CONVOLUTIONAL NEURAL NETWORKS FOR MULTI- AND HYPER-SPECTRAL IMAGE CLASSIFICATION

by

Jacob John Senecal

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

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

April 2019

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. John Sheppard for his support and advice throughout my time in the Numerical Intelligent Systems Laboratory (NISL). I have learned a great deal and become a better researcher throughout my time in the NISL lab.

I would also like to thank Dr. Dave Millman for serving on my committee, and Dr. Joe Shaw for all of the advice, information, and help throughout this project. I certainly know much more about optics and imagery now than when I started this research.

Finally, I would like to thank my fellow members of the Numerical Intelligent Systems Lab as well as my friends and family for all of their support and ideas.

Funding for this work was provided by the Intel Corporation through IL/ER/Montana-Intel SRA 10-18-17 and by the S2 Corporation. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the state of Montana or the Intel Corporation.

TABLE OF CONTENTS

TABLE OF CONTENTS – CONTINUED

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES – CONTINUED

ABSTRACT

While a great deal of research has been directed towards developing neural network architectures for classifying RGB images, there is a relative dearth of research directed towards developing neural network architectures specifically for multi-spectral and hyper-spectral imagery. The additional spectral information contained in a multi-spectral or hyper-spectral image can be valuable for land management, agriculture and forestry, disaster control, humanitarian relief operations, and environmental monitoring.

However, the massive amounts of data generated by a multi-spectral or hyper-spectral instrument make processing this data a challenge. Machine learning and computer vision techniques could automate the analysis process of these rich data sources. With these benefits in mind we have adapted recent developments in small efficient convolutional neural networks (CNNs), to create a small CNN architecture capable of being trained from scratch to classify 10 band multi-spectral images, using much fewer parameters than popular deep architectures, such as the ResNet or DenseNet architectures. We show that this network provides higher classification accuracy and greater sample efficiency than the same network using RGB images.

We also show that it is possible to employ a transfer learning approach and use a network pre-trained on multi-spectral satellite imagery to increase accuracy on a second much smaller multi-spectral dataset, even though the satellite imagery was captured from a much different perspective (high altitude, overhead vs. ground based at close stand-off distance). This results demonstrates that it is possible to train our small network architectures on small multi-spectral datasets and still achieve high classification accuracy. This is significant as labeled hyper-spectral and multi-spectral datasets are generally much smaller than their RGB counterparts.

Further we approximate a Bayesian version of our CNN architecture using a recent technique known as Monte Carlo dropout. By keeping dropout in place during test time we can perform a Monte Carlo procedure using multiple forward passes of our network to generate a distribution of network outputs which can be used as a measure of uncertainty in the predictions a network is making. Large variance in the network output corresponds to high uncertainty and vice versa. We show that a network that is capable of working with multi-spectral imagery significantly reduces the uncertainty associated with class predictions compared to using RGB images. This analysis reveals that the benefits of an architecture that works effectively with multi-spectral or hyper-spectral imagery extends beyond higher classification accuracy. Multi-spectral and hyper-spectral imagery allows us to be more confident in the predictions that a deep neural network is making.

CHAPTER ONE

INTRODUCTION

Sources of hyper-spectral and multi-spectral imagery are becoming increasingly prevalent. For example, the European Space Agency launched the Sentinel-2A satellite in June of 2015, and the Sentinel-2B satellite was launched in March of 2017. Both satellites are capable of imaging in 13 different spectral bands, ranging from 443 nm to 2190 nm [European Space Agency, 2018]. These satellites have a high revisit rate, with every area of Earth beneath these satellites' orbital path being imaged every five days. The full Sentinel constellation is expected to generate approximately 8 terabytes of data every day in operation. Further, the proliferation of small satellites (i.e. cubesats) is likely to add to the deluge of imagery that has high resolution both spatially and spectrally. The company Planet Labs currently has approximately 150 cubesats in orbit, imaging in both RGB and near infrared spectral bands [Baylor, 2018].

Ground and aerial based platforms also exist for acquiring high dimensional imagery. An example of such an instrument is the Resonon Pika L, used in some of the work presented in this thesis. The Pika L is small enough to be mounted on a small drone or can be used as a ground based platform. The Pika L is a hyper-spectral instrument that acquires even more spectral information than a typical satellite based multi-spectral system, capturing spectral bands ranging from 400 nm to 1000 nm at approximately two nanometer intervals, for a total of 300 spectral channels.

This type of data can be incredibly valuable in land management, agriculture and forestry, disaster control, humanitarian relief operations, and environmental

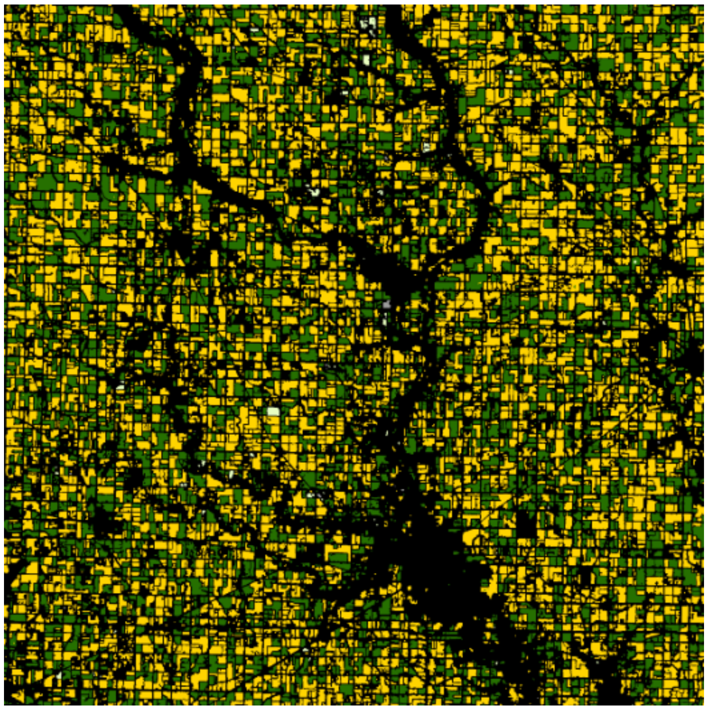


Figure 1.1: Predicting corn and soy yields in Humbolt county, Iowa using historical satellite imagery. Green is soy and yellow is corn. Source: [Descartes Labs, 2016]

monitoring [Miller et al., 2011]. For example, Descartes Labs, a spin-off from Los Alamos National Laboratory, has used a petabyte of freely available satellite imagery to train a machine learning model to predict corn and soy yields across the U.S., (Figure 1.1). [Nugent et al., 2018] combined hyper-spectral imaging with a support vector machine to discriminate between susceptible and herbicide resistant weeds, with the goal of enabling precision application of herbicides.

Processing and interpreting the amount of data that multi-spectral and hyper-spectral imaging generates is a challenge. To help mitigate these challenges machine learning and computer vision techniques could be used to automate the process of extracting useful information from the raw multi-dimensional imagery provided by sources such as the Pika L imager, the Sentinel satellite constellation, and NASA's LandSat mission.

1.1 Problem Statement

While the use of machine learning to process and classify generic RGB images has received a great deal of attention, and in general has been quite successful, much less energy has been devoted to extending machine learning models like convolutional neural networks (CNN) to process the type of multi-spectral and hyper-spectral imagery provided by space based, aerial, and ground based systems. One issue has been the lack of large, labeled training datasets for the variety of types of multi- and hyper-spectral imagery that can be acquired. This is in contrast to RGB images where there exist several large publicly available datasets such as ImageNet [Deng et al., 2009], and CIFAR-10 [Krizhevsky and Hinton, 2010]. Even given an application that requires more specific image classes or additional training data, large datasets like ImageNet and CIFAR-10 provide a useful starting point for researchers and practitioners dealing with RGB images, as these large datasets can be used to pre-train neural networks that can later be fine-tuned for more specific tasks.

The vast amount of available satellite imagery (the LandSat mission has been operating since the 70's), the multi-spectral nature of these datasets, and the insights that can be unlocked by combining machine learning with these datasets motivates the need for computationally efficient and effective machine learning models that can be applied to multi-spectral and hyper-spectral imagery.

In this work, we first adapt and extend convolutional neural network architectures originally developed with embedded systems applications in mind, to work well with multi-spectral imagery. We also show that if we have a convolutional neural network capable of effectively processing the full spatial-spectral input space from a multi-spectral image, we can make comparisons of the uncertainty characteristics of our models and how these characteristics vary when using multi-spectral imagery vs standard RGB images. This analysis relies on a new development known as Monte Carlo dropout [Gal and Ghahramani, 2016].

Finally, while convolutional neural networks have almost become the *de facto* method for image classification tasks, there still exist situations where it is difficult to deploy these models. Particularly in this thesis, we study a produce (i.e. fruits, vegetables) analysis task where we have a small number of multi-spectral images in each of the classes. Popular deep neural network architectures would likely overfit to this small of a dataset. When dealing with small datasets, it is common to turn to transfer learning [Caruana, 1995] and use a network trained on a different but related task as a starting point for training on the new target task. However, standard transfer learning approaches using ImageNet as the source domain are difficult or impossible to use in our case due to differences in architectures between a network trained on RGB images vs multi-spectral or hyper-spectral images. Instead we study the efficacy of transfer learning from multi-spectral satellite data to our target task, and we also test the feasibility of training our small network architectures from scratch.

1.2 Contributions

This work makes a number of contributions regarding the application of small and efficient neural network architectures to multi-spectral and hyper-spectral imagery, in addition to studying the uncertainty characteristics of deep neural network architectures when applied to multi-spectral imagery vs. RGB images.

- In this work we adapt recent developments in compressed, efficient convolutional networks (e.g., MobileNets [Howard et al., 2017] and SqueezeNets [Iandola et al., 2016]) to create a small (in terms of number of parameters) and computationally efficient convolutional neural network, capable of being trained to process the full spectral-spatial input space from multi-spectral imagery.

- We show that it is possible to train such a network end-to-end on multi-spectral imagery, using no data augmentation, that achieves greater classification accuracy and higher sample efficiency than an identical network trained on RGB images.

