unsupervised unmixing of hyperspectral imagery using multiscale representation”, *IEEE*

J. Sel. Topics Appl. Earth Observ. Remote Sens., vol. 7, no. 6, pp. 1985–1993, Jun. 2014.

[54] L. Miao and H. Qi, “Endmember extraction from highly mixed data using minimum volume constrained nonnegative matrix factorization”, *IEEE Trans. Geosci. Remote Sens.*, vol. 45, no. 3, pp. 765–777, Mar. 2007.

[55] K. T. Shahid and I. D. Schizas, “Unsupervised hyperspectral unmixing via kernelized correlations”, in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Valencia, Spain, Jul. 2018, pp. 6388–6391.

[56] G. A. Licciardi and F. Del Frate, ”Pixel Unmixing in Hyperspectral Data by Means of Neural Networks”, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 11, pp. 4163-4172, 2011.

[57] X. Feng, H. Li, J. Li, Q. Du, A. Plaza and W. J. Emery, ”Hyperspectral Unmixing Using Sparsity-Constrained Deep Nonnegative Matrix Factorization With Total Variation”, *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 10, pp. 6245-6257, 2018.

[58] S. Ozkan, B. Kaya and G. B. Akar, ”EndNet: Sparse AutoEncoder Network for Endmember Extraction and Hyperspectral Unmixing,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 1, pp. 482-496, 2019.

[59] B. Hapke, “Bidirectional reflectance spectroscopy. 1. Theory”, *J. Geophys. Res.*, vol. 86, pp. 3039–3054, 1981.

[60] N. Dobigeon, L. Tits, B. Somers, Y. Altmann, and P. Coppin, “A comparison of nonlinear mixing models for vegetated areas using simulated and real hyperspectral data”, *IEEE J. Sel. Topics Appl. Earth Observat. Remote Sens.*, vol. 7, no. 6, pp. 1869–1878, Jun. 2014.

[61] K. T. Shahid, and I. D. Schizas, ”Unsupervised Kernelized Correlation-Based Hyperspectral Unmixing With Missing Pixels”, *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 7, pp. 4509-4520, Feb. 2019.

[62] B. Palsson, J. Sigurdsson, J.R. Sveinsson and M.O. Ulfarsson, ”Hyperspectral unmixing using a neural network autoencoder”, *IEEE Access*, vol. 6, pp.25646-25656, 2018.

[63] Y. Su, A. Marinoni, J. Li, J. Plaza, and P. Gamba, ”Stacked nonnegative sparse autoencoders for robust hyperspectral unmixing”, *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 9, pp. 1427-1431, Jun. 2018.

[64] R. A. Borsoi, T. Imbiriba, and J. C. M. Bermudez, ”Deep generative endmember modeling: An application to unsupervised spectral unmixing”, *IEEE Trans. Comput. Imag.*, vol. 6, pp. 374-384, Oct. 2019.

[65] F. Khajehrayeni and H. Ghassemian, ”Hyperspectral unmixing using deep convolutional autoencoders in a supervised scenario”, *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 567-576, Feb. 2020.

[66] Y. Qian, F. Xiong, Q. Qian, and J. Zhou, ”Spectral Mixture Model Inspired Network Architectures for Hyperspectral Unmixing”, *IEEE Trans. Geosci. Remote Sens.*, pp 1-17, Apr. 2020

[67] A. Marinoni, J. Plaza, A. Plaza and P. Gamba, “Integrating multiple nonlinear estimators into hyperspectral unmixing”, in *Proc. 6th Workshop Hyperspectral Image Signal Process., Evol. Remote Sens. (WHISPERS)*, Lausanne, Switzerland, Jun. 2014.

[68] B.-C. Kuo, H.-H. Ho, C.-H. Li, C.-C. Hung, and J.-S. Taur, “A kernel-based feature selection method for SVM with RBF kernel for hyperspectral image classification”, *IEEE J. Sel. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 1, pp. 317–326, Jan. 2014.

[69] S. Yang, H. Jin, M. Wang, Y. Ren, and L. Jiao, “Data-driven compressive sampling and learning sparse coding for hyperspectral image classification”, *IEEE Geo. Rem. Sens. Lett.*, vol. 11, no. 2, pp. 479–483, 2014

[70] Y. Altmann, N. Dobigeon, S. McLaughlin, and J.-Y. Tourneret, “Nonlinear unmixing of hyperspectral images using radial basis functions and orthogonal least squares”, in *Proc. IEEE Int. Conf. Geosci. Remote Sens. (IGARSS)*, Vancouver, Canada, Jul. 2011, pp. 1151–1154.

