# Low-Shot Learning for the Semantic Segmentation of Remote Sensing Imagery

by

Ronald Kemker

B.S. in Computer Engineering, Michigan Technological University, 2010

B.S. in Electrical Engineering - Photonics, Michigan Technological University, 2010

M.S. in Electrical Engineering, Michigan Technological University, 2011

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Chester F. Carlson Center for Imaging Science
College of Science
Rochester Institute of Technology

August 17, 2018

Signature of the Author ______________________________

Accepted by ______________________________
Coordinator, Ph.D. Degree Program Date

CHESTER F. CARLSON CENTER FOR IMAGING SCIENCE

COLLEGE OF SCIENCE

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

---

Ph.D. DEGREE DISSERTATION

---

The Ph.D. Degree Dissertation of Ronald Kemker
has been examined and approved by the
dissertation committee as satisfactory for the
dissertation required for the
Ph.D. degree in Imaging Science

Dr. Christopher Kanan, Dissertation Advisor

Dr. Pengcheng Shi, External Chair

Dr. Carl Salvaggio

Dr. Michael Gartley

Date

*To my wife Christine.*

Low-Shot Learning for the Semantic Segmentation
of Remote Sensing Imagery

by

Ronald Kemker

Submitted to the
Chester F. Carlson Center for Imaging Science
in partial fulfillment of the requirements
for the Doctor of Philosophy Degree
at the Rochester Institute of Technology

## Abstract

Deep-learning frameworks have made remarkable progress thanks to the creation of large annotated datasets such as ImageNet, which has over one million training images. Although this works well for color (RGB) imagery, labeled datasets for other sensor modalities (e.g., multispectral and hyperspectral) are minuscule in comparison. This is because annotated datasets are expensive and man-power intensive to complete; and since this would be impractical to accomplish for each type of sensor, current state-of-the-art approaches in computer vision are not ideal for remote sensing problems. The shortage of annotated remote sensing imagery beyond the visual spectrum has forced researchers to embrace unsupervised feature extracting frameworks. These features are learned on a per-image basis, so they tend to not generalize well across other datasets. In this dissertation, we propose three new strategies for learning feature extracting frameworks with only a small quantity of annotated image data; including 1) self-taught feature learning, 2) domain adaptation with synthetic imagery, and 3) semi-supervised classification. "Self-taught" feature learning frameworks are trained with large quantities of unlabeled imagery, and then these networks extract spatial-spectral features from annotated data for supervised classification. Synthetic remote sensing imagery can be used to boot-strap a deep convolutional neural network, and then we can fine-tune the network with real imagery. Semi-supervised classifiers prevent overfitting by jointly optimizing the supervised classification task along side one or more unsupervised learning tasks (i.e., reconstruction). Although obtaining large quantities of annotated image data would be ideal, our work shows that we can make due with less cost-prohibitive methods which are more practical to the end-user.

## Acknowledgements

First and foremost, I would like to thank my wife Christine. She has made many sacrifices for our family including a deployment two months after we were married and now a time-intensive PhD program. I love you very much and I look forward to spending the rest of my life making it up to you.

Second, I would like to thank my parents, Brian and Terri; parents in-law, Tom and Karen; my sister Stephanie, her wife Courtney, and my brothers-in-law Travis and Jimmy. You have always loved and supported Christine and I, which we are eternally grateful for.

Third, I would like to thank the countless number of people from previous assignments that have invested so much time into my development and made this opportunity possible. I would like to give a special shout-out to all of my military and civilian mentors, including Michael Roggemann, Chris Middlebrook, Glen Archer, Nikola Subotic, Tyler Erickson, Colin Brooks, Joel LeBlanc, John Valenzuela, Frank Scenna, Brian McKellar, Andrew Hoff, Ian Cameron, Wendy Kunkle, Carl Brenner, Jeffrey Kwok, Paul Sebold, Michael Grogan, Michael Belko, Bobby Bankston, and the many others that have and continue to shape my personal and professional career. Good luck to all of you, and your families, in your future endeavors.

Fourth, I would like to thank the people here that made this all possible. First, I would like to thank my advisor Dr. Kanan for his mentorship and for pushing me over the last three years to do above and beyond what was expected of me. You put a lot of trust and faith in my work and ability to get things done. I would also like to give a special thanks to Dr. Salvaggio for helping me build the RIT-18 dataset and to Dr. Gartley for helping me with my DIRSIG problems. We would like to thank Nina Raqueno, Paul Sponagle, Timothy Bausch, Michael McClelland II, and other members of the RIT Signature Interdisciplinary Research Area, UAS Research Laboratory that supported with data collection efforts for RIT-18. I would like to thank Kushal Kafle and Tyler Hayes for being a sounding board during this process and for putting up with me in the lab.

Finally, Christine and I would both like to thank the US/Canadian Air Force students, and all of our friends, that have either already moved on to their next big thing or are still grinding away here at RIT today. You have provided a community that made our time here a bit easier and more enjoyable. Good luck to all of you and I hope that our paths cross again really soon!

## Author Publications

† indicates that a modified version of this publication is included in this dissertation.

**Refereed Publications**

- **Kemker, R.** and Kanan, C. (2018) FearNet: Brain-Inspired Model for Incremental Learning. In the *Proceedings for the Sixth International Conference for Learning Representations*.
- **Kemker, R.** , McClure, M., Abitino, A., Hayes, T., and Kanan, C. (2018) Measuring Catastrophic Forgetting in Neural Networks. In the *Proceedings for the Thirty-Second Association for the Advancement of Artificial Intelligence (AAAI)*.

† **Kemker, R.** and Kanan C. (2017) Self-Taught Feature Learning for Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing (TGRS)*, 55(5): 2693-2705. 10.1109/TGRS.2017.2651639.

**Submitted/In-Review**

† **Kemker, R.**, Salvaggio C., and Kanan C. (In Review) Algorithms for Semantic Segmentation of Multispectral Remote Sensing Imagery using Deep Learning. In review at the *ISPRS Journal of Photogrammetry and Remote Sensing - "Deep Learning for Remotely Sensed Data"*.

† **Kemker, R.**, Luu, R., and Kanan C. (In Review) Low-Shot Learning for the Semantic Segmentation of Remote Sensing Imagery. In review at the *IEEE Transactions on Geoscience and Remote Sensing (TGRS)*.

- Parisi, G., **Kemker, R.**, Part, J., Kanan, C., and Wermter, S. (In Review) Continual Lifelong Learning with Neural Networks: A Review. In review at *Neural Networks*.

**Technical Reports**

† **Kemker, R.**, Salvaggio, C, and Kanan C. (2017) High-Resolution Multispectral Dataset for Semantic Segmentation. *arXiv preprint* arXiv:1703.06452.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction and Motivation

## 1.1 Context

Unsupervised and supervised feature learning has received a lot of attention over the past few years thanks to advancements in machine- and deep-learning. In computer vision, deep convolutional neural networks (DCNNs) have shattered previous state-of-the-art frameworks for nearly every supervised learning task such as object recognition, object detection, semantic segmentation, etc. These breakthroughs have been made possible by advancements in computer hardware (e.g. graphical processing units (GPUs), more efficient learning optimization schemes [3], fast and effective non-linear activations (e.g. rectified linear units (ReLU)), and massive labeled datasets such as ImageNet[4] and MSCOCO[5]. Although computer vision has made significant progress over the past several years, these frameworks still require large labeled datasets to learn discriminative features that generalize well.

### 1.1.1 Self-Taught Learning

The ImageNet Challenge uses an object recognition dataset that contains 1.2 million training images distributed across 1,000 object classes. This dataset was considered by the computer vision community to be particularly challenging; and in 2012, the authors in [6] released the AlexNet DCNN which crushed the competition. This network-architecture opened the door to even deeper networks that have continued to push state-of-the-art development on the ImageNet challenge[7–9]. These DCNN frameworks have also been re-purposed to learn additional tasks with orders-of-magnitude less labeled data[10, 11].

Although deep-learning frameworks have been remarkably successful with color (RGB) imagery, supervised deep-learning frameworks for non-RGB sensors tend to not perform as well. This is because there is not an ImageNet-sized dataset for non-RGB image modalities such as multispectral (MSI) and hyperspectral imagery (HSI). Previous attempts to train DCNN frameworks to classify HSI resulted in mediocre performance because there are not enough samples to learn generalized features, so remote sensing researchers have embraced unsupervised spatial-spectral feature extractors such as stacked autoencoders. Although these frameworks have historically achieved state-of-the-art performance on individual datasets, no work has demonstrated that these features transfer across multiple datasets/sensors. A real-time deployable framework should not learn a feature representation on a per-image basis because of time and memory constraints, so it will need extract generalized features that are also discriminative enough to perform well across many images.

In our work, we used self-taught learning to build spatial-spectral feature extractors for HSI classification. Instead of learning features that are specific to a single labeled dataset, self-taught learning uses large quantities of unlabeled data to learn discriminative features that generalize well. Once the framework is learned, it can be deployed to work across a variety of different datasets.

### 1.1.2 Domain Adaptation

The self-taught learning paradigm works well for some remote sensing scenes; however, very high resolution imagery requires the discriminative power of DCNN-based semantic segmentation frameworks. These models use combinations of convolution and spatial downsampling (pooling) layers that allow for the network to learn semantic information from large-scale scenes and to form associations between different objects present in the scene.

In our work, we used large quantities of synthetic remote sensing imagery to augment the requirement for large annotated MSI/HSI datasets. We then fine-tuned these DCNN frameworks using real imagery, which is a technique commonly referred to as domain adaptation. This strategy prevents overfitting in DCNN-based segmentation frameworks, which will allow the network to provide superior classification performance.

### 1.1.3 Semi-supervised Learning

A third approach to training deep learning frameworks using only a small quantity of annotated image data is semi-supervised learning. Semi-supervised frameworks jointly train neural networks to perform supervised (i.e., classification) and unsupervised tasks (i.e., reconstruction) to prevent over-fitting. These models learn a more meaningful feature representation from the image data; and as a result, they better generalize on the test data.

In our work, we introduce a semi-supervised classification framework that is jointly optimized to classify individual pixels and reconstruct the image data.

## 1.2 Objectives

This dissertation will address the training of supervised machine- and deep-learning frameworks with relatively small quantities of annotated remote sensing imagery (i.e. low-shot learning). The main objectives of our work include:

1. Develop universal feature extracting frameworks from large quantities of unlabeled data that work well across multiple datasets and sensors (i.e. self-taught learning).

   (a) Evaluate the transferability of low- and high-level features built by self-taught learning frameworks (Chapters 2 and 4)

   (b) Demonstrate that the discriminative power of these frameworks are more powerful when trained on large quantities of unlabeled data versus on the labeled data alone (Chapter 2 and 4)

   (c) Demonstrate that these generalized features transfer across sensors (Chapter 2 and 4)

2. Establish new training/testing methodologies that enable the development of deployable classification frameworks for remote sensing imagery

   (a) Identify issues with current training/testing methodologies in remote sensing literature which can affect the future deployability of these classification frameworks. (Chapter 2)

   (b) Establish a new benchmark dataset that address these issues. (Chapter 3 and Appendix A)

3. Develop universal feature extracting frameworks from large quantities of synthetic data (i.e. domain adaptation). (Chapter 3)
   (a) Evaluate performance on new benchmark established in Chapter 3
   (b) Demonstrate the utility of our learning strategy when model capacity increases
4. Develop modular self-taught feature learning paradigm for the semantic segmentation of remote sensing imagery. (Chapter 4)
   (a) Improve original self-taught learning frameworks from Chapter 2 by incorporating multiple reconstruction losses throughout the framework.
   (b) Evaluate the discriminative power of these learned features learned and compare it to previous feature extraction methods presented in Chapter 2.
   (c) Develop semi-supervised classification framework that works well with small quantities of annotated image data.
   (d) Demonstrate that the improved feature extraction framework and a semi-supervised classifier yield state-of-the-art performance for the semantic segmentation of non-RGB remote sensing imagery.

## 1.3 Dissertation Layout

This dissertation consists of five chapters, including the introduction (Chapter 1) and conclusion (Chapter 5).

### 1.3.1 Chapter 2: Self-Taught Feature Learning for Hyperspectral Image Classification (Objectives #1 and #2)

In Chapter 2, we study self-taught learning for hyperspectral image classification. Supervised deep learning methods are currently state-of-the-art for many machine learning problems, but these methods require large quantities of labeled data to be effective. Unfortunately, existing labeled hyperspectral image benchmarks are too small to directly train a deep supervised network. Alternatively, we used self-taught learning which is an unsupervised method to learn feature extracting frameworks from unlabeled hyperspectral imagery. These models learn how to extract generalizable features by training on sufficiently large quantities of unlabeled data that is distinct from the target dataset. Once trained, these models can extract features from smaller labeled target datasets. We studied two self-taught learning frameworks for hyperspectral image classification. The first is a shallow approach that uses independent component analysis, and the second is a three-layer (stacked) convolutional autoencoder. Our models are applied to the Indian Pines, Salinas Valley, and Pavia University datasets, which were captured by two separate sensors at different altitudes. Despite large variation in scene-type, our algorithms achieve state-of-the-art results across all three datasets.

### 1.3.2 Chapter 3: Algorithms for the Semantic Segmentation of Remote Sensing Imagery (Objective #2 and #3)

Deep convolutional neural networks (DCNNs) have been used to achieve state-of-the-art performance on many computer vision tasks (e.g., object recognition, object detection, semantic segmentation) thanks to a large repository of annotated image data. Large labeled datasets for other sensor modalities, e.g., multispectral imagery (MSI), are not available due to the large cost and manpower required. In this chapter, we adapt state-of-the-art DCNN frameworks in computer vision for semantic segmentation for MSI imagery. To overcome label scarcity for MSI data, we substitute real MSI for generated synthetic MSI in order to initialize a DCNN framework. We evaluate our network initialization scheme on the new RIT-18 dataset that we present in this chapter. This dataset contains very-high resolution MSI collected by an unmanned aircraft system. The models initialized with synthetic imagery were less prone to over-fitting and provide a state-of-the-art baseline for future work.

### 1.3.3 Chapter 4: Low-Shot Learning for the Semantic Segmentation of Remote Sensing Imagery (Objective #4)

Recent advances in computer vision using deep learning with RGB imagery (e.g., object recognition and detection) have been made possible thanks to the development of large annotated RGB image datasets. In contrast, multispectral image (MSI) and hyperspectral image (HSI) datasets contain far fewer labeled images, in part due to the wide variety of sensors used. These annotations are especially limited for semantic segmentation, or pixel-wise classification, of remote sensing imagery because it is labor intensive to generate image annotations. Low-shot learning algorithms can make effective inferences despite smaller amounts of annotated data. In this paper, we study low-shot learning using self-taught feature learning for semantic segmentation. We introduce 1) an improved self-taught feature learning framework for HSI and MSI data and 2) a semi-supervised classification algorithm. When these are combined, they achieve state-of-the-art performance on remote sensing datasets that have little annotated training data available. These low-shot learning frameworks will reduce the manual image annotation burden and improve semantic segmentation performance for remote sensing imagery.

## 1.4 Novel Contributions

**Chapter 2: Self-Taught Feature Learning for Hyperspectral Image Classification**

The main contribution of this chapter was to introduce the self-taught learning paradigm to the remote sensing community. We trained spatial-spectral feature extracting frameworks on large quantities of unlabeled hyperspectral imagery (HSI). The large quantities of data allow our frameworks to learn discriminative features that can transfer across datasets.

We used two feature extracting frameworks to demonstrate that our self-taught learning paradigm is capable of transferring features across multiple datasets by achieving state-of-the-art classification results on the Indian Pines, Salinas Valley, and Pavia University datasets. One of these frameworks uses independent component analysis to build low-level feature extracting filters that resemble Gabor patterns (i.e. bar/edge detectors). The other is a stacked convolutional autoencoder framework that is capable of extracting deep spatial-spectral features from image data. We demonstrated that features from both frameworks can be transferred across different sensors through band-resampling.

We identified two problems with how HSI classification is currently performed in the remote sensing community. First, training and testing folds are traditionally built from the same HSI cube. This is ok with spectral-only classification; but if we want to include neighboring pixel information, there is a high chance of overlap between training and testing data. This can artificially inflate performance. Second, training/testing folds are built by randomly sampling available labeled data, and the results are reported as the mean/standard-deviation of $N$ runs.

**Chapter 3: Algorithms for the Semantic Segmentation of Remote Sensing Imagery**

We are the first to adapt recent fully-convolutional DCNNs to semantic segmentation of multispectral remote sensing imagery. We demonstrated that pre-training these networks on synthetic imagery can prevent overfitting and significantly improve their performance on the semantic segmentation task.

We released the new RIT-18 dataset for evaluating MSI semantic segmentation algorithms. RIT-18 addresses the problems listed in Chapter 2. RIT-18 was built using high-resolution (4.7 cm) multispectral imagery captured by an unmanned aerial system and has been pre-split into training, validation, and testing folds. This dataset contains 18 labeled object classes and we have shown that this dataset is very difficult to perform well on due to the highly unbalanced class distribution. We are working on making RIT-18 available through the IEEE Geoscience and Remote Sensing Society (GRSS) evaluation server. This will standardize evaluation and push state-of-the-art development.

**Chapter 4: Low-Shot Learning for the Semantic Segmentation of Remote Sensing Imagery**

We introduced the semantic segmentation framework SuSA (**s**elf-ta**u**ght **s**emi-supervised **a**utoencoder). SuSA is designed to perform well on MSI and HSI data where image annotations are scarce.

We developed the stacked multi-loss convolutional auto-encoder (SMCAE) model for spatial-spectral feature extraction in non-RGB remote sensing imagery. SMCAE uses unsupervised self-taught learning to acquire a deep bank of feature extractors. SMCAE is used by SuSA for feature extraction.

We propose the semi-supervised multi-layer perceptron (SS-MLP) model for the semantic segmentation of non-RGB remote sensing imagery. SuSA uses SS-MLP to classify the feature representations from SMCAE, and SS-MLP's semi-supervised mechanism enables it to perform well at low-shot learning.

We demonstrate that SuSA achieves state-of-the-art results on the Indian Pines and Pavia University datasets hosted on the IEEE GRSS Data and Algorithm Standard Evaluation website.

## 1.5 Related Publications

Portions of this dissertation have been published in the following outlets:

- **Kemker, R.**, Kanan C. (2017) Self-Taught Feature Learning for Hyperspectral Image Classification. *IEEE Transactions on Geoscience and Remote Sensing (TGRS)*, 55(5): 2693-2705. 10.1109/TGRS.2017.2651639

- **Kemker, R.**, Salvaggio, C, and Kanan C. (2017) High-Resolution Multispectral Dataset for Semantic Segmentation. *arXiv preprint* arXiv:1703.06452.
- **Kemker, R.**, Salvaggio C., and Kanan C. (2017) Algorithms for Semantic Segmentation of Multispectral Remote Sensing Imagery using Deep Learning. In review at *ISPRS Journal of Photogrammetry and Remote Sensing - "Deep Learning for Remotely Sensed Data"*.
- **Kemker, R.**, Luu, R., and Kanan C. (2018) Low-Shot Learning for the Semantic Segmentation of Remote Sensing Imagery. In review at *IEEE Transactions on Geoscience and Remote Sensing (TGRS).*

# Chapter 2

# Self-Taught Feature Learning for Hyperspectral Image Classification

Hyperspectral image (HSI) classification, known in computer vision as semantic segmentation, involves assigning class labels to pixels. In this task, one of the greatest challenges is determining what types of features should be extracted from the pixels. In the past, researchers used have hand-engineered features, e.g., Gabor filters [12, 13], wavelets [14–16], extended morphological profiles [17], morphological attribute profiles [18], extended multi-attribute profiles (EMAP) [1], and rotation invariant spatial-spectral feature representations [19]. These features have major limitations and often require a great deal of tuning to get them to work well on a particular dataset. An alternative approach is to train an algorithm that learns how to extract useful features directly from the pixels. Deep convolutional neural networks (CNNs) have been widely adopted in computer vision for this purpose, but they require large labeled datasets to train; otherwise, they will work poorly on test data. Existing labeled HSI datasets for remote sensing are minuscule in comparison to the color (RGB) datasets used in computer vision. This problem is compounded by the far greater dimensionality of pixels in HSI compared to RGB, making it difficult to use supervised feature learning techniques on today's HSI datasets. However, unsupervised feature learning can be used because these techniques do not require labels. A variation of this approach is self-taught learning [20], which uses unsupervised learning to train a model by extracting features from large unlabeled datasets. The self-taught model is then applied to a target labeled dataset. The assumption behind self-taught models is that the features they learn to extract are generalizable, i.e., they will work well across datasets if their underlying natural statistics are similar. In this chapter, we study two distinct self-taught learning algorithms for classifying pixels in HSI.

A number of researchers have used unsupervised learning to extract features from HSI data. The earliest of these did this by learning linear combinations of individual pixels using principal component analysis (PCA) [21] and independent component analysis (ICA) [22–25]. These methods were pixel-specific, and did not include information about the neighborhood around the pixel. As the ground sample distance (GSD) for HSI improved, it was shown that spatial and spectral features could be combined to improve classification performance [26]. A number of algorithms have been used to extract spatial-spectral features from HSI data, including stacked autoencoders [2, 27–31], dictionary learning [32, 33], and ICA [34]. These methods discover salient information buried in the data.

The majority of published unsupervised feature learning frameworks involve learning spatial-spectral features from a single HSI dataset, breaking these feature responses up into test and train data, and then classifying that dataset [2, 15, 35–37]. In some cases, the features are learned on both training and testing data, which may skew results compared to what can be expected when the system is deployed [15, 36]. In self-taught learning for HSI classification, three steps are needed: 1) learn a set of filters from dataset #1, 2) use these filters to generate features from dataset #2, and then 3) classify dataset #2. Self-taught learning frameworks are more representative of an operational or deployed system since the test data is not used for unsupervised feature extraction or classifier training.

In this chapter, we propose two distinct self-taught learning frameworks for HSI classification: 1) multi-scale ICA (MICA) and 2) the stacked convolutional autoencoder (SCAE). Both models learn a set of generalizable filters from a diverse set of unlabeled HSI data, but MICA learns a low-level feature representation whereas SCAE learns deeper features. These filters are then applied to classify three popular labeled HSI benchmark datasets: Indian Pines, Salinas Valley, and Pavia University. Although these datasets have different GSDs, we show that MICA and SCAE are still able to produce state-of-the-art results. We also demonstrate that their features can be transferred across different sensors through band-resampling.

## 2.1 Background

### 2.1.1 Unsupervised Feature Learning

There are many limitations to using engineered features, especially with HSI. A way to overcome these limitations is using unsupervised feature learning algorithms that analyze the data in order to create a filter bank. These algorithms identify common features such as spatial edges and spectral transitions. These features take the form of spatial-spectral filters that can be applied to the labeled data prior to classification. Three of the most common algorithms for unsupervised feature learning in RGB images are $k$-means [38], sparse coding[39–41], and ICA [42, 43], with ICA generally yielding the best results in the literature.

Figure 2.1 shows filters learned using ICA on natural color images [44]. Each filter resembles Gabor filters that have been widely used to model simple cells in the human visual cortex. Red-green, blue-yellow, and light-dark opponency is an emergent phenomenon with ICA filters learned from RGB images, and this same kind of opponency is found in the human visual system [43]. The ICA algorithm used is a sparse coding algorithm that was designed to separate a signal into its statistically independent (non-Gaussian) components. ICA builds sparse features from whitened data by learning a set of orthonormal basis vectors.

ICA has been heavily used for HSI data analysis problems, including spectral unmixing [25, 45], target detection [46], clustering [34] (sometimes called unsupervised classification), and classification [22, 23]. While spectral features learned from HSI data with ICA have been widely used, little work has been done to learn spatial-spectral features from neighborhood pixel information.

An autoencoder is an unsupervised neural network that learns useful encodings from the input data [47]. The learned feature representation can be used for unsupervised feature extraction in semi-supervised classification networks [48] and for dimensionality reduction [49]. Different variations of these autoencoders, including sparse [50], denoising [51], and convolutional [52] have been used to improve image classification performance by extracting discriminative features from data and feeding them to a classifier. The convolutional autoencoder (CAE) uses a combination of convolution and max-pooling hidden-layers, similar to that of a traditional CNN, to learn the input coding. Stacked autoencoders (SAE) use several autoencoders to learn a hierarchical feature representation, resulting in more discriminative features[52, 53]. The first-layer of the CAE will have a similar bar/edge detector structure like the ones shown in Figure 2.1, but the SAE

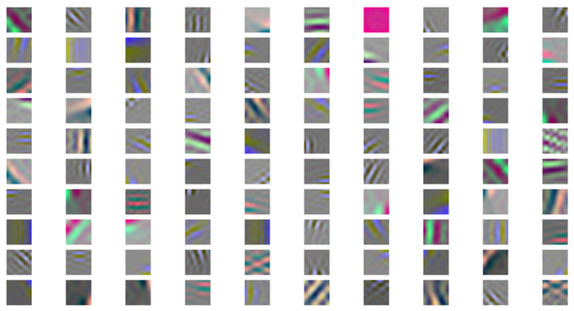


Figure 2.1: ICA filters learned from color (RGB) imagery. Most of the filters resemble Gabor filters. The green-red, dark-light, and yellow-blue opponency are an emergent phenomenon.

will also learn higher-level features not present in ICA. SAEs have become extremely prevalent in HSI classification literature [27,28] including sparse [2], denoising [29], and convolutional [30,31].

### 2.1.2 Self-Taught Learning

Traditional supervised classification methods use only labeled data, which is often scarce or difficult to obtain. To compensate, several alternative frameworks have been proposed that harness unlabeled data or data from other datasets. Semi-supervised methods use both labeled and unlabeled data from a dataset [54]. A commonly used semi-supervised approach in remote sensing is active learning. Active learning uses an initial prediction from a classifier to feed more training data back into the classifier [55–57]. Transfer-learning frameworks exploit labeled data from other datasets to improve the performance of the supervised classifier. The additional labeled data does not necessarily need to share class labels with the original dataset, but it should be sufficiently representative of the data that will be classified [58–60]. For RGB image analysis in computer vision, training CNNs on very large labeled datasets and then using them on smaller labeled datasets has been enormously successful.

Self-taught learning seeks to leverage the advantages of both the semi-supervised and transfer learning paradigms [20], and it has been widely adopted for unsupervised feature learning [20,43,61,62]. Self-taught learning uses a large quantity of unlabeled data from alternative datasets to improve supervised classification on a target dataset. Additionally, the unlabeled data does not necessarily need to have the same classes present as the labeled data. It only requires that the data share the same underlying statistics, e.g., natural images.

### 2.1.3 Feature Learning for HSI Classification

Early frameworks only leveraged spectral information during the classification process, but many modern techniques use spatial-spectral features that harness neighboring pixel information to improve performance. In [1], supervised spectral feature extraction and EMAPs were used to automatically extract spatial-spectral features and classify the Indian Pines and Pavia University datasets. In [15], the authors used combinations of three-dimensional Gabor wavelets to extract spatial-spectral features from the Indian Pines dataset. These fused filters reduced the number of redundant features, resulting in more discriminative features. Other papers have extended these successes by fusing large combinations of different spatial-spectral features. One of the most successful papers iterated through 54 different sets of spectral (raw spectrum, PCA, linear discriminant analysis (LDA), and non-parametric weighted feature extraction) and spatial (morphological profiles, Gabor, grey-level co-occurrence matrices (GLCMs), and segmented GLCMs) features [36]. This chapter still holds some of the best results for the Salinas Valley and Pavia University datasets; however, each of their state-of-the-art results were achieved with a different combination of spatial-spectral features. This highlights a need for methods that can extract features that generalize across datasets with minimal tuning.

Unsupervised feature learning frameworks require less tuning than hand-crafted features. These methods learn a set of spatial-spectral filters by analyzing natural scenes. Some of the most successful methods are sparse coding algorithms and stacked-sparse autoencoders (SSAEs). Spatially-weighted sparse coding uses neighboring pixel information to improve the performance of spectral unmixing in HSI data [33]. The author used the dominant end-member to classify a given pixel, achieving an impressive performance on the Indian Pines dataset. In [35], the author used SSAEs to extract deep features from the three labeled datasets featured in this chapter. Other frameworks have been developed to automatically tune spatial-spectral filters that were previously hand-crafted to provide the optimal class separation [63]. While ICA has been extensively used for unsupervised spatial-spectral feature learning with RGB imagery, less work has been done with HSI data and most of it is concentrated solely on spectral features of a single pixel and ignore neighborhood information [22–25, 34].

Each unsupervised feature learning framework listed above learned a set of spatial-spectral features from a single data source; however, these filters may not be representative of spatial-spectral features found in other datasets. In [2], the author transferred a set of filters learned from one HSI dataset onto another. Both of these datasets had the same GSD and were imaged by the same sensor, so the ability of the features to generalize to other sensors or GSDs is unclear.

## 2.2 Self-Taught Learning Frameworks

We propose two distinct self-taught learning frameworks. MICA uses ICA to learn a shallow filter bank at multiple scales. SCAE is a deep neural network approach, which can potentially learn richer non-linear features than MICA. We use models with shallow and deep feature representations to demonstrate the utility of our self-taught training methodology. Each model is trained on multiple unlabeled HSI datasets to add spatial-spectral variation to filters learned by the models. After training, both models are then used to extract features from the three labeled datasets, which are then fed to a classification algorithm. In this section, we give high-level descriptions for both models. Specific implementation details are provided in Section 2.3.

Acquiring large quantities of unlabeled HSI data is straight-forward thanks to government and university sponsored websites containing open-source airborne/satellite imagery. For this chapter, we used unlabeled data from the Airborne Visible / Infrared Imaging Spectrometer (AVIRIS) and Hyperion HSI sensors. The goal is to show that we can use self-taught learning to train a model that can extract features which generalize across datasets and sensors.

### 2.2.1 Multi-scale ICA (MICA)

Our proposed MICA model, illustrated in Figure 2.2, will learn a set of low-level feature extracting filters at multiple scales. When ICA is applied to image data, it tends to learn bar/edge, gradient, and corner detectors. MICA obtains robustness to multiple scales because it is trained on datasets collected at different altitudes. For example, a horizontal bar filter will detect the edge of a house or the outline of a long highway. These objects may be captured at different scales, but the same filter may respond similarly to them.