- We extend our analysis to a Bayesian version of the CNN architecture in question, using Monte Carlo dropout [Gal and Ghahramani, 2016], and show that with a multi-spectral network, epistemic uncertainty in the model is significantly reduced compared to an identical network using 3-channel RGB images. This allows us to be more confident in the predictions a multi-spectral network is making. This is the first analysis that we are aware of that studies how additional spectral channels impact uncertainty characteristics of deep neural networks.

- We demonstrate the application of our CNN architecture to a produce (e.g. fruits, vegetables) quality classification task.

- Additionally, we show the efficacy of transfer learning from multi-spectral satellite imagery to the produce task, as well as demonstrate the capability of our networks to train effectively on the small number of images present in the produce dataset.

1.3 Organization

The remainder of this thesis is organized as follows. In Chapter 2 we will cover necessary information to make the reader familiar with the various topics discussed in this thesis, including multi- and hyper-spectral imagery, neural networks, transfer learning, and uncertainty quantification.

Chapter 3 details the design and implementation of a custom convolutional neural network architecture for multi-spectral imagery. This network architecture is evaluated on the multi-spectral and hyper-spectral datasets also described in Chapter 3.

We then cover the problem of transfer learning from our multi-spectral satellite imagery dataset to a hyper-spectral produce evaluation dataset in Chapter 4.

Finally, in Chapter 5 we explore a method for quantifying the uncertainty of the predictions made by a deep neural network using a method based on Monte Carlo dropout, and to the best of our knowledge present some of the first work related to studying the uncertainty characteristics of a multi-spectral deep neural network.

CHAPTER TWO

BACKGROUND

This work combines aspects of the fields of remote sensing and machine learning; specifically, multi-spectral and hyper-spectral imagery, convolutional neural networks, and fully connected feed-forward networks.

2.1 Hyper-Spectral & Multi-Spectral Imagery

A hyper-spectral or multi-spectral imager collects information from a wide range of the electromagnetic spectrum in contrast to a common camera that generally only acquires a narrow band of wavelengths in the blue range (440 nm - 485 nm), the green range (500 nm - 565 nm), and the red range (625 nm - 740 nm). Hyper-spectral and multi-spectral images can be characterized by their spatial and spectral resolution. Spatial resolution refers to the real-world spatial distance covered by a single pixel in an image, and spectral resolution refers to the number of acquired wavelengths as well as the spacing and width of those wavelengths. At each pixel in the image, a hyper-spectral or multi-spectral instrument acquires the light intensity (radiance) for a large number of wavelengths. A radiance measure can be converted to a reflectance value ranging from 0 (no light reflected at the given wavelength) to 100 (all light reflected at the given wavelength).

While multi-spectral and hyper-spectral imagery are very similar to one another, they can be distinguished by the width of the spectral bands acquired and how contiguous the acquired spectral bands are. A hyper-spectral imager acquires narrow spectral bands at a much finer spectral spacing (usually close enough to be considered contiguous) than a multi-spectral imager that generally acquires wider spectral bands

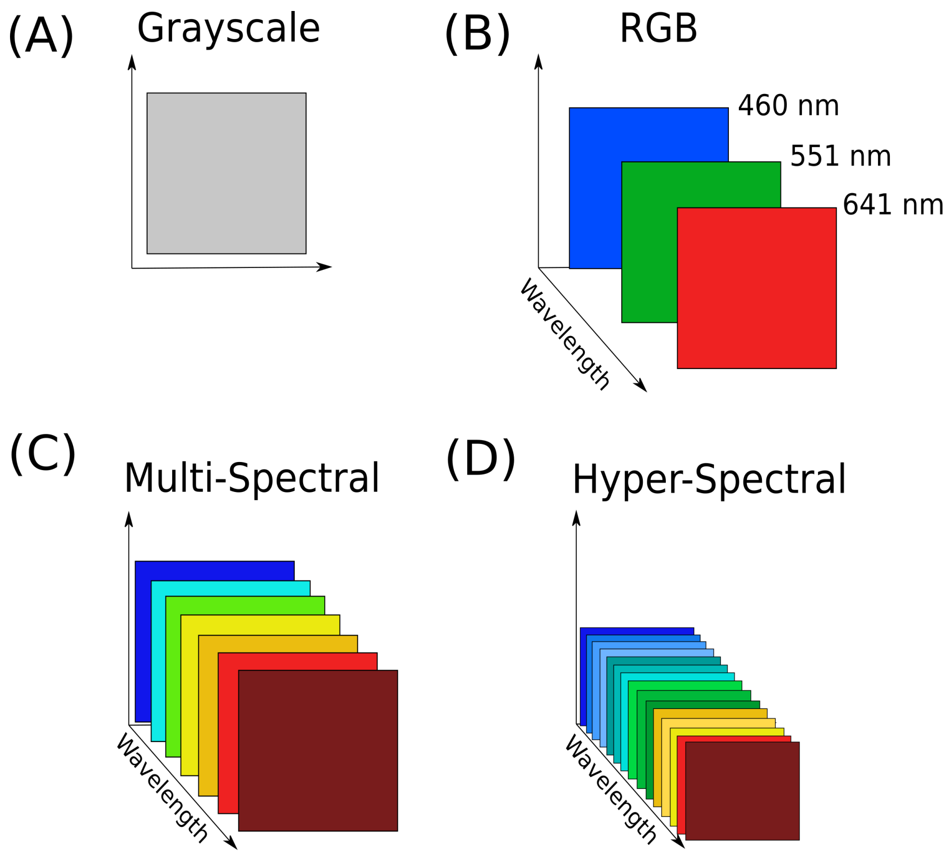


Figure 2.1: (A) An example of a single channel grayscale image. (B) A data cube representation of an image with three spectral channels: red, green, and blue. (C) A multi-spectral image with many more wavelengths compared to RGB, but with significant gaps between different channels along the wavelength axis. (D) A hyper-spectral image consisting of finely spaced spectral bands.

at a smaller number of targeted wavelengths.

Regardless of whether we are dealing with a multi-spectral or hyper-spectral instrument, when an image is captured the resulting data structure is what is known as a “data cube,” consisting of spatial x and y dimensions and the third wavelength dimension, λ. Figure 2.1 illustrates the “data cube” and the difference between a multi-spectral and hyper-spectral image. In Figure 2.2 we can see a reflectance curve for a single pixel in the image, highlighted in red. These reflectance curves can be used for a variety of an analyses, including pixel-wise segmentation and classification.

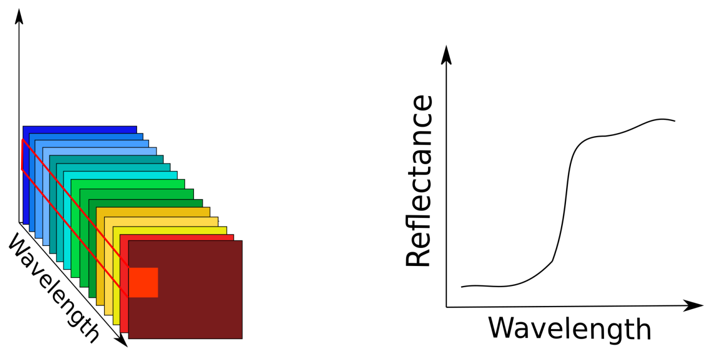


Figure 2.2: A reflectance curve can be produced from each pixel in a hyper-spectral image.

2.1.1 Acquisition Methods

A hyper-spectral or multi-spectral image can be acquired in a variety of manners that can generally be divided into four different categories: spatial scanning, spectral scanning, spatial-spectral scanning, and non-scanning.

A spatial scanning instrument acquires a 2-D strip of the scene consisting of a single spatial dimension and the wavelength dimension. These types of systems are often called "pushbroom scanners" as they repeatedly collect 2-D strips of a scene using some form of mechanical stage to pan the instrument across the scene of interest to form the final spatial dimension of the data cube. Alternatively, depending on the application, the scene to be imaged may scroll past the instrument, as is the case if we are imaging items on a conveyor belt or the instrument is mounted on an aerial or space based platform.

In a spectral scanning system each instrument output contains the two spatial dimensions and a single wavelength, forming a monochromatic image. The instrument performs spectral scanning by switching optical filters to select different wavelengths of light from the spectrum.

Spatial-spectral scanning is a fairly recent development, using a slit spectroscope to acquire a diagonal slice along the wavelength dimension of a hyper-spectral data cube [Grusche, 2014]. To get the full spectrum of a single pixel, repeated scanning is still required. Refer to Figure 2.3 for a visualization of how the data cube is collected.

A non-scanning system acquires the full data cube directly, including the two spatial dimensions and the full wavelength dimension. This type of imager is sometimes called a "snapshot" instrument for its ability to acquire the full data cube at once. This method can capture images faster than the other methods, but there is a trade-off between spectral or spatial resolution, in addition to higher computational and manufacturing costs.

2.1.2 Application Areas

Hyper-spectral and multi-spectral imaging has application in a wide variety of fields, including geology, medicine, physics, and biology among others. The advantage of a multi-spectral or hyper-spectral image compared to an RGB image lies in the numerous and varied responses to different wavelengths of the electromagnetic spectrum that different materials exhibit beyond just the visual range. The features present in the reflectance characteristics of a particular object or material can be indicative of the type of material or specific attributes of a given object or material.

Multi-spectral and hyper-spectral remote sensing applications originated with NASA in the early 1970's with the launch of the Landsat-1 satellite [Goetz, 2009]. Initially the primary use case for this type of imagery was the detection and mapping of different types of minerals and materials on the surface of the Earth. As alluded to earlier, different types of materials typically exhibit varied responses at different wavelengths such that a measured reflectance curve can form a type of "fingerprint" for a particular material [Clark et al., 1990].