[71] M. Wang, M. Zhao, J. Chen, and S. Rahardja, ”Nonlinear unmixing of hyperspectral data via deep autoencoder networks”, *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 9, pp. 1467-1471, Sep. 2019.

[72] R. Heylen and P. Scheunders, “A multilinear mixing model for nonlinear spectral unmixing”, *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 1, pp. 240–251, Jan. 2016.

[73] X. Chen and L. Vierling, “Spectral mixture analysis of hyperspectral data acquired using a tethered balloon”, *Remote Sens. Environ.*, vol. 103, pp. 338–350, 2006.

[74] N. Raksuntorn and Q. Du, “Nonlinear spectral mixture analysis for hyperspectral imagery in an unknown environment”, *IEEE Geosci. Remote Sens. Lett.*, vol. 7, no. 99, pp. 836–840, 2010.

[75] Y. Altmann, N. Dobigeon, and J.-Y. Tourneret, “Unsupervised postnonlinear unmixing of hyperspectral images using a Hamiltonian Monte Carlo algorithm”, *IEEE Trans. Image Process.*, vol. 23, no. 6, pp. 2663–2675, Jun. 2014.

[76] J. MacQueen, ”Some Methods for classification and Analysis of Multivariate Observations”, *Proc. 5th Berkeley Symp. Math. Statist. Probability*, Vol. 1, No. 14, pp. 281-297, 1967.

[77] Samson and Jasper Ridge Dataset. Accessed: Jul. 5, 2020. [Online]. Available: https://rslab.ut.ac.ir/data

[78] D. P. Kingma and J. Ba. (Dec. 2014). “Adam: A method for stochastic optimization.” [Online]. Available: https://arxiv.org/abs/1412.6980

[79] Nonlinear Hyperspectral Unmixing Autoencoder. [Online]. Available: https://github.com/KaziTShahid/Nonlinear-Hyperspectral-Unmixing-Autoencoder/blob/master/autoencoder_main.py

[80] Nonlinear Hyperspectral Unmixing Spatial Filters Autoencoder. [Online]. Available: https://github.com/KaziTShahid/Nonlinear-Hyperspectral-Unmixing-Spatial-Filters-Autoencoder/blob/master/autoencoder_main.py

[81] F. Khajehrayeni and H. Ghassemian, “Hyperspectral unmixing using deep convolutional autoencoders in a supervised scenario,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 567–576, Feb. 2020

[82] B. Palsson, M. O. Ulfarsson, J. R. Sveinsson, ”Convolutional Autoencoder for Spectral–Spatial Hyperspectral Unmixing”, *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 1, pp. 535-549, May 2020.

[83] A. Malhotra and I.D. Schizas. ”On unsupervised simultaneous kernel learning and data clustering”, *Pattern Recognition*, vol. 108, no. 107518, Dec. 2020

[84] K. T. Shahid and I. D. Schizas, “Unsupervised Hyperspectral Unmixing Via Nonlinear Autoencoders”, *IEEE Trans. Geosci. Remote Sens.*, (Accepted)

[85] R. Heylen, M. Parente, and P. Scheunders, “Estimation of the number of endmembers in a hyperspectral image via the hubness phenomenon,” *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 4, pp. 2191–2200, Apr. 2017.

[86] X. Tao, T. Cui, A. Plaza, and P. Ren, “Simultaneously counting and extracting endmembers in a hyperspectral image based on divergent subsets,” *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 12, pp. 8952–8966, Dec. 2020

[87] X. Zhang, Y. Sun, J. Zhang, P. Wu, and L. Jiao, “Hyperspectral unmixing via deep convolutional neural networks,” *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 11, pp. 1755–1759, Nov. 2018

[88] K. T. Shahid and I. D. Schizas, “Unsupervised Mitral Valve Tracking for Disease Detection in Echocardiogram Videos”, *J. of Imaging*, vol. 6, no. 9, pp. 93, Sep. 2020

[89] K. T. Shahid and I. D. Schizas, “Spatial-Aware Hyperspectral Nonlinear Unmixing Autoencoder With Endmember Number Estimation”, *IEEE Trans. Geosci. Remote Sens.*, (Accepted)

BIOGRAPHICAL STATEMENT

Kazi Tanzeem Shahid has finished his B.Sc. in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology in 2014. He has worked for Synergic Improvement Solutions Pty Ltd from January 2015 to July 2015 as a Forex Trader, and began his direct B.Sc.-to-Ph.D. program in the University of Texas at Arlington in Fall 2015. His research focuses on machine learning, unsupervised learning and artificial neural networks.