The input to MICA is a contrast-stretched image, i.e., the pixels in each image are normalized to be between 0 and 1. To cope with the enormous luminance variance in the scene, these normalized pixels are pushed through a non-linear function that pro-

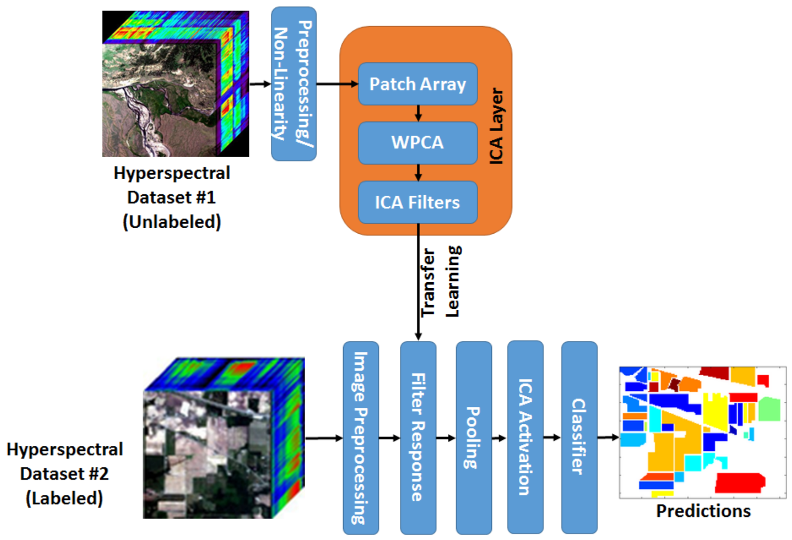


Figure 2.2: Our self-taught learning model using multiscale ICA (MICA) filters. This framework learns low-level feature extractors from one (or more) hyperspectral dataset(s) and then applies them to a separate target dataset. Multiple datasets with varying GSDs can be used to make the filters scale invariant.

duces better behaved first-order natural image statistics [64]. This kind of transformation is common when learning filters using ICA in RGB, LMS, and monochrome colorspaces [42, 43, 61, 62, 65]. These functions resemble transformations done in the retina to help the human visual system handle large changes in luminance. Typically, a monotonically increasing function is used, with its shape resembling a logarithm. Here, we used the cumulative distribution function (CDF) of an exponential distribution for this pre-processing step, which others have also used with ICA [65]:

$$r_i = 1 - \exp(-\lambda_i |x_i|) \tag{2.1}$$

where $x_i$ is the $i$ spectral band of pixel $\mathbf{x} \in \mathbb{R}^z$, and $\lambda_i$ is a parameter fit to the pixels from band $i$ using the unlabeled datasets. In preliminary experiments, we found this worked better than using two alternative functions that used logarithms [42, 61].

To learn ICA filters, we extracted $k$ patches of size $p \times p \times z$ at random locations from $N$ unlabeled HSI datasets, where $z$ is the number of spectral bands. Each patch is vectorized to form a column vector and then stacked along the second dimension to create an $zp^2 \times kN$ patch array $\mathbf{R}$. Whitened PCA (WPCA) is then used to whiten the data and reduce the dimensionality:

$$\mathbf{X} = \mathbf{D}^{-\frac{1}{2}}\mathbf{E}^T\mathbf{R} \tag{2.2}$$

where $\mathbf{X}$ is the whitened patch array, $\mathbf{E}$ is the eigenvectors of the covariance matrix $\mathbf{RR}^T$, and $\mathbf{D}$ is a diagonal matrix containing the corresponding eigenvalues of the $\mathbf{E}$ matrix [42]. Consistent with others [43,62,66], the first principal component (PC) is discarded because it mainly corresponds to the brightness across patches, and the corresponding eigenvalue is orders of magnitude larger than the other PCs, which hinders the ability of ICA to learn discriminative features. These steps result in $\mathbf{X}$ being a $d \times kN$ patch array, where $d$ specifies the reduced dimension where $d \ll zp^2$. The value for $d$ is found using cross-validation.

Next, ICA is used to learn a $d \times d$ linear transformation matrix $\mathbf{A}$ from the whitened data, which makes the data linearly statistically independent. Our spatial-spectral filters are built by multiplying the ICA transformation matrix $\mathbf{A}$ with the WPCA transformation, i.e., $\mathbf{W} = \mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{E}^T$, where $\mathbf{W}$ will be a $d \times zm^2$ matrix, where the rows of $\mathbf{W}$ contain the filters. Each row of $\mathbf{W}$ is then reshaped to the correct spatial dimension, producing a collection of $d$ filters of size $p \times p \times z$ [43]. A visualization of the filters learned using this procedure is shown in Figure 2.3. These low-level ICA feature extractors include Gabor-like bar/edge detectors, image gradients, and various spatial textures. Convolving a $m \times n \times z$ HSI with the learned filter bank, while using symmetric border padding, will yield a $m \times n \times d$ array of feature responses.

Subsequently, mean-pooling is applied to the feature response array to incorporate translation robustness into MICA. We convolve each channel of the ICA feature responses with an $a \times a$ mean pooling filter. We used padding to preserve the spatial dimensions of the ICA filter response map.

Finally, we increased the discriminative utility of the MICA feature responses by applying a non-linear function to them. Previous work has shown that the discriminative power of ICA filters for images can be increased by taking the absolute value of the filter responses and then applying a CDF-like function [43, 61, 67]. Following [61], we apply the CDF of an exponential distribution to the ICA filter responses

$$g_i = 1 - \exp(-\lambda_i |q_i|) \tag{2.3}$$

where $q_i$ is the mean-pooled feature response of channel $i$ and $\lambda_i$ is the learned scale parameter of the exponential distribution. The $\lambda_i$ parameters were fit using the unlabeled HSI datasets.

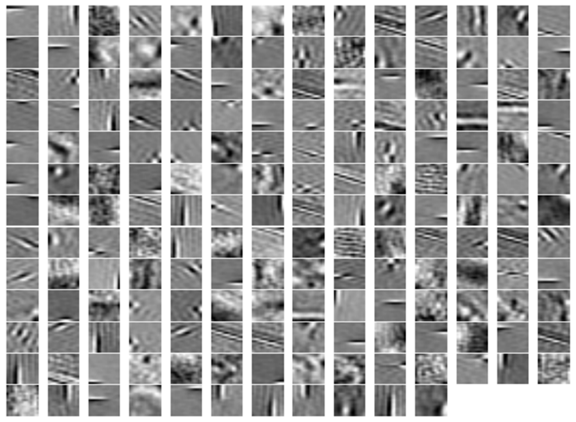


Figure 2.3: A visualization of the learned $15 \times 15$ ICA filters from the Indian Pines dataset. It was generated by summing across the spectral dimension of the filters (hundred of bands). Despite summing across all spectral dimensions, this visualization still resembles the Gabor-like filters found in primary visual cortex.

### 2.2.2 Stacked Convolutional Autoencoders (SCAE)

As a form of deep neural network, SCAE can extract higher-level features than the ones extracted by MICA. SCAE is made up of several autoencoders, so it is important to understand how they operate.

An autoencoder is a type of neural network that can be trained in an unsupervised manner to learn an encoded representation of the data. A typical autoencoder $f$ is given by $\hat{\mathbf{x}} = f(\mathbf{x})$, where $\mathbf{x}$ is the input. It is trained to try to make $\hat{\mathbf{x}}$ as close to $\mathbf{x}$ as possible, i.e., to learn an identity function. Typically, an autoencoder will have internal constraints so that the hidden layers of its neural network will learn interesting features, e.g., sparsity constraints or a bottle neck. After training, the output of the hidden layers can be used as an alternative encoding of the data. While an autoencoder could be trained end-to-end as a deep network, which is often done when using bottle-neck constraints, each autoencoder in an SAE is typically trained individually. An autoencoder with a single hidden layer is given by

$$\mathbf{h} = \sigma\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right) \tag{2.4}$$

where $\sigma$ is the activation function, $\mathbf{W}$ are the learned weights, and $\mathbf{b}$ are the learned biases. Typically, $\mathbf{h}$ is referred to as the encoding of $\mathbf{x}$ produced by the hidden layer.

Recovering $\hat{\mathbf{x}}$ is then given by

$$\hat{\mathbf{x}} = \sigma\left(\mathbf{W}'\mathbf{h} + \mathbf{b}'\right) \tag{2.5}$$

where $\mathbf{W}'$ is typically constrained such that $\mathbf{W}' = \mathbf{W}^T$.

A CAE is an autoencoder variation in which the hidden layers are convolutional layers [52]. This allows them to efficiently process image data. The output of a convolutional hidden layer is given by

$$\mathbf{H} = \sigma\left(\mathbf{W} * \mathbf{X}\right), \tag{2.6}$$

where $*$ denotes the convolution operation, $\mathbf{X}$ is the input (e.g., a 3-tensor HSI image), $\mathbf{W}$ is a learned filter bank (a 4-tensor), and $\mathbf{H}$ is the output feature response map (a 3-tensor). For brevity, we have dropped the biases in the equations.

Autoencoders, including CAEs, can be stacked to learn a hierarchical features [30, 31, 52, 53]. When this is done with CAEs, it gives rise to the SCAE. The output of layer $k$ in an SCAE is then given by

$$\mathbf{H}^k = \sigma\left(\mathbf{W}^k * \mathbf{H}^{k-1}\right), \tag{2.7}$$

where $\mathbf{H}^0 = \mathbf{X}$. Each layer in an SCAE is trained individually to minimize reconstruction error and then its encoding is used as the input to the next CAE layer.

In this chapter, our SCAE uses three stacked CAE models; but as shown in Figure 2.4, the hidden layers we use are more complex than traditional CAEs. The feed-forward network resembles a traditional CNN architecture by incorporating max-pooling layers, which requires our model to upsample for reconstruction. In Figure 2.4, convolution #3 is a fully-connected layer followed by refinement modules [68], which are used to reconstruct the image. These modules use skip connections to merge low-level features from the feed-forward network with the high-level semantic features at the end of the network. This is followed by an upsampling operation until the image has been restored to its original dimensionality. As shown in Figure 2.4, each refinement module contains three separate convolution layers with trainable weights.

As shown in Figure 2.5, the feature responses $\mathbf{H}^k$ from each CAE are concatenated and mean-pooled to introduce translation invariance. These features are then made unit length by dividing by the $L_2$-norm. Since the dimensionality is much higher than the number of training samples, we use WPCA to reduce the dimensionality prior to classification.

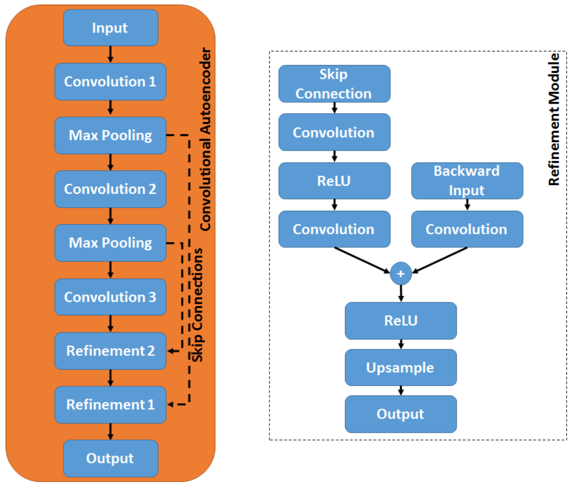


Figure 2.4: This figure shows the CAE modules used in this chapter. The refinement layer combines features from different parts of the network to provide a better reconstruction.

## 2.3 Experiments and Results

We evaluated MICA and SCAE against three standard HSI benchmark datasets: Indian Pines, Salinas Valley, and Pavia University. The ground truth maps are shown in Figure 2.6. Section 2.3.1 discusses these datasets and the training sets that we will use to compare our frameworks against.

Section 2.3.2 lays out the specific network architectures proposed for this chapter and any other relevant information regarding training and evaluation. Section 2.3.3 lists the experimental results for both MICA and SCAE on the three benchmark datasets. Section 2.3.4 highlights some additional experiments that justify some of the decisions made for our proposed models.

The scikit-learn library [69] was used to implement ICA and the support vector machine (SVM) classifier. Keras [70], with the Theano backend, was used to implement the SCAE. The Spectral Python library [71] was used to perform band-resampling. This library maps the band centers and full width at half maximum values of the source spectrum to the target spectrum by assuming a Gaussian sensor response.

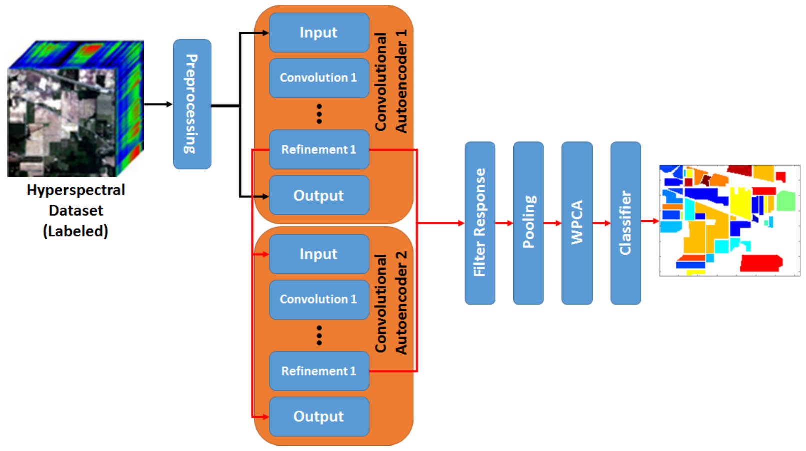


Figure 2.5: A two-layer stacked convolutional autoencoder. This figure shows how a pre-trained network can extract features from labeled data and classify them.

### 2.3.1 HSI Datasets

We used unlabeled data from the AVIRIS and Hyperion HSI sensors to train our self-taught learning frameworks. We trained two MICA and SCAE models from each of the unlabeled data sources; MICA-AVIRIS, MICA-Hyperion, SCAE-AVIRIS, and SCAE-Hyperion. We used three unlabeled datasets to train the MICA models and 20 datasets to train SCAE.

AVIRIS is an airborne sensor operated by NASA's Jet Propulsion Laboratory. It has 224 visual/near-infrared (VNIR) and short-wave infrared (SWIR) spectral bands. The GSD varies with the elevation in which the data was collected.

Hyperion is a HSI sensor located on NASA's EO-1 satellite. Hyperion data contains 242 VNIR/SWIR spectral bands; however, this experiment only used the 198 calibrated bands. The GSD of each HSI dataset is 30.5 meters, which is larger than all of the labeled datasets used in this chapter. The AVIRIS and Hyperion data is radiometrically calibrated using gain factors provided by NASA.

We denote the dimensionality of each labeled dataset by # of pixels $\times$ # of pixels $\times$ # of bands. Bands that were in the atmospheric absorption region or bands with low signal-to-noise ratio (SNR) were removed.

Indian Pines, Figure 2.6(a), is a $145 \times 145 \times 200$ dataset that was collected over Northwestern Indiana [72]. The original Indian Pines dataset has 224-bands, but bands

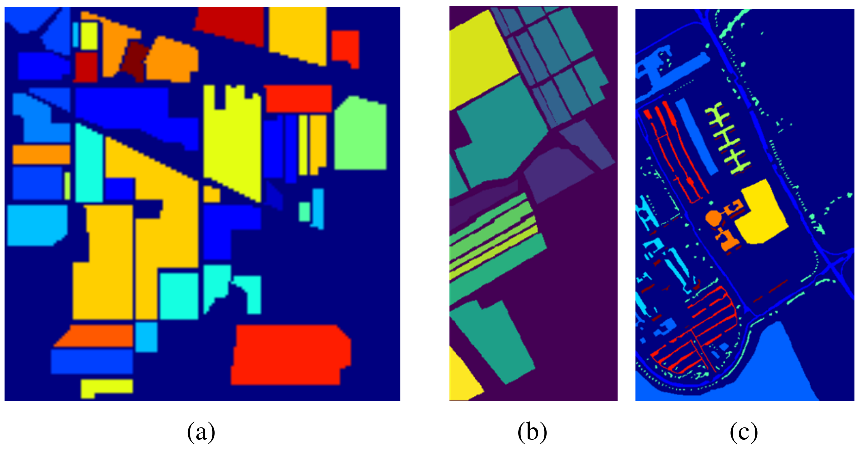


Figure 2.6: The ground truth maps for Indian Pines 2.6(a), Salinas Valley 2.6(b), and Pavia University 2.6(c) HSI datasets.

104-108, 150-163, and 220-224 were removed due to atmospheric absorption or low SNR. The ground-truth contains 16 classes of different crops and crop-mixtures. Indian Pines has a GSD of 20 meters. Our self-taught learning algorithms were compared against three common training sets including #1) 5% per class[15], #2) 10% per class[33], and #3) a common training set containing 50 samples per class (except with a few smaller classes where 15 samples are reserved for training[1]).

Salinas Valley, Figure 2.6(b), is a $512 \times 217 \times 204$ dataset that contains 16 ground-truth classes related to crops at different stages in their growth. The original dataset has 224-bands, but bands 108-112, 154-167, and 224 were removed due to atmospheric absorption or low SNR. The GSD for Salinas Valley is 3.7 meters. Our algorithms were compared against three standard training sets including #1) 1% per class[35], #2) 5% per class[36], and #3) 50 samples per class [36].

Pavia University, Figure 2.6(c), is a $610 \times 610 \times 103$ dataset that was collected during a campaign flown over Northern Italy. The original dataset was band-resampled from 115 bands to 103 bands. Pavia University has 9 ground truth classes containing a mixture of man-made and natural scenery. The GSD for Pavia University is 1.3 meters. Our algorithms were compared against three common training sets including #1) 5% per class[36], #2) 10% per class[16], and #3) a custom training set commonly used in literature [2].

The Indian Pines and Salinas Valley datasets were both collected by the AVIRIS sensor. The Pavia University dataset was collected by the Reflective Optics System Imaging

Spectrometer (ROSIS) sensor which only operates in the VNIR spectral range.

### 2.3.2 Experimental Setup

#### MICA Setup and Parameters

The MICA ICA filter size is fixed to $m = 15$ across every dataset, which worked well in preliminary experiments. MICA's mean pooling layer uses a $11 \times 11$ filters. We trained MICA using unlabeled datasets that came from different sensors. MICA-AVIRIS is solely trained with data from the AVIRIS sensor and MICA-Hyperion solely from data from the Hyperion sensor. In both cases, the filters were created by extracting 5,000 random patches from each of the three unlabeled datasets (15,000 patches total). The unlabeled AVIRIS datasets are orthorectified, so these rotated images have gaps in the corners with no data, and we did not extract patches from these empty regions. This quantity of data was sufficient to learn low-level bar/edge filters since these features can be found in most scenes.

#### SCAE Setup and Parameters

We trained two separate SCAE networks from AVIRIS (SCAE-AVIRIS) and Hyperion (SCAE-Hyperion) data. Each SCAE network stacks three CAEs with an identical architecture. This architecture is outlined in Table 2.1. The last hidden layer of the preceding CAE becomes the input of the following CAE.

Each convolution module in Table 2.1 uses convolution followed by batch normalization and ReLU activation. Batch normalization is a regularization technique that speeds up and stabilizes training by reducing internal covariate shift [73]. Every convolutional layer is randomly initialized with a zero-mean normal distribution [74].

The only pre-processing step for the unlabeled data is subtracting the channel mean and dividing by the channel standard deviation. This mean and standard deviation is stored and applied to the labeled data. SCAE was trained by extracting 2,000 $16 \times 16$ patches from 20 unlabeled datasets and feeding it in batches of 256 through the network. Theoretically, we could have trained SCAE on entire HSI datasets; but due to graphical processing unit (GPU) memory limitations, this was not feasible. The amount of data used to train SCAE is larger than the quantity used to train MICA because it has more trainable parameters. This is because MICA is only learning a single-layer network; whereas, SCAE is stacking three multi-layer networks together. Twenty unlabeled datasets seemed to be sufficiently large for SCAE to learn discriminative filter banks.

Table 2.1: The convolutional autoencoder model used for this chapter where $z$ is either the number of spectral bands in the input image (first CAE) or the number of features extracted from the previous CAE (subsequent CAEs). The 'Refinement 1' layer is the feature response that will be passed to other CAEs and classified. Each convolutional layer consists of a convolution followed by batch normalization and ReLU activation. The refinement layer is outlined in Section 2.2.2.

| **Layer Name** | **Output Shape** | **Filter Size** |
|---|---|---|
| Input | $m \times n \times z$ | N/A |
| Convolution 1 | $m \times n \times 256$ | $3 \times 3 \times 256$ |
| Max Pooling 1 | $\frac{m}{2} \times \frac{n}{2} \times 256$ | $2 \times 2$ |
| Convolution 2 | $\frac{m}{2} \times \frac{n}{2} \times 512$ | $3 \times 3 \times 512$ |
| Max Pooling 2 | $\frac{m}{4} \times \frac{n}{4} \times 512$ | $2 \times 2$ |
| Convolution 3 | $\frac{m}{4} \times \frac{n}{4} \times 512$ | $3 \times 3 \times 512$ |
| Max Pooling 3 | $\frac{m}{8} \times \frac{n}{8} \times 512$ | $2 \times 2$ |
| Convolution 4 | $\frac{m}{8} \times \frac{n}{8} \times 1024$ | $1 \times 1 \times 1024$ |
| Refinement 3 | $\frac{m}{4} \times \frac{n}{4} \times 512$ | $3 \times 3 \times 512$ |
| Refinement 2 | $\frac{m}{2} \times \frac{n}{2} \times 512$ | $3 \times 3 \times 512$ |
| Refinement 1 | $m \times n \times 256$ | $3 \times 3 \times 256$ |
| Output | $m \times n \times z$ | N/A |

Each CAE was trained independently using the Adam optimizer with Nesterov momentum [3] and a mean-squared-error cost function. We used an initial learning rate of 2e-3 and dropped the learning rate by a factor of 10 (four times) as the validation loss plateaued.

After learning its filters, each labeled dataset is passed through the SCAE network. These feature responses are concatenated, fed through a $5 \times 5$ mean-pooling layer, and normalized with the $L_2$-norm. WPCA is used to reduce the dimensionality of the feature responses to 97.5% (SCAE-Hyperion) and 97% (SCAE-AVIRIS) of the original variance. This variance was tuned to Indian Pines and found to work well on the other two labeled datasets as well.

### Band-Resampling

Band-resampling is used to maximize large quantities of unlabeled, open-source data. For MICA, filters are learned from the unlabeled data and then resampled to match the spectral bands of the target (labeled) data. For SCAE, the network is trained on the unlabeled data and the labeled data is resampled so it can be passed through the network. When deployed, unlabeled data from the sensor would be used to build these feature extracting frameworks.

Band-resampling could also be useful to take advantage of existing training data. For example, if we increased the spectral resolution of our sensor, we may be able to upsample existing labeled datasets in order to increase the quantity of annotated data. Labeled data is expensive and difficult to come by.

### Classification

The training/testing sets were built by randomly sampling the feature responses generated by our MICA and SCAE frameworks. The number of samples extracted for training data was selected based on current state-of-the-art solutions found in literature, and a plot for each labeled dataset was generated to illustrate mean per-class accuracy as a function of percent of training samples. These feature responses were classified using a radial basis function SVM (RBF-SVM). Cross-validation was used to determine the optimal penalty term $C$ and kernel width ($\gamma$) hyperparameters. For MICA, the feature responses were flipped across the horizontal axis to augment the training data.

The peak performance for the MICA framework in Section 2.3.3 was achieved by using a different number of learned filters, which corresponds to the number of PCs retained during Section 2.2.1. If there are too many PCs, then the learned ICA filters will be noisy and less discriminative. We found that the ideal number differed for each labeled dataset, since they all cover different spectral bands and have a different quantity of labeled data available. For a deployable system, where the size of the image and spectral bands are fixed, the number of filters can be easily determined through cross-validation and fixed for future classification. In this chapter, we cross-validated by a step-size of 5 filters to determine the optimal quantity; however, adding or removing a few filters will not have a major impact on classification performance.

The major advantage that the SCAE network has over MICA is that it requires no tuning for different datasets, which means we are universally applying the same SCAE network across all three benchmark datasets.

### 2.3.3 Experimental Results

In this section, we compare MICA and SCAE against the best algorithms in literature found for each of the standard benchmark datasets we used. This comparison is challenging since each paper uses a different training methodology for their algorithm, so we selected the training sets we found to be the most common. All of our results are reported as the mean and standard deviation of 30 sample runs in terms of overall accuracy (OA), mean-class accuracy (AA), and kappa statistic ($\kappa$). Additionally, we provide kappa statistic plots as a function of training size (percent samples per class). The kappa statistic was selected because it takes both overall and mean-class accuracy into account. Finally, we provide classification maps for the three state-of-the-art comparisons for each dataset.

#### Indian Pines

Table 2.2 compares MICA and SCAE to state-of-the-art solutions for the Indian Pines dataset found in literature. We used the first 210 feature extracting filters for the MICA framework. The first training set, 5% training data, is compared against a spatial/spectral feature extraction method that uses three-dimensional Gabor wavelets (3DGW) [15]. The second training set, 10% training data, is compared against a spatially weighted sparse coding (SWSC) unmixing approach [33]. The third training set, 50 samples per class (except 3 smaller classes that use 15 samples per class), is compared to a paper that uses extended morphological attribute profiles for supervised feature extraction (DAFE) [1]. Contextual deep learning multinomial logistic regression (CDL-MLR) had a percent higher mean class accuracy for the second training set. Our classifier was cross-validated by using overall accuracy as the metric, and our algorithm yielded state-of-the art results for this metric. Overall, our self-taught learning frameworks, especially SCAE, proved to be more discriminative than the algorithms listed above.

Figure 2.7 shows classification maps for the three training sets in Table 2.2 using our SCAE models. The majority of the errors reside in the pixel edges between classes, especially between contiguous classes. Possible causes include adjacency effect and feature blurring with the mean-pooling layer. The errors between frameworks are not consistent, suggesting that ensembling may improve classification performance.

The kappa statistic for the Indian Pines dataset is shown in Figure 2.8. The smallest training percentage used for this plot is 1%; however, some of the classes will provide only one training sample at that level. Even at 5% training data, four classes have five or fewer samples to train on, but both frameworks still achieve impressive results. SCAE has a slight advantage over MICA likely due to the requirement for higher-level features

Table 2.2: The classification results for MICA and SCAE on the Indian Pines dataset. The three training sets include #1) 5% per class, #2) 10% per class and #3) 50 samples per class (15 for smaller classes). We compare against state-of-the-art algorithms found in literature. Some statistics, including standard deviations, were not provided by the authors, so they could not be included.

| | **OA** | **AA** | $\kappa$ |
|---|---|---|---|
| **Training Set #1** | | | |
| 3DGW [15] | 96.04 | 92.81 | Unk |
| MICA-AVIRIS | 95.42±0.76 | 92.29±1.68 | 0.9478±0.0087 |
| MICA-Hyperion | 95.71±0.63 | 91.56±1.52 | 0.9510±0.0072 |
| SCAE-AVIRIS | 96.21±0.67 | 93.32±1.76 | 0.9567±0.0077 |
| SCAE-Hyperion | **96.61±0.50** | **94.58±1.50** | **0.9613±0.0057** |
| **Training Set #2** | | | |
| CDL-MLR [35] | 98.23 | **98.70** | 0.9799 |
| SWSC [33] | 98.30 | 97.70 | 0.981 |
| MICA-AVIRIS | 98.32±0.23 | 96.60±1.28 | 0.9809±0.0026 |
| MICA-Hyperion | 98.31±0.39 | 97.10±1.15 | 0.9807±0.0044 |
| SCAE-AVIRIS | 98.66±0.22 | 97.07±1.56 | 0.9848±0.0026 |
| SCAE-Hyperion | **98.68±0.27** | 97.83±0.84 | **0.9849±0.0031** |
| **Training Set #3** | | | |
| DAFE [1] | 93.27 | 95.86 | 0.923 |
| MICA-AVIRIS | 94.63±1.00 | 97.31±0.37 | 0.9385±0.0114 |
| MICA-Hyperion | 94.26±0.99 | 96.84±0.51 | 0.9340±0.0113 |
| SCAE-AVIRIS | 95.35±0.68 | 97.61±0.43 | 0.9466±0.0078 |
| SCAE-Hyperion | **96.12±0.78** | **98.03±0.35** | **0.9554±0.0089** |

on larger GSD imagery, which is why there is a slight improvement on smaller training sets.

The quantity of data used for smaller training sets could cause the classifier to suffer from the curse of dimensionality. We mitigated this issue by selecting the RBF kernel for our SVM. The large margin assumption in the RBF kernel helps compensate for this issue through regularization. The excellent performance of MICA and SCAE on smaller training sets demonstrates that our classifier is capable of discriminating between classes.

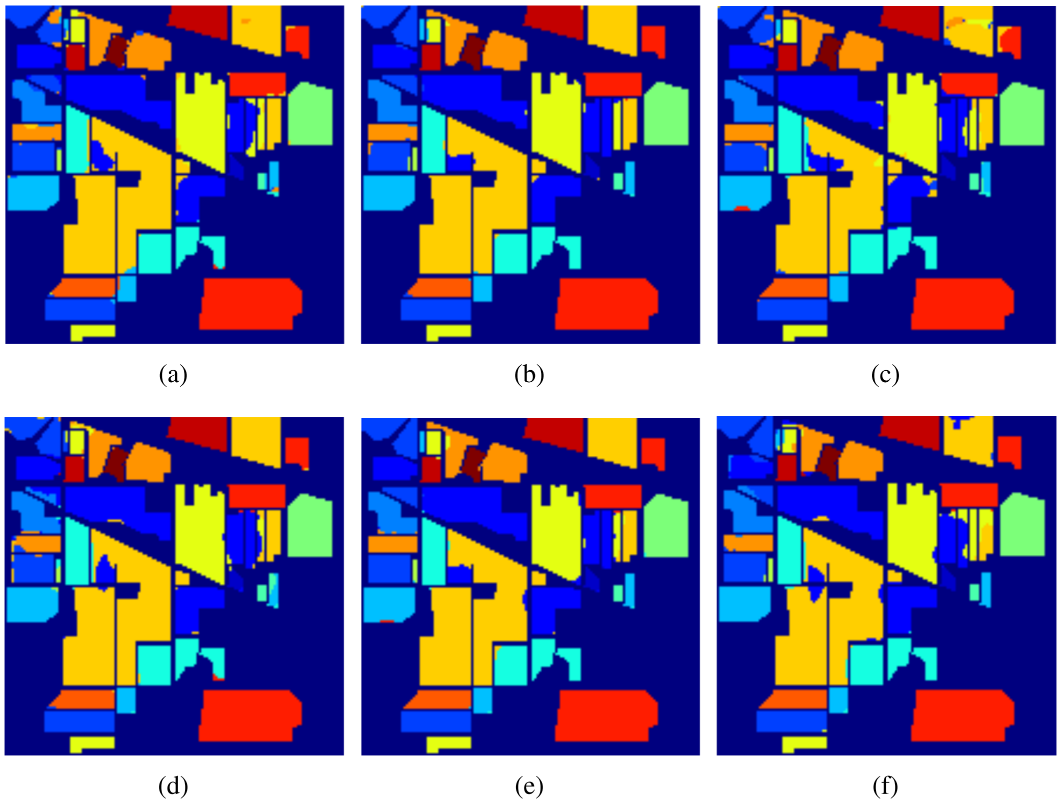


Figure 2.7: The classification maps for the Indian Pines dataset. Figures a-c were generated with the SCAE-AVIRIS filters while Figures d-f were generated with the SCAE-Hyperion filters. These class maps correspond, respectively, to the following three training sets: 5% training data, 10% training data, and a training set commonly found in literature[1].