Multi-spectral and hyper-spectral imagery is also useful for assessing the

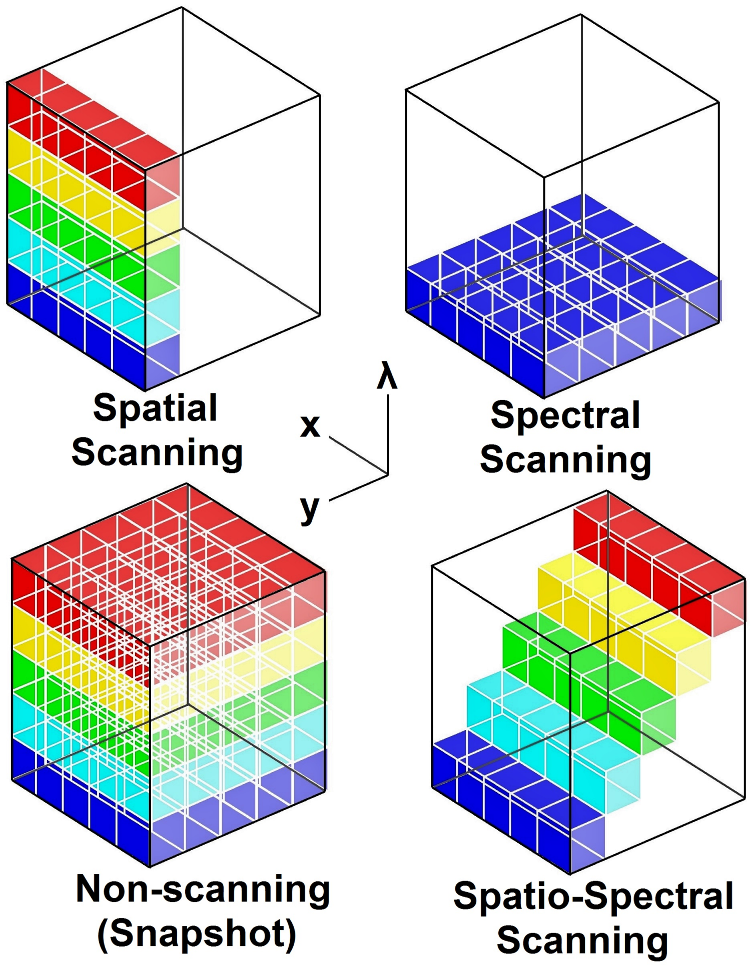


Figure 2.3: The difference in data cube acquisition using spatial scanning, spectral scanning, spatial-spectral scanning, and a non-scanning method.

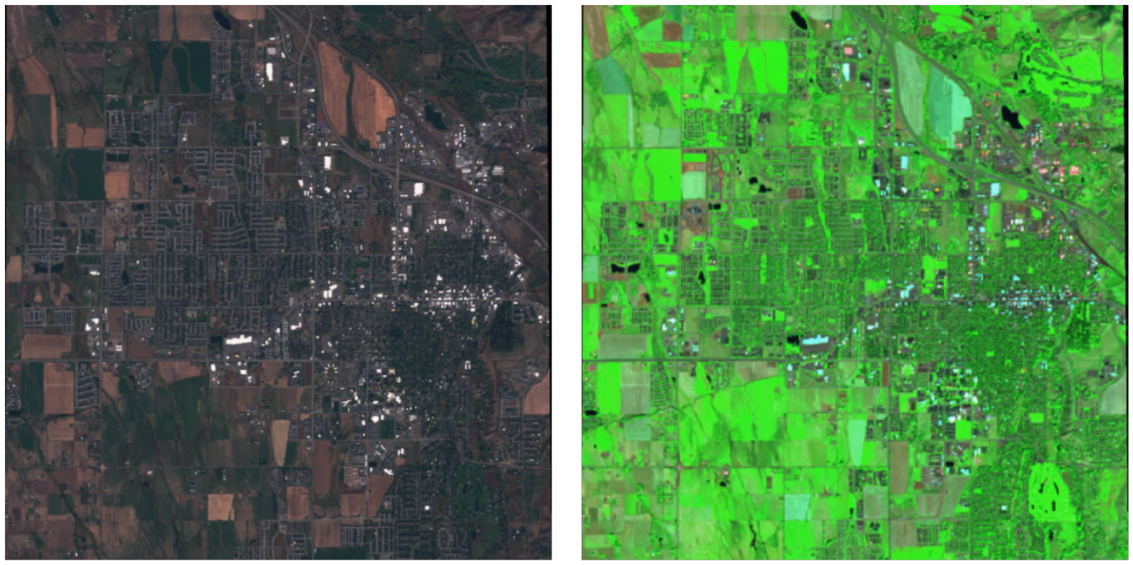


Figure 2.4: Bozeman, MT viewed in RGB on the left and short wave infrared (SWIR) on the right. Moisture levels in vegetation can be particularly sensitive to SWIR spectral bands.

properties of vegetation. Vegetation here refers to a broad class of land cover types that can be observed from a remote sensing platform, as well as individual plants that may be observed at much closer stand-off distances. Reflectance characteristics of vegetation can be sensitive to physiological properties distinguishing different species of plants, in addition to the various chemical and biological processes naturally occurring in plants, such as chlorophyll levels and water concentration that can provide indicators of vegetation health [Govender et al., 2007]. For example, in Figure 2.4 we can see a satellite view of Bozeman, MT viewed in RGB wavelengths on the left, and a short wave infrared wavelength on the right. Short wave infrared bands are sensitive to moisture levels in vegetation. This is apparent in Figure 2.4 where boundary lines between fields and development is more apparent.

The use of hyper-spectral imaging in medical applications has also been growing. When light strikes biological materials, absorption at different wavelengths is primarily governed by the concentrations of factors such as hemoglobin, melanin, and water. In some cases these tissue properties change during the progression of disease, allowing hyper-spectral imaging to provide a useful diagnostic [Lu and Fei, 2014].

Hyper-spectral imaging has even been applied in the area of art conservation to study such properties as material composition and degradation, in addition to pigment identification [Liang, 2012].

2.2 Feed Forward Neural Networks

Parts of this work apply feed forward neural networks [Rumelhart et al., 1986] to make classification or regression predictions based on an input spectral reflectance curve. A neural network can be thought of as a series of nested mathematical functions transforming an input into some output. We will first give a more intuitive high level description of the operations taking place in a neural network, followed by a more precise mathematical example.

Figure 2.5 illustrates a fully connected, feed forward neural network with one internal layers, known as a “hidden” layer, in addition to the input and output layer. A network that is fully connected means that each neuron or node (the circles in Figure 2.5) is connected to each node in the subsequent layer. The terms neuron and node are used interchangeably in the context of artificial neural networks. The descriptor “feed forward” refers to a network that only propagates input information to subsequent layers from input to output along the network. A node will not pass information to a node in the same layer or a node in a previous layer.

If we view what is happening at each node in a neural network in detail such as the one shown in Figure 2.6, we can see that it consists of a linear combination of the inputs to the neuron multiplied by the network “weights,” followed by an activation

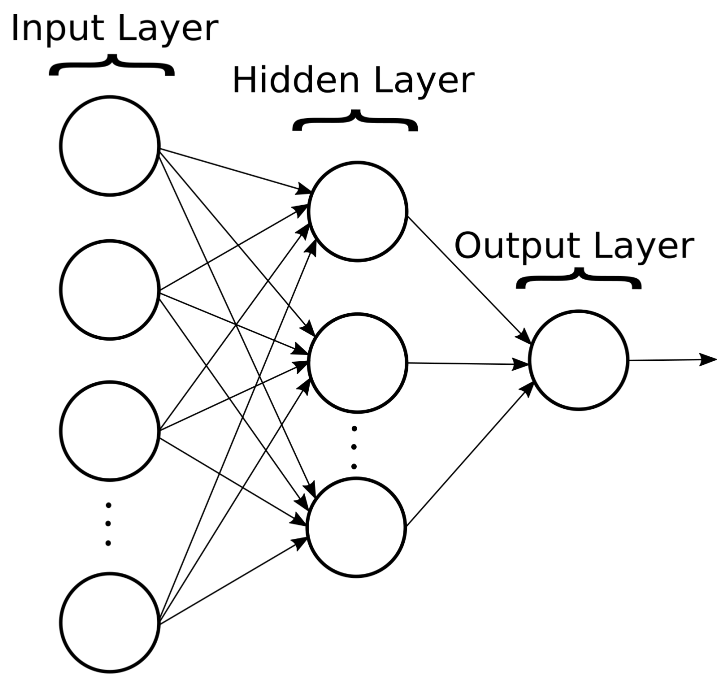


Figure 2.5: A fully connected, feed forward neural network with one hidden layer.

function.

For the example in Figure 2.6, the mathematical expression would be,

$$o_j = \psi\big(\sum_{i=1}^{n} x_i w_i\big), \tag{2.1}$$

where ψ denotes the activation function. An affine transformation known as a "bias," is not shown in Figure 2.6, but it can also be applied in Equation 2.1 such that the final equation would be,

$$o_j = \psi\big(b + \sum_{i=1}^{n} x_i w_i\big), \tag{2.2}$$

where b is the bias. The bias is another trainable parameter similar to the weights in a network.

The activation function is an essential component of artificial neural networks, transforming a purely linear operation into a non-linear one, and giving neural

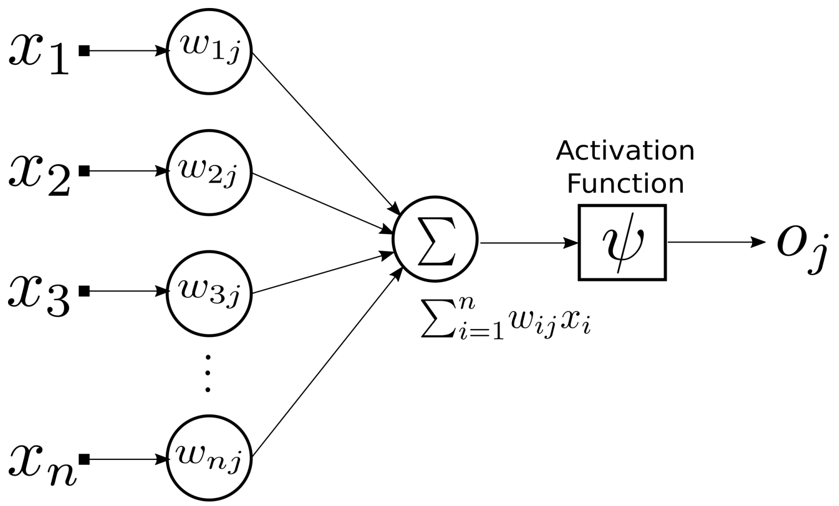


Figure 2.6: The mathematical operations occurring at a neuron in an artificial neural network

networks the ability to be universal function approximators [Hornik et al., 1989]. Common activation functions include the sigmoid, the hyperbolic tangent, the more recently developed (and currently most popular) rectified linear unit (ReLU), and variants of the ReLU like the leaky ReLU. Common activation functions are illustrated in Figure 2.7.