## Salinas Valley

Table 2.3 compares MICA and SCAE to state-of-the-art solutions for the Salinas Valley dataset found in literature. We used the first 130 and 145 feature extracting filters for MICA-AVIRIS and MICA-Hyperion respectively. The first training set, 1% training data, was compared to the CDL-MLR framework built by [35]. The second and third training sets, 5% training data and 50 samples per class respectively, were compared against a feature combination strategy (GLCM+) developed by [36]. The second training set combines

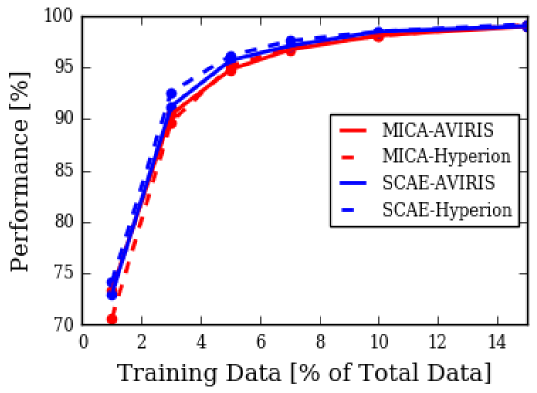


Figure 2.8: Results for the Indian Pines dataset in terms of kappa statistic as a function of percent training samples per class.

the HSI data with features from morphological profiles and segmented GLCM features. The third training set combines the features from the second training set along with Gabor and non-segmented GLCM features. MICA and SCAE both exceeded all earlier methods.

Figures 2.9 and 2.10 show the classification maps for the three training sets in Table 2.3 using both our SCAE and MICA models respectively. The results indicate that our self-taught learning frameworks perform well on the fine-grain material identification task (lettuce at different growth stages and fallow conditions). The small error that is present mostly resides in the untrained grapes and untrained vineyards classes, which may have similar spectral characteristics.

The kappa statistic for the Salinas Valley dataset is shown in Figure 2.11. Both of our frameworks work well for the Salinas Valley dataset, even with a low number of training samples. The close performance between MICA and SCAE indicates that low-level spatial-spectral features from MICA may be sufficient for this particular dataset.

In Section 2.2, we stated that the band-resampling function assumes that the source and target sensors have a Gaussian spectral response function. Although this is not always the case, Figure 2.11 quantitatively shows that MICA-AVIRIS and MICA-Hyperion perform equivalently well. Since MICA-Hyperion was band-resampled and MICA-AVIRIS was not, it could be concluded that band-resampling had a negligible effect in performance.

Table 2.3: The classification results for MICA and SCAE on the Salinas Valley dataset. The three training sets include #1) 1% per class, #2) 5% per class and #3) 50 samples per class. We compare against state-of-the-art algorithms found in literature. Some statistics, including standard deviations, were not provided by the author, so they could not be included.

| | **OA** | **AA** | $\kappa$ |
|---|---|---|---|
| **Training Set #1** | | | |
| CDL-MLR [35] | 98.26 | 98.72 | 0.9806 |
| MICA-AVIRIS | **98.36±0.40** | **98.56±0.31** | **0.9817±0.0044** |
| MICA-Hyperion | 98.15±0.31 | 98.34±0.48 | 0.9794±0.0034 |
| SCAE-AVIRIS | 98.20±0.34 | 97.41±0.56 | 0.9799±0.0038 |
| SCAE-Hyperion | 97.98±0.36 | 98.41±0.30 | 0.9776±0.0040 |
| **Training Set #2** | | | |
| GLCM+ [36] | 98.61 | Unk | Unk |
| MICA-AVIRIS | 99.87±0.03 | 99.82±0.12 | 0.9985±0.0008 |
| MICA-Hyperion | **99.90±0.05** | **99.88±0.06** | **0.9989±0.0005** |
| SCAE-AVIRIS | 99.75±0.06 | 99.69±0.09 | 0.9972±0.0007 |
| SCAE-Hyperion | 99.75±0.07 | 99.71±0.08 | 0.9973±0.0008 |
| **Training Set #3** | | | |
| GLCM+ [36] | 95.41 | Unk | Unk |
| MICA-AVIRIS | 97.15±0.56 | 98.57±0.29 | 0.9683±0.0062 |
| MICA-Hyperion | 95.95±0.66 | 98.34±0.24 | 0.9549±0.0073 |
| SCAE-AVIRIS | **98.06±0.45** | **98.94±0.22** | **0.9784±0.0050** |
| SCAE-Hyperion | 97.50±0.54 | 98.85±0.22 | 0.9722±0.0060 |

**Pavia University**

Table 2.4 compares MICA and SCAE to state-of-the-art methods on the Pavia University dataset. The first training set, 5% training data, was compared to the same GLCM+ feature combination framework developed by [36]. The second training set, 10% training data, was compared to a three-dimensional scattering wavelet (3DSW) transform developed by [16]. The third training set, a common training set used widely throughout literature, was compared to a SSAE developed by [2]. The results for MICA involved the use of 100 feature extracting filters. Both of our frameworks achieve state-of-the art results on all three training sets.

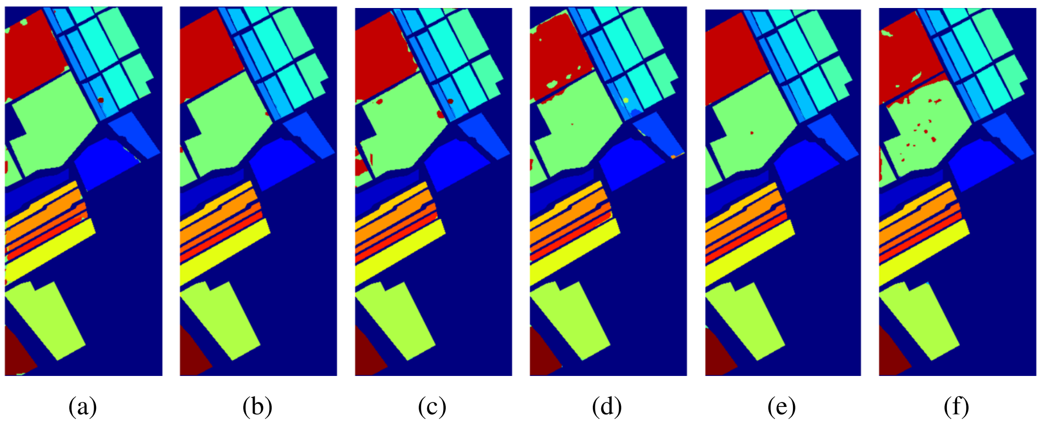


Figure 2.9: The classification maps for the Salinas Valley dataset. Figures a-c were generated with the SCAE-AVIRIS filters while Figures d-f were generated with the SCAE-Hyperion filters. These class maps correspond, respectively, to the following three training sets: 1% training data, 5% training data, and 50 samples per class.

Figure 2.12 shows the classification maps for the three training sets in Table 2.4 using SCAE features. There are only minor errors, which include confusing shadows for trees and bare soil for meadows.

The kappa statistic for this dataset is shown in Figure 2.13. MICA-AVIRIS and MICA-Hyperion performed equally well with the exception of the smaller training sets where MICA-Hyperion yielded optimal results. This near-equivalent performance could indicate that spatial features are more important than spectral information for this dataset.

Table 2.5 shows a comparison of MICA and SCAE against the transfer learning experiment done in [2]. In this experiment, the author learned filters using a SSAE from the Pavia Center dataset and applied them to the Pavia University dataset. The Pavia Center and Pavia University datasets were both captured with the ROSIS sensor; and since both of these datasets were captured during the same campaign, they have approximately the same GSD. The results in Table 2.5 were calculated using a training set of 50 samples per class. Although MICA and SCAE were trained from data captured by completely separate sensors, our classification yielded better results because these filters were able to extract more salient information from the labeled data than the SSAE.

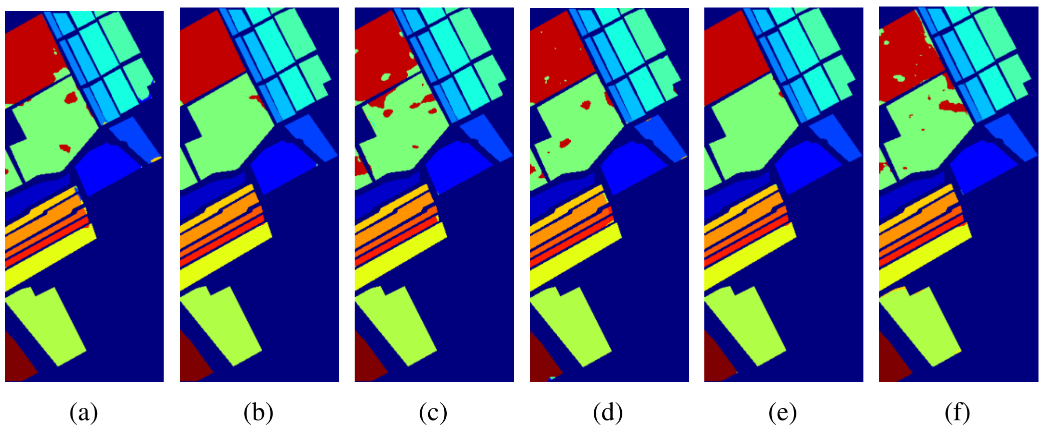


Figure 2.10: The classification maps for the Salinas dataset. Figures a-c were generated with the MICA-AVIRIS filters while Figures d-f were generated with the MICA-Hyperion filters. These class maps correspond to the following three training sets: 1% training data, 5% training data, and 50 samples per class respectively.

### 2.3.4 Additional Experiments

#### Scale-Invariance of MICA

Earlier, we stated that the MICA filter-band is somewhat scale-invariant. Despite all three labeled datasets having different GSDs, MICA performed well on them. This is because some of the filters learned by MICA-AVIRIS and MICA-Hyperion resemble Gabor-type edge detectors at various sizes. One of these filters can work well at detecting a building from an image taken by an airborne sensor and could work just as well at detecting a long road imaged by a satellite sensor. MICA-AVIRIS is even more scale-invariant because it learns filters from a variety of datasets taken at different GSDs.

To prove our hypothesis, we re-sized the Indian Pines data and ground-truth map by half the original size. As a fair comparison, we used the same MICA-AVIRIS filters discussed in Section 2.3.3. In Figure 2.14, we see that re-sizing the image had a minimal impact on the classification performance, which could indicate that the filters work just as well across different scales. The minor differences in performance are likely due to re-size interpolation errors.

In Figure 2.14, the quantity of training data is equivalent to the training data from Section 2.3.3. To further demonstrate that MICA is scale-robust, Figure 2.15 shows the clas-

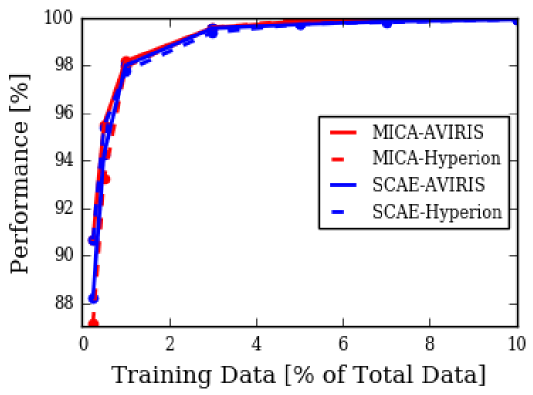


Figure 2.11: Results for the Salinas Valley dataset in terms of kappa statistic as a function of percent training samples per class.

sification performance when the GSD of the Salinas Valley dataset is artificially changed through image re-sizing. During this experiment, the training size was fixed at 50 samples per class. There was a minor increase in performance due to image blurring; however, the overall performance of the MICA filter bank indicates that there is some resistance to scale-invariance.

### Representation Similarity Analysis

How similar are the features learned across sensors and models? To address this question, we performed a representation similarity analysis (RSA) [75, 76] to compare the features extracted by the MICA and SCAE frameworks across sensors. RSA is a quantitative measure used by neuroscientists to assess the similarity of features from multiple models (or sources). Since MICA and SCAE have different dimensionality, we used reduced-rank regression (RRR) [77, 78] to do our RSA analysis. RRR is ordinary least-squares (OLS) regression with a low-rank constraint. This approach allows us to build a linear regression model from feature representation X to feature representation Y. We can then assess goodness of fit on test data to determine how well Y's features can be reconstructed from X's. If this can be done well, then X is considered to contain the same feature information as Y. Note that Y may only contain a subset of the features in X, so this regression analysis must be done from Y to X as well.

We use the standard RRR formulation to assess how well Y's features can be predicted

Table 2.4: The classification results for MICA and SCAE on the Pavia University dataset. The three training sets include #1) 5% per class, #2) 10% per class, and #3) a standard training set commonly found in literature. We compare against state-of-the-art algorithms found in literature. Some statistics, including standard deviations, were not provided by the author, so they could not be included.

| | **OA** | **AA** | $\kappa$ |
|---|---|---|---|
| **Training Set #1** | | | |
| GLCM+ [36] | 98.95 | Unk | Unk |
| MICA-AVIRIS | 99.21±0.18 | 98.40±0.36 | 0.9895±0.0023 |
| MICA-Hyperion | 99.20±0.17 | 98.31±0.43 | 0.9894±0.0022 |
| SCAE-AVIRIS | **99.50±0.13** | **99.07±0.30** | **0.9934±0.0018** |
| SCAE-Hyperion | 99.41±0.15 | 99.01±0.31 | 0.9921±0.0021 |
| **Training Set #2** | | | |
| 3DSW [16] | 99.30±0.12 | 98.63±0.23 | 0.9907±0.0016 |
| MICA-AVIRIS | 99.70±0.07 | 99.39±0.17 | 0.9961±0.0009 |
| MICA-Hyperion | 99.78±0.08 | 99.51±0.18 | 0.9971±0.0010 |
| SCAE-AVIRIS | **99.88±0.05** | **99.77±0.12** | **0.9984±0.0006** |
| SCAE-Hyperion | 99.81±0.06 | 99.65±0.13 | 0.9975±0.0008 |
| **Training Set #3** | | | |
| SSAE [2] | 98.63±0.17 | 97.64±0.22 | 0.9822±0.0013 |
| MICA-AVIRIS | 99.69±0.11 | 99.72±0.09 | 0.9957±0.0014 |
| MICA-Hyperion | 99.72±0.12 | 99.79±0.11 | 0.9962±0.0016 |
| SCAE-AVIRIS | 99.85±0.06 | **99.89±0.06** | 0.9980±0.0008 |
| SCAE-Hyperion | **99.88±0.05** | 99.87±0.06 | **0.9984±0.0006** |

Table 2.5: A comparison of the filter transfer learning experiment done in [2] and our MICA and SCAE models. The training set included 50 samples per class.

| | **OA** | **AA** | $\kappa$ |
|---|---|---|---|
| SSAE [2] | 91.96±0.87 | 93.52±0.42 | 0.9025±0.0112 |
| MICA-AVIRIS | 92.87±1.11 | 94.79±0.64 | 0.9067±0.0142 |
| MICA-Hyperion | 93.92±1.38 | 95.58±0.64 | 0.9203±0.0177 |
| SCAE-AVIRIS | 94.30±1.13 | **96.82±0.54** | 0.9253±0.0145 |
| SCAE-Hyperion | **95.84±0.94** | 96.56±0.51 | **0.9451±0.0123** |

from X's. First the regularized OLS regression is computed, i.e.,

$$\mathbf{B}_{OLS} = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^\top \mathbf{Y} \tag{2.8}$$

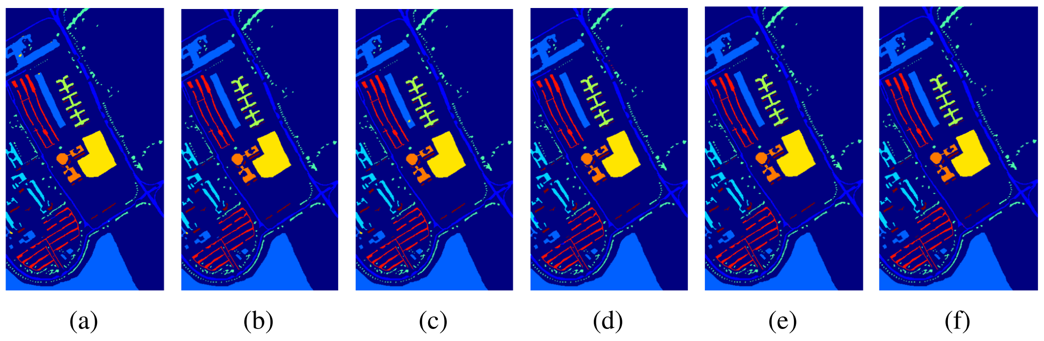


Figure 2.12: The classification maps for the Pavia University dataset. Figures a-c were generated with the SCAE-AVIRIS filters while Figures d-f were generated with the SCAE-Hyperion filters. These class maps correspond, respectively, to the following three training sets: 5% training data, 10% training data, and a training set commonly found in literature [2].

where $\mathbf{X}$ is the data we are fitting, $\mathbf{Y}$ is the data we are making the prediction on, and $\lambda$ is the L2-regularization penalty term. Then, PCA is performed on $\hat{\mathbf{Y}}_{OLS} = \mathbf{B}_{OLS}\mathbf{X}$ to obtain the first $r$ PCs $\mathbf{U}_r$. The RRR solution is given by

$$\mathbf{B}_{RRR} = \mathbf{B}_{OLS}\mathbf{U}_r\mathbf{U}_r^\top, \tag{2.9}$$

where $\mathbf{B}_{RRR}$ is used as a one-way linear mapping between feature representations.

Table 2.6 shows the coefficients of determination ($R^2$) for the input $\mathbf{X}$ and the predicted output $\hat{\mathbf{Y}}_{RRR} = \mathbf{B}_{RRR}\mathbf{X}$. We performed a 10-fold cross-validation using the MICA and SCAE feature responses from the Pavia University dataset and reported the $R^2$ values for the test data in terms of mean and standard deviation across all folds. The vertical and horizontal axes are the feature responses we fit and predicted respectively. We cross-validated for the optimal rank $r$ and L2 regularization penalty term $\lambda$.

Table 2.6 shows that the low-level MICA features learned from both AVIRIS and Hyperion sensors are very similar to each other. Also, the MICA features can be predicted from the SCAE fairly accurately ($R^2 > 0.9$ in all cases). However, MICA does a relatively poor job predicting the output of SCAE. This suggests that SCAE produces many of the feature responses of MICA, but MICA omits some of the higher-level features that SCAE generates.

The SCAE models trained on different sensors learn fairly similar features ($R^2 > 0.73$), but there are differences. There are two possible factors, which are not mutually

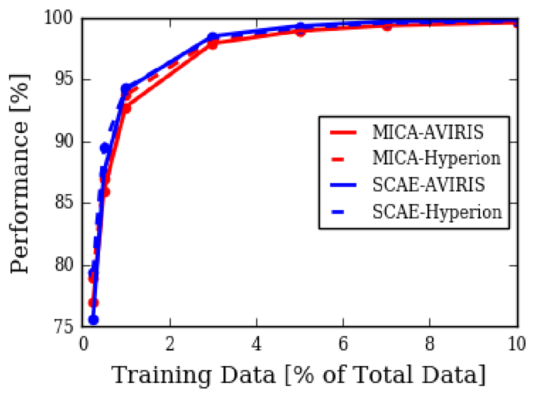


Figure 2.13: Results for the Pavia University dataset in terms of kappa statistic as a function of percent training samples per class.

Table 2.6: RSA between MICA and SCAE feature responses built from Pavia University (see Section 2.3.4). This table shows $R^2$ values for predicting a column (Y) from a row (X). A value of 1 means a perfect fit. The results are presented as the mean and standard deviation $R^2$ on the test data from 10 cross-validation folds. Because test data is being studied, the diagonal is not necessarily all 1s.

| | **MICA Framework** | | **SCAE Framework** | |
|---|---|---|---|---|
| | **#1 AVIRIS** | **#2 Hyperion** | **#3 AVIRIS** | **#4 Hyperion** |
| **#1** | 1.000±0.000 | 0.982±0.000 | 0.411±0.004 | 0.388±0.006 |
| **#2** | 0.974±0.001 | 1.000±0.000 | 0.431±0.004 | 0.421±0.006 |
| **#3** | 0.921±0.002 | 0.935±0.002 | 0.999±0.000 | 0.740±0.002 |
| **#4** | 0.915±0.002 | 0.927±0.002 | 0.730±0.001 | 0.999±0.000 |

exclusive. The first is that the information produced by the different sensors is somewhat different so they learn somewhat different features. An alternative explanation is that SCAE is difficult to optimize because it is a deep model, so the model obtained will significantly differ based on random weight initialization and the data fed to it. We tried to tease these issues apart by training two new SCAE-AVIRIS models with separate training sets, but with the same network initialization. Table 2.7 shows that using different training data yields slightly different feature responses. We also generated a third model by combining the training sets to determine if additional training data could yield additional

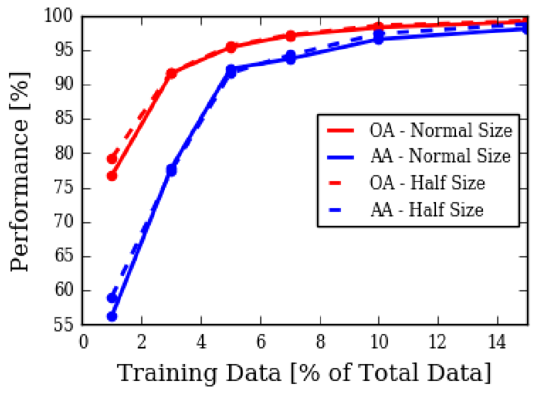


Figure 2.14: This figure shows that the learned MICA filter bank is scale-invariant. The solid lines represent the overall (red) and mean-class (blue) accuracies for the original data. The dashed lines represent the overall (red) and mean-class (blue) accuracies for the re-sized data.

Table 2.7: RSA for SCAE-AVIRIS trained with an identical network initialization. The results are presented as the mean and standard deviation $R^2$ on the test data from 10 cross-validation folds. This table shows $R^2$ values for predicting a column from a row.

| | **Set #1** | **Set #2** | **Combined** |
|---|---|---|---|
| **Set #1** | 0.9987±0.0000 | 0.8269±0.0010 | 0.7966±0.0014 |
| **Set #2** | 0.8157±0.0010 | 0.9987±0.0000 | 0.8074±0.0015 |
| **Combined** | 0.8090±0.0011 | 0.8290±0.0012 | 0.9987±0.0000 |

features, but there was only a small decrease in similarity.

## 2.4 Discussion and Conclusions

In this chapter, we introduced two self-taught learning frameworks that work well with a small amount of labeled training data. MICA has solely a low-level feature representation, while SCAE can learn higher-level feature representations. We evaluated on several common HSI benchmark datasets. Both models yielded state-of-the-art or near state-of-

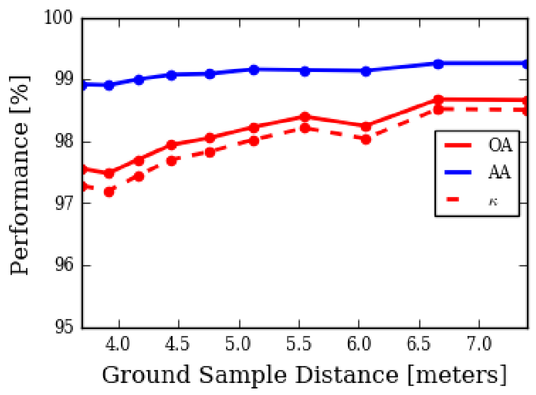


Figure 2.15: This figure shows the classification performance of the Salinas Valley dataset as the GSD is artificially changed through image resizing in terms of overall accuracy (OA), mean-class accuracy (AA), and kappa statistic ($\kappa$). The training set was fixed at 50 samples per class.

the-art results across all of the datasets. MICA and SCAE can learn filters from datasets with different GSDs making them robust to changes in scale.

The low-level features learned by MICA yielded superior results compared to the deeper stacked autoencoder architectures found in [2, 27–30]. This further demonstrates, as is true in any machine or deep learning problem, that the diversity and quantity of the training data can be just as important as the depth of our network. In most cases, our SCAE model yielded superior performance to our MICA model, showing that higher-level features can be advantageous in some cases.

The framework that achieved the best results also tells us something about the types of features that are necessary for good performance. SCAE-Hyperion, built from 30 meter GSD imagery, performed the best for the Indian Pines dataset, which may indicate that spectral features may be more important for high-GSD datasets. Salinas and Pavia University seemed to prefer spatial features (MICA and SCAE-AVIRIS). A deployable system could ensemble multiple spatial-spectral feature extracting frameworks to be more robust.

A good self-taught filter bank for HSI remote sensing data will have features that are common to any HSI remote sensing dataset. Low-level feature extractors, such as MICA, will look for edges, bars, gradients, and basic textures that can be commonly found in image data. High-level feature extractors, such as SCAE, will identify deeper features, such as parts, objects, and the semantic relationship between pixels and spectral channels. High-level feature extractors will likely have more discriminative power than shallow ones, but they take longer to train and are often slower to run.

As we have demonstrated, self-taught learning can be useful for HSI classification. It allows massive quantities of unlabeled HSI data to be used for training. This enables good performance to be achieved across datasets using only a small amount of labeled data.

## Acknowledgement

This chapter is a reprint of the material as it appears in IEEE Transactions on Geoscience and Remote Sensing, Volume 55, Issue 5, pages 2693 - 2705, 2017. I was the primary investigator on this paper.

We would like to thank Purdue University, Pavia University, and the Hysens project for making their labeled hyperspectral image (HSI) data available to the public. We would also like to thank NASA/JPL-Caltech and the United States Geological Survey for their repository of unlabeled HSI data.

# Chapter 3

# Algorithms for Semantic Segmentation of Multispectral Remote Sensing Imagery using Deep Learning

Semantic segmentation algorithms assign a label to every pixel in an image. In remote sensing, semantic segmentation is often referred to as image classification, and semantic segmentation of non-RGB imagery has numerous applications, such as land-cover classification [79], vegetation classification [80], and urban planning [81, 82]. Semantic segmentation has been heavily studied in both remote sensing and computer vision. In recent years, the performance of semantic segmentation algorithms for RGB scenes has rapidly increased due to deep convolutional neural networks (DCNNs). To use DCNNs for semantic segmentation, they are typically first trained on large image classification datasets that have over one million labeled training images. Then, these pre-trained networks are then adapted to the semantic segmentation task. This two-step procedure is necessary because DCNNs that process high-resolution color (RGB) images have millions of parameters, e.g., VGG-16 has 138 million parameters [7]. Semantic segmentation datasets in computer vision are too small to find good settings for the *randomly initialized* DCNN parameters (weights), and over-fitting would likely occur without the use of pre-trained networks. For example, to evaluate a semantic segmentation method on the RGB PASCAL VOC datasets [83], state-of-the-art methods use a DCNN pre-trained on ImageNet (1.28 million training images), fine-tune it for semantic segmentation on the COCO dataset (80K training images) [5], and then fine-tune it again on PASCAL VOC (1,464 training images) [84, 85].

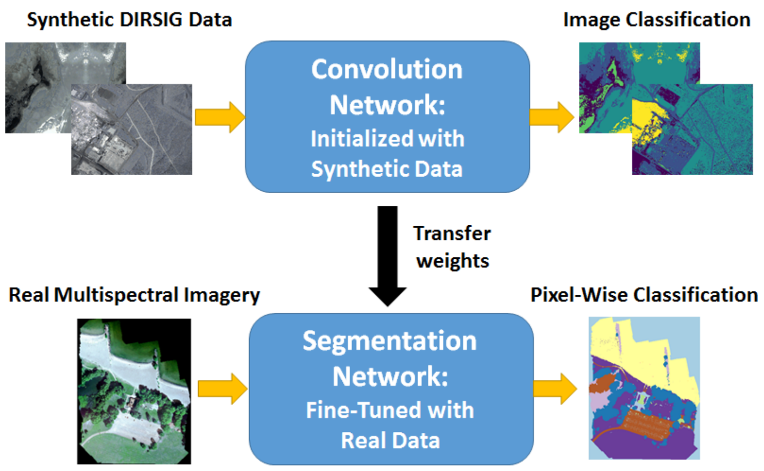

Synthetic DIRSIG Data
Image Classification
Convolution Network: Initialized with Synthetic Data
Transfer weights
Real Multispectral Imagery
Pixel-Wise Classification
Segmentation Network: Fine-Tuned with Real Data

Figure 3.1: Our proposed model uses synthetic multispectral imagery to initialize a DCNN for semantic segmentation. This model is then fine-tuned on real imagery.

Utilizing pre-trained networks to prevent overfitting works well for RGB imagery because massive labeled datasets are available; but in the non-RGB domain, label scarcity is a far greater problem. For example, existing semantic segmentation benchmarks for hyperspectral imagery consist of a single image mosaic. Therefore, pre-training DCNNs on hand-labeled datasets consisting of real images is not currently possible in non-RGB domains. In this chapter, we explore an alternative approach: using vast quantities of automatically-labeled *synthetic* multispectral imagery (MSI) for pre-training DCNN-based systems for semantic segmentation.

We propose to use the Digital Imaging and Remote Sensing Image Generation (DIRSIG) modeling software to generate large quantities of synthetic MSI and corresponding label maps. We use DIRSIG to build a large, diverse scene model, in which we can simulate various weather and lighting conditions. We then capture synthetic aerial images of the scene with a MSI sensor model. We use the synthetic data to initialize a DCNN for object recognition, and then we combine the pre-trained DCNN with two different fully-convolutional semantic segmentation models using real MSI.

In the past, researchers have used DCNNs pre-trained on ImageNet to yield state-of-the-art results for the semantic segmentation of high-resolution multispectral aerial imagery [86, 87] because the most widely used benchmarks [81] only use a single non-RGB band. What happens when the spectral range of the dataset increases? The real MSI used

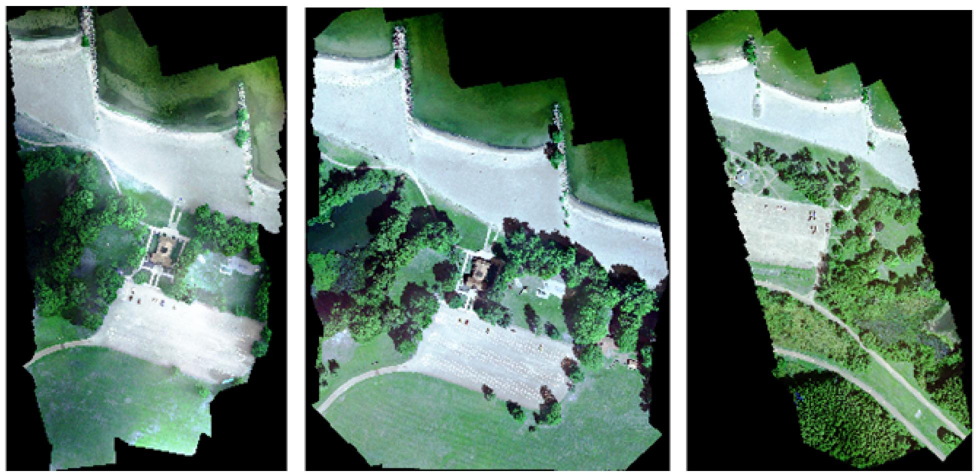

(a) Train (b) Validation (c) Test

Figure 3.2: RGB visualization of RIT-18 dataset. This dataset has six spectral bands.

to evaluate our network initialization scheme comes from a new semantic segmentation dataset that we built called RIT-18[1]. RIT-18 consists of high-resolution MSI (six bands) acquired by an unmanned aircraft system (UAS). The primary use of this dataset is for evaluating semantic segmentation frameworks designed for non-RGB remote sensing imagery. The dataset, shown in Fig. 3.2, is split into training, validation, and testing folds to 1) provide a standard for state-of-the-art comparison, and 2) demonstrate the feasibility of deploying algorithms in a more realistic setting. Baseline results demonstrate that the large spatial variability commonly associated with high-resolution imagery, large sample (pixel) size, small and hidden objects, and unbalanced class distribution make this a difficult dataset to perform well on, making it an excellent dataset for evaluating our DCNN frameworks for semantic segmentation.