When we refer to the training of neural networks we are referring to the adjustment of the weights, w_i, in Equation 2.2 in order to minimize error at the

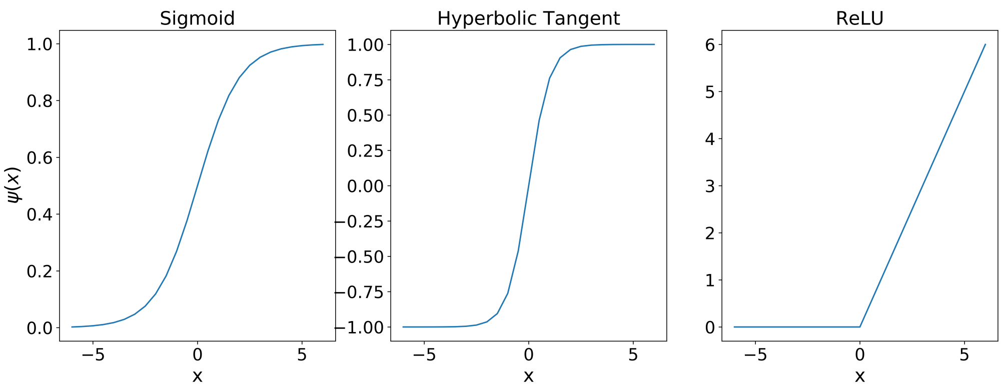


Figure 2.7: Commonly used activation functions in neural networks.

output of the neural network, whether that be for a regression or classification task.

Now we that we have an overview of the operations taking place in a neural network operation, we can write all of these operations in terms of matrices for a one layer network. The conventions shown here easily generalize to more layers. We will use $\mathbf{x}$ to denote an input vector of length N. Let $\mathbf{W_1}$ denote a weight matrix with N rows and M columns, and $\mathbf{b}$ to denote the bias vector of length M, such that the input $\mathbf{x}$ is transformed as, $\mathbf{xW_1} + \mathbf{b}$. Following this operation there is a final linear combination computed to map the hidden layer to an output vector with length C. $\mathbf{W_2}$ will denote the final weight matrix with M rows and C columns. A forward pass through the network then consists of,

$$\hat{y} = \psi(\mathbf{x}\mathbf{W}_1 + \mathbf{b})\mathbf{W}_2,$$

where ψ again denotes an element-wise activation function.

When training a neural network, we would then compute the error of the output of the network. During training we split our dataset into a train set $\mathbf{X}_{\text{train}}$, $\mathbf{Y}_{\text{train}}$ and a test set $\mathbf{X}_{\text{test}}$, $\mathbf{Y}_{\text{test}}$ [1]. With $\mathbf{X}$ denoting the input set and $\mathbf{Y}$ denoting the correct output for a regression task or correct label for a classification task[2]. A variety of loss functions can then be used to measure the error of our network. For a regression task we might use mean squared error for example,

$$\mathbf{E}^{\mathbf{W}_1,\mathbf{W}_2,\mathbf{b}}(\mathbf{X},\mathbf{Y}) = \frac{1}{N}\sum_{i=1}^{N} ||\mathbf{y}_i - \hat{\mathbf{y}}_i||^2 \tag{2.3}$$

where $\{\mathbf{y}_1, ..., \mathbf{y}_N\}$ are the N true outputs and $\{\hat{\mathbf{y}}_1, ..., \hat{\mathbf{y}}_N\}$ are our N model outputs.

[1]The notation here assumes images as input which are matrices. Other types of input may be vectors which would be denoted with a lower case variable.

[2]In practice we also typically have a validation set to tune our model, or may perform something like cross-validation for training in addition to facilitating statistical testing.

For a classification task where we are attempting to label our inputs as one of C classes or categories, cross-entropy is a commonly used loss function,

$$\mathbf{E}^{\mathbf{W}_1,\mathbf{W}_2,\mathbf{b}}(\mathbf{X},\mathbf{Y}) = -\sum_{c=1}^{C} \mathbf{y}_{i,c} \log \hat{\mathbf{y}}_{i,c} \tag{2.4}$$

where the loss is summed across all C output classes.

Using a process known as "backpropagation," we can then compute the gradient of our loss function with respect to the weights in our network and adjust the network weights such that the error of the network would be lower if we were to feed an input into our network again. By iteratively performing this process over preferably a large number of training examples, we hope to produce a network that will generalize well to unseen test data. In practice generalization depends in large part on how well the training set distribution approximates the true distribution of the underlying process generating the data we are attempting to make predictions over.

2.3 Convolutional Neural Networks

Convolutional neural networks [LeCun et al., 1989] are a more specialized type of neural network originally developed for image recognition tasks, though they have also been used in audio processing and natural language processing applications, [Zhang and Wallace, 2015].

If we were to apply the fully connected networks from Section 2.2 to a generic RGB image of a fairly small size (e.g. 256×256 pixels), each node in the network would have $256 * 256 * 3 = 196,608$ weights. The number of parameters in this type of network architecture would balloon quickly to number in the several millions.

A convolutional neural network instead constrains the number of parameters in the network in a sensible way. In addition to the fully connected layers discussed

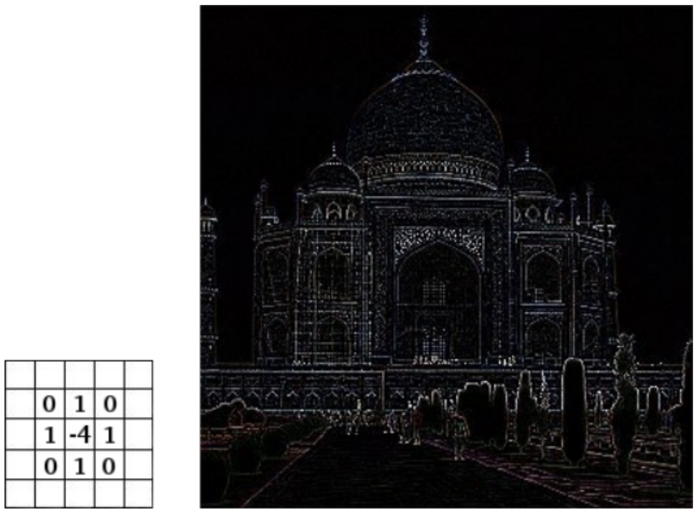


Figure 2.8: An edge detection filter applied to an image. The matrix on the left is the filter itself.

in the previous section, a convolutional neural network (CNN) primarily consists of convolution layers, and pooling layers. The convolutional layers are composed of what are alternately known as filters, kernels, or feature detectors. The filters are simply matrices of weights similar to the weights in a fully connected network.

The filters are sometimes referred to as feature detectors as they extract or detect certain details of images. For example Figure 2.8 shows the result of an edge detection filter applied to an image. Certain algorithms like the Sobel operator manually set the filters that are used. CNN's in contrast learn the filters (the weights of the network) through training by backpropagation in the same manner as a fully connected neural network.

However, in contrast to the weights in a fully connected neural network, a convolutional neural network makes use of weight sharing. That is, each network weight is not connected to a single input but is rather connected to multiple inputs. See the example of a one dimensional filter in Figure 2.9. This filter slides over the pixels in the image and is convolved with each successive group of 3 inputs.

The general formulation for a convolution between two functions f and g is,

$$(f \circledast g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau.$$

We use the symbol "$\circledast$" to denote the convolution operator, and we use "$*$" to denote the multiplication operator. Convolution produces a third function that describes how the behavior of one function, f is modified by the other, g.

In the context of a CNN, the convolution operation is two-dimensional and discrete, and it can be defined as,

$$(f \circledast g)(x, y) = \sum_{\tau=-a}^{a} \sum_{\gamma=-b}^{b} f(\tau, \gamma)g(x-\tau, y-\gamma)$$

The operation essentially consists of an element-wise multiplication between the weights and the pixel values followed by a summation to produce a single value for each convolution operation.

The convolutions preserve the spatial information of an input. That is, information regarding the relative placement of pixels in an image is preserved by the convolution operation. The distance along the input that the filter slides after each convolution operation is known as the stride. The spatial dimensions of the filter are sometimes referred to as the receptive field of the filter. Figure 2.9 shows an example of a one-dimensional discrete convolution. The filter slides across the input with stride $= 1$ to produce the output on the left, while the filter is applied with stride $= 2$ to produce the output on the right. So on the left the first operation would be $[-1, 0, 2] \circledast [1, 0, -1] = -1 * 1 + 0 * 0 + 2 * -1 = -3$, then the convolution strides to the right by one input such that the next operation is $[0, 2, -2] \circledast [1, 0, -1] = 0 * 1 + 2 * 0 + -2 * -1 = 2$, and so on.

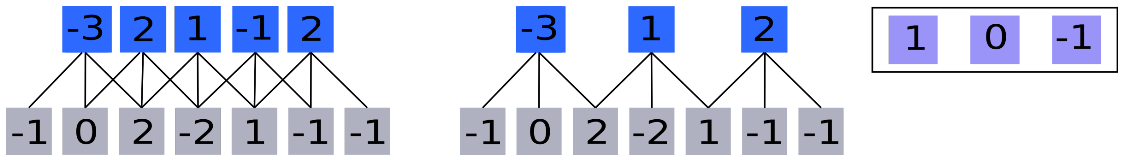


Figure 2.9: A one-dimensional discrete convolution operation.

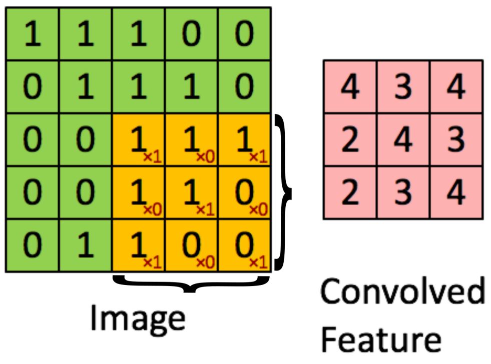


Figure 2.10: A convolution kernel applied to a matrix. If we were to perform an elementwise multiplication with the matrix input in the shaded region delineated by the braces with the small numbers in the bottom corners of the cells in the shaded region and sum the result we would get 4, the result in the bottom right box of the convolved feature.

In figure 2.10 we see an example of a two dimensional filter applied to an input matrix. The filter is denoted by the small numbers in the bottom corner of the part of the input matrix that is shaded and delineated by the braces.

Each convolutional layer also incorporates non-linear activation functions applied to the result of the convolution operation. Any activation function that can be applied to a fully connected network can also be used in a convolutional neural network.

The second main type of layer in a convolutional neural network is the pooling layer. A pooling layer serves to aggregate feature detectors, and is what provides a CNN with invariance to translation, and rotation in images [Goodfellow et al., 2016]. Pooling is a sub-sampling technique, and the most common pooling operations are max pooling and average pooling. Similar to the filters in the convolutional layers,

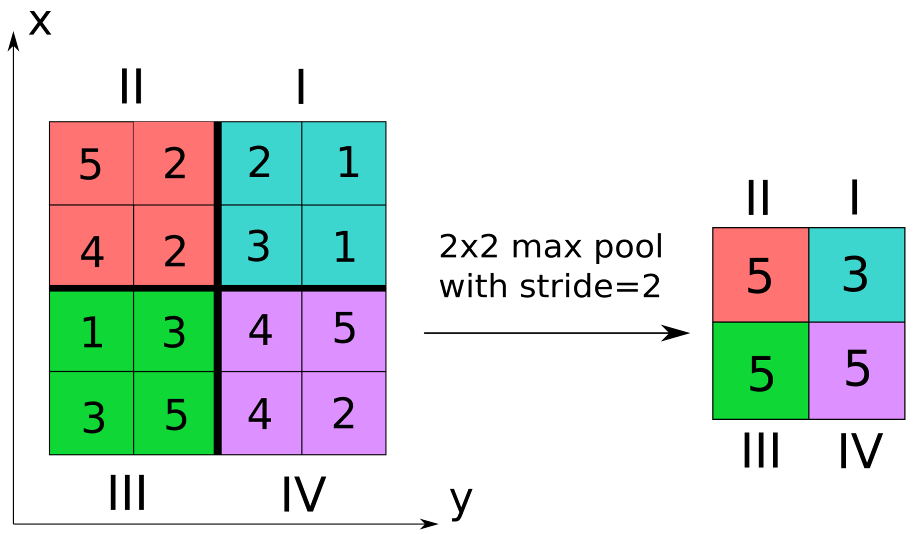


Figure 2.11: A max pooling operation applied to one channel of an input. We see that in quadrant I the maximum value is 3, in quadrant II it is 5, in quadrant III it is 5, and in quadrant IV the maximum value is 5.

a pooling filter also has spatial extents, typically similar in size to the filters in the convolutional layers. However, instead of performing convolution a max pooling layer will take the maximum value from the area of the input covered by the pooling filter dropping all others. An average pooling layer will average all values of the input covered by the pooling filter. Figure 2.11 gives an example of how maximum pooling would be applied to an input. Average pooling operates in exactly the same manner, except that instead of taking the maximum value the average of the values in each shaded region would be taken.

In addition to serving as feature aggregators, pooling layers also downsample the input, further reducing the number of parameters in the neural network. The pooling operation can also be used to ensure that we have a fixed size output. Regardless of an input filter size, a pooling operation could be used to ensure we have a single value from each input.

The final concept we will discuss is the existence of channels in a CNN. When a CNN receives a typical RGB image as input there is not just one matrix of pixel values, there are three: the red channel, the green channel, and the blue channel. So our filters are not two-dimensional but three-dimensional and are applied to all input channels simultaneously, but they slide across the two spatial dimensions of the input. Figure 2.12 shows a single convolutional filter without pooling being applied to an input RGB image. The filter has dimensions of $3 \times 3 \times 3$ pixels. Imagine the filter placed on a portion of the input image such that a 3×3 spatial region is covered and all input channels are covered by the filter. We then perform element-wise multiplication among the weights in the filter and the pixels covered by those weights then sum all these values to produce a single final output, effectively collapsing the three channel input into a single channel. The filter then repeatedly slides to a new location specified by the stride parameter until the full 2D spatial dimensions of the image have been covered. Notice that the application of the filter results in a two-dimensional output, as each convolution operation results in a single output from the three channel input.

In each convolutional layer we will have many filters, each resulting in a two-dimensional output. The two-dimensional outputs of the filters are then stacked to form the input to the next convolutional layer, such that the depth of the input into a subsequent layer will be equal to the number of filters in the previous layer.

A typical CNN architecture consists of multiple convolution layers followed by non-linear activation functions, and interspersed with pooling layers. Fully-connected networks are commonly added at the end of the network with a final softmax output layer to produce the classification predictions. The softmax layer is used to produce a valid probability distribution over the outputs of the network and is defined as,

$$S(\mathbf{y}) = \frac{e^{\mathbf{y}}}{\sum_j e^{\mathbf{y}_j}}$$

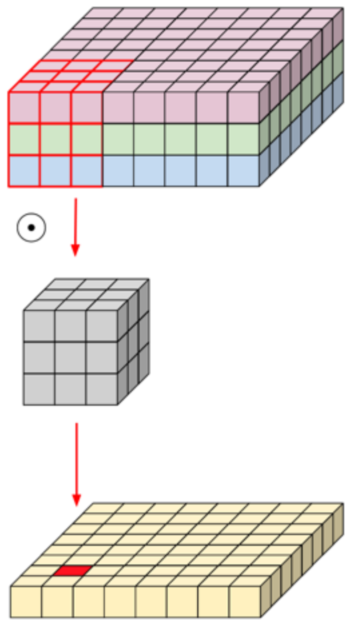


Figure 2.12: A convolutional filter applied to an RGB image input.

where $\sum_j$ represents the sum over the individual elements of the vector $\mathbf{y}$.

Over the past few years, a large number of variations and modifications have been proposed to this basic architecture. For example Google's Inception network [Szegedy et al., 2017] supports multiple parallel columns of convolutional layers that concatenate filters from the different columns at multiple points along the network. The residual network architecture (ResNet) [He et al., 2016] incorporates skip connections, forwarding earlier layer inputs to be concatenated to the inputs of later layers along the network. The SqueezeNet architecture [Iandola et al., 2016], which we will discuss in detail later, does not use a fully connected layer on top of the convolutional layers. In this work we focus specifically on how convolutional neural network architectures can be made to be computationally efficient, how to avoid overfitting models on small datasets, and how such architectures might be adapted to work well with multi-spectral and hyper-spectral images.

2.4 Transfer Learning

At a high level, transfer learning is the process of transferring knowledge from a previously learned task to some new related task [Caruana, 1995]. In the context of neural networks, transfer learning typically involves first training a network on a *source* task, and the learned feature detectors from the source task are then transferred to a different but related *target* task.

Learned feature detectors are transferred by using the weights of a network trained on the source domain to initialize the weights of the network to be trained on the target domain, such that the new target network does not have to be trained completely from scratch. This form of transfer learning relies on the generality of low level features in neural networks. For example, the initial layers in a neural network have been observed to form general feature detectors that are similar to color gradient detectors, edge detectors, etc. [Yosinski et al., 2014]. These low level feature detectors are applicable to a variety of tasks, and as such they typically have good transferability. As we proceed through a network towards the output layers, feature specificity increases until we arrive at the final softmax layer (for a classification task), at which point each neuron is entirely specific to an individual class [Yosinski et al., 2014].

When transferring the weights learned by the source network, we copy the weights of up to the first $l - z$ layers of the source network into the target network. The remaining z layers of the source network are discarded, and the target network replaces the final z layers with some number of untrained layers. We can then choose to freeze the weights from the pre-trained network and train only the newly added layers on the target domain, or we may backpropagate errors into the pre-trained weights as well to fine tune them.

Transfer learning is an attractive approach when the dataset for a task of interest is small or computational resources are limited. Transfer learning allows us to take advantage of the computational time that has already been expended to train a network from scratch on a large, diverse dataset. Networks that have been previously trained are referred to as "pre-trained" networks in this context.

If we desire to use transfer learning to train a new network, this imposes certain constraints and considerations. It is possible to train on images that are of a different size than what the source domain network was originally trained with. Convolutional layers can be applied to images of any size as long as the stride and the filter size fit the image. As we are replacing the final layers of our target network during the transfer learning process, we can adapt that layer to handle a different number of inputs from the original pre-trained network, and we can update the final layers to output a different number of class predictions corresponding to the new task in our target domain.

However, we are limited by the pre-trained network architectures available to us. If using a pre-trained network, we cannot remove intermediate network layers nor accept inputs with a different number of channels than what the original network was configured for. For many applications, this does not pose a serious constraint. The pre-trained models that are available to researchers and practitioners through libraries like Tensorflow and PyTorch have been trained exclusively on ImageNet [Krizhevsky et al., 2012], which consists of RGB images. When transferring to a target task where the input domain is also RGB images, ImageNet has proven to be an excellent source domain. When the target domain consists of something other than RGB images, like the multi-spectral and hyper-spectral applications considered in this thesis, transferring from a network pre-trained on ImageNet limits the target domain network to three channel inputs. We argue in a later chapter that while using

ImageNet as the source domain for multi-spectral and hyper-spectral applications can be useful, this is not a desirable situation as we discard massive amounts of useful information that is acquired by a multi-spectral or hyper-spectral instrument.