**Contributions**: This chapter makes three major contributions: 1) We are the first to adapt recent fully-convolutional DCNNs to semantic segmentation of multispectral remote sensing imagery; 2) We demonstrate that pre-training these networks on synthetic imagery can significantly improve their performance; and 3) We describe the new RIT-18 dataset for evaluating MSI semantic segmentation algorithms.

[1]The dataset is available at `https://github.com/rmkemker/RIT-18`

## 3.1 Related Work

### 3.1.1 Semantic Segmentation of RGB Imagery with Deep Networks

In this chapter, pixel-wise classification and semantic segmentation are synonymous. Semantic segmentation is the term more commonly used in computer vision and is becoming increasingly used in remote sensing. State-of-the-art semantic segmentation frameworks for RGB imagery are trained end-to-end and consist of convolution and segmentation subnetworks. The convolution network is usually a pre-trained DCNN designed to classify images from ImageNet [68, 84, 88, 89], and current state-of-the-art performers use VGG-16 [7] or ResNet [4]. The segmentation network is appended to the convolution network and is designed to reconstruct the feature response to the same spatial dimensions as the input before assigning semantic labels. The resulting semantic segmentation network can be fine-tuned with orders of magnitude fewer training images (thousands versus millions of images) because the convolutional network is already trained. We describe some of the best performing recent models below, all of which used non-aerial RGB scenes.

The first fully-convolutional network (FCN) designed for semantic segmentation [88] used the VGG-16 network [7], which has approximately 138 million parameters. VGG-16 is trained to do image classification on ImageNet [4], rather than directly for semantic segmentation. Their FCN model used coarse upsampling and deconvolution in the segmentation network to classify each pixel. The net's major disadvantage was that VGG-16's 5 max-pooling layers shrunk the original image by a factor of 32, resulting in a coarse label map [88].

In [89], they proposed to improve the FCN model by building a symmetric (deconvolution) network using spatial unpooling and deconvolution layers. This increased performance when classifying objects at multiple resolutions (i.e. small or large objects in the image); however, it still produced a coarse label map. As a post-processing step, the authors used a conditional random field (CRF) to sharpen the classification boundaries [90]. The major downside to this deconvolution network was that it required more memory and time to train compared to [88].

The DeepLab semantic segmentation network [84] was built with the ResNet DCNN. DeepLab mitigates downsampling issues and makes segmentation boundaries sharper by replacing conventional convolution layers with atrous convolutions. An atrous convolution filter is filled with zeros between the sample points; so although the effective size of the filter increases, the number of trainable parameters remains constant. When these filters are convolved with an image, it can preserve the original dimensions. The authors found that using atrous filters throughout the entire network was inefficient, so they used both conventional and atrous filters, reducing the image only by a factor of eight. A dense CRF was used as a post-processing step to make the predicted label map sharper.

Many of these earlier models used a CRF as a post-processing step to sharpen the classification masks, but it may be better to allow the network to directly optimize itself towards creating a sharper label mask. Two recent models that did this, Sharpmask [68] and RefineNet [85], used skip-connections to incorporate image refinement into the end-to-end model. Sharpmask used a refinement module to combine features from the convolution network with the upsampled features from the segmentation network. RefineNet improved the Sharpmask model with multi-resolution fusion (MRF) to combine features at different scales, chained residual pooling (CRP) to capture background context, and residual convolutional units (RCUs) to improve end-to-end training. In this chapter, we adapt the Sharpmask and RefineNet models to multispectral remote sensing imagery and use them to evaluate our proposed initialization procedure. These DCNN algorithms are described in more detail in Sections 3.2.2 and 3.2.2.

### 3.1.2 Deep-Learning for Non-RGB Sensors

Deep learning approaches to classify and analyze RGB remote sensing imagery have advanced considerably thanks to deep learning, including the use of DCNNs pre-trained on ImageNet and unsupervised feature extraction [91–93]. Deep-learning frameworks for the semantic segmentation of multispectral and hyperspectral images have been explored by the remote sensing community; however, the paucity of annotated data available for these sensor modalities has pushed researchers to embrace unsupervised feature extraction methods [94] and object based image analysis [95] (see Section 3.1.3). Deep features extracted from every pixel of the labeled data are then used with a classifier, often a support vector machine, to generate a pixel-wise classification map.

The authors in [96] determined that spatial (texture) information influenced classification performance the most, which is why current state-of-the-art methods extract spatial-spectral features from the image data. Early spatial-spectral feature extractors had hyper-parameters that required tuning (e.g. gray-level co-occurrence matrices [36], Gabor [97], sparse coding [33], extended morphological attribute profiles [1], etc). These hand-crafted features could fail to generalize well across multiple datasets, so they were replaced with learned features that were automatically tuned from the data itself.

Arguably, the most successful of these learned feature extraction methods for remote sensing imagery is the stacked autoencoder [2, 27, 29, 30, 35]. An autoencoder is an unsupervised neural network that learns an efficient encoding of the training data. These autoencoders can be stacked together to learn higher feature representations.

In [2], stacked sparse autoencoders (SSAE) were used to learn feature extracting filters from one hyperspectral image, and then these filters were used to classify the same image as well as a different target image. The SSAE features learned on the one dataset failed to properly transfer to another dataset because these features failed to generalize well. One possible way to overcome this limitation would be to do the unsupervised training on multiple images, which may lead to more discriminative features that generalize between different target datasets.

This idea was explored in [94], where the authors showed that training on large quantities of unlabeled hyperspectral data, which is known as self-taught learning [20], improved classification performance. The authors proposed two different frameworks for semantic segmentation that used self-taught learning: multi-scale independent component analysis (MICA) and stacked convolutional autoencoders (SCAE). SCAE outperformed MICA in their experiments, but both approaches advanced the state-of-the-art across three benchmark datasets, showing that self-taught features can work well for multiple images.

### 3.1.3 Semantic Segmentation of High Resolution Remote Sensing Imagery

As the resolution of available image data increased, researchers explored geographic object based image analysis (GEOBIA) to deal with the high spatial variability in the image data [98]. According to [99], GEOBIA involves the development of "automated methods to partition remote sensing imagery into meaningful image-objects, and assessing their characteristics through spatial, spectral and temporal scales, so as to generate new geographic information in GIS-ready format." The model can generate superpixels through clustering (unsupervised) or segmenting (supervised) techniques, extrapolate information (features) about each superpixel, and use that information to assign the appropriate label to each superpixel. Clustering/Segmenting the image into superpixels could prevent the salt-and-pepper misclassification errors that are characteristic of high-resolution imagery. Another strategy for mitigating this type of error is post-processing the classification map with a Markov Random Field [100] or CRF [101].

Classifying superpixels is another semantic segmentation strategy found in the computer vision literature [102,103], but it is less used with the development of fully-convolutional neural networks for end-to-end semantic segmentation [88]. In contrast, remote sensing researchers have been very successful employing this strategy, via GEOBIA, for segmenting high resolution imagery. Our DCNN approach had multiple advantages: 1) it did not require a time-consuming unsupervised segmentation, 2) it could be trained end-to-end, 3) it increased classification performance, and 4) it can be done much faster. Incorporating a CRF or employing GEOBIA methods could tighten classification map boundaries and reduce salt-and-pepper label noise [90]. In addition, skip-connections that pass low-level features to the segmentation network were shown to tighten classification boundaries and reduce salt-and-pepper errors in end-to-end semantic segmentation frameworks [68, 85]. In our work, we chose to use end-to-end, fully-convolutional networks with skip-connections for classification, but future work could include a CRF and/or GEOBIA methods to further increase performance.

### 3.1.4 MSI Semantic Segmentation Datasets for Remote Sensing

Most remote sensing semantic segmentation datasets have been imaged by airborne or satellite platforms. The existing publicly available MSI datasets are shown in Table 3.1. The gold-standard benchmark for the semantic segmentation of visual near-infrared (VNIR) MSI are the Vaihingen and Potsdam datasets hosted by the International Society for Photogrammetry and Remote Sensing (ISPRS) [81]. These datasets have comparably high spatial resolution, but only five classes. Newer datasets, such as Zurich [82] and EvLab-SS [104], have sacrificed some spatial resolution but include additional labeled classes. Additional competitions hosted by the IEEE Geoscience and Remote Sensing Society (GRSS) [79] and Kaggle [105] involve the fusion of multi-modal imagery captured (or resampled) to different ground sample distances (GSDs).

In this chapter, we describe a new semantic segmentation dataset named RIT-18. Some of the advantages of our dataset over existing benchmarks include:

1. RIT-18 is built from **very-high resolution** (4.7 cm GSD) UAS data. The ISPRS datasets are the only comparable datasets, but these datasets only use 3-4 spectral bands and 5 object classes.

2. RIT-18 has **18 labeled object classes**. The 2017 IEEE GRSS Data Fusion has 17 object classes, but all of the data has been spatially resampled to 100 meter GSD (land-cover classification).

3. RIT-18 has **6 VNIR spectral bands**, including two additional NIR bands not included in the ISPRS benchmarks. These additional bands increase the discriminative power of classification models in vegetation heavy scenes (see Table 3.5.)

4. RIT-18 was **collected by a UAS platform**. A UAS is cheaper and easier to fly than manned aircraft which can allow the end-user to collect high spatial resolution imagery more frequently.

5. RIT-18 is a **very difficult dataset** to perform well on. It has an unbalanced class distribution. A system that is capable of performing well on RIT-18 must be capable of low-shot learning.

| **Dataset** | **Year** | **Sensor(s)** | **GSD** | **Classes** |
|---|---|---|---|---|
| ISPRS Vaihingen [81] | 2012 | Green/Red/IR | 0.09 | 5 |
| ISPRS Potsdam [81] | 2012 | 4-band (VNIR) | 0.05 | 5 |
| Zurich Summer [82] | 2015 | QuickBird | 0.61 | 8 |
| EvLab-SS [104] | 2017 | World-View-2, GF-2, QuickBird, & GeoEye | 0.1-1.0 | 10 |
| GRSS Data Fusion [79] | 2017 | Landsat & Sentinel 2 | 100 | 17 |
| Kaggle Challenge [105] | 2017 | World-View 3 | 0.3-7.5 | 10 |
| **RIT-18** | **2017** | **6-band (VNIR)** | **0.047** | **18** |

Table 3.1: Benchmark MSI semantic segmentation datasets, the year they were released, the sensor that collected it, its ground sample distance (GSD) in meters, and the number of labeled object classes (excluding the background class). Our RIT-18 dataset is in bold.

### 3.1.5 Deep Learning with Synthetic Imagery

Synthetic data has been used to increase the amount of training data for systems that use deep neural networks, and this has been done for many applications, including object detection [106], pose estimation [107], face and hand-writing recognition [108], and semantic segmentation [109]. Past work used various methods to generate large quantities of synthetic data including geometric/color transformations, 3D modeling, and virtual reality emulators.

The major upside to synthetic imagery is that it is normally cheaper and easier to obtain than images that are manually annotated by humans; however, the difference in feature-space distributions, also known as the synthetic gap, can make it difficult to transfer features from synthetic to real imagery. Researchers have adopted domain adaptation techniques [110] to mitigate this phenomenon, including training autoencoders to shift the distribution of the synthetic data to the distribution of the real data [108, 111]. This can allow for a classifier to be trained using synthetic images and then make predictions on real data.

Network fine-tuning, has been widely-adopted in the computer vision community for re-purposing DCNNs trained for image classification for a variety of alternative applications, such as semantic segmentation. Typically, a portion of the network is pre-trained on a large image classification dataset (e.g., ImageNet), and then adapted to a dataset with different class labels and feature distributions. However, it is possible to use synthetic data instead. In [109], the authors built a synthetic dataset using a virtual reality generator for the semantic segmentation of autonomous driving datasets, and then they combined the synthetic and real imagery to train their semantic segmentation model. In our work, we initialized the ResNet-50 DCNN to classify synthetic MSI, and then we fine-tuned these weights to perform semantic segmentation on real MSI.

## 3.2 Methods

In this section, we first describe how we generated synthetic imagery for pre-training DCNNs to classify MSI data. Then we describe two fully convolutional semantic segmentation algorithms, which we adapt for MSI imagery. Lastly, we describe both simple baseline and state-of-the-art algorithms for semantic segmentation for remote sensing imagery, which will serve as a comparison to the FCN models.

### 3.2.1 Synthetic Image Generation using DIRSIG

Because there are no publicly-available ImageNet sized datasets for non-RGB sensor modalities, we used DIRSIG to build a large synthetic labeled dataset for the semantic segmentation of aerial scenes. DIRSIG is a software tool used heavily in the remote sensing industry to model imaging sensors prior to development. It can be used to simulate imaging platforms and sensor designs, including monochromatic, RGB, multispectral, hyperspectral, thermal, and light detection and ranging (LIDAR).

DIRSIG images an object/scene using physics-based radiative transfer/ propagation modeling [112,113]. A realistic object can be defined in the DIRSIG environment with 3D geometry, textures, bi-directional reflection distribution function (BRDF), surface temperature predictions, etc. A custom sensor (e.g. MSI sensor) and platform (e.g. UAS) can

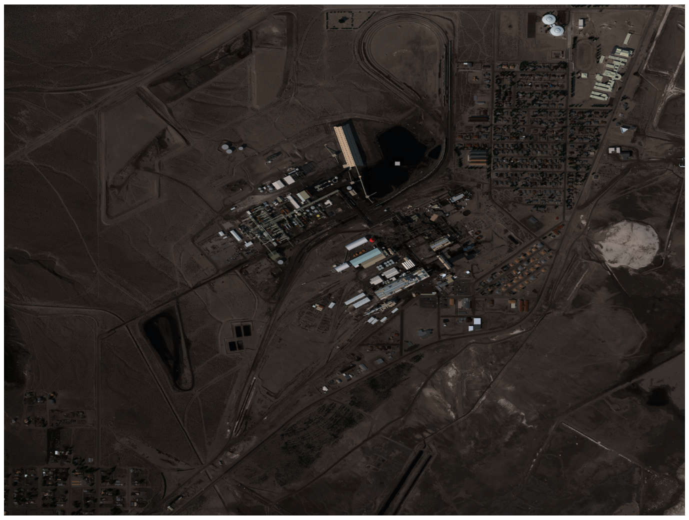


Figure 3.3: The DIRSIG Trona Scene.

be defined in the DIRSIG environment to “fly-over” a scene and image it. The position of the sun/moon, the atmosphere profile, and the flight plan can all be modified to generate realistic data as well as provide a pixel-wise classification map.

We used the synthetic scene shown in Fig. 3.3, which resembles Trona, an unincorporated area in Southern California. It is an industrial and residential scene containing 109 labeled classes, including buildings, vehicles, swimming pools, terrain, and desert plant life.

Trona is only accurate to a GSD of 0.5 meters, and creating images with smaller GSDs produces image artifacts. The GSD of RIT-18 is about ten times higher; however, Trona was the only available scene that was 1) sufficiently large, 2) had enough object classes, and 3) had an accurate ground truth map. To mitigate this problem, we forced our networks to learn scale-invariant features by generating MSI at multiple GSDs (0.5, 0.75, and 1 meter). This is accomplished by flying the simulated UAS at different elevations.

We “flew” the drone across the entire synthetic scene with some overlap to make sure all objects located at the edges of some images are located near the center of other images. We varied the time-of-year (summer and winter), time-of-day (morning and afternoon), and the corresponding atmospheric conditions (mid-latitude summer and winter). The semantic label for each pixel is the dominant class that lies within the instantaneous field of view of that pixel. We built the final synthetic dataset by breaking the scene into smaller image patches, resulting in 4.7 million training and 520 thousand validation 80x80 MSI. The label for each image is the majority category present.

### 3.2.2 Fully-Convolutional Deep Networks for Semantic Segmentation

There are several ways to sharpen the object boundaries in a classification map. Post-processing techniques such as a fully-connected CRF can be used to eliminate small errors and sharpen object boundaries. GEOBIA methods use the superpixel boundaries to help sharpen the output mask. The authors in [114] combined CNN features with superpixel features to increase the detail in the object boundaries and reduce salt-and-pepper noise. Both CRFs and GEOBIA methods require an additional step to sharpen the output mask which adds additional processing time; and in many cases, requires tuning of additional hyperparameters. In our work, we adapted two recent fully-convolutional deep neural networks for the semantic segmentation of MSI: SharpMask [68] and RefineNet [85]. Both of them produced state-of-the-art results on standard semantic segmentation benchmarks in computer vision. The Sharpmask and RefineNet models were designed to learn the mask sharpening operation in their respective end-to-end frameworks without post-processing with a CRF or clustering/segmenting the image. This is done by passing lower level features from the DCNN to the parts of the segmentation network that are responsible for upsampling the feature map. This restores high spatial frequency information (such as object boundaries or other detail) that was lost when the feature map was downsampled.

The DCNN used for both of these models was ResNet-50 [8] with the improved network architecture scheme proposed in [115], where batch-normalization and ReLU are applied prior to each convolution. Both models were implemented using Theano/Keras [70] and trained on computers with a NVIDIA GeForce GTX Titan X (Maxwell) graphical processing unit (GPU), Intel Core i7 processor, and 64 GB of RAM. Our goal is to compare the performance of these algorithms to baseline and state-of-the-art methods for semantic segmentation of remote sensing imagery, and to measure the benefit of pre-training with synthetic imagery.

We describe the architectural details of these two models in this section, but details regarding training of these networks is given in Section 3.4.

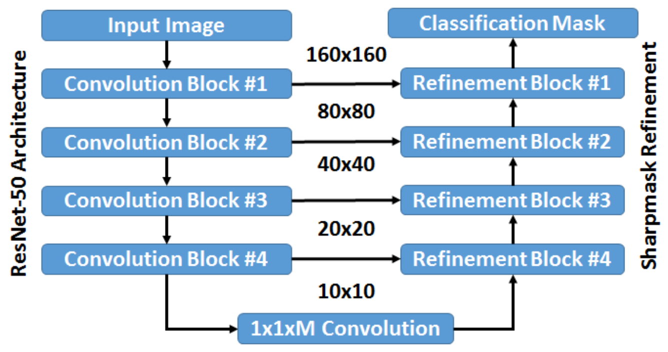


Figure 3.4: Our Sharpmask model. The spatial dimensions of each module's output are provided to illustrate that the refinement layer combines features from the feed-forward network (convolutional blocks) and the upsampled features in the reverse network to restore the original image dimensions. The output is a class probability matrix for each pixel.

### SharpMask

The Sharpmask model [68] used for this chapter is illustrated in Fig. 3.4. The network is broken into the convolution, bridge, and segmentation sub-networks. Note that Sharpmask does not use all of the ResNet-50 model. As shown in Fig. 3.4, it only uses the first four macro-layers. A macro-layer contains the convolution, batch-normalization, and ReLU activation layers right up to where the feature map is down-sampled by a factor of 2x. This corresponds to the first 40 ResNet-50 convolution layers. SharpMask was developed by Facebook to be lightweight and fast, possibly for deployment on a platform where size, weight, and power (SWaP) constraints are limited (e.g. UAS, embedded platform). They tested and compared the classification performance and prediction time for models with various capacities including models that used only the first three and four ResNet macro-layers. This trade-off study showed that SharpMask with four macro-layers provided the desired classification accuracy while also providing a 3x speed-up to their previous state-of-the-art DeepMask model[116]. We slightly modified the SharpMask model to retain the batch normalization layers for regularization.

The bridge network is a $M \times 1 \times 1$ convolution layer between the convolution and segmentation networks, where $M$ is selected as a trade-off between performance and speed. The main goal of this network is to add some variability to the features fed into

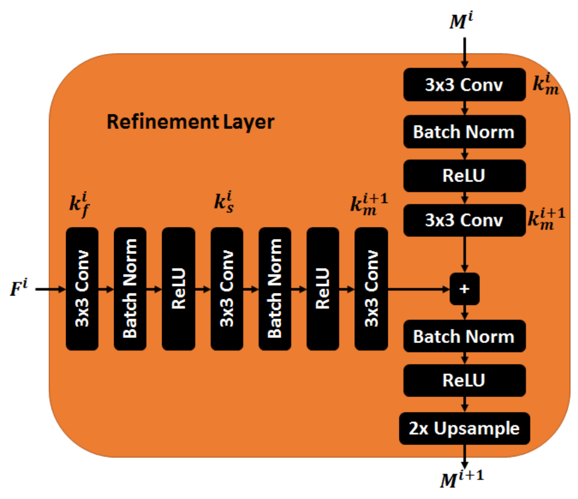


Figure 3.5: Sharpmask refinement with batch normalization. This layer learns mask refinement by merging features from the convolution $F^i$ and segmentation $M^i$ networks.

the segmentation network at refinement module #4. We use a value of $M = 512$, which worked well in preliminary experiments.

The segmentation network uses refinement modules to restore the bridge layer output to the original dimensionality of the input data. Instead of using a fully-connected CRF, the segmentation sharpening was learned as a part of the end-to-end network. The refinement module merges low-level spatial features $F_i$ from the convolution network with high-level semantic content in the segmentation network $M_i$, as illustrated in Fig. 3.5.

Each refinement module uses convolution and sum-wise merge layers prior to upsampling the feature response. The upsampled feature response is fed into the refinement module at the next higher dimension. The number of filters used in the $i$-th refinement module are given by

$$k_s^i = k_m^i = \frac{128}{2^{i-1}}, i \leq 4. \tag{3.1}$$

These parameters differ slightly from [68] because the higher dimensionality of RIT-18 required a slightly larger model capacity.

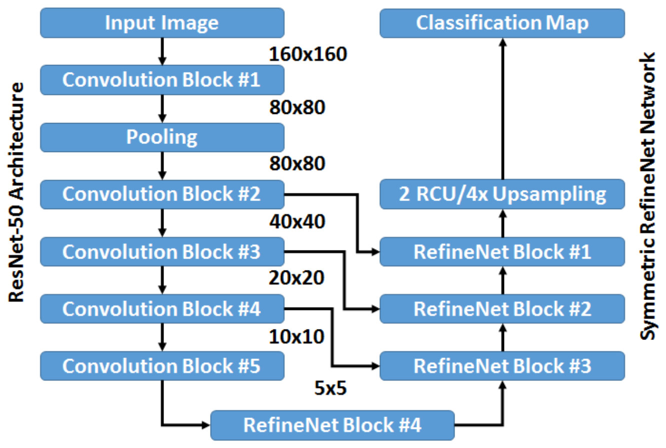
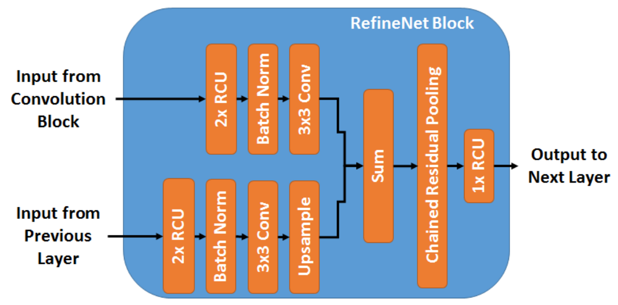


Figure 3.6: Network Architecture incorporating RefineNet modules for semantic segmentation.

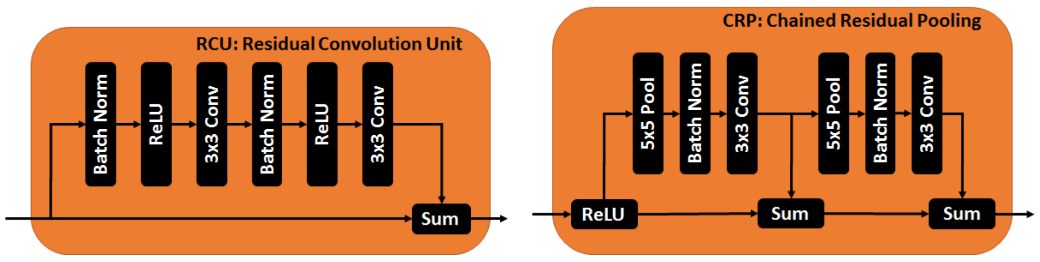
## RefineNet

The RefineNet model [85] used in this chapter (Fig. 3.6) follows the same basic structure as the Sharpmask model with a few minor changes. The refinement block in Fig. 3.5 is replaced with a more complex block called RefineNet (Fig. 3.7(a)), which is broken up into three main components: residual convolution units (RCUs), multi-resolution fusion (MRF), and chained residual pooling (CRP). Our model uses batch normalization for regularization. The convolutional network uses all five ResNet-50 macro-layers, so it is using every convolution layer in ResNet-50 except for the softmax classifier.

The RCUs (Fig. 3.7(b)) are used to propagate the gradient across short- and long-range connections, which makes end-to-end training more effective and efficient. The MRF is used to combine features at multiple scales, which in our work, will be two. The CRP module (Fig. 3.7(c)) pools features across multiple window sizes to capture background context, which is important for discriminating classes that are spatially and spectrally similar. Fig. 3.7(c) uses two window sizes to illustrate how CRP works; however, our model pools features across four window sizes.



(a) RefineNet Block



(b) Residual Convolution Unit

(c) Chained Residual Pooling with two window sizes

Figure 3.7: RefineNet architecture.

## 3.2.3 Comparison Semantic Segmentation Algorithms

Because RIT-18 is a new dataset, we compared the two FCN models to a number of classic approaches, including classifying individual or mean-pooled pixels. We also used two spatial-spectral feature extraction methods, MICA and SCAE, which recently achieved state-of-the-art performance on the semantic segmentation of hyperspectral imagery. These methods use unsupervised learning to acquire spatial-spectral feature representations making them sample efficient. As another baseline, we also used a method inspired by GEOBIA.

### Simple Classification Algorithms

One simple approach to semantic segmentation that has been widely used in remote sensing is running a classifier directly on each individual pixel. Using this approach, we establish baseline results using three different classifiers: $k$-nearest neighbor (kNN), linear support vector machine (SVM), and multi-layer perceptron (MLP). We also used spatial mean-pooling (MP) as a simple way of incorporating neighboring spatial information into the classifier.

The kNN classifier used the Euclidean distance metric and we cross-validated for $k$ over the range of 1-15.

For the linear SVM, we used the LIBLINEAR implementation [117], which works well with large datasets. During training, it uses L2 regularization and adopts the one-vs-rest paradigm to multi-class classification. The input was scaled to zero-mean/unit-variance, where the mean and standard deviation were computed using the training data. The SVM used RIT-18's validation set to cross-validate for its cost parameter $C$ over the range of $2^{-9} - 2^{16}$, and then the training and validation sets were combined to fit the final model. The loss function was weighted by the inverse class frequency.

Our MLP implementation is a fully-connected neural network with a single hidden-layer of 64 units, chosen through cross-validation. This hidden layer is preceded by a batch-normalization layer and followed by a ReLU activation. To compensate for class unbalance, we assign each class distinct weights, which are given for class $i$ by

$$w_i = \mu * \log_{10} \frac{\sum_1^N h_i}{h_i}, \tag{3.2}$$

where $h_i$ is the number of pixels labeled as class $i$, $N$ is the number of classes ($N = 18$), and $\mu$ is a tunable parameter. The MLP was trained using the NAdam optimizer [3], with a batch size of 256, L2 regularization value of $10^{-4}$ in the convolution and batch normalization layers, and the class weighted update in Equation 3.2 where $\mu = 0.15$.

Running classifiers on individual pixels ignores neighboring pixel information. This will negatively impact performance due to the high spatial variability commonplace in high resolution imagery. To address this, we also test the MP model. We convolve the original data with a $5 \times 5$ mean-pooling filter and then pass the response to the same Linear SVM described earlier.

## MICA

MICA achieved excellent performance at semantic segmentation of HSI data [94], and we use it here as one of our baseline algorithms. MICA uses the unsupervised learning algorithm independent component analysis (ICA) to learn a spatial-spectral filter bank from images captured by the sensor. The filters that it acquires exhibit color opponency and resemble Gabor-type (bar/edge) filters. These filters are then convolved with the image, like a single layer convolutional neural network, and their responses are normalized using a non-linear activation function. These responses are then pooled to incorporate translation invariance, and then a classifier is applied to each of the pooled responses. While the work in Chapter 2 used an RBF-SVM to classify these responses, that is not feasible due to the size of RIT-18. Instead, we fed these responses into an MLP classifier.

The MICA model used in this chapter had $F = 64$ learned filters of size $25 \times 25 \times 6$ and a mean pooling window of $13$. The MLP model used to classify the final MICA feature responses used one hidden layer with 256 ReLU units and a softmax output layer. The MICA filters were trained with 30,000 $25 \times 25$ image patches randomly sampled from the training and validation folds. Additional details for the MICA algorithm can be found in [94].

### SCAE

SCAE is another unsupervised spatial-spectral feature extractor that achieved excellent results on the semantic segmentation of hyperspectral imagery [94]. SCAE extracts features using stacked convolutional autoencoders, which are pre-trained using unsupervised learning. SCAE has a deeper neural network architecture than MICA and is capable of extracting higher-level semantic information. The SCAE model used in this chapter is almost identical to the model proposed in [94] with a few notable exceptions. First, SCAE was trained with 30,000 $128 \times 128$ image patches randomly extracted from the training and validation datasets. This increase in receptive field size was done to compensate for higher GSD imagery, which assisted the model in learning local relationships between object classes.

Second, the network capacity of each convolutional autoencoder (CAE) was decreased to compensate for the reduced dimensionality of RIT-18 (only six bands). For each CAE, there are 32 units in the first convolution block, 64 units in the second convolution block, 128 units in the third, and 256 units in the $1 \times 1$ convolution. The refinement blocks have the same number of units as their corresponding convolution block, so the last hidden layer of each CAE has 32 features.

Third, as was done with MICA, the RBF-SVM classifier was replaced with a MLP classifier in order to speed up training. An entire orthomosaic is passed through the SCAE network to generate three $N \times 32$ feature responses. These feature responses are concatenated, convolved with a $5 \times 5$ mean-pooling filter, and then reduced to 99$\%$ of the original variance using whitened principal component analysis (WPCA). The final feature response is passed to a MLP classifier with the same architecture used by the MICA model in Section 3.2.3.

We also introduce a multi-resolution segmentation (MRS) method that uses SCAE features. We combined object-based features with learned spatial-spectral features extracted from the SCAE deep learning framework. For object-based features, we hyper-segmented the orthomosaic images using the mean-shift clustering algorithm. These orthomosaics were hyper-segmented at different scales to improve the segmentation of objects at different sizes. Mean-shift was chosen because it automatically determines how many clusters should be in the output image. The inputs to the mean-shift algorithm were the six spectral channels, the pixel location, and computed normalized-difference vegetation index (NDVI) for each pixel. The object based features for the pixels in each cluster are the area and both spatial dimensions for each cluster. We denote this method MRS+SCAE.