2.5 Uncertainty Quantification

In Chapter 5, we will discuss a possible way of quantifying uncertainty in the predictions that a neural network is making. In general, uncertainty quantification deals with capturing or quantifying the variability or error that may be present in some process. In numerical simulations for example, small differences in initial conditions can lead to a variety of output distributions. Uncertainty quantification might then seek to quantify the variance in output distributions that may be present [Bose et al., 2006].

Uncertainty quantification can also refer to assigning confidence levels to model predictions. We may want our model to provide some indication if has received an input that is significantly outside of the training distribution.

Uncertainty can generally be categorized into aleatoric and epistemic uncertainty [O'Hagan, 2004]. Aleatoric uncertainty is the stochasticity present in a system and is sometimes referred to as irreducible uncertainty, corresponding to things such as measurement noise or random motion of a measurement platform. Epistemic uncertainty refers to reducible uncertainty, such as uncertainty in a model's parameters that could be improved with the collection of additional data. The focus in Chapter 5 is determining how epistemic uncertainty of a deep learning model changes when using RGB images vs. multi-spectral or hyper-spectral images.

Multiple methods for quantifying uncertainty have been developed, including Monte Carlo simulation [Hastings, 1970], fuzzy methods [Klir and Yuan, 1995], and functional expansion based methods such as polynomial chaos expansion [Xiu and

Karniadakis, 2003] among others. The uncertainty method used in Chapter 5 is a Monte Carlo method that uses repeated random sampling to develop an estimate of variability in a process.

CHAPTER THREE

AN EFFICIENT CNN ARCHITECTURE FOR MULTI-SPECTRAL IMAGE CLASSIFICATION

Several issues compound when attempting to apply convolutional neural networks to multi-spectral or hyper-spectral imagery. It is very difficult to train a deep convolutional neural network, which has a large number of parameters, on a typically small, labeled, multi-spectral image dataset without overfitting to the training data. The increased number of spectral channels in a multi-spectral or hyper-spectral image compared to a three channel RGB image increases the number of network parameters significantly, further increasing the already significant amount of computational resources required to train a typical deep CNN. For example, 300 channel hyper-spectral images that we have acquired as part of our study of the aging characteristics of produce in grocery stores are ≈ 1 gigabyte in size, much larger than the few megabytes of an average RGB image.

3.1 Related Work

Previous approaches to creating CNNs for multi-spectral or hyper-spectral imagery have focused on fine-tuning CNNs that were pre-trained on RGB images, typically the ImageNet repository. Three band combinations had to be selected manually from multi-spectral images due to the architectural constraints of the original RGB network. If a new three band combination was desired, the network had to be re-trained [Xie et al., 2015, Penatti et al., 2015, Castelluccio et al., 2015]. This approach discards a significant portion of the spectral information present in a multispectral image, almost defeating the purpose of acquiring high dimensional imagery in the first place.

When only selecting three-band combinations, the number of useful patterns and correlations between spectral channels that can be extracted and exploited for classification is severely reduced. Further, the imaging systems that acquire multi-spectral and hyper-spectral imagery are much more expensive than a common camera capturing the red, green, and blue bands in the visible spectrum. If the algorithm or model we are using for analysis, in this case a convolutional neural network, is not using the additional spectral information provided by multi-spectral and hyper-spectral imagers, then the added cost of acquiring the additional spectral channels is not justified.

Other approaches that attempt to use more than three spectral bands often employ complex, multi-stage strategies to separate individual bands, reduce the dimensionality of the spectral bands, then perform some form of concatenation of the transformed spectral bands for later stages of a classifier [Santara et al., 2017, Chen et al., 2015b]. If we could have a small network architecture that could be trained from scratch on multi-spectral images, we could extract features from all spectral bands simultaneously, more effectively utilizing the information present in all of the available spectral bands, while at the same time keeping computational costs low. A small network with few parameters would also reduce the risk of overfitting to the smaller labeled datasets that are common with high dimensional imagery.

CNNs have been used to classify individual pixels in remote sensing imagery by performing 1-D convolutions on the spectrum of each pixel [Slavkovikj et al., 2015]. A 1-D convolution works similarly to a 2-D convolution except the filter is a vector that is convolved with another vector. An example is shown in Chapter 2, Figure 2.9. In [Chen et al., 2015a] this work was extended to be a spectral-spatial approach by applying 1-D convolutions on a flattened vector formed from the spectra of a group of adjacent pixels. None of these approaches maintains the physical distribution of spatial features when performing inference; the shape of spatial features is completely lost.

[Zhao and Du, 2016] used compressed spectral features from a local discriminant embedding method that were concatenated with spatial features from a CNN and fed into a multi-class classifier. In [Santara et al., 2017], a relatively complex network was created that split a high-dimensional image into separate channels that were then fed to individual sub-networks to learn band specific features. These features were then concatenated and fed into a series of fully connected layers for final classification.

3.2 Adapting Efficient Architectures

The primary difficulties in working with multi-spectral imagery lie in its high dimensionality and the lack of large labeled datasets for training. Both of these factors make training recent deep network architectures from scratch on multi-spectral imagery difficult, often to the point of being impractical. A modern deep architecture such as a ResNet [He et al., 2016], or DenseNet [Huang et al., 2017], contains enough parameters to overfit to a small dataset easily. The high dimensionality of multi-spectral imagery can also increase computational requirements significantly.

To overcome these issues, we adopt strategies that maximize accuracy with a limited budget of parameters. Several relevant strategies to this goal are presented in recent work on small and efficient convolutional neural networks (e.g. SqueezeNets [Iandola et al., 2016] and MobileNets [Howard et al., 2017]). While both SqueezeNets and MobileNets were conceived originally as a way to reduce the number of parameters in convolutional neural networks and make them more efficient, we hypothesize that, in addition to improving efficiency, some of the techniques used in those architectures will also be effective for processing high dimensional imagery.

3.2.1 The SqueezeNet Architecture

The original motivation for the development of the SqueezeNet architecture was to create a network with a small number of parameters that could still achieve comparable classification accuracy to a much larger network. The authors developed this network with deployment on embedded systems in mind.

SqueezeNet employs a module design. A module is a higher level building block, consisting of multiple convolution layers with a fixed organization. Multiple modules can be combined with other custom individual layers if desired to create a convolutional neural network. A network made up of modules simplifies the design of the network by limiting the number of manual selections required to define filter dimensions for every layer in a network.

Beyond the module design, SqueezeNet incorporates three main architectural design strategies.

1. Make the majority of filters in the network 1×1, rather than 3×3 since a 1×1 filter has nine times fewer parameters than a 3×3 filter.

2. Decrease the number of input channels to layers made up of 3×3 filters. Since the total number of parameters in a layer is (number of input channels) * (number of filters) * (3*3) limiting the number of input channels into layers made up of 3×3 filters can significantly reduce the number of parameters in a network.

3. The final strategy is to downsample late in the network to maintain large activation maps. Past research has shown that delayed downsampling results in higher classification accuracy [He and Sun, 2015].

SqueezeNet is made up of “Fire” modules which use the strategies outlined above. A Fire module consists of a “squeeze” layer and an “expand” layer. The squeeze layer is solely comprised of 1×1 filters. This layer computes a channel-wise linear combination from the inputs to the layer to reduce the total number of channels that are fed to the subsequent expand layer. The expand layer includes a mix of 1×1 filters and 3×3 filters. By including a mix of filter dimensions many feature detectors can be included in an expand layer while limiting the number of parameters relative to a layer with the same number of feature detectors composed only of 3×3 filters. Since current deep learning frameworks do not natively support layers with mixed filter sizes the expand layer implementation actually consists of two separate layers, one with 1×1 filters and one with 3×3 filters. The output from squeeze layer is fed individually through each of these layers and then concatenated to form the final output of the expand layer.

3.2.2 The MobileNet Architecture

The MobileNet architecture was originally envisioned for deployment on mobile phones, hence its name. MobileNet eschews the convolutional layers we have been discussing up to this point in favor of “depthwise separable convolutions”. Depthwise separable convolutions split standard convolutions into two stages; a depthwise convolution stage and a pointwise convolution stage. In the depthwise stage 2-D filters are applied to each channel in an input individually after which pointwise convolutions compute a linear combination from the outputs of the depthwise convolutions to create the final output. Section 3.3.2 and Figure 3.3 provide a detailed description and illustration. Depthwise separable convolutions are attractive because they limit the number of parameters in a network and reduce the number of multiply-add operations that must be performed relative to standard convolutions.

MobileNet also incorporates batch normalization layers [Ioffe and Szegedy, 2015] after convolution layers. The original paper on batch normalization states “training deep neural networks is complicated by the fact that the distribution of each layers inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating non-linearities.". Batch normalization alleviates this issue by normalizing the inputs into the layer immediately after the batch normalization layer. While not a parameter reduction strategy these layers are an important piece of the MobileNet architecture and we incorporate them into a version of our custom convolutional neural network architecture.

3.3 The SpectrumNet Architecture

We are calling our network SpectrumNet, and it closely mimics the macro-architectural design of the SqueezeNet architecture [Iandola et al., 2016] while adapting the micro-architectural elements to work well with high-dimensional imagery. We have used a module based design for SpectrumNet. The macro-architecture of the network defines "spectral" modules that consist of the squeeze and expand layers described in Section 3.2.1. A depiction of the spectral module is shown in Figure 3.1. Figure 3.1 also shows a high level modular view of the layers that make up the SpectrumNet architecture. As this figure shows spectral modules make up the majority of the network with the exception of a single convolution layer at the beginning and end of the network. The spectral macro-architectural element was referred to as a "fire" module in [Iandola et al., 2016]. We made several adaptations to the structure of the layers in these modules that we believed would be useful when dealing with multi-spectral imagery.