## 3.3 RIT-18 Dataset

RIT-18 is a high-resolution (4.7 cm) benchmark, designed to evaluate the semantic segmentation of MSI, collected by a UAS. UAS collection of non-RGB imagery has grown in popularity, especially in precision agriculture, because it is more cost effective than manned flights and provides better spatial resolution than satellite imagery. This cost savings allows the user to collect data more frequently, which increases the temporal resolution of the data as well. The applications for UAS with MSI payloads include crop health sensing, variable-rate nutrient application prescription, irrigation engineering, and crop-field variability [118].

### 3.3.1 Collection Site

The imagery for this dataset was collected at Hamlin Beach State Park, located along the coast of Lake Ontario in Hamlin, NY. The training and validation data was collected at one location, and the test data was collected at a different location in the park. These two locations are unique, but they share many of the same class-types. Table 3.2 lists several other collection parameters.

| | **Train** | **Validation** | **Test** |
|---|---|---|---|
| **Date** | 29 Aug 16 | 6 Sep 16 | 29 Aug 16 |
| **Time (UTC)** | 13:37 | 15:18 | 14:49 |
| **Weather** | | Sunny, clear skies | |
| **Solar Azimuth** | 109.65° | 138.38° | 126.91° |
| **Solar Elevation** | 32.12° | 45.37° | 43.62° |

Table 3.2: Collection parameters for training, validation, and testing folds for RIT-18.

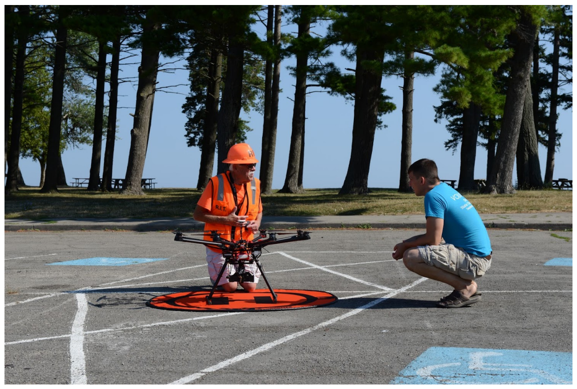


Figure 3.8: Tetracam Micro-MCA6 mounted on-board the DJI-S1000 octocopter prior to collection.

### 3.3.2 Collection Equipment

The equipment used to build this dataset and information about the flight is listed in Table 3.3. The Tetracam Micro-MCA6 MSI sensor has six independent optical systems with bandpass filters centered across the VNIR spectrum. The Micro-MCA6 has been used on-board UASs to perform vegetation classification on orthomosaic imagery [80] and assess crop stress by measuring the variability in chlorophyll fluorescence [119] and through the acquisition of other biophysical parameters [120]. Fig. 3.8 shows an image of the Micro-MCA6 mounted on-board the DJI-S1000 octocopter.

| **Imaging System** | |
|---|---|
| Manufacturer/Model | Tetracam Micro-MCA6 |
| Spectral Range [nm] | 490-900 |
| Spectral Bands | 6 |
| RGB Band Centers [nm] | 490/550/680 |
| NIR Band Centers [nm] | 720/800/900 |
| Spectral f [nm] | 10 (Bands 1-5) |
| | 20 (Band 6) |
| Sensor Form Factor [pix] | 1280x1024 |
| Pixel Pitch [$\mu$m] | 5.2 |
| Focal Length [mm] | 9.6 |
| Bit Resolution | 10-bit |
| Shutter | Global Shutter |
| **Flight** | |
| Elevation [m] | 120 (AGL) |
| Speed [m/s] | 5 |
| Ground Field of View [m] | $\approx$60x48 |
| GSD [cm] | 4.7 |
| Collection Rate [images/sec] | 1 |

Table 3.3: Data Collection Specifications

### 3.3.3 Dataset Statistics and Organization

The RIT-18 dataset (Fig. 3.2) is split up into training, validation, and testing folds. Each fold contains an orthomosaic image and corresponding classification map. Each orthomosaic contains the six-band image described in Section A along with a mask where the image data is valid. The spatial dimensionality of each orthomosaic image is 9,393$\times$5,642 (train), 8,833$\times$6,918 (validation), and 12,446$\times$7,654 (test).

Figure 3.9 lists the 18 class labels in RIT-18 and their corresponding color map (used throughout this chapter). Each orthomosaic was hand-annotated using the region-of-interest (ROI) tool in ENVI. Several individuals took part in the labeling process. More information about these classes can be found in B.

The class-labeled instances are, as illustrated in Fig. 3.10, orders of magnitude dif-

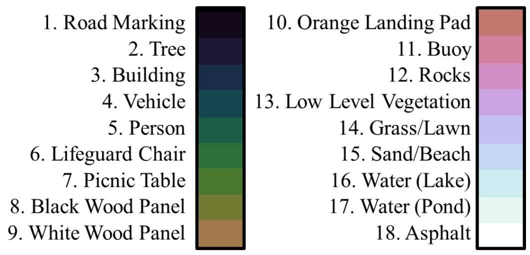


Figure 3.9: Class labels for RIT-18.

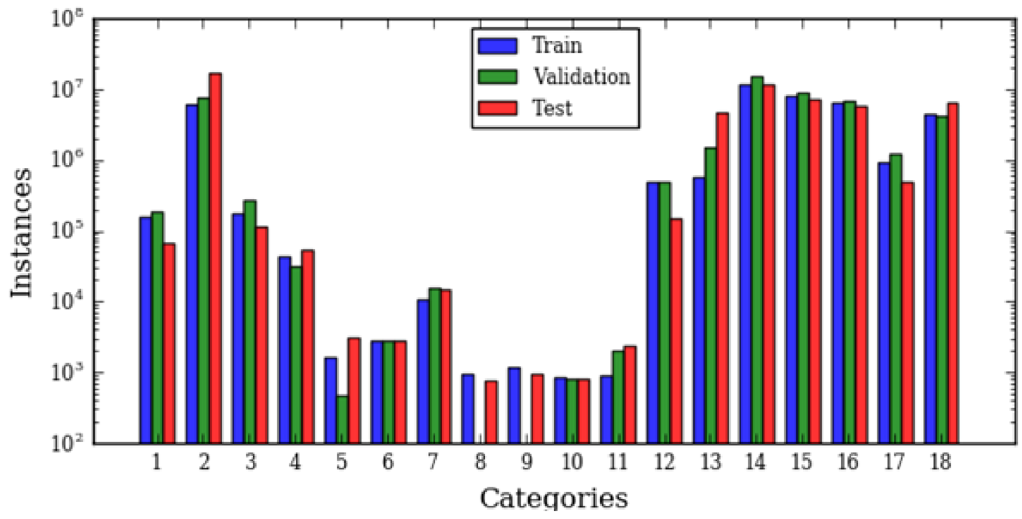


Figure 3.10: Class-label instances for the RIT-18 dataset. Note: The y-axis is logarithmic to account for the number disparity among labels.

ferent from one-another. These underrepresented classes should make this dataset more challenging.

## 3.4 FCN Training Procedures

### 3.4.1 Pre-Training the ResNet-50 DCNN on Synthetic MSI

The two FCN models are assessed with and without DCNNs that are pre-trained on synthetic DIRSIG imagery. For the pre-trained model, the ResNet-50 DCNN was initialized using 80$\times$80 pixel patches, which is approximately 4.7 million DIRSIG training images. The network was trained using a mini-batch size of 128 and a weight-decay of 1e-4. The weights were randomly initialized from a zero-mean normal distribution. The labels provided by DIRSIG are a class-label for each pixel; however, the ResNet-50 DCNN is trained to perform image classification. We assigned a label to each image patch based on the most common label.

We compute the channel-mean and standard-deviation using the entire DIRSIG training set, and these parameters were used to scale each image to zero-mean/unit-variance. During training, the images are shuffled for each epoch, and we use random horizontal and vertical flips for data augmentation. Because the class distribution for the DIRSIG data is unbalanced, we use the class-weights $w_i$ for the entire training set to build sample (per-pixel) weights. The class-weights were assigned using Equation 3.2 with $\mu = 0.15$. We optimized the network using Nadam with an initial learning rate of 2e-3. We then dropped the learning rate when the validation loss plateaued.

### 3.4.2 DCNN Fine-Tuning

We trained both semantic segmentation frameworks using randomly initialized DCNNs and with the DCNN pre-trained on the DIRSIG dataset. Because of the high-resolution of each orthomosaic, the data is broken into $160 \times 160$ patches and fed to the segmentation frameworks. The randomly initialized models are trained end-to-end in one stage because the filters of its DCNN need to be tuned. All weights that do not come from the pre-trained DCNN are randomly initialized from a zero-mean Gaussian distribution. For the models that did not use pre-training, an initial learning rate of 2e-3 is used, and then the learning rate is dropped by a factor of 10 four times as the validation loss plateaued.

For the models that use the pre-trained DCNN, we train them in two stages. First, the pre-trained portion of the model is frozen and the remaining layers are trained to adapt the weights of the segmentation network to the pre-trained weights in the convolution network. An initial learning rate of 2e-3 is used for this stage, and then it is dropped by a factor of 10 when validation loss plateaus. Second, we fine-tune both the pre-trained portion and the segmentation network jointly using an initial learning rate of 2e-5; and again, drop it by a factor of 10 four times as the validation loss plateaued.

All models are optimized using the Nadam optimizer, batch size of 32, weight decay of 1e-4, and class-weight parameter of $\mu = 0.25$, which optimizes for AA. Plots for the training and validation accuracy and loss, for both FCN models, are provided in D. We observed that the pre-trained models, especially RefineNet, saturated at the peak accuracy more quickly (fewer epochs) than the randomly initialized networks because most of the weights are close to their final solution. The fine-tuned models continue to make minor improvements over a longer period of time because it employs a lower learning rate to prevent from completely overwriting the pre-trained DIRSIG weights; whereas, the random weight initialization stops training more quickly because it is not attempting to preserve any pre-trained weights.

## 3.5 Experimental Results

### 3.5.1 RIT-18 Results

Results for all algorithms on the RIT-18 test set are listed in Table 3.4. The table shows the classification performance of our DCNN segmentation models, with (Sim) and without (Rdm) initializing the network using synthetic data. Pretraining the ResNet-50 DCNN on the synthetic imagery required three weeks on our GPU. The number of batch updates performed which was higher than the number performed in [35]. The network fine-tuning operation took 17.1 hours for Sharpmask and 56.7 hours for RefineNet. Each algorithm is evaluated on per-class accuracy and mean-class accuracy (AA). AA is used as our primary metric due to the disparity in class distribution, and all algorithms except kNN were trained to compensate for class unbalance. Random chance for assigning a class to a given pixel is 5.6%. Models that perform poorly on some classes, especially classes that perform worse than chance, have over-fit to perform better on classes with more training samples.

When synthetic image initialization was used, both Sharpmask and RefineNet outperform all of the other algorithms in mean-class accuracy performance, demonstrating that advanced techniques used in computer vision can be effectively used with remote sensing MSI data. For mean-class accuracy, the best performing model is RefineNet-Sim. Both of the FCN models that do not use pre-training perform better on classes with more samples, but the mean-class accuracy shows that these models are not as discriminative as their counterparts that are initialized with synthetic data. RefineNet-Rdm overfit to the classes with the most samples, whereas RefineNet-Sim was more discriminative with the pre-trained ResNet-50 weights.

Sharpmask-Rdm performed only slightly worse than Sharpmask-Sim, whereas RefineNet-Sim's results were far greater than RefineNet-Rdm. This discrepancy is likely because RefineNet has 5.8 times as many trainable parameters compared to Sharpmask. Sharpmask is a relatively shallow semantic segmentation framework, it only uses the early layers of ResNet-50, and it only has 11.9 million trainable parameters. In comparison, RefineNet has 69 million trainable parameters. The more parameters a model has the greater its capacity, but more parameters also mean more labeled data is needed to prevent overfitting. We suspect that if a deeper pre-trained network was used, e.g., ResNet-152, then we would see even greater differences between pre-trained and randomly initialized models.

Comparing the baseline models, we see that MICA yielded a 4.8 percent increase in mean-class accuracy from the simpler MP experiment, demonstrating that unsupervised feature extraction can boost classification performance for high-resolution imagery. Unlike earlier work [94], MICA outperformed SCAE showing that low-level features over a larger receptive field could be more important than higher-level features over a smaller spatial extent. MRS+SCAE improved performance over SCAE for objects where the confusion between class predictions was based on the object's size, which means that object based methods, such as CRF or GEOBIA, could increase segmentation performance in future frameworks, especially when the dominant source of error is salt-and-pepper label errors.

The main issue with unsupervised feature extraction methods like SCAE is that they are unable to learn object-specific features without image annotations and will instead learn the dominant spatial-spectral features present in the training imagery (i.e., they may miss small objects and other classes not commonly present). In contrast, supervised DCNN frameworks, if properly trained and regularized, can better learn the spatial-spectral feature representation for each object class; however, these models need to be pre-trained with large annotated datasets to generalize well in the first place. This is why the end-to-end FCN frameworks pre-trained with synthetic imagery is important for the remote sensing community. It is entirely possible to train these FCN models with object based features in the end-to-end framework, but this comes with an additional computation burden in contrast to end-to-end FCN models that *learn* to sharpen object boundaries.

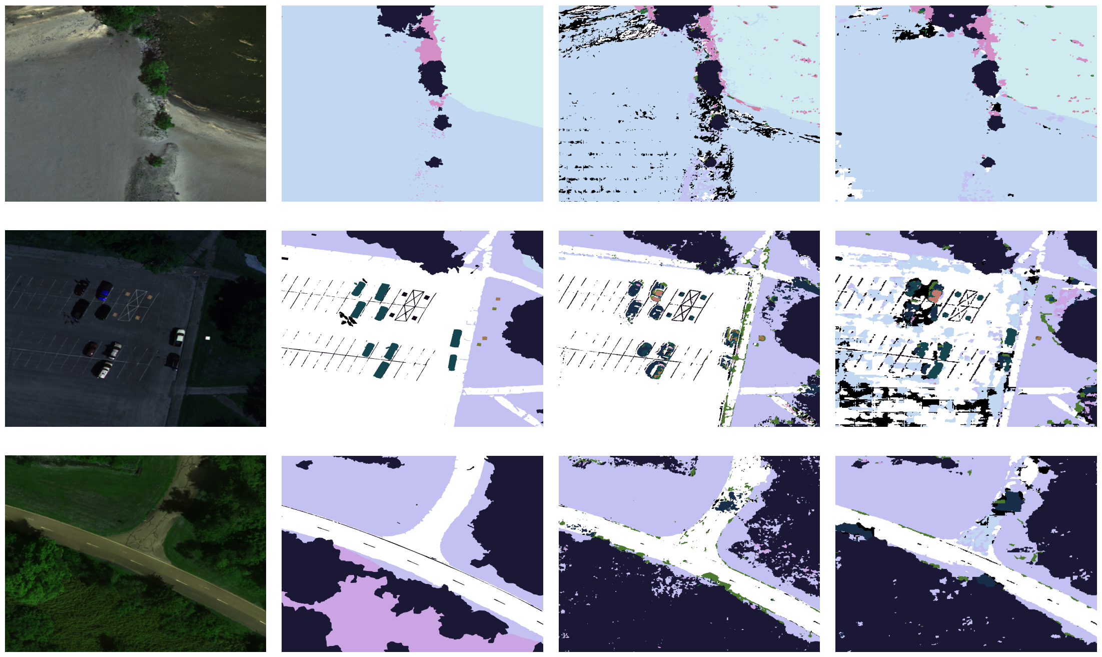
The FCN models failed to classify the black panel, low-level vegetation, and the pond. According to the confusion matrices (D), the black-panel was predominantly misclassified as asphalt, likely because 1) black-panel had very few training samples and 2) they shared

| | kNN | SVM | MLP | MP | MICA | SCAE | MRS+ SCAE | Sharpmask Rdm | Sharpmask Sim | RefineNet Rdm | RefineNet Sim |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Road Markings** | 65.1 | 51.0 | 75.6 | 29.6 | 43.2 | 37.0 | 63.3 | 78.3 | **92.5** | 0.0 | 59.3 |
| **Tree** | 71.0 | 43.5 | 62.1 | 44.1 | 92.6 | 62.0 | 90.5 | **93.0** | 89.8 | 91.0 | 89.8 |
| **Building** | 0.3 | 1.5 | 3.7 | 0.6 | 1.0 | 11.1 | 0.7 | 6.3 | 16.5 | 0.0 | **17.0** |
| **Vehicle** | 15.8 | 0.2 | 1.0 | 0.2 | 47.5 | 11.8 | 53.6 | 51.7 | 20.8 | 0.0 | **61.9** |
| **Person** | 0.1 | 19.9 | 0.0 | 31.2 | 0.0 | 0.0 | 2.3 | 29.9 | 14.9 | 0.0 | **36.8** |
| **Lifeguard Chair** | 1.0 | 22.9 | 3.1 | 16.9 | 50.8 | 29.4 | 2.8 | 44.5 | **85.1** | 0.0 | 80.6 |
| **Picnic Table** | 0.6 | 0.8 | 0.0 | 0.6 | 0.0 | 0.0 | 30.3 | 11.4 | 32.9 | 0.0 | **62.4** |
| **Black Panel** | 0.0 | **48.3** | 0.0 | 47.9 | 0.0 | 0.0 | 18.9 | 0.0 | 0.0 | 0.0 | 0.0 |
| **White Panel** | 0.1 | 0.3 | 0.0 | 0.8 | 0.0 | 0.4 | 0.0 | 4.6 | 9.8 | 0.0 | **47.7** |
| **Orange Pad** | 14.6 | 15.2 | 77.4 | 22.1 | 66.3 | 99.3 | 0.0 | **100.0** | 97.1 | 0.0 | **100.0** |
| **Buoy** | 3.6 | 0.7 | 1.8 | 10.1 | 0.0 | 7.2 | 55.6 | 71.0 | 82.4 | 0.0 | **85.8** |
| **Rocks** | 34.0 | 20.8 | 38.8 | 33.4 | 66.5 | 36.0 | 42.8 | 79.0 | **87.8** | 87.3 | 81.2 |
| **Low Vegetation** | 2.3 | 0.4 | 0.3 | 0.1 | 13.3 | 1.1 | 5.1 | **22.9** | 0.5 | 0.0 | 1.2 |
| **Grass/Lawn** | 79.2 | 71.0 | 85.4 | 73.1 | 84.8 | 84.7 | 83.6 | 84.8 | 87.4 | 88.4 | **90.3** |
| **Sand/Beach** | 56.1 | 89.5 | 36.4 | **95.2** | 78.2 | 85.3 | 8.0 | 73.4 | 91.2 | 6.8 | 92.2 |
| **Water (Lake)** | 83.6 | 94.3 | 92.6 | 94.6 | 89.1 | **97.5** | 90.0 | 96.2 | 93.3 | 90.4 | 93.2 |
| **Water (Pond)** | 0.0 | 0.0 | 0.0 | 0.2 | **3.4** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **Asphalt** | 80.0 | 82.7 | 93.1 | 93.3 | 46.9 | 59.8 | 72.0 | **96.2** | 92.1 | 95.9 | 77.8 |
| **AA** | 27.7 | 29.6 | 30.4 | 31.3 | 36.2 | 32.1 | 34.4 | 52.4 | 57.3 | 30.1 | **59.8** |

Table 3.4: Per-class accuracies as well as mean-class accuracy (AA) on the RIT-18 test set. The two initializations used for our Sharpmask and RefineNet models include random initialization (Rdm) and a network initialized with synthetic data (Sim). We compare our results against the benchmark classification frameworks listed in Section 3.5.

similar spatial/spectral characteristics in the VNIR spectrum. The low-level vegetation was misclassified as trees and the pond water was misclassified as grass/lawn for similar reasons. Large portions of the pond, especially in the test image, contain some vegetation in and on-top of the water which is why it could have been confused for grass. A possible solution to correcting this problem is to initialize the DCNN with DIRSIG imagery that replicates these conditions (e.g., vegetation in a body of water).

Fig. 3.11 shows a sample of the predictions made by our Sharpmask-Sim and RefineNet-Sim frameworks. Sharpmask does a better job at classifying the road, road markings, and vehicles; and RefineNet had fewer classification artifacts over the beach



(a) Test Image (b) Ground Truth (c) Sharpmask (d) RefineNet

Figure 3.11: Experimental results for Sharpmask and RefineNet models. These images are small patches taken from the test orthomosaic.

area. This shows that certain model architectures can be robust to illumination invariance in rough surfaces caused by BRDF effects. RefineNet is likely more robust because it extracts features from deeper convolutional layers and the chained residual pooling aids in capturing background context. Both models did a good job classifying the grass and, for the most part, the lake; however, the low-level vegetation area seemed to be mis-classified as trees - which has orders of magnitude more training samples. Sharpmask and RefineNet would also tend to mis-classify the parts of Lake Ontario, where the wave-crest is whiter than the rest of the body of water, as rocks. This is likely because the rocks in RIT-18 are dominantly surrounded by darker lake water, so the DCNN models have been trained to associate the relatively brighter patches in the lake as rocks. This could be remedied during training by revisiting areas in the lake that have a higher spatial variability or collecting additional examples where the white wave-crests occur.

| | **AA** |
|---|---|
| **SVM-RGB** | 19.9 |
| **SVM-NIR** | 26.0 |
| **SVM-CIR** | 25.3 |
| **SVM-VNIR** | 25.7 |
| **SVM** | **29.6** |

Table 3.5: The effect of band-selection on mean-class accuracy (AA). For comparison, the last entry is the experiment from Table 3.4 which used all six-bands.

### 3.5.2 Band Analysis for RIT-18

The focus of our work is on semantic segmentation of MSI data using FCN models with and without pre-training, but how useful are the non-RGB bands? To assess this, we conducted two experiments using RIT-18. First, we trained the SVM model on different channels. This analysis, shown in Table 3.5, includes only the RGB bands (SVM-RGB), only the three NIR bands (SVM-NIR), a false-color image (SVM-CIR), and a four band RGB-NIR (SVM-VNIR). The 720 nm band was used for the SVM-CIR and SVM-VNIR experiments.

This analysis suggests that the additional NIR bands have a large impact on performance; consistent with the fact that most of the scene is vegetation. To further test this hypothesis, we pre-trained ResNet-50 using 4-band DIRSIG data, and then it was used to fine-tune RefineNet on the first four spectral channels of RIT-18. The results of the 4-band model compared to the full 6-band model are shown in Table 3.6. These results indicate that the Micro-MCA6 increase classification performance compared to simpler 4-band MSI systems. The 4-band solution overfits to the dominant classes in RIT-18. This type of sensor could, in the future, provide an option for the fine-grained classification of various plant life.

## 3.6 Discussion

In this chapter, we demonstrated the utility of FCN architectures for the semantic segmentation of remote sensing MSI. An end-to-end segmentation model, which uses a combination of convolution and pooling operations, is capable of learning global relationships

| | **4-Band** | **6-Band** |
|---|---|---|
| **Road Markings** | 0.0 | **59.3** |
| **Tree** | **95.8** | 89.8 |
| **Building** | 0.0 | **17.0** |
| **Vehicle** | 0.0 | **61.9** |
| **Person** | 0.0 | **36.8** |
| **Lifeguard Chair** | 0.0 | **80.6** |
| **Picnic Table** | 0.0 | **62.4** |
| **Black Panel** | 0.0 | 0.0 |
| **White Panel** | 0.0 | **47.7** |
| **Orange Pad** | 0.0 | **100.0** |
| **Buoy** | 0.0 | **85.8** |
| **Rocks** | 0.0 | **81.2** |
| **Low Vegetation** | 0.0 | **1.2** |
| **Grass/Lawn** | 82.8 | **90.3** |
| **Sand/Beach** | 18.4 | **92.2** |
| **Water (Lake)** | 84.0 | **93.2** |
| **Water (Pond)** | 0.0 | 0.0 |
| **Asphalt** | 6.9 | **77.8** |
| **AA** | 15.2 | **59.8** |

Table 3.6: Performance of RefineNet-Sim on the RIT-18 test set using 4-band (VNIR) and the full 6-band images. These results include per-class accuracies and mean-class accuracy (AA).

between object classes more efficiently than traditional classification methods. Table 3.4 showed that an end-to-end semantic segmentation framework provided superior classification performance on fourteen of the eighteen classes in RIT-18, demonstrating that the learned features in supervised DCNN frameworks are more discriminative than features built from unsupervised learning methods.

We showed that generated synthetic imagery can be used to effectively initialize DCNN architectures to offset the absence of large quantities of annotated image data. Models that were initialized randomly showed degraded mean-class accuracy, a good metric for datasets with unbalanced class distributions.

DIRSIG could be used to generate large custom datasets for other imaging modalities such as multispectral, hyperspectral, LIDAR, or a combination of all the above. These types of scenes could be developed thanks to the increase in UAS collection of remote sensing data. Our work could be adapted to these sensors. Evolving from multispectral to hyperspectral data will likely require modifications to our current models in order to deal with the higher dimensionality. Initializing networks using DIRSIG could improve, not only semantic segmentation, but also networks for other tasks, such as object detection or target tracking in hyperspectral imagery.

Finally, we introduced the RIT-18 dataset as a benchmark for the semantic segmentation of MSI. This dataset benefits from higher spatial resolution, large number of object classes, and wider spectral coverage to improve performance in vegetation-heavy scenes. Datasets that are built from UAS platforms could be more practical for any commercial or research purpose. The orthomosaic generation pipeline provided in A could be utilized to quickly generate remote sensing products with little intervention.

The absolute accuracy of the orthomosaic images in RIT-18 are limited to 10 feet due to the accuracy of the on-board GPS. This keeps us from being able to overlay this dataset with other imagery; however, this will not affect our semantic segmentation results since the images have the same GSD and are registered relative to one another. Our lab has recently acquired a higher-accuracy GPS and inertial navigation system (INS) to make overlaying multiple sensors (e.g., LIDAR, multispectral, and hyperspectral) possible for future collections. In addition, the ability to quickly and accurately register multi-modal remote sensing data could enable the construction of larger DIRSIG scenes with improved spatial and spectral resolution. The increase in quantity and quality of synthetic imagery could improve the network initialization scheme proposed in our work for any sensor modality.

## 3.7 Conclusion

We have shown that synthetic imagery can be used to assist the training of end-to-end semantic segmentation frameworks when there there is not enough annotated image data. Our network initialization scheme has been shown to increase semantic segmentation performance when compared to traditional classifiers and unsupervised feature extraction techniques. The features learned from the synthetic data successfully transfered to real-world imagery and prevented our RefineNet-Sim model from overfitting during training. This work will enable remote sensing researchers to take advantage of advancements in deep-learning that were not previously available to them due to the lack of annotated image data.

In addition, we have introduced the RIT-18 dataset as an improved and more challenging benchmark for the semantic segmentation of MSI. We will make this data available on the IEEE GRSS evaluation server in order to standardize the evaluation of new semantic segmentation frameworks. Although not the largest, RIT-18 is more practical because UAS platforms are easier/cheaper to fly. RIT-18 is difficult to perform well on because of the high-spatial variability and unbalanced class distribution.

In the future, we hope to improve our model by 1) exploring deeper ResNet models; 2) using newer state-of-the-art convolution (ResNeXt [121]) and segmentation models; 3) improving the inherent GSD of the DIRSIG scene; and 4) including additional diverse classes to the synthetic data. The DCNN models we explored still produce classification maps with some salt-and-pepper label noise. This could be remedied in future work by newer end-to-end DCNN segmentation frameworks, GEOBIA methods, or CRF-based algorithms. e These techniques should aid the development of more discriminative frameworks that yield superior performance.

## Acknowledgements

This chapter is a reprint of the material as it appears in our submission to the ISPRS Journal of Photogrammetry and Remote Sensing. I was the primary investigator on this paper. We would like to thank Nina Raqueno, Paul Sponagle, Timothy Bausch, Michael McClelland II, and other members of the RIT Signature Interdisciplinary Research Area, UAS Research Laboratory that supported with data collection. We would also like to thank Dr. Michael Gartley with DIRSIG support.

# Chapter 4

# Low-Shot Learning for the Semantic Segmentation of Remote Sensing Imagery

Semantic segmentation is a computer vision task that involves assigning a categorical label to each pixel in an image (i.e., pixel-wise classification). For color (RGB) imagery, deep convolutional neural networks (DCNNs) are continually pushing the state-of-the-art for this task. This is enabled by the availability of large annotated RGB datasets. When small amounts of data are used, conventional DCNNs generalize poorly, especially deeper models. This has made it difficult to use models designed for RGB data with multispectral imagery (MSI) and hyperspectral imagery (HSI) that are widely used in remote sensing, since publicly available annotated data is scarce. Due to the limited availability of annotated data for these "non-RGB" sensors, adapting DCNNs to remote sensing problems requires using low-shot learning. Low-shot learning methods seek to accurately make inferences using a small quantity of annotated data. These methods typically build meaningful feature representations using unsupervised or semi-supervised learning to cope with the reduced amount of labeled data.

Many researchers have explored unsupervised feature extraction as a way to boost performance in semantic segmentation of MSI and HSI. They have tried shallow features (e.g., gray-level co-occurrence matrices [36], Gabor [97], sparse coding [33], and extended morphological attribute profiles [1]), and deep-learning models (e.g., autoencoders [2,27,29,30,35]) that learn spatial-spectral feature extractors directly from the data. Recently, self-taught learning models have been introduced to build feature-extracting

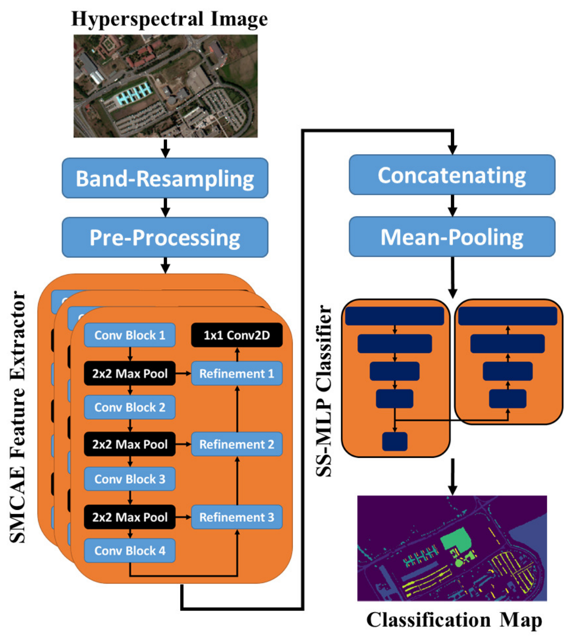


Figure 4.1: Our proposed SuSA architecture for semantic segmentation of remote sensing imagery. For feature extraction, SuSA uses our SMCAE model, a stacked multi-loss convolutional autoencoder that has been trained on unlabeled data using unsupervised learning. For classification, SuSA uses our semi-supervised multi-layer perceptron (SS-MLP) model.

frameworks that generalize well across multiple datasets [94]. In self-taught learning, spatial-spectral feature extractors are trained using a large quantity of unlabeled HSI and then used to extract features from other datasets that we may want to classify (i.e. the target datasets). Self-taught learning for HSI semantic segmentation was pioneered in [94].

As the dimensionality of each feature vector increases, the performance for many deterministic models (e.g., support vector machine (SVM)) will degrade [122]. The most common method for preventing this is to reduce the dimensionality of the feature space (e.g., using principal component analysis (PCA)); however, this involves tuning at least one more hyperparameter (i.e., number of dimensions to retain) through cross-validation.