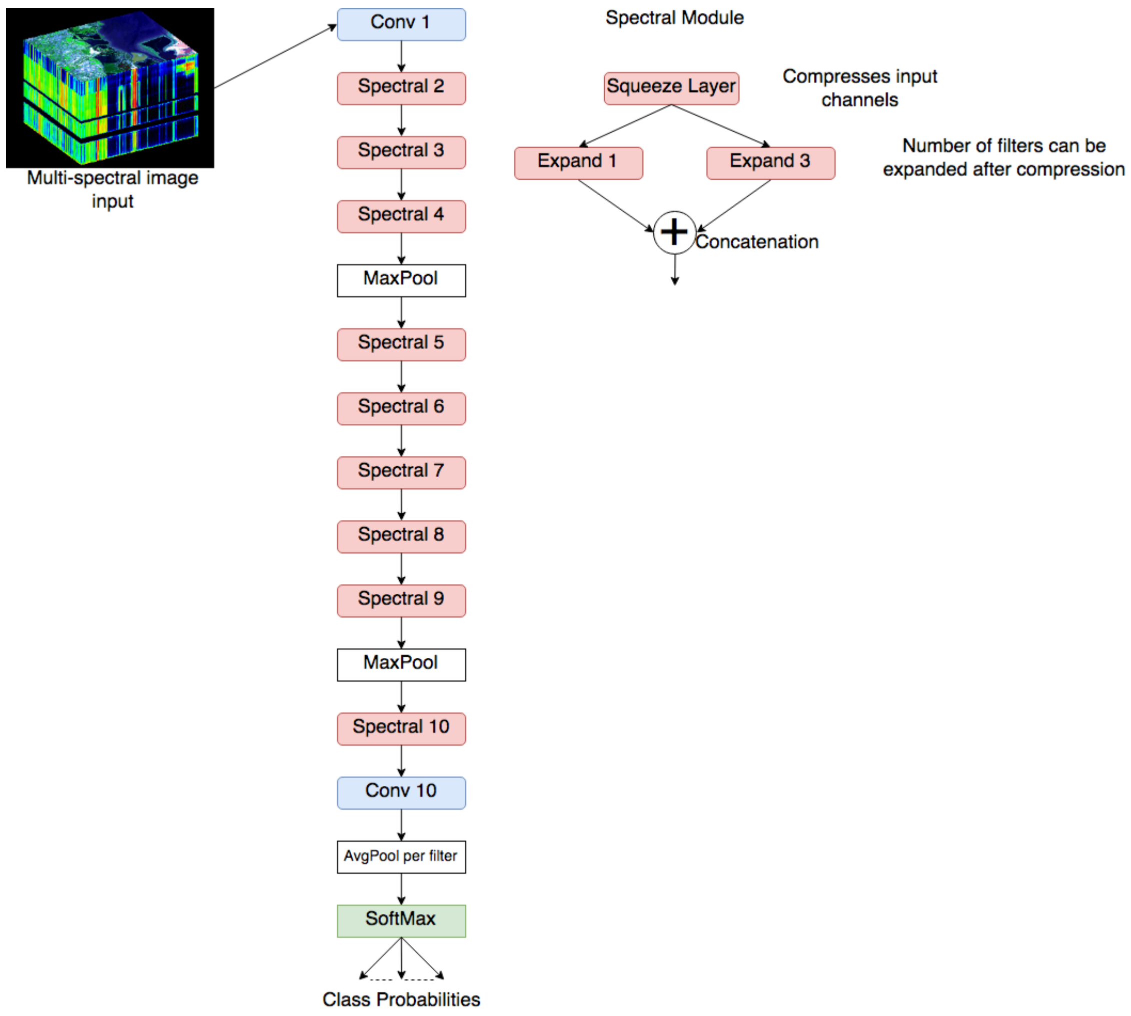


Figure 3.1: An illustration of the components that make up the macro-architecture of SpectrumNet.

3.3.1 Parameter Reduction Strategies

In the original SqueezeNet architecture “squeeze” layers were introduced to reduce the number of input channels to “expand” layers that consisted of a mix of 3×3 filters and 1×1 filters. The squeeze layers consisted entirely of 1×1 convolutions and were originally used as a parameter reduction strategy. We adapt the squeeze layer in our architecture to reduce the number of parameters in the network and to compress the high dimensional input. Since the squeeze layer consists entirely of 1×1 convolutions, it is not really a true convolution. Instead, it essentially computes linear combinations between spectral bands.

This strategy worked well with RGB images [Iandola et al., 2016], but it is not obvious that these “squeeze” layers would perform well with high-dimensional images where rapidly collapsing the initial input represents a major information bottleneck. The challenge then is to balance this bottleneck effect with the number of parameters in the network.

We further limit the number of 3×3 convolutions in the network, by setting the percentage of 1×1 convolutions in the network to 75%. This adaptation is motivated by the additional computational requirements that a multi-spectral image introduces, being over three times as large as the average RGB image. By configuring the network modules to have a majority of 1×1 convolutions this significantly reduces the number of parameters and multiply-add operations that take place in the network. Additionally we delay any pooling operations until midway through the network, with one final max pooling operation before the final spectral module. Part of the motivation for the delayed pooling is to further increase the size of the activation maps compared to the original SqueezeNet architecture. In [He and Sun, 2015] it is reported that delayed downsampling can result in higher classification accuracy. [Iandola et al., 2016] employed delayed downsampling in an effort to maintain classification accuracy with a reduced number of network parameters. As we are further reducing the number of parameters in the SpectrumNet architecture we reason that further delaying downsampling may help to maintain a given level of classification accuracy. Also delaying downsampling until late in the network allows us to process relatively small images (64×64 pixels) while still being able to incorporate a large number of layers in the network. We also forgo placing a fully connected network on top of the convolutional layers. Instead we use the same approach as the original SqueezeNet, and use a final convolutional layer with the number of output filters equal to the number of classes. A global average pooling is performed over these final filters to produce a logit for each class. A logit is defined as the input to a softmax layer. This approach was inspired by [Lin et al., 2013]. Our adapted SpectrumNet architecture is shown in Table 3.1.

3.3.2 Depthwise Separable Convolutions

The second version of our network incorporates depthwise separable convolutions and batch normalization layers into the squeeze and expand portions of the spectral module. Table 3.2 details the reduction in network parameters and multiply-add operations that can be achieved with such an architecture. We define a new module, "Spectral-DWS" that incorporates the depthwise separable convolutions and batch normalization layers in Figure 3.2.

The introduction of depthwise separable convolutions further reduces the computational requirements when dealing with high dimensional imagery. Depthwise separable convolutions are composed of depthwise and pointwise convolutions. As shown in Figure 3.3, the depthwise convolutions first apply the filter to each input channel individually producing outputs that have the same x and y spatial dimensions as the original input, but only have one channel. These outputs are then stacked to

Table 3.1: The SpectrumNet architecture.

Layer name/type	**Output size/filters**	**Filter/ stride** (if not a spectral module)	1×1 **Squeeze**	1×1 **Expand**	3×3 **Expand**
Input	$64 \times 64 \times 10$				
conv1	$32 \times 32 \times 96$	$2 \times 2/1$			
spectral2	$32 \times 32 \times 128$		16	96	32
spectral3	$32 \times 32 \times 128$		16	96	32
spectral4	$32 \times 32 \times 256$		32	192	64
maxpool4	$16 \times 16 \times 256$	$2 \times 2/2$			
spectral5	$16 \times 16 \times 256$		32	192	64
spectral6	$16 \times 16 \times 384$		48	288	96
spectral7	$16 \times 16 \times 384$		48	288	96
spectral8	$16 \times 16 \times 512$		64	385	128
maxpool8	$8 \times 8 \times 512$	$2 \times 2/2$			
spectral9	$8 \times 8 \times 512$		64	385	128
conv10	$8 \times 8 \times 10$	$1 \times 1/1$			
avgpool10	$1 \times 1 \times 10$	$8 \times 8/1$			

Table 3.2: Number of parameters and multiply-add operations in SpectrumNet, ResNet, and DenseNet.

Network	**No. Parameters** RGB/Multi-Spectral	**Million Mult-Adds** RGB/Multi-Spectral
SpectrumNet w/standard convolution	727.59K / 730.28K	203.47 / 206.22
SpectrumNet w/depthwise separable convolution	266.47K / 269.16K	75.59 / 78.34
ResNet-50	25.56M / NA	4120 / NA
DenseNet-161	28.68M / NA	7820 / NA

form an input of the same spatial x and y dimensions as the original input and the same number of channels. Then a 1×1 pointwise convolution is applied pixel-wise to create a linear combination of the outputs from the depthwise filters such that the final output is a single channel.

Depthwise convolution can be written as,

$$\hat{\mathbf{G}}_{k,l,m} = \sum_{i,j} \hat{\mathbf{K}}_{i,j,m} \times \mathbf{F}_{k+i-1,l+j-1,m}$$

where M is the number of channels, $\hat{\mathbf{K}}$ is the depthwise convolutional kernel of size $D_K \times D_K \times M$, with D_K being the spatial dimension of the filter, and $\mathbf{F}$ is a set of filters. This convolution is referred to as depthwise because the m_{th} channel in $\hat{\mathbf{K}}$ is applied to the m_{th} filter in $\mathbf{F}$.

The cost of a depthwise convolution is equal to the number of parameters in the filter, D_K^2 multiplied by number of values in a square input D_F^2 multiplied by the number of channels in the input, M,

$$O(MD_K^2D_F^2)$$

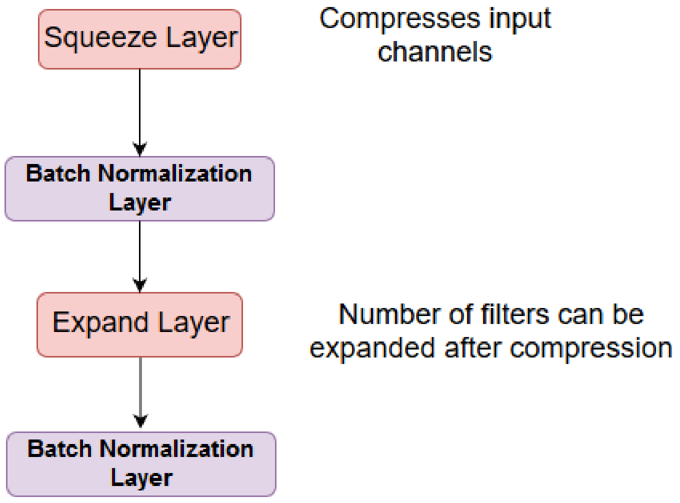

Squeeze Layer
Compresses input channels
Batch Normalization Layer
Expand Layer
Number of filters can be expanded after compression
Batch Normalization Layer

Figure 3.2: The design of the Spectral-DWS module, incorporating depthwise separable convolutions and batch normalization layers.