Multi-layer perceptron (MLP) neural networks can learn which features are the most important for classification; however, they normally require a large quantity of annotated data to generalize well. Semi-supervised learning uses an unsupervised task to regularize classifiers that do not have enough annotated data to work with. For example, the ladder network architecture proposed by Rasmus et al. [123] trains on labeled and unlabeled data simultaneously to boost segmentation performance on smaller training sets. Semi-supervised frameworks give the model the ability to increase the dimensionality in the feature space, which allows them to learn what features are most important for optimal performance, and also enables them to perform well with little annotated data.

In this paper, we describe the semantic segmentation framework SuSA (**s**elf-ta**u**ght **s**emi-supervised **a**utoencoder) shown in Fig. 4.1. SuSA is designed to perform well on MSI and HSI data where image annotations are scarce. SuSA is made of two modules. The first module is responsible for extracting spatial-spectral features, and the second module classifies these features.

We evaluated SuSA across multiple training/testing paradigms, and we compared our performance against state-of-the-art solutions for each respective paradigm found in literature, including two recent self-taught feature learning frameworks: MICA-SVM and SCAE-SVM [94]. We describe these in more detail in later sections.

**This paper's major contributions are**:

- We describe the stacked multi-loss convolutional auto-encoder (SMCAE) model (Fig. 4.4) for spatial-spectral feature extraction in non-RGB remote sensing imagery. SMCAE uses unsupervised self-taught learning to acquire a deep bank of feature extractors. SMCAE is used by SuSA for feature extraction.

- We propose the semi-supervised multi-layer perceptron (SS-MLP) model (Fig. 4.5) for the semantic segmentation of non-RGB remote sensing imagery. SuSA uses SS-MLP to classify the feature representations from SMCAE, and SS-MLP's semi-supervised mechanism enables it to perform well at low-shot learning.

- We demonstrate that SuSA achieves state-of-the-art results on the Indian Pines and Pavia University datasets hosted on the IEEE GRSS Data and Algorithm Standard Evaluation website.

## 4.1 Related Work

### 4.1.1 Self-Taught Feature Learning

The self-taught feature learning paradigm was recently introduced as an unsupervised method for improving the performance for the semantic segmentation of HSI [94]. In the past, researchers learned spatial-spectral features directly from the target data and then passed them to a classifier [2, 16, 35, 36]. Learning spatial-spectral features on a per-image basis is computationally expensive, which may not be ideal for near-real-time analysis. Self-taught feature learning uses large quantities of unlabeled image data to build discriminative feature extractors that generalize well across many datasets, so there is no need to re-train these types of feature extracting frameworks [20].

The authors in [94] introduced two self-taught learning frameworks for the semantic segmentation of HSI. The first model, multi-scale independent component analysis (MICA) learned low-level feature extracting filters corresponding to bar/edge detectors, color opponency, image gradients, etc. The second model, the stacked convolutional autoencoder (SCAE), is a deep learning approach that is able to extract deep spatial-spectral features from HSI. These pre-trained models would extract features from the source image (i.e., the image we want to classify) and pass them to a support vector machine (SVM) classifier. Since MICA-SVM and SCAE-SVM provide state-of-the-art performance across multiple benchmark datasets, we compare our proposed work against them.

The SCAE model consisted of three separate convolutional autoencoder (CAE) modules trained in sequence. The training loss for each CAE was the mean-squared error (MSE) between the input data and the reconstructed output (also known as the data layer). It was shown in [124] that backpropagation is better at optimizing trainable parameters that are closer to where the training loss is computed (i.e., training error signal) than the trainable parameters in deeper layers. The solution was to take a weighted sum of the reconstruction loss for every encoder/decoder pair, which allowed the network to reduce reconstruction errors that occur in deeper layers.

In this paper, we introduce the stacked multi-loss convolutional autoencoder (SMCAE) spatial-spectral feature extracting framework (Fig. 4.4). It is made up of multiple MCAE modules, where each uses multiple loss functions to incorporate and correct reconstruction errors from both shallow and deeper CAE layers. SMCAE trains, extracts, and concatenates feature responses from the individual MCAEs in the way SCAE is built from individual CAEs. SMCAE allows the user to extract deep spatial-spectral features directly from the image data.

### 4.1.2 Semi-Supervised Learning

Self-taught feature learning focuses on unsupervised learning of features on additional data, and then use these features with a supervised system. Semi-supervised algorithms use supervised and unsupervised learning to improve generalization on supervised tasks; which in turn, improves classification performance on test data [123–125]. In both cases, unsupervised learning helps these algorithms to avoid overfitting when given only a small number of labeled HSI samples.

A number of discriminative semi-supervised methods have been adapted for HSI classification. The transductive support vector machine (TSVM) is a low-density separation algorithm that saw early success. TSVM seeks to choose a decision boundary that maximizes the margin between classes using both labeled and unlabeled data [126]. The TSVM outperformed the inductive SVM when evaluated on the Indian Pines HSI dataset [127]. TSVM is computationally expensive and has a tendency to fall into a local minima.

Camps-Valls et al. [128] trained graph-based models for HSI classification using labeled and unlabeled data. Their model iteratively assigned labels to unlabeled pixels that were clustered near labeled pixels. Their model outperformed a standard SVM on the Indian Pines dataset. Using manifold regularization, the Laplacian support vector machine (LapSVM) expanded the graph-based model and showed promise in MSI classification and cloud screening [129]. LapSVM was later modified to incorporate spatial-spectral information [130] and semi-supervised kernel propagation with sparse coding [131]. Ratle et al. [132] recognized the shortcomings of using an SVM and replaced it with a semi-supervised neural network. This neural network outperformed LapSVM and TSVM on the Indian Pines and Kennedy Space Center HSI datasets in both classification accuracy and computational efficiency.

Dopido et al. [133] introduced a semi-supervised model that jointly learned the classification and spectral unmixing task to help improve classification performance on training sets with only a few labeled samples. Liu et al. [134] used the ladder network architecture proposed by [123] to semantically segment HSI. Their ladder network model used convolutional hidden layers in order to learn spatial-spectral features directly from the image. Both of these frameworks introduce an unsupervised task that is jointly optimized with the classification task to help regularize the model, which helped the model generalize and perform well with smaller training sets.

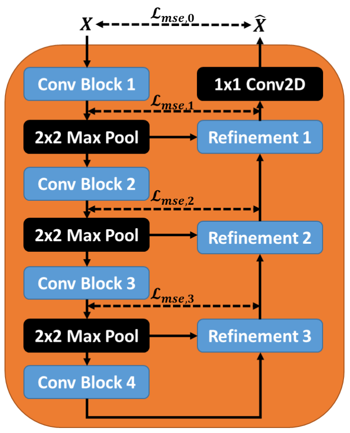


Figure 4.2: MCAE model architecture. Dashed lines indicate where the mean-squared error loss $\mathcal{L}_j$ is calculated for layer $j$, and solid lines are the feed-forward and lateral network connections where information is passed. The refinement layers (Fig. 2.4 are responsible for reconstructing the downsampled feature response.

## 4.2 Methods

### 4.2.1 Multi-Loss Convolutional Autoencoder

Here, we describe the MCAE model (Fig. 4.2), a significant improvement over the original CAE model [94]. Formally, an autoencoder $f$ is an unsupervised neural network that attempts to reconstruct the input $\mathbf{x}$ (e.g., sample from HSI) such that $\hat{\mathbf{x}} = f(\mathbf{x})$, where $\hat{\mathbf{x}}$ is the autoencoder's reconstruction of the original input $\mathbf{x}$. An autoencoder can be trained with various constraints to learn a meaningful feature representation that can still be used for reconstruction. Typically, autoencoders include a separate encoder network that learns a compressed feature representation of the data and a symmetrical decoder network that reconstructs the compressed feature representation back into an estimate of the original input. These networks have hidden layers that use trainable weights $\mathbf{W}$ and biases $\mathbf{b}$ to compress and then reconstruct the input. Since this is an unsupervised learning method, the MSE between $\mathbf{x}$ and $\hat{\mathbf{x}}$ is the loss used to train the network. Once trained, we can extract the features $\mathbf{h}$ from an autoencoder with a single hidden-layer such that,

$$\mathbf{h} = \sigma\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right) \tag{4.1}$$

where $\sigma$ is the non-linear activation function (e.g., CAE used the Rectified Linear Unit (ReLU) activation). A CAE replaces multiply/add operations with 2-D convolution operations,

$$\mathbf{H} = \sigma\left(\mathbf{W} * \mathbf{X}\right) \tag{4.2}$$

where $*$ denotes the 2-D convolution operation and $\mathbf{X}$ is the 2-D image data that will be convolved. 2-D convolution operations learn position invariant feature representations; that is, the feature response for a given object in an image is independent of the pixel location. It slides learned convolution filters across the target image, so the number of trainable parameters are $k^2 \times F_{in} \times F_{out}$, where $k$ is the number of pixels along the edge of the convolution filter (e.g., typically $k = 3$), and $F_{in}$ and $F_{out}$ are the number of input/output features respectively . Standard multi-layer perceptron (MLP) neural networks have a trainable parameter relating every pixel to every input/output feature, resulting in $N_{pixels}^2 \times F_{in} \times F_{out}$ trainable parameters, where $N_{pixels}$ is the number of pixels in the image data. DCNNs almost always have fewer trainable parameters than MLPs of equivalent depth, which can prevent the model from overfitting. In [94], the stacked CAE (SCAE) model is built using several CAEs, where the input to the $k$-th CAE is the output from the last hidden-layer of the $k-1$-th CAE,

$$\mathbf{H}^k = \sigma\left(\mathbf{W}^k * \mathbf{H}^{k-1}\right) \tag{4.3}$$

where $\mathbf{H}^0 = \mathbf{X}$. This allowed the model to learn a deeper feature representation from the input data. Each CAE contains multiple hidden-layers and the down-sampled feature response is reconstructed by the refinement layer shown in Fig. 4.3.

Valpola [124] showed that, for an autoencoder with multiple hidden-layers, errors in deeper layers had a harder time being corrected during back-propagation because they are too far from the training signal. To fix this, we train each CAE using a weighted sum of the reconstruction losses for each hidden-layer,

$$\mathcal{L} = \sum_{j=1}^{M} \lambda_{mcae,j} \mathcal{L}_{mse,j} \tag{4.4}$$

where $M$ is the number of hidden-layers, $\mathcal{L}_{mse,j}$ is the MSE of the encoder and decoder at layer $j$, and $\lambda_{mcae,j}$ is the loss weight at layer $j$. We refer to this new feature extracting model as MCAE. The SMCAE model (Fig. 4.4) trains, extracts, and concatenates feature responses from the individual MCAEs in the same manner as SCAE.

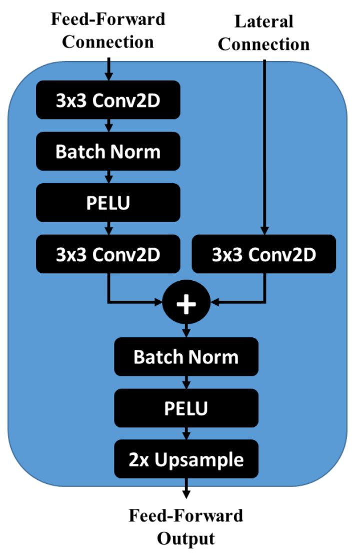


Figure 4.3: Refinement layer used in CAE and MCAE.

### 4.2.2 Semi-Supervised Multi-Layer Perceptron Neural Network

In [94], a major bottleneck in their self-taught learning model was that it used PCA to reduce the feature dimensionality prior to being classified by an SVM. This was necessary because SVMs can suffer from the curse of dimensionality when the feature dimensionality is too high. The ideal number of principal components varied across datasets and required cross-validation. In contrast, MLP-based neural networks are able to learn what features are most important for semantic segmentation. The downside is that standard MLPs require large quantities of labeled data or they will overfit.

To overcome this problem we propose a semi-supervised MLP (SS-MLP). As shown in Fig. 4.5, SS-MLP has a symmetric encoder-decoder framework. The feed-forward encoder network segments the original input and the decoder reconstructs the compressed feature representation back to the original input. The reconstruction serves as an additional regularization operation that can prevent the model from overfitting when there are only a few training samples available. SS-MLP is trained by minimizing the total supervised and unsupervised loss

$$\mathcal{L} = \mathcal{L}_{class} + \sum_{j=1}^{M} \lambda_{recon,j} \cdot \mathcal{L}_{recon,j} \tag{4.5}$$

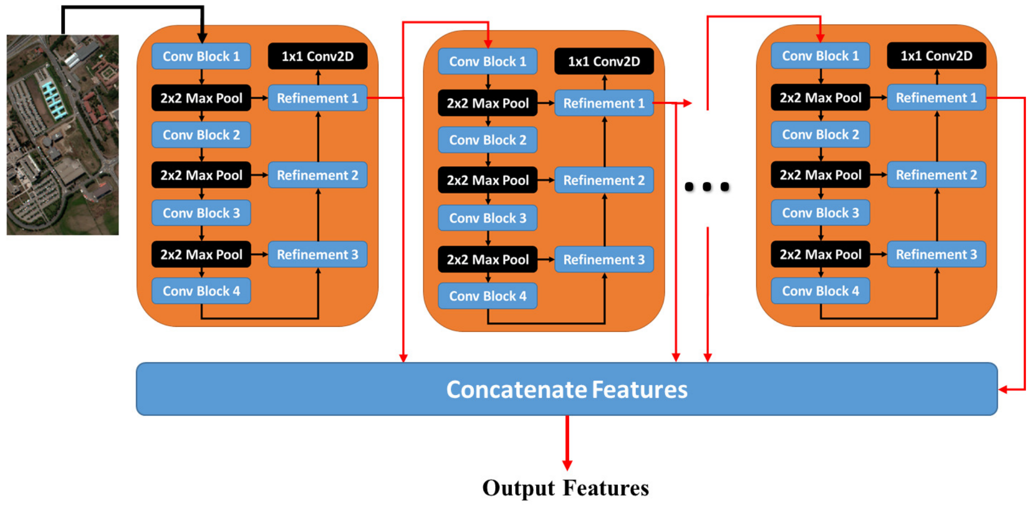


Figure 4.4: The stacked multi-loss convolutional autoencoder (SMCAE) spatial-spectral feature extractor used in this paper consists of two or more MCAE modules. The red lines denote where features are being extracted, transferred to the next MCAE, and concatenated into a final feature response.

where $\mathcal{L}_{class}$ is the cross-entropy loss for classification, $\mathcal{L}_{recon,j}$ is the MSE of the reconstruction at layer $j$, $\lambda_{recon,j}$ is the importance of the unsupervised loss term at layer $j$, and $M$ is the number of hidden layers in SS-MLP. The $\lambda_{recon,j}$ weights are set empirically. This optimization strategy is similar to the ladder network introduced in [134], where the network uses convolutional units to learn spatial-spectral features from a single HSI cube. In this case, the learned spatial-spectral features are specific to this dataset alone and may not transfer well to other HSI we wish to classify. Self-taught learning features, which are learned from a large quantity of imagery, can be more discriminative and generalize well across multiple datasets. In this paper, we will use pre-trained SMCAE models to extract features from the labeled data and then pass them to SS-MLP to generate the final classification map.

### 4.2.3 Adaptive Non-Linear Activations

Kemker and Kanan [94] showed that classification performance with low-level features could be improved by applying an adaptive non-linearity to the feature response. The

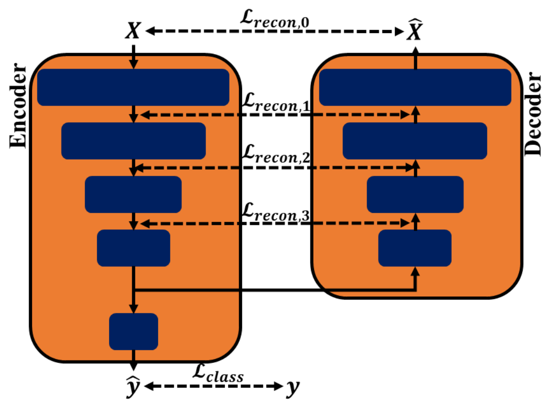


Figure 4.5: The semi-supervised multi-layer perceptron (SS-MLP) classification framework used in this paper.

SCAE is a deep-feature extractor, but it only used a Rectified Linear Unit (ReLU) activation which just sets all negative values to zero. Fixed activations like this may not be the ideal non-linearity required for every network layer; so in this paper, we use the Parametric Exponential Linear Unit (PELU) activation [135],

$$\sigma\left(\mathbf{h};a,b\right)=\begin{cases}\frac{a}{b}\mathbf{h} & \text{if } \mathbf{h}\geq 0\\ a\left(\exp\left(\frac{\mathbf{h}}{b}\right)-1\right) & \text{otherwise}\end{cases} \tag{4.6}$$

where $a$ and $b$ are positive trainable parameters. PELU was shown to increase performance by learning the ideal activation function for each network layer [135]. Depending on the values of $a$ and $b$, PELU can approximate a ReLU activation function or a number of other commonly used activation functions (e.g., LeakyReLU [136] and exponential linear units [137]). In this paper, we use PELU activations with our SMCAE feature extractor and SS-MLP classifier.

## 4.3 Experimental Setup

### 4.3.1 Data Description

The SCAE and SMCAE frameworks were trained using publicly-available HSI data collected by three different NASA sensors: 1) NASA Jet Propulsion Laboratory's Airborne

Table 4.1: Various HSI sensors used in this paper to train and evaluate our SMCAE SS-MLP framework.

| | **AVIRIS** | **Hyperion** | **GLiHT** | **ROSIS** |
|---|---|---|---|---|
| **Platform** | Airborne | Satellite | Airborne | Airborne |
| **Spectral Range [nm]** | 400-2500 | 400-2500 | 400-1000 | 430-838 |
| **Spectral Bands [#]** | 224 | 220 | 402 | 115 |
| **FWHM [nm]** | 10 | 10 | 5 | 5 |
| **GSD [m]** | Varies | 30 | $< 1$ | 0.3-0.7 (best) |
| **Sensor Type** | Whisk Broom | Grating Image Spectrometer | 2-D CCD Imager | Grating Image Spectrometer |

AVIRIS - Airborne Visible/Infrared Imaging Spectrometer
CCD - Charged Couple Device
FWHM - Full-width, Half-Max
GSD - Ground Sample Distance
GLiHT - Goddard's LiDAR, Hyperspectral & Thermal Imager
ROSIS - Reflective Optics System Imaging Spectrometer

Visible/Infrared Imaging Spectrometer (AVIRIS), 2) EO-1 Hyperion imaging spectrometer, and 3) Goddard's LiDAR, Hyperspectral & Thermal Imager (GLiHT). Relevant technical specifications for each sensor are available in Table 4.1. We attempted to collect data from a wide variety of different locations and climates (e.g., urban, forest, farmland, etc.) so that the frameworks would learn spatial-spectral features that generalize across multiple labeled datasets. Samples from all three sensors can be seen in Fig. 4.6.

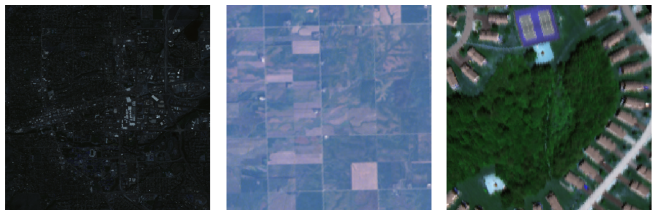

(a) AVIRIS (b) Hyperion (c) GLiHT

Figure 4.6: RGB visualization of HSI from all three sensors used to train SCAE and SMCAE.

Table 4.2: Benchmark HSI datasets used in this paper to evaluate the algorithms.

| | **Indian Pines** | **Pavia University** | **Salinas Valley** |
|---|---|---|---|
| **Sensor** | AVIRIS | ROSIS | AVIRIS |
| **Spatial Dimensions [pix]** | $145 \times 145$ | $610 \times 340$ | $512 \times 217$ |
| **GSD [m]** | 20 | 1.3 | 3.7 |
| **Spectral Bands** | 224 | 103 | 224 |
| **Spectral Range [nm]** | 400-2500 | 430-838 | 400-2500 |
| **Number of Classes** | 16 | 9 | 16 |

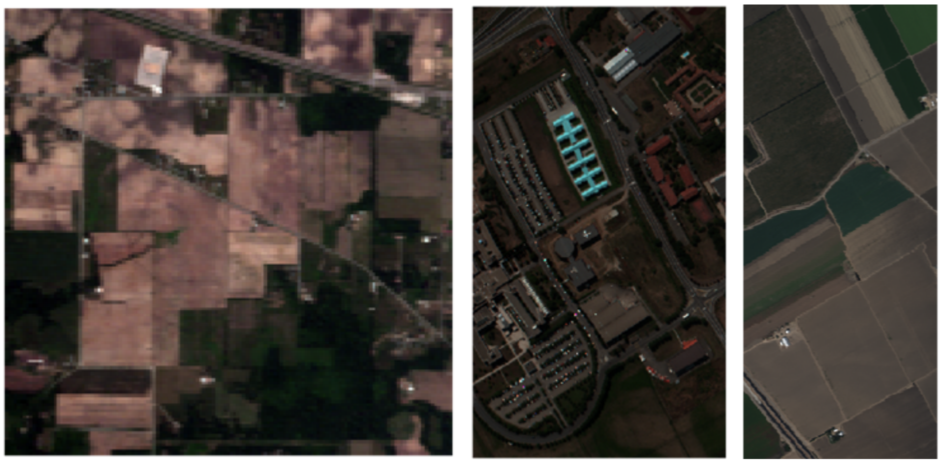
GSD - Ground Sample Distance
ROSIS - Reflective Optics System Imaging Spectrometer
AVIRIS - Airborne Visible/Infrared Imaging Spectrometer

The three annotated HSI datasets used to evaluate the SuSA framework are Indian Pines (Fig. 4.7(a)), Pavia University (Fig. 4.7(b)), and Salinas Valley (Fig. 4.7(c)). Indian Pines and Salinas Valley were captured by the AVIRIS HSI sensor and contain mostly agricultural scenes. Pavia University was collected by the Reflective Optics System Imaging Spectrometer (ROSIS) airborne sensor and is an urban scene with several man-made objects. Fig. 4.7 shows a RGB visualization of all three datasets and Fig. 4.8 shows their corresponding ground truth maps.

### 4.3.2 Training Parameters

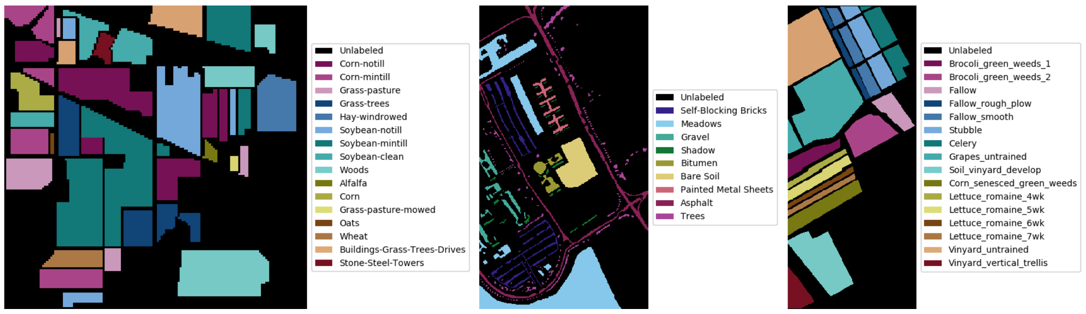
#### MCAE

The CAE and MCAE frameworks use the parameters listed in Table 4.3 throughout this paper. We used the same layer shape found to work well in [94] for CAE and MCAE to provide a fair comparison between the two models. These networks were trained using the open-source imagery listed in Table 4.6. We randomly sampled a total of 50,000 $32{\times}32{\times}B$ image patches from these different HSI images, where $B$ is the number of spectral bands that correspond to each sensor. Of the 50,000 image patches, 45,000 are reserved for training and 5,000 are reserved for validation. Bands that correspond to low SNR and atmospheric absorption are removed. We center each feature in the patch array to zero-mean and unit-variance prior to training the model. The weights are initialized with Xavier initialization [138](i.e., drawn from a normal distribution with its variance



(a) Indian Pines (b) Pavia Univ. (c) Salinas

Figure 4.7: RGB visualization for Indian Pines, Pavia University, and Salinas Valley HSI datasets. See Table 4.2 for scale.



(a) Indian Pines (b) Pavia Univ. (c) Salinas

Figure 4.8: Classification truth maps for Indian Pines, Pavia University, and Salinas Valley HSI datasets.

chosen based on the number of units), and the biases and PELU parameters are initialized with ones.

SCAE and SMCAE were trained using the Nadam optimizer, which is a common variant of stochastic gradient descent used to speed up training of deep learning models [139]. During training, the learning rate was dropped by a factor of 10 when the validation loss did not improve for five consecutive epochs. The models were also trained using early

Table 4.3: Training parameters for CAE and MCAE.

| | **CAE** | **MCAE** |
|---|---|---|
| **Multi-Loss Weights** | None | $1, 10^{-1}, 10^{-2}, 10^{-2}$ |
| **Convolution Layer** | 256,512,512,1024 | |
| **Refinement Layer** | 512,512,256 | |
| **Activation** | ReLU | PELU |
| **Initial Learning Rate** | $2 \times 10^{-3}$ | |
| **Batch Size** | 512 | |

stopping, where training terminated when the validation loss did not improve for ten consecutive epochs. The output of the last hidden layer is then fed to the next CAE/MCAE to build the corresponding SCAE/SMCAE frameworks.

After training SCAE and SMCAE, we use them to extract features from the annotated datasets. First, we re-sample the data to match the same spectral-bands and full-width, half-maxes (FWHMs) as the data used to train the corresponding feature extracting framework. Throughout this paper, we use the band resampling method used in [71], which has been made publically available. This method assumes that the target sensor has a (per-band) Gaussian response. For each target band and corresponding full-width, half-max (FWHM), the algorithm searches for the source bands that overlap and then integrates those responses over the region of overlap in the target sensor.

Next, we center each feature in the data to zero-mean/unit-variance. Finally, we pass this data through the first CAE/MCAE and extract the features from the last hidden layer. These features are fed to the second CAE/MCAE, and so on. The output from each CAE/MCAE is concatenated along the feature dimension. Each feature in the feature response is centered to zero-mean, unit-variance. Finally, we incorporate translation invariance into our final feature response by pooling the feature response with a $5 \times 5$ mean-pooling filter. The receptive field of this filter is considerably smaller than the one used in [94], which will prevent the mean-pooling operation from blurring out small objects and will also preserve sharp boundaries between object classes.

### SS-MLP

The input to the SS-MLP classifier is the extracted features from SMCAE. The HSI cube is reshaped into a 2-dimensional vector (i.e., number of pixels $\times$ number of features). The parameters for the SS-MLP classifier used in this paper are shown in Table 4.4. The relatively high weight decay term was shown in [135] to work well for the PELU activation.

Table 4.4: Training parameters for SS-MLP.

| | |
|---:|:---:|
| **Hidden Layer Shapes** | $[1600, 950, 250, 225]$ |
| **Activation** | PELU |
| $\lambda_{recon}$ | $[1, 1, 0.1, 0.1, 0.1, 0.1]$ |
| **Initial Learning Rate** | $2 \cdot 10^{-3}$ |
| **Mini-Batch Size** | $8$ |
| **Weight Decay** | $10^{-3}$ |

The weights are initialized with Xavier initialization [138], and the biases and PELU parameters are initialized with ones. We optimize the joint loss function using the Nadam optimizer. The initial learning rate is the default $2 \cdot 10^{-3}$. We drop the learning rate by a factor of 10 when the validation accuracy plateaus for 25 consecutive epochs; and we stop training the model when the validation accuracy plateaus for 50 consecutive epochs. The training/validation folds are built by randomly sampling the available training data 90%/10% respectively.

## 4.4 Experimental Results and Discussion

We conducted experiments to measure the performance of our proposed SuSA framework. All of the results are reported as the mean and standard deviation of 30 trials. In each trial, we randomly sample $L$ labeled samples from the HSI dataset for training. The reported performance is the semantic segmentation result on all available labeled samples. The three reported metrics used for this section are overall accuracy (OA), mean-class (average) accuracy (AA), and Cohen's kappa coefficient ($\kappa$).

Before giving the results of the full model across three datasets in Section 4.4.3, we first describe preliminary experiments to compare single- vs. multi-loss CAE and study the effect of stacking features using the Pavia University dataset.

### 4.4.1 Single- vs. Multi-Loss CAE

In this section, we compare the CAE model proposed earlier in [94] to the MCAE model proposed in this paper using the Pavia University dataset for both $L = 10$ and $L = 50$ samples per class. In this experiment, we extracted the features from a single CAE/MCAE trained on unlabeled AVIRIS HSI. The results are given in Table 4.5. We also show performance on the raw spectrum (i.e., pass the original HSI to SS-MLP). MCAE outperforms

Table 4.5: Classification results on the Pavia University dataset using a single CAE and MCAE model trained on unlabeled AVIRIS data. These results were generated by training SS-MLP on $L$ labeled samples per class. Best performance for each experiment is in bold.

| | **OA** | **AA** | $\kappa$ |
|---|---|---|---|
| $L = 10$ | | | |
| **Raw Spectrum** | $77.58 \pm 2.41$ | $76.37 \pm 3.31$ | $0.7041 \pm 0.0306$ |
| **CAE** | $83.47 \pm 2.66$ | $84.78 \pm 2.99$ | $0.7844 \pm 0.0325$ |
| **MCAE** | $\mathbf{84.07 \pm 2.49}$ | $\mathbf{84.95 \pm 2.54}$ | $0.\mathbf{7923 \pm 0.0305}$ |
| $L = 50$ | | | |
| **Raw Spectrum** | $87.93 \pm 0.92$ | $87.55 \pm 1.15$ | $0.8411 \pm 0.0117$ |
| **CAE** | $92.08 \pm 1.23$ | $92.53 \pm 1.55$ | $0.8960 \pm 0.0158$ |
| **MCAE** | $\mathbf{94.19 \pm 0.99}$ | $\mathbf{94.74 \pm 1.16}$ | $\mathbf{0.9234 \pm 0.0130}$ |

its CAE predecessor, although the gap is not large. In the next sections, we increase this gap by including features from stacked MCAEs trained by HSI from three different sensors.

### 4.4.2 Stacked Feature Representations

In this experiment, we examine the impact of stacking MCAE feature representations on classification performance. We extracted features from the SMCAE model, trained on unlabeled AVIRIS HSI, and fed it to four different classifiers: linear kernel SVM, radial basis function (RBF) SVM, standard MLP, and our SS-MLP. For the SVM experiments, we cross-validate for the optimal cost $C$ and kernel width $\gamma$ (RBF only) hyperparameters. We use the same hyperparameters in Table 4.4 for the standard and semi-supervised MLP classifiers.

Each model was trained on Pavia University using $L = 50$ samples per class. Fig. 4.9 shows the mean-class test accuracy of each classifier (as a mean of 30 runs) as additional stacked MCAE feature representations are added. SS-MLP model outperformed these standard classification methods and the performance improves as more MCAE features are added. Since the performance saturates at 4-5 MCAEs, we will use 5 MCAEs from each sensor for the remainder of this paper. The SVM classifier's peak performance occurs at 2-3 CAEs and then decreases when additional CAE features are added due to overfitting.