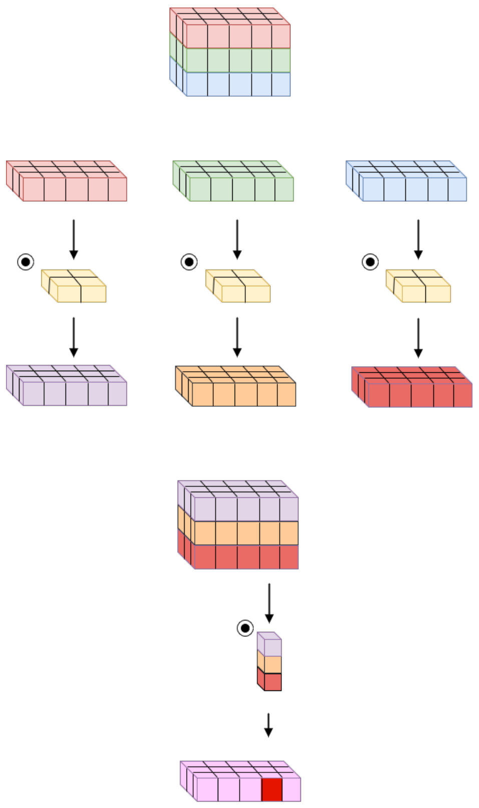


Figure 3.3: Depthwise separable convolution.

with D_F being the spatial shape of the square input. The number of operations in the pointwise convolution is equal to the number of input channels M, multiplied by the number of output channels N, multiplied by the number values in the square input D_F^2,

$$O(MND_F^2).$$

The computational cost of depthwise separable convolutions is the sum of the depthwise and pointwise convolutions,

$$O(MD_K^2D_F^2 + MND_F^2),$$

where D_F is the spatial shape of a square input, M is the number of input channels, and N is the number of output filters. Regular convolutional filters consist of a number of operations equal to the number of input channels M, multiplied by the number of output channels N, multiplied by the number of parameters in the filter D_K^2, multiplied by the number of values in the square input D_F^2,

$$O(MND_K^2D_F^2).$$

Thus, depthwise separable convolutions reduce computation by a factor of,

$$\frac{MD_K^2D_F^2 + MND_F^2}{MND_K^2D_F^2} = \frac{1}{N} + \frac{1}{D_K^2}$$

There is actually a significant amount of similarity between elements of the squeeze layers presented earlier and depthwise separable convolutions. The pointwise convolutions in the depthwise separable convolutions are similar to the original squeeze layers. They act to create linear combinations of input channels. Given this similarity, depthwise separable convolutions can be substituted for the squeeze layers in our CNN architecture.

In our first network architecture, a single convolutional kernel is used to extract cross channel correlations and spatial correlations simultaneously. The key difference between the first network architecture and this architecture which incorporates depthwise separable convolutions is the decoupling between spectral bands and the spatial correlations. The Xception architecture [Chollet, 2017], which consists entirely of depthwise separable convolutions, hypothesizes that cross-channel correlations and spatial correlations are sufficiently decoupled that it is actually preferable not to map them jointly. It is not clear if this decoupling is preferable when dealing with multi-spectral imagery. In an RGB classification task like ImageNet, which has many classes that are easily separable even if the images are grayscale and contain no color information (e.g., space shuttle and soccer ball), simultaneously mapping the coupling between spectral and spatial correlations likely matters little. However, when working on a task where spectral information almost always provides useful additional information, such as land cover classification, it may be desirable to map spatial and spectral correlations together, as is the case when using standard convolutions.

3.4 Datasets

The task under consideration is multi-spectral and hyper-spectral image classification, which we will collectively refer to as "high-dimensional" image classification. We test the neural network architectures developed in this thesis on two diverse datasets. The first dataset comes from the Sentinel satellite constellation operated by the European Space Agency [European Space Agency, 2018]. The second dataset comes from a ground based instrument the Resonon Pika L. This second dataset has been built from scratch at Montana State University over the course of this work.

3.4.1 EuroSat Dataset

The EuroSat dataset [Helber et al., 2017] consists of images from the Sentinel-2A and Sentinel-2B satellites. It is a land cover classification problem, consisting of ten different land cover classes: annual crop, permanent crop, highway, herbaceous vegetation, pasture, forest, sea and lake, river, residential, and industrial.

The Sentinel-2 satellites carry instruments capable of imaging the thirteen bands shown in Table 1. The thirteen bands range from the low end of the visible spectrum at 443 nm to the short wave infrared part of the spectrum at 2,190 nm. This is fairly typical of a space-based multi-spectral instrument. The spatial resolution varies from 10 meters/pixel up to 60 meters/pixel. The pixel values of the images are reported as top of atmosphere reflectance. Top of atmosphere reflectance consists of the combined surface and atmospheric reflectance and is useful as it reduces in-between scene variability through a normalization for solar irradiance.

In this work, we do not use bands B01, B10, or B11, as they are low resolution at 60 meters/pixel, and are typically used for detecting things like water vapor, or aerosols in the atmosphere. The remaining spectral bands that have resolution lower than 10 meters/pixel are upsampled to this resolution using cubic spline interpolation. The dataset consists of 27,000 different images with 2,000 to 3,000 images per class. Each image is rather small at 64×64 pixels. To the best of our knowledge this is the largest multi-spectral dataset that has been created for use by the general machine learning community. Example images from each of the classes are shown in Figure 3.4.

3.4.2 Produce Dataset

The second tested dataset we used is a hyper-spectral rather than multi-spectral dataset, captured using the Pika L imager. As part of a collaboration with the Intel

Table 3.3: Spectral bands acquired by the Sentinel-2A and Sentinel-2B satellites.

Band	**Spatial Resolution (m)**	**Wavelength (nm)**
B01 - Aerosols	60	443
B02 - Blue	10	490
B03 - Green	10	560
B04 - Red	10	665
B05 - Red Edge 1	20	705
B06 - Red Edge 2	20	740
B07 - Red Edge 3	20	783
B08 - NIR	10	842
B09 - Red Edge 4	20	865
B10 - Water Vapor	60	945
B11 - Cirrus	60	1375
B12 - SWIR 1	20	1610
B13 - SWIR 2	20	2190

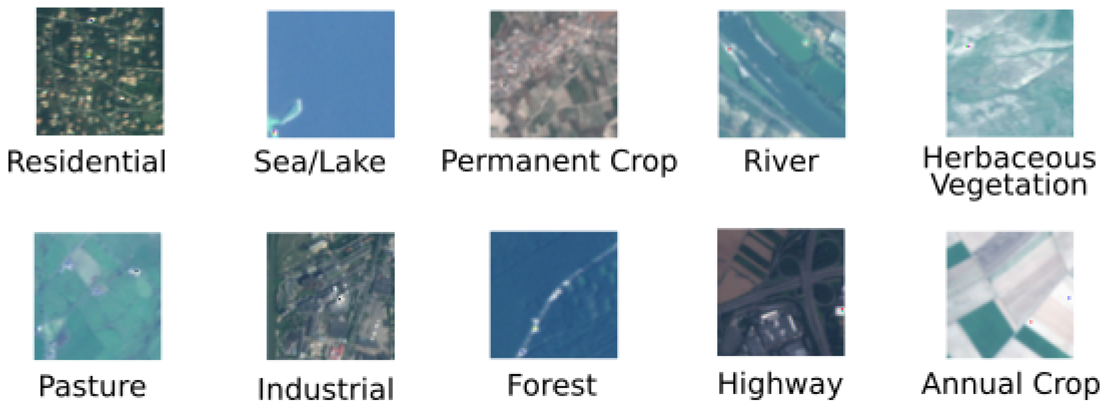


Figure 3.4: Example images from the ten classes in the EuroSat land cover dataset [Helber et al., 2017].

Corporation, we have been monitoring the aging process of various species of produce (i.e fruits, vegetables) in grocery stores over the course of a year (2018) with the goal of predicting the quality and age of produce using these measurements. Multiple varieties of produce were purchased and analyzed including avocados, tomatoes, potatoes, apples, bananas, and four varieties of bell peppers.

Typically, a given batch of produce is imaged for up to two weeks. Multiple data collection runs for each species of produce have been collated into one dataset. By collating multiple data collection runs, we introduce variability in the dataset that should help our models generalize, and we also reduce the possibility of spurious correlations being present in a dataset that a neural network could detect to make classifications that would not be present in an operational environment (e.g., differences in lighting between a fresh set and non-fresh set of produce, or some background detail that was present when produce was fresh but was not present after a few days of aging). We also crop the initially acquired image such that each resulting image is limited to the bounding box for the piece of produce it contains.

After purchase, the produce is placed in a 5-sided box with a uniform black background, with the only open end of the box facing the hyper-spectral imager. The produce is placed on a stage at ambient temperature and typically is imaged once per day over the lifetime of the produce. The produce is illuminated by two halogen bulbs set in light boxes in order to provide diffuse broadband illumination. Since halogen bulbs generate a great deal of heat, they are turned on only while capturing an image. The illumination source is a particularly important consideration, as a hyper-spectral instrument can only capture wavelengths that are present in the illumination source to begin with. Outdoors, this rarely presents a problem as the Sun is an excellent broadband illumination source.

The Pika L instrument from Resonon is used to acquire the hyper-spectral images. The Pika L is a pushbroom scanner. It acquires a single vertical strip at a time in the y spatial dimension while acquiring wavelengths from 400 nm to 1000 nm at 2 nm increments. The instrument is panned across the scene using a mechanical stage to capture the full spatial x dimension. Figure 3.5 displays the experimental set-up, while Figure 3.6 shows an example RGB image extracted from a hyper-spectral measurement taken by the Pika L. Initially, the Pika L reports data in terms of digital number, which is a unitless magnitude. Digital number can be converted to reflectance which is defined as the ratio of the amount of light leaving a target to the amount of light hitting a target. Reflectance can