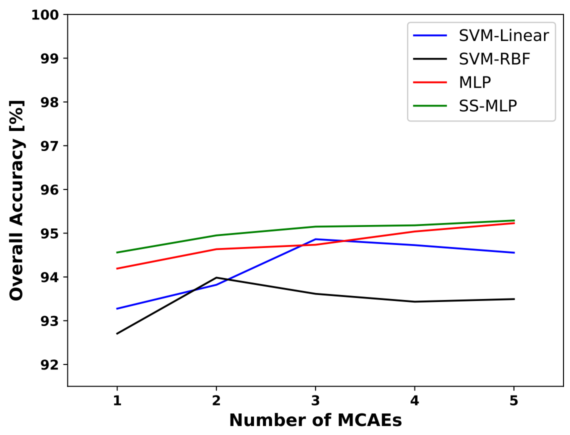


Figure 4.9: SMCAE performance on four different classifiers: linear kernel SVM, radial basis function (RBF) SVM, standard MLP, and our SS-MLP. Our SS-MLP model does the best.

### 4.4.3 Multi-Sensor Fusion

In this section, we show how combining features from SMCAE models trained on HSI collected from different sensors can significantly improve semantic segmentation performance. In this experiment, we evaluate performance using the Pavia University dataset, where our framework is trained using $L = 50$ samples per class. We trained three variants of SMCAE, where the model is trained on HSI from the AVIRIS, Hyperion, and GLiHT sensors. We also tested each possible combination of SMCAE frameworks, where the output of each SMCAE is concatenated along the feature axis. Table 4.6 shows the impact that each SMCAE has on performance. Performance across models differs noticeably, and combining features from multiple sensors yields the best performance. This could indicate that each SMCAE model learns novel information that is not available from the SMCAE models trained on different sensors (see Section 4.4.5 for more details). The SMCAE model trained on GLiHT yielded superior results than the other two SMCAE models. This is likely because Pavia University and GLiHT share similar spectral range and bands; whereas AVIRIS and Hyperion expand beyond the range covered by the ROSIS sensor that collected Pavia University.

Table 4.6: Classification performance using features extracted from SMCAE models that were trained with data from different HSI sensors.

| **Data Source(s)** | **OA** | **AA** | $\kappa$ |
|---|---|---|---|
| **AVIRIS** | $95.07 \pm 0.69$ | $96.03 \pm 0.85$ | $0.9350 \pm 0.0090$ |
| **Hyperion** | $95.93 \pm 0.90$ | $96.47 \pm 0.56$ | $0.9463 \pm 0.0117$ |
| **GLiHT** | $97.83 \pm 0.67$ | $98.03 \pm 0.57$ | $0.9713 \pm 0.0088$ |
| **AVIRIS/Hyperion** | $96.51 \pm 0.91$ | $96.85 \pm 1.06$ | $0.9538 \pm 0.0120$ |
| **AVIRIS/GLIHT** | $97.96 \pm 0.57$ | $98.13 \pm 0.49$ | $0.9730 \pm 0.0075$ |
| **Hyperion/GLiHT** | $98.13 \pm 0.36$ | $98.21 \pm 0.38$ | $0.9752 \pm 0.0048$ |
| **AVIRIS/Hyperion/ GLiHT** | **98.18 $\pm$ 0.53** | **98.29 $\pm$ 0.38** | **0.9759 $\pm$ 0.0069** |

### 4.4.4 State-of-the-Art Comparison

In this section, we use the same SMCAE configuration discussed in Section 4.4.3, where we stacked features from all three sensors listed in Table 4.6. Table 4.7 shows the classification performance when $L = 10$ samples per class. We compared against models found to work well using this training paradigm. For Indian Pines and Pavia University, we compare against a semi-supervised classification approach that uses spectral-unmixing to help improve classification performance [133]. They showed that introducing the unsupervised task helped regularize the model, thus improving generalization when only small quantities of annotated image data are available. Their results were reported as the mean and standard deviation of 10 separate runs. Imani and Ghassemian [140] proposed a model that was supposed to work well on all three of the annotated HSI datasets evaluated in this paper; however, they showed that a SVM classifier yielded the best results. They only reported the mean (no standard deviation) of the mean-class accuracy over three runs. To generate more detailed results, we reproduced this experiment using an SVM-RBF classifier. We reported the overall accuracy, mean-class accuracy, and kappa statistic as the mean and standard deviation over 30 trials. Our SuSA framework achieved superior results compared to each of these frameworks.

Table 4.8 directly compares against previous self-taught and semi-supervised frameworks discussed in this paper. The SCAE-SVM framework introduced by [94] performed well on the $L = 50$ samples per class training paradigms. Note, the Indian Pines dataset used $L = 50$ samples per class except for the three classes that had the smallest number of annotated training samples available, where we only used $L = 15$ samples per class. Liu et al. [134] only evaluated their ladder network on Pavia University with $L = 200$

Table 4.7: Results of low-shot learning experiment where the training set contains only $L$=10 samples per class.

| **Model** | **OA** | **AA** | $\kappa$ |
|---|---|---|---|
| **Indian Pines** | | | |
| Dopido et al. [133] | $75.29 \pm 2.40$ | $79.05 \pm 2.00$ | $0.7184 \pm 0.0275$ |
| MICA-SVM [94] | $66.32 \pm 2.17$ | $81.47 \pm 1.27$ | $0.6261 \pm 0.0237$ |
| SCAE-SVM [94] | $80.58 \pm 2.13$ | $89.24 \pm 1.08$ | $0.7816 \pm 0.0234$ |
| SuSA | **81.16** $\pm$ **1.85** | **89.01** $\pm$ **1.26** | **0.7874** $\pm$ **0.0205** |
| **Pavia University** | | | |
| Dopido et al. [133] | $84.14 \pm 1.97$ | $84.48 \pm 1.04$ | $0.7923 \pm 0.0237$ |
| MICA-SVM [94] | $75.74 \pm 3.81$ | $78.85 \pm 4.01$ | $0.6907 \pm 0.0446$ |
| SCAE-SVM [94] | $84.67 \pm 3.36$ | $87.32 \pm 2.04$ | $0.8028 \pm 0.0405$ |
| SuSA | **89.28** $\pm$ **2.64** | **89.58** $\pm$ **1.62** | **0.8595** $\pm$ **0.0330** |
| **Salinas Valley** | | | |
| SVM-RBF [140] | $82.65 \pm 1.49$ | $90.01 \pm 0.92$ | $0.8075 \pm 0.0165$ |
| MICA-SVM [94] | $90.08 \pm 1.50$ | $94.17 \pm 0.99$ | $0.8899 \pm 0.0165$ |
| SCAE-SVM [94] | $92.74 \pm 1.42$ | $95.56 \pm 0.80$ | $0.9193 \pm 0.0157$ |
| SuSA | **93.47** $\pm$ **1.27** | **96.46** $\pm$ **0.71** | **0.9274** $\pm$ **0.0142** |

samples per class. In every case, SuSA outperforms the previous state-of-the-art classification frameworks.

We performed a statistical significance test (using a 99% confidence interval) on the mean-class accuracy results in Table 4.8. We chose mean-class accuracy because the class distributions are imbalanced, so this is a more meaningful measurement of model performance. The results for all four training/testing paradigms were shown to be statistically significant.

Finally, Table 4.9 shows that SuSA yielded state-of-the-art performance on the Indian Pines and Pavia University HSI datasets hosted on the IEEE GRSS Data and Algorithm Standard Evaluation (DASE) website. The training/testing folds are pre-defined, and the server provides the classification performance on the test set. This dataset is more difficult to perform well on because the training samples are co-located instead of being randomly sampled across the image. At this time, the server only lists the top-10 performers, so we are unable to ascertain the identity of the previous state-of-the-art performer or what method they used. It also only lists their overall accuracy; however, we have provided all of the relevant statistics, including the classification maps in Fig. 4.10. The main performance degradation for Pavia University occurred when SuSA predicted meadows

Table 4.8: Performance comparison of SuSA against the other semi-supervised and self-taught learning frameworks discussed in this paper.

| **Model** | **OA** | **AA** | $\kappa$ |
|---|---|---|---|
| **Indian Pines** ($\mathbf{L = 50/15}$) | | | |
| DAFE [1] | 93.27 | 95.86 | 0.923 |
| MICA-SVM [94] | $94.63 \pm 1.00$ | $97.31 \pm 0.37$ | $0.9385 \pm 0.0114$ |
| SCAE-SVM [94] | $96.12 \pm 0.78$ | $94.58 \pm 0.31$ | $0.9554 \pm 0.0078$ |
| SuSA | $\mathbf{96.49 \pm 0.69}$ | $\mathbf{98.34 \pm 0.31}$ | $\mathbf{0.9602 \pm 0.0089}$ |
| **Pavia University** ($\mathbf{L = 50}$) | | | |
| SSAE [2] | $91.96 \pm 0.87$ | $93.52 \pm 0.42$ | $0.9025 \pm 0.0112$ |
| MICA-SVM [94] | $93.92 \pm 1.38$ | $95.58 \pm 0.64$ | $0.9203 \pm 0.0177$ |
| SCAE-SVM [94] | $95.84 \pm 0.94$ | $96.56 \pm 0.51$ | $0.9451 \pm 0.0123$ |
| SuSA | $\mathbf{98.18 \pm 0.53}$ | $\mathbf{98.29 \pm 0.38}$ | $\mathbf{0.9759 \pm 0.0069}$ |
| **Pavia University** ($\mathbf{L = 200}$) | | | |
| SS-CNN [134] | 98.32 | 98.47 | Unknown |
| MICA-SVM [94] | $98.20 \pm 0.33$ | $98.96 \pm 0.17$ | $0.9763 \pm 0.0043$ |
| SCAE-SVM [94] | $98.57 \pm 0.24$ | $99.07 \pm 0.17$ | 0.9812±0.0032 |
| SuSA | $\mathbf{99.66 \pm 0.11}$ | $\mathbf{99.70 \pm 0.09}$ | $\mathbf{0.9954 \pm 0.0014}$ |
| **Salinas Valley** ($\mathbf{L = 50}$) | | | |
| GLCM+ [36] | 95.41 | Unknown | Unknown |
| MICA-SVM [94] | $97.15 \pm 0.56$ | $98.57 \pm 0.29$ | $0.9683 \pm 0.0062$ |
| SCAE-SVM [94] | $98.06 \pm 0.45$ | $98.94 \pm 0.22$ | $0.9784 \pm 0.0050$ |
| SuSA | $\mathbf{98.10 \pm 0.61}$ | $\mathbf{99.11 \pm 0.26}$ | $\mathbf{0.9788 \pm 0.0068}$ |

(largest object class) when it should have predicted bare soil. There was also a problem predicting trees when it should have predicted meadows. For Indian Pines, SuSA mispredicted corn for corn no-till and corn-min, and pasture/mowed grass was confused for soybeans-min.

### 4.4.5 Dissimilarity Between Learned Features

In this paper, we show that our SuSA framework yields state-of-the-art performance when only a few training samples are available. We also show that transferring spatial-spectral features from multiple sensors can improve classification performance. This would mean that SMCAE learns different features from different data and sensor modalities. To quantify the dissimilarity between different SMCAE models, we used the dissimilarity metric proposed by [75]. The authors computed the dissimilarity of two feature representations

Table 4.9: Classification results for the Indian Pines and Pavia University datasets from the IEEE GRSS Data and Algorithm Standard Evaluation website.

| | **Indian Pines** | **Pavia University** |
|---|---|---|
| **State-of-Art Performer:** | | |
| **OA** | 90.73 | 73.06 |
| **SuSA:** | | |
| **OA** | **91.32** | **81.86** |
| **AA** | 81.17 | 74.09 |
| $\kappa$ | 0.90 | 0.77 |

Table 4.10: Dissimilarity between the feature responses from all three SMCAE models. The higher the value, the more dissimilar the two feature representations are.

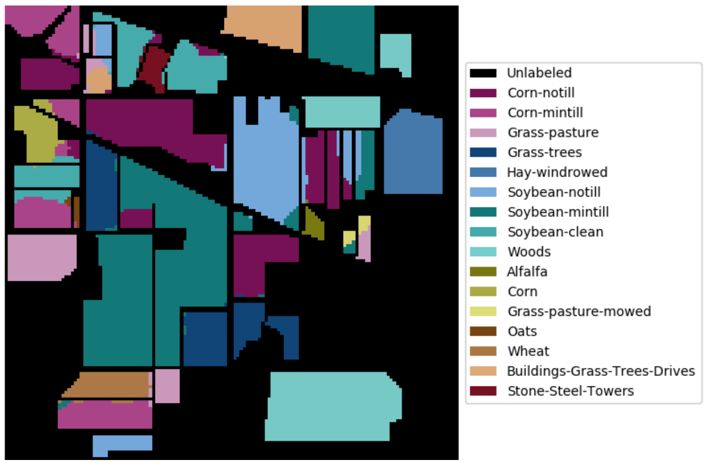
| | **AVIRIS** | **GLiHT** | **Hyperion** |
|---|---|---|---|
| **AVIRIS** | 0.000 | 0.108 | 0.093 |
| **GLiHT** | | 0.000 | 0.105 |
| **Hyperion** | | | 0.000 |

**X**, **Y** such that,

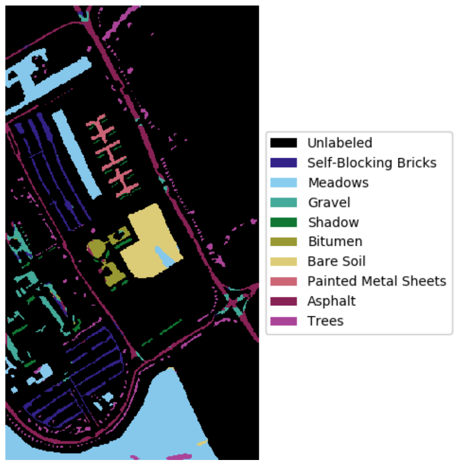
$$d\left(\mathbf{X},\mathbf{Y}\right) = 1 - \frac{1}{N}\sum_{i}^{N}\max_{j} r\left(\mathbf{X}_{i,j},\mathbf{Y}_{i,j}\right) \tag{4.7}$$

where $r$ is the Spearman-correlation matrix and $N$ is the number of rows in $r$. We select a random AVIRIS HSI, generate SMCAE features from all three sensors, and then compute the dissimilarity metrics for every feature response pair (Table 4.10). Although there is some feature overlap between different SMCAE models, there is some new information that comes from combining learned features from multiple sensors. The SMCAE models trained on Hyperion and AVIRIS are more similar than any combination with the SMCAE trained on GLiHT because AVIRIS and Hyperion span the short-wave infrared spectrum whereas GLiHT only spans through the near infrared.

The annotated benchmarks evaluated in this paper all have dramatically different ground sample distances (GSDs) ranging from 1.3 meters to 20 meters; yet, the state-of-the-art performance on each of these datasets could indicate that SMCAE is learning scale-invariant features. In addition, the collection of HSI from different climates, scenes, and weather/atmosphere conditions further improve learned feature generalization; and



(a) Indian Pines



(b) Pavia University

Figure 4.10: Classification maps for SuSA on the Indian Pines and Pavia University datasets from the IEEE GRSS Data and Algorithm Standard Evaluation website.

ultimately, could enable the seamless transfer of spatial-spectral features across different sensors, environments, and machine learning tasks.

## 4.5 Conclusion

In this paper, we demonstrated that SMCAE learns more discriminative self-taught learning features by correcting errors in both shallow and deeper layers during training. We have also shown that our SS-MLP classifier is effective at low-shot learning and able to handle high-dimensional inputs. Our SuSA framework achieved state-of-the-art performance on both IEEE GRSS benchmarks for HSI semantic segmentation and have established a high bar for low-shot learning of HSI datasets. Future work will include scaling these frameworks to other data modalities (e.g., MSI, thermal, synthetic aperture radar, etc.), higher GSD imagery (e.g., centimeter resolution imagery taken from drones), and other remote sensing tasks (e.g., target detection, crop health estimation, etc.).

## Acknowledgements

This chapter is a reprint of the material as it appears in our submission to the IEEE Transactions on Geoscience and Remote Sensing. I was the primary investigator on this paper. We would like to thank Purdue, Pavia University, the Hysens project, and NASA/JPL-Caltech for making their remote sensing data publicly available.

# Chapter 5

# Conclusion

In this dissertation, we focused on different strategies that compensate for the lack of sensor-specific remote sensing imagery with corresponding annotations. First, our self-taught feature learning frameworks (i.e., MICA, SCAE, and SMCAE) used large quantities of unlabeled imagery to learn a set of spatial-spectral feature extractors that transfer between different sensors and imagery (i.e. they generalize well) and they improve the discriminative power required for supervised classification. In SMCAE, the additional reconstruction losses between the encoder and decoder layers deeper in the network allowed the model to learn more discriminative features; which in turn, increase classification performance over SCAE on every benchmark. Also, fusing features learned from different sensors can also increase the discriminative power of the features extracted from the labeled data.

Second, we designed a new benchmark to assess performance for the semantic segmentation of non-RGB remote sensing imagery (i.e., RIT-18). We have made this dataset publicly-available and hope that the remote sensing community begins to adopt better training and evaluation practices, which will in-turn push state-of-the-art performance for this task.

Third, we used large quantities of synthetic MSI to initialize a DCNN-based semantic segmentation framework so it can be fine-tuned on a much smaller quantity of real MSI. Domain adaptation strategies like this could enable the deployment of semantic segmentation frameworks that can make predictions quickly and also learn the mask sharpening operation during end-to-end training without the need for some computationally expensive post-processor (e.g., CRF).

Finally, our semi-supervised classifier was able to yield state-of-the-art classification results with training sets that have little annotated data available. When used with our self-taught feature learning frameworks, we beat the state-of-the-art performers on the Indian Pines and Pavia University HSI datasets hosted on the IEEE DASE evaluation server.

Future work for self-taught feature learning should focus on incorporating information from multiple senor modalities (e.g., LIDAR, synthetic aperture radar (SAR), thermal, etc.) and smart feature selection strategies that are capable of reducing the increasingly large feature dimensionality to something that trains and predicts more quickly. We could also expand our frameworks to operate on other non-semantic segmentation remote sensing tasks such as object recognition, target detection and tracking, spectral unmixing, etc. We could also theoretically expand our self-taught learning frameworks to different domains and data modalities such as medical imaging and non-RGB robotic vision. We demonstrated that self-taught feature learning still works when we use frameworks trained on other sensor data, but we could possibly improve performance by using data captured by the sensor we are planning to use in a more relevant environment.

The domain adaptation techniques used in this paper could be greatly improved as the quantity and quality of DIRSIG scenes improves as well. In the future, we could pull data from multiple DIRSIG scenes, increase the number of object classes in a given scene, increase the variability of the objects in the scene, and also improve the GSD of the scenes themselves. The RIT Signature Interdisciplinary Research Area, UAS Research Laboratory has made some sizable investments in UAS technology, including a platform capable of simultaneously carrying a high-res RGB imaging system, LWIR microbolometer, LIDAR, and VNIR HSI push-broom sensor. This UAS platform also carries a GPS/IMU with 1.5-3 cm resolution and a data acquisition unit that logs data from all of the sensors. Their future acquisitions include a new platforms capable of carrying payloads up to 24 lbs, a HSI sensor that extends spectral coverage into SWIR bands, a better thermal imaging system, among others. A UAS with this type of configuration could be seen as a high-fidelity DIRSIG scene generator; and with a bit of annotation, these scenes could be used to generate higher quality/quantity synthetic images, which could enable the next-generation of semantic segmentation frameworks for any remote sensing sensor.

# Bibliography

[1] P. Ghamisi, J. A. Benediktsson, G. Cavallaro, and A. Plaza, "Automatic framework for spectral–spatial classification based on supervised feature extraction and morphological attribute profiles," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2147–2160, 2014.

[2] C. Tao, H. Pan, Y. Li, and Z. Zou, "Unsupervised spectral–spatial feature learning with stacked sparse autoencoder for hyperspectral imagery classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 12, pp. 2438–2442, 2015.

[3] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "ImageNet Large Scale Visual Recognition Challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[5] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision*, 2014.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., 2012, pp. 1097–1105.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[9] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[10] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[11] R. Girshick, “Fast r-cnn,” in *International Conference on Computer Vision*, 2015.

[12] A. Jain and G. Healey, “A multiscale representation including opponent color features for texture recognition,” *IEEE Trans. Image Proc.*, vol. 7, no. 1, pp. 124–128, 1998.

[13] O. Rajadell Rojas, P. García Sevilla, and F. Pla Bañón, “Spectral–spatial pixel characterization using gabor filters for hyperspectral image classification,” *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 4, pp. 860–864, 2013.

[14] L. M. Bruce, C. H. Koger, and J. Li, “Dimensionality reduction of hyperspectral data using discrete wavelet transform feature extraction,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 10, pp. 2331–2338, 2002.

[15] L. Shen and S. Jia, “Three-dimensional gabor wavelets for pixel-based hyperspectral imagery classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 12, pp. 5039–5046, 2011.

[16] Y. Y. Tang, Y. Lu, and H. Yuan, “Hyperspectral image classification based on three-dimensional scattering wavelet transform,” *IEEE Transactions on Geoscience and Remote sensing*, vol. 53, no. 5, pp. 2467–2480, 2015.

[17] M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. R. Sveinsson, “Spectral and spatial classification of hyperspectral data using svms and morphological profiles,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 11, pp. 3804–3814, 2008.

[18] M. Dalla Mura, J. Atli Benediktsson, B. Waske, and L. Bruzzone, “Extended profiles with morphological attribute filters for the analysis of hyperspectral data,” *International Journal of Remote Sensing*, vol. 31, no. 22, pp. 5975–5991, 2010.

[19] C. Tao, Y. Tang, C. Fan, and Z. Zou, “Hyperspectral imagery classification based on rotation-invariant spectral–spatial feature,” *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 5, pp. 980–984, 2014.

[20] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 759–766.

[21] A. Plaza, P. Martinez, J. Plaza, and R. Perez, "Dimensionality reduction and classification of hyperspectral image data using sequences of extended morphological transformations," *IEEE Transactions on Geoscience and remote sensing*, vol. 43, no. 3, pp. 466–479, 2005.

[22] N. Falco, J. A. Benediktsson, and L. Bruzzone, "A study on the effectiveness of different independent component analysis algorithms for hyperspectral image classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2183–2199, 2014.

[23] I. Dópido, A. Villa, A. Plaza, and P. Gamba, "A quantitative and comparative assessment of unmixing-based feature extraction techniques for hyperspectral image classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 2, pp. 421–435, 2012.

[24] F. Yan, H. Mingyi, S. Jianghong, and W. Jiang, "ICA-based neural network approach to classification of hyperspectral image," in *Proc. Artificial Intelligence*, 2006, pp. 835–839.

[25] J. Wang and C.-I. Chang, "Independent component analysis-based dimensionality reduction with applications in hyperspectral image analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 6, pp. 1586–1600, 2006.

[26] M. Fauvel, Y. Tarabalka, J. A. Benediktsson, J. Chanussot, and J. C. Tilton, "Advances in spectral-spatial classification of hyperspectral images," *Proceedings of the IEEE*, vol. 101, no. 3, pp. 652–675, 2013.

[27] Z. Lin, Y. Chen, X. Zhao, and G. Wang, "Spectral-spatial classification of hyperspectral image using autoencoders," in *Information, Communications and Signal Processing*. IEEE, 2013, pp. 1–5.

[28] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 2094–2107, 2014.

[29] Y. Liu, G. Cao, Q. Sun, and M. Siegel, "Hyperspectral classification via deep networks and superpixel segmentation," *International Journal of Remote Sensing*, vol. 36, no. 13, pp. 3459–3482, 2015.

[30] W. Zhao, Z. Guo, J. Yue, X. Zhang, and L. Luo, "On combining multiscale deep learning features for the classification of hyperspectral remote sensing imagery," *International Journal of Remote Sensing*, vol. 36, no. 13, pp. 3368–3379, 2015.

[31] B. Du, W. Xiong, J. Wu, L. Zhang, L. Zhang, and D. Tao, "Stacked convolutional denoising auto-encoders for feature representation," *IEEE transactions on cybernetics*, vol. 47, no. 4, pp. 1017–1027, 2017.

[32] H. Leigang, F. Xiangchu, H. Chunlei, and P. Chunhong, "Learning deep dictionary for hyperspectral image denoising," *IEICE Transactions on Information and Systems*, vol. 98, no. 7, pp. 1401–1404, 2015.

[33] A. Soltani-Farani and H. R. Rabiee, "When pixels team up: spatially weighted sparse coding for hyperspectral image classification," *IEEE Geoscience Remote Sensing Letters*, vol. 12, no. 1, pp. 107–111, 2015.

[34] P. Du, H. Zhao, B. Zhang, and L. Zheng, "Independent component analysis for hyperspectral imagery plant classification," in *Electronic Imaging 2005*. International Society for Optics and Photonics, 2005, pp. 71–81.

[35] X. Ma, J. Geng, and H. Wang, "Hyperspectral image classification via contextual deep learning," *EURASIP Journal on Image and Video Processing*, vol. 2015, no. 1, pp. 1–12, 2015.

[36] F. Mirzapour and H. Ghassemian, "Improving hyperspectral image classification by combining spectral, texture, and shape features," *International Journal of Remote Sensing*, vol. 36, no. 4, pp. 1070–1096, 2015.

[37] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 2, pp. 22–40, 2016.

[38] A. Coates, H. Lee, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," *Ann Arbor*, vol. 1001, no. 48109, p. 2, 2010.

[39] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, p. 607, 1996.

[40] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," *Advances in neural information processing systems*, vol. 19, p. 801, 2007.

[41] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 1794–1801.

[42] M. S. Caywood, B. Willmore, and D. J. Tolhurst, "Independent components of color natural scenes resemble v1 neurons in their spatial and color tuning," *Journal of Neurophysiology*, vol. 91, no. 6, pp. 2859–2873, 2004.

[43] C. Kanan and G. Cottrell, "Robust classification of objects, faces, and flowers using natural image statistics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[44] A. Olmos and F. A. Kingdom, "A biologically inspired algorithm for the recovery of shading and reflectance images," *Perception*, vol. 33, no. 12, pp. 1463–1473, 2004.

[45] J. M. Nascimento and J. M. B. Dias, "Does independent component analysis play a role in unmixing hyperspectral data?" *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 1, pp. 175–187, 2005.

[46] K. Tiwari, M. Arora, and D. Singh, "An assessment of independent component analysis for detection of military targets from hyperspectral images," *International Journal of Applied Earth Observation and Geoinformation*, vol. 13, no. 5, pp. 730–740, 2011.

[47] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," DTIC Document, Tech. Rep., 1985.

[48] M. A. Ranzato and M. Szummer, "Semi-supervised learning of compact document representations with deep networks," in *Proc Int. Conf. Mach. Learn.*, 2008, pp. 792–799.

[49] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[50] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-based model," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*. MIT Press, 2006, pp. 1137–1144.

[51] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[52] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," *Artificial Neural Networks and Machine Learning–ICANN 2011*, pp. 52–59, 2011.

[53] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[54] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine learning*, vol. 39, no. 2, pp. 103–134, 2000.

[55] S. Rajan, J. Ghosh, and M. M. Crawford, "An active learning approach to hyperspectral data classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 4, pp. 1231–1242, 2008.

[56] D. Tuia, M. Volpi, L. Copa, M. Kanevski, and J. Munoz-Mari, "A survey of active learning algorithms for supervised remote sensing image classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 3, pp. 606–617, 2011.

[57] B. Du, Z. Wang, L. Zhang, L. Zhang, W. Liu, J. Shen, and D. Tao, "Exploring representativeness and informativeness for active learning," *IEEE transactions on cybernetics*, vol. 47, no. 1, pp. 14–26, 2017.

[58] R. K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *J. Mach. Learn. Research*, vol. 6, pp. 1817–1853, 2005.

[59] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[60] S. Thrun, "Is learning the n-th thing any easier than learning the first?" *Adv. neural info. Proc. Sys.*, pp. 640–646, 1996.

[61] C. Kanan, "Active object recognition with a space-variant retina," *ISRN Machine Vision*, vol. 2013, 2013.

[62] P. Wang, G. W. Cottrell, and C. Kanan, "Modeling the object recognition pathway: A deep hierarchical model using gnostic fields," in *Cognitive Sci. Soc.* Cognitive Science Society, July 2015, pp. 2601–2606.

[63] D. Tuia, M. Volpi, M. Dalla Mura, A. Rakotomamonjy, and R. Flamary, "Automatic feature learning for spatio-spectral image classification with sparse svm," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 10, pp. 6062–6074, 2014.

[64] D. L. Ruderman, "The statistics of natural images," *Network: computation in neural systems*, vol. 5, no. 4, pp. 517–548, 1994.

[65] T. Wachtler, E. Doi, T.-W. Lee, and T. J. Sejnowski, "Cone selectivity derived from the responses of the retinal cone mosaic to natural scenes," *Journal of Vision*, vol. 7, no. 8, pp. 6–6, 2007.

[66] A. J. Bell and T. J. Sejnowski, "The independent components of natural scenes are edge filters," *Vision research*, vol. 37, no. 23, pp. 3327–3338, 1997.

[67] H. Shan and G. W. Cottrell, "Looking around the backyard helps to recognize faces and digits," in *Proc. IEEE Comp. Vis. Patt. Recog.* IEEE, 2008, pp. 1–8.

[68] P. H. O. Pinheiro, T. Lin, R. Collobert, and P. Dollár, "Learning to refine object segments," in *European Conference on Computer Vision*, 2016.

[69] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[70] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[71] T. Boggs, "Spectral python," http://www.spectralpython.net, 2014.

[72] M. Baumgardner, L. Biehl, and D. Landgrebe, "220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3," Sep 2015. [Online]. Available: https://purr.purdue.edu/publications/1947/1

[73] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015.

[74] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artificial Intelligence Stats.*, 2010.

[75] N. Kriegeskorte, M. Mur, and P. A. Bandettini, "Representational similarity analysis-connecting the branches of systems neuroscience," *Frontiers in systems neuroscience*, vol. 2, p. 4, 2008.

[76] C. F. Cadieu, H. Hong, D. L. Yamins, N. Pinto, D. Ardila, E. A. Solomon, N. J. Majaj, and J. J. DiCarlo, "Deep neural networks rival the representation of primate it cortex for core visual object recognition," *PLoS Comput Biol*, vol. 10, no. 12, p. e1003963, 2014.

[77] T. W. Anderson, "Estimating linear restrictions on regression coefficients for multivariate normal distributions," *Annals Math. Stats.*, pp. 327–351, 1951.

[78] A. J. Izenman, "Reduced-rank regression for the multivariate linear model," *J. multivariate analysis*, vol. 5, no. 2, pp. 248–264, 1975.

[79] (2017) 2017 IEEE GRSS Data Fusion Contest. [Online]. Available: http://www.grss-ieee.org/community/technical-committees/data-fusion/

[80] A. S. Laliberte, M. A. Goforth, C. M. Steele, and A. Rango, “Multispectral Remote Sensing from Unmanned Aircraft: Image Processing Workflows and Applications for Rangeland Environments,” *Remote Sensing*, vol. 3, no. 11, pp. 2529–2551, 2011.

[81] F. Rottensteiner, G. Sohn, J. Jung, M. Gerke, C. Baillard, S. Benitez, and U. Breitkopf, “The isprs benchmark on urban object classification and 3d building reconstruction,” *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci*, vol. 1, no. 3, pp. 293–298, 2012.

[82] M. Volpi and V. Ferrari, “Semantic segmentation of urban scenes by learning local class interactions,” in *Earth Vision Workshop*, 2015, pp. 1–9.

[83] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes (VOC) Challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[84] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *arXiv preprint*, vol. arXiv:1606.00915, 2016. [Online]. Available: http://arxiv.org/abs/1606.00915

[85] G. Lin, A. Milan, C. Shen, and I. D. Reid, “RefineNet: Multi-path refinement networks for high-resolution semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[86] H. Wang, Y. Wang, Q. Zhang, S. Xiang, and C. Pan, “Gated convolutional neural network for semantic segmentation in high-resolution images,” *Remote Sensing*, vol. 9, no. 5, 2017.

[87] D. Marmanis, J. D. Wegner, S. Galliani, K. Schindler, M. Datcu, and U. Stilla, “Semantic Segmentation of Aerial Images with an Ensemble of CNNS,” *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2016*, vol. 3, pp. 473–480, 2016.

[88] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[89] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” in *International Conference on Computer Vision*, 2015.

[90] P. Krähenbühl and V. Koltun, “Efficient inference in fully connected crfs with gaussian edge potentials,” in *Proceedings of the International Conference on Neural Information Processing Systems*, 2011.

[91] G. Cheng, J. Han, and X. Lu, “Remote sensing image scene classification: Benchmark and state of the art,” *IEEE*, vol. 105, no. 10, pp. 1865–1883, Oct 2017.

[92] G. Cheng, P. Zhou, and J. Han, “Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 12, pp. 7405–7415, 2016.

[93] X. Yao, J. Han, G. Cheng, X. Qian, and L. Guo, “Semantic annotation of high-resolution satellite images via weakly supervised learning,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 6, pp. 3660–3671, 2016.

[94] R. Kemker and C. Kanan, “Self-taught feature learning for hyperspectral image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 5, pp. 2693–2705, 2017.

[95] G. Chen, Q. Weng, G. J. Hay, and Y. He, “Geographic object-based image analysis (geobia): emerging trends and future opportunities,” *GIScience & Remote Sensing*, vol. 55, no. 2, pp. 159–182, 2018.

[96] R. Khatami, G. Mountrakis, and S. V. Stehman, “A meta-analysis of remote sensing research on supervised pixel-based land-cover image classification processes: General guidelines for practitioners and future research,” *Remote Sensing of Environment*, vol. 177, pp. 89 – 100, 2016.

[97] L. Shen *et al.*, “Discriminative gabor feature selection for hyperspectral image classification,” *IEEE Geoscience Remote Sensing Letters*, vol. 10, no. 1, pp. 29–33, 2013.

[98] T. Blaschke, “Object based image analysis for remote sensing,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 65, no. 1, pp. 2–16, 2010.

[99] G. Hay and G. Castilla, “Geographic Object-Based Image Analysis (GEOBIA): A new name for a new discipline,” pp. 75–89, 01 2008.

[100] U. B. Gewali and S. T. Monteiro, “Spectral angle based unary energy functions for spatial-spectral hyperspectral classification using markov random fields,” *arXiv preprint arXiv:1610.06985*, 2016.

[101] P. Zhong and R. Wang, “A multiple conditional random fields ensemble model for urban area detection in remote sensing optical images,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 12, pp. 3978–3988, 2007.

[102] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[103] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *European Conference on Computer Vision*, 2014, pp. 297–312.

[104] M. Zhang, X. Hu, L. Zhao, Y. Lv, M. Luo, and S. Pang, "Learning dual multi-scale manifold ranking for semantic segmentation of high-resolution images," *Remote Sensing*, vol. 9, no. 5, 2017.

[105] (2017) DSTL Satellite Imagery Feature Detection. [Online]. Available: https://www.kaggle.com/c/dstl-satellite-imagery-feature-detection

[106] X. Peng, B. Sun, K. Ali, and K. Saenko, "Learning deep object detectors from 3d models," in *International Conference on Computer Vision*, 2015, pp. 1278–1286.

[107] W. Chen, H. Wang, Y. Li, H. Su, Z. Wang, C. Tu, D. Lischinski, D. Cohen-Or, and B. Chen, "Synthesizing training images for boosting human 3d pose estimation," in *3D Vision*. IEEE, 2016, pp. 479–488.

[108] X. Zhang, Y. Fu, S. Jiang, L. Sigal, and G. Agam, "Learning from synthetic data using a stacked multichannel autoencoder," in *International Conference on Machine Learning*. IEEE, 2015, pp. 461–464.

[109] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3234–3243.

[110] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1-2, pp. 151–175, 2010.

[111] I. Guyon, G. Dror, V. Lemaire, G. Taylor, and D. W. Aha, "Unsupervised and transfer learning challenge," in *Neural Networks (IJCNN)*. IEEE, 2011, pp. 793–800.

[112] E. Ientilucci and S. Brown, "Advances in wide area hyperspectral image simulation," *Proc SPIE*, vol. 5075, pp. 110–121, 2003. [Online]. Available: http://www.dirsig.org/docs/megascene.pdf

[113] J. Schott, S. Brown, R. Raqueo, H. Gross, and G. Robinson, "An advanced synthetic image generation model and its application to multi/hyperspectral algorithm development," *Canadian Journal of Remote Sensing*, vol. 25, no. 2, pp. 99–111, 1999.

[114] W. Zhao, S. Du, and W. J. Emery, "Object-based convolutional neural network for high-resolution imagery classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 7, pp. 3386–3396, July 2017.

[115] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision*, 2016.

[116] P. O. Pinheiro, R. Collobert, and P. Dollár, "Learning to segment object candidates," in *Advances in Neural Information Processing Systems*, 2015, pp. 1990–1998.

[117] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A Library for Large Linear Classification," *J. Mach. Learn. Rsrch.*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[118] Y. Huang, S. J. Thomson, Y. Lan, and S. J. Maas, "Multispectral Imaging Systems for Airborne Remote Sensing to Support Agricultural Production Management," *Int. J. Agricultural & Bio. Eng.*, vol. 3, no. 1, pp. 50–62, 2010.

[119] P. J. Zarco-Tejada, J. A. Berni, L. Suárez, G. Sepulcre-Cantó, F. Morales, and J. R. Miller, "Imaging Chlorophyll Fluorescence with an Airborne Narrow-band Multispectral Camera for Vegetation Stress Detection ," *Remote Sens. of Env.*, vol. 113, no. 6, pp. 1262 – 1275, 2009.

[120] J. A. Berni, P. J. Zarco-Tejada, L. Suárez, and E. Fereres, "Thermal and Narrowband Multispectral Remote Sensing for Vegetation Monitoring from an Unmanned Aerial Vehicle," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 3, pp. 722–738, 2009.

[121] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[122] R. Bellman, *Dynamic programming*. Courier Corporation, 2013.

[123] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko, "Semi-supervised learning with ladder networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 3546–3554.

[124] H. Valpola, "From neural pca to deep unsupervised learning," *Advances in Independent Component Analysis and Learning Machines*, pp. 143–171, 2015.

[125] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[126] L. Bruzzone, M. Chi, and M. Marconcini, “A novel transductive svm for semisupervised classification of remote-sensing images,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 11, pp. 3363–3373, 2006.

[127] ——, “Transductive svms for semisupervised classification of hyperspectral data,” in *Geoscience and Remote Sensing Symposium, 2005. IGARSS’05. Proceedings. 2005 IEEE International*, vol. 1. IEEE, 2005, pp. 4–pp.

[128] 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.

[129] L. Gómez-Chova, G. Camps-Valls, J. Munoz-Mari, and J. Calpe, “Semisupervised image classification with laplacian support vector machines,” *IEEE Geoscience and Remote Sensing Letters*, vol. 5, no. 3, pp. 336–340, 2008.

[130] L. Yang, S. Yang, P. Jin, and R. Zhang, “Semi-supervised hyperspectral image classification using spatio-spectral laplacian support vector machiney,” *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 3, pp. 651–655, 2014.

[131] L. Yang, M. Wang, S. Yang, R. Zhang, and P. Zhang, “Sparse spatio-spectral lapsvm with semisupervised kernel propagation for hyperspectral image classification,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 5, pp. 2046–2054, 2017.

[132] F. Ratle, G. Camps-Valls, and J. Weston, “Semisupervised neural networks for efficient hyperspectral image classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 5, pp. 2271–2282, 2010.

[133] I. Dópido, J. Li, P. Gamba, and A. Plaza, “A new hybrid strategy combining semisupervised classification and unmixing of hyperspectral data,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 8, pp. 3619–3629, 2014.

[134] B. Liu, X. Yu, P. Zhang, X. Tan, A. Yu, and Z. Xue, “A semi-supervised convolutional neural network for hyperspectral image classification,” *Remote Sensing Letters*, vol. 8, no. 9, pp. 839–848, 2017.

[135] L. Trottier, P. Giguère, and B. Chaib-draa, “Parametric exponential linear unit for deep convolutional neural networks,” *CoRR*, vol. abs/1605.09332, 2016. [Online]. Available: http://arxiv.org/abs/1605.09332

[136] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the International Conference on Machine Learning*, vol. 30, no. 1, 2013.

[137] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *Proceedings of the International Conference on Learning Representations*, 2016.

[138] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[139] T. Dozat, "Incorporating nesterov momentum into adam," 2016.

[140] M. Imani and H. Ghassemian, "Boundary based supervised classification of hyperspectral images with limited training samples," *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, no. 3, pp. 203–207, 2013.

[141] Agisoft LLC, *Agisoft PhotoScan User Manual Professional Edition, Version 1.2*. Agisoft, 2016. [Online]. Available: http://www.agisoft.com/pdf/photoscan-pro_1_2_en.pdf

# Appendix A

# RIT-18: Dataset Creation Details

In 2016, the Chester F. Carlson Center for Imaging Science established a new UAS laboratory to collect remote sensing data for research purposes. This laboratory is equipped with several UAS payloads including RGB cameras, MSI/HSI sensors, thermal imaging systems, and light detection and ranging (LIDAR). This section will provide detail on how the RIT-18 dataset was created. Fig. A.1 provides a macro-level view of our orthomosaic generation pipeline.

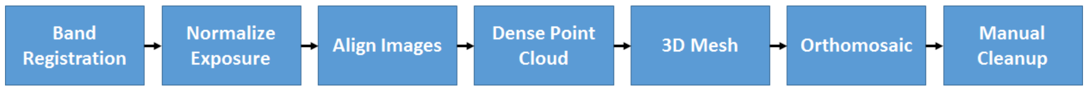


Figure A.1: Orthomosaic Processing Pipeline

The first step is to **co-register the images** from all six spectral bands. For each collection campaign, we filtered out data not collected along our desired flight path (*i.e.* takeoff and landing legs). The six spectral images come from independent imaging systems, so they need to be registered to one another. The manufacturer provided an affine transformation matrix that was not designed to work at the flying height this data was collected at, which caused noticeable registration error. We used one of the parking lot images to develop a global perspective transformation for the other images in our dataset. Fig. A.2 illustrates the registration error caused by the affine transformation provided by the manufacturer and how this error can be reduced with our perspective transformation.

The global transformation worked well for some of the images, but there were registration errors in other parts of the scene, indicating that the transformation needs to be

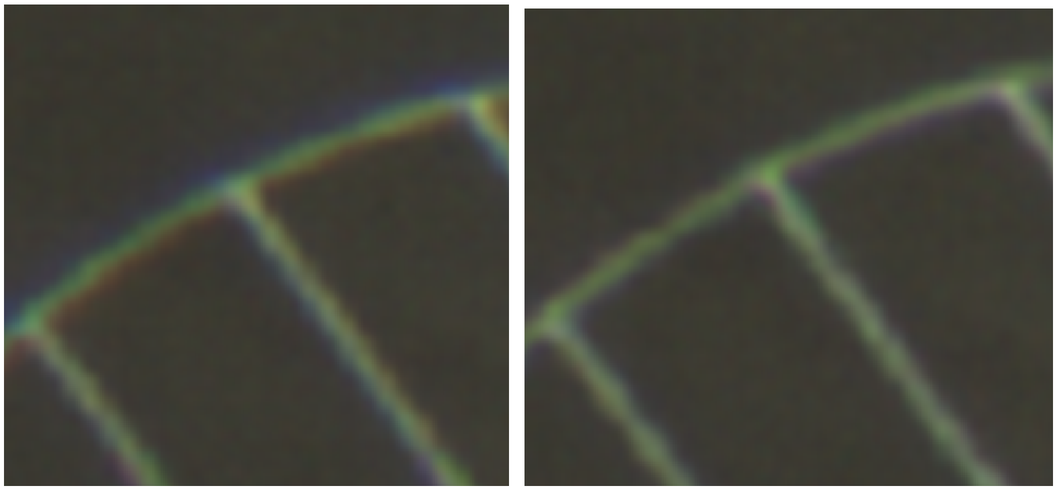

(a) Affine (b) Perspective

Figure A.2: Difference between manufacturer's affine transformation and our perspective transformation. The registration error in the affine transformation looks like a blue and red streak along the top and bottom of the parking lines, respectively.

performed on a per-image basis. This is critical when the wind causes excessive platform motion or when the UAV flies over trees. This was done by 1) extracting SIFT features from each image, 2) using k-Nearest Neighbor to find the best matches, 3) keep the best matches, 4) use RANSAC to find the best homography, and 5) transforming the images with this homography using nearest-neighbor interpolation. If there were no good matches, then we used the global perspective transformation. This was done by matching SIFT features from each band to build custom homographies. If a good homography could not be found, the global transformation was used instead. This was common for homogenous scene elements, such as water, or repeating patterns, such as an empty parking lots.

The second step was to **normalize the exposure** for each band. Each collected frame was acquired with a unique integration time (*i.e.* auto exposure) and each band of the Tetracam Micro-MCA6 uses a different integration time proportional to the sensor's relative spectral response. Another issue is that each image, including each band, is collected with a different integration time. The longer integration times required for darker images, especially over water scenes, resulted in blur caused by platform motion. We normalized each image with its corresponding integration time and then contrast-stretched the image back to a 16-bit integer using the global min/max of the entire dataset. The original images are 10-bit, but the large variation in integration time groups most of the data to lower intensity ranges. We extended the dynamic range of the orthomosaic by stretching the possible quantized intensity states. We generated the othomosaics mostly because of time-constraints; however, we are also interested in evaluating models that train on small quantities of annotated imagery.

The remaining steps were performed using Agisoft PhotoScan [141]. Photoscan uses the 720nm band to perform most of the procedures, and then the remaining spectral channels are brought back in the orthomosaic generation step. The PhotoScan workflow involves:

1. **Image Alignment:** Find key points in the images and match them together as tie-points.
2. **Dense Point Cloud Reconstruction:** Use structure from motion (SfM) to build a dense point cloud from the image data.
3. **3D Mesh:** Build a 3D mesh and corresponding UV texture map from the dense point cloud.
4. **Orthomosaic:** Generate an orthomosaic onto the WGS-84 coordinate system using the mesh and image data.
5. **Manual Clean-up:** Manually correct troublesome areas by removing photographs caused by motion blur or moving objects.

PhotoScan can generate high-quality orthomosaics, but manual steps were taken to ensure the best quality. First, not all of the images were in focus; and although PhotoScan has an image quality algorithm, we opted to manually scan and remove the defocused images. Second, the 3D model that the orthomosaic is projected onto is built from structure-from-motion. Large objects that move over time, such as tree branches blowing in the wind, or vehicles moving throughout the scene, will cause noticeable errors. This is corrected by highlighting the affected region and manually selecting a single (or a few) alternative images that will be used to generate that part of the orthomosaic, as opposed to those automatically selected.

# Appendix B

# RIT-18: Class Descriptions

## B.1 Water/Beach Area

The two classes for water are lake and pond. The lake class is for Lake Ontario, which is north of the beach. The pond water class is for the small inland pond, present in all three folds, which is surrounded by marsh and trees. Along Lake Ontario is a sand/beach class. This class also includes any spot where sand blew up along the asphalt walking paths. Along the beach are some white-painted, wooden lifeguard chairs. The buoy class is for the water buoys present in the water and on the beach. They are very small, primarily red and/or white, and assume various shapes. The rocks class is for the large breakwater along the beach.

## B.2 Vegetation

There are three vegetation classes including grass, trees, and low-level vegetation. The tree class includes a variety of trees present in the scene. The grass includes all pixels on the lawn. There are some mixtures present in the grass (such as sand, dirt, or various weeds), so the classification algorithm will need to take neighboring pixel information into account. The grass spots on the beach and asphalt were labeled automatically using a normalized-difference vegetation index (NDVI) metric. Grass spots that were missed were manually added. The low-level vegetation class includes any other vegetation, including manicured plants, around the building or the marsh next to the pond.

## B.3 Roadway

The asphalt class includes all parking lots, roads, and walk-ways made from asphalt; but the cement and stone paths around the buildings are not consistent between different folds, so they remain labeled as the background class. The road marking class is for any painted asphalt surface including parking/road lanes. This class was automatically labeled with posteriori, but there were a few parking lines in the shade that needed to be manually added. The road markings in the validation image are sharper than those depicted in the training image since the park repainted the lines between collects. The vehicle label includes any car, truck, or bus.

## B.4 Underrepresented Classes

Underrepresented classes, which may be small and/or appear infrequently, will be difficult to identify. Since some of the land cover classes are massive in comparison, the mean-class accuracy metric will be the most important during the classification experiments in Table 3.4. Small object classes, such as person and picnic table, represent only a minute fraction of the image and should remain very difficult to correctly classify. These small objects will be surrounded by larger classes and may even hide in the shade.

There are also a few classes that are only present in the scene a couple of times, such as the white/black wood targets, orange UAS landing pad, lifeguard chair, and buildings. The building class is primarily roof/shingles of a few buildings found throughout the scene. The similarity between the white wooden target and the lifeguard chair should make semantic information in the scene vital to classification accuracy. There is only a single instance of the orange UAS landing pad in every fold. The black and white targets are not present in the validation fold, which could make it difficult to cross-validate for a model that can correctly identify them.

# Appendix C

# SCAE Architecture Used for RIT-18

SCAE, illustrated in Fig. C.1, is another unsupervised spatial-spectral feature extractor. SCAE has a deeper neural network architecture than MICA and is capable of extracting higher-level features [94]. The architecture used in this paper involves three individual convolutional autoencoders (CAEs) that are trained independently. The input and output of the first CAE is a collection of random image patches from the training data, and the input/output of the subsequent CAEs are the features from the last hidden layer of the previous CAE. The output of all three CAEs are concatenated in the feature domain, mean-pooled, and the dimensionality is reduced to 99% of the original variance using WPCA. The final feature response is scaled to zero-mean/unit-variance and then passed to a traditional classifier. Here, we use an MLP with one hidden layer.

The architecture of each CAE is illustrated in Fig. C.2. Each CAE contains a small feed-forward network consisting of multiple convolution and max-pooling operations. The feature response is reconstructed with symmetric convolution and upsampling operations. The reconstruction error is reduced by using skip connections from the feed-forward network, inspired by [68].

The SCAE model used in this paper has the same architecture shown in Fig. 2.5 and C.2. It was trained with 30,000 $128 \times 128$ image patches randomly extracted from the training and validation datasets. Each CAE was trained individually with a batch size of 128. There are 32 units in the first convolution block, 64 units in the second convolution block, 128 units in the third, and 256 units in the $1 \times 1$ convolution. The refinement blocks have the same number of units as their corresponding convolution block, so the last hidden layer has 32 features.

After training, the whole image is passed through the SCAE network to generate three

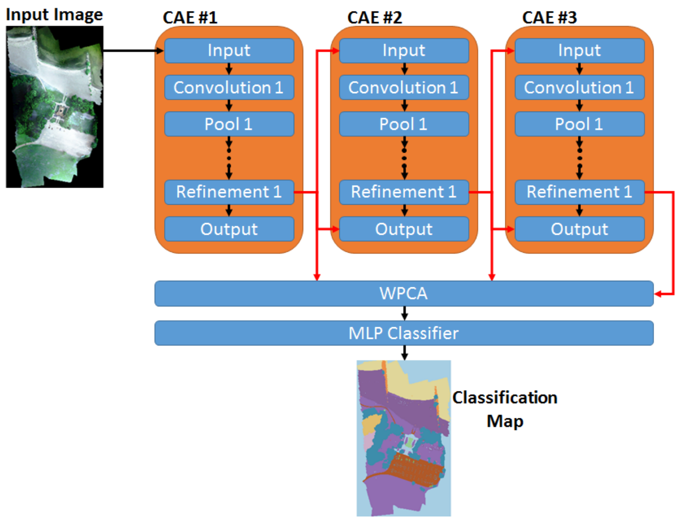


Figure C.1: The SCAE model used in this paper. Architecture details for each CAE are shown in Fig. C.2.

$N \times 32$ feature responses. These feature responses are concatenated, convolved with a $5 \times 5$ mean-pooling filter, and then reduced to 99% of the original variance using WPCA. The final feature response is passed to a MLP classifier with the same architecture used by the MICA model.

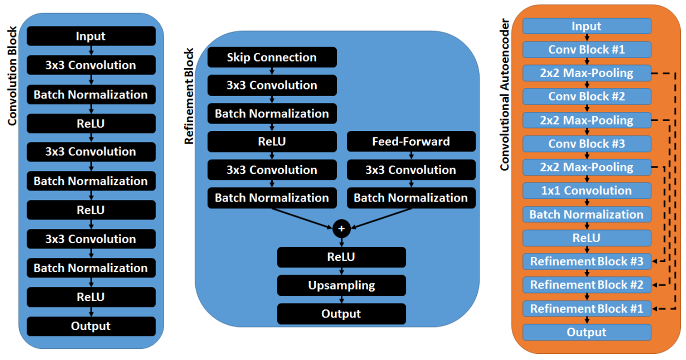


Figure C.2: The convolutional autoencoder (CAE) architecture used in SCAE. This CAE is made up of several convolution and refinement blocks. The SCAE model in this paper uses three CAEs.

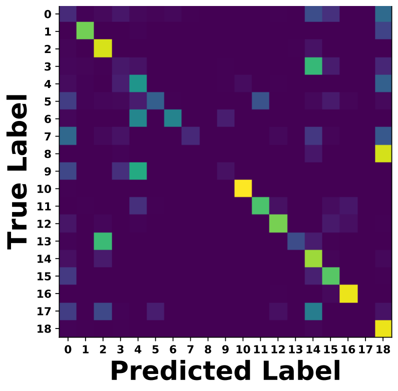
# Appendix D

# Additional Results

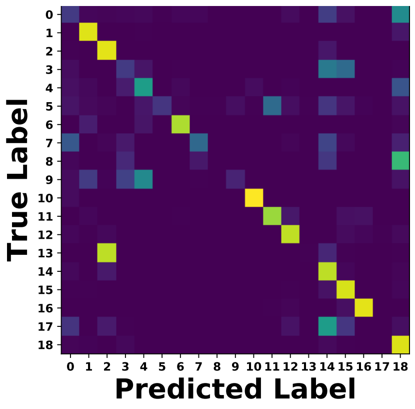
Fig. D.1 shows a heat-map of the confusion matrices for SharpMask and RefineNet - with and without DIRSIG pre-training. These results were generated from the prediction map on the RIT-18 test set. Each row is normalized to show the most common prediction for each class. The brighter (yellow) squares indicate a high classification accuracy for that particular class. A strong (bright) diagonal for each confusion matrix shows that the model is doing well, and strong off-diagonal elements indicate that where the model is commonly misclassifying a particular class. The RefineNet-Rdm model (Fig. D.D.1(c)) is clearly overfitting to the classes with more training samples; but when the model is pre-trained with DIRSIG data (Fig. D.D.1(d); it does a better job classifying the test set.

Fig. D.2 shows the loss and accuracy curves for SharpMask-Sim (solid) and RefineNet-Sim (dashed). Fig. D.D.2(a) shows that the training and validation loss stay very close, which indicates that the model is not overfitting to the training data.

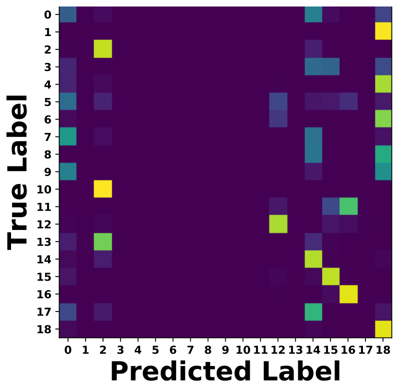
Fig. D.3 shows the class distribution of the generated DIRSIG dataset. This dataset is also unbalanced, which is why we use a weighted loss function (Equation 3.2) to pre-train the ResNet-50 DCNN.



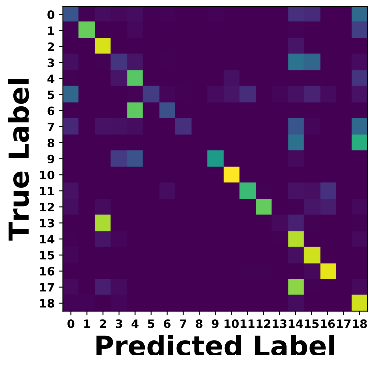
(a) Sharpmask-Rdm



(b) Sharpmask-Sim

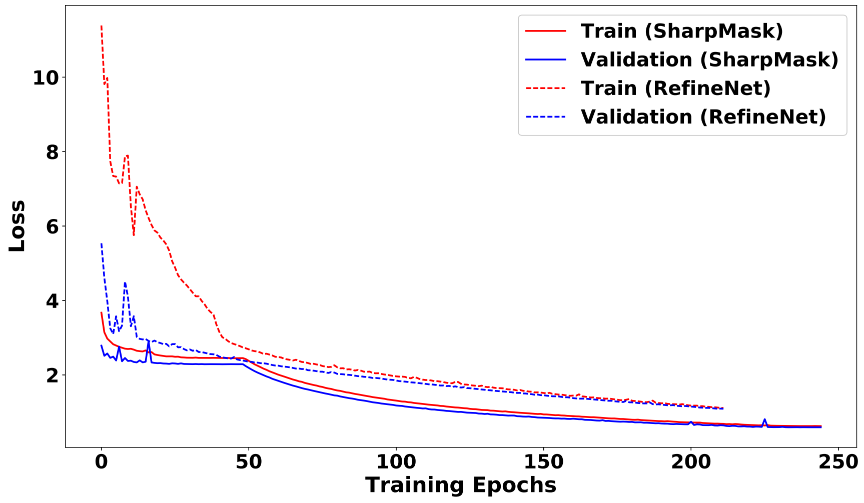


(c) RefineNet-Rdm

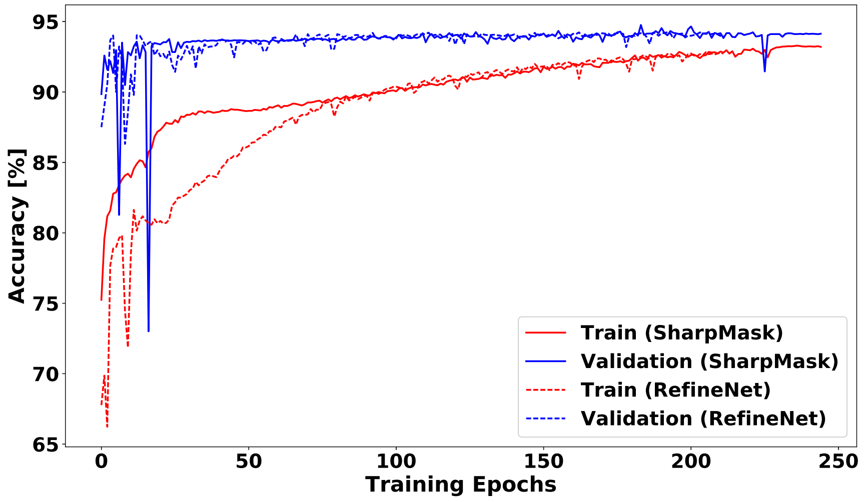


(d) RefineNet-Sim

Figure D.1: Heatmap visualization of the confusion matrices for all four models. Each row is normalized to itself to highlight the most common errors.



(a) Loss



(b) Overall Accuracy

Figure D.2: Training (red) and validation (blue) loss and accuracy plots for SharpMask (solid line) and RefineNet (dashed line) models with DIRSIG weight initialization.

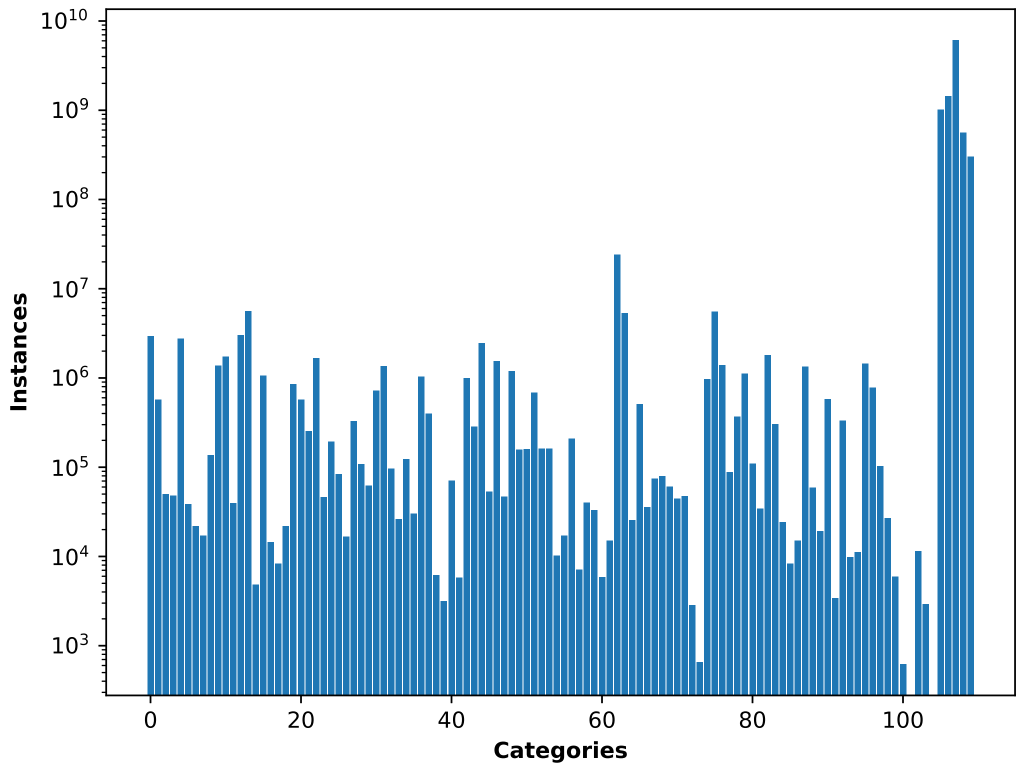


Figure D.3: Histogram of class distribution for DIRSIG training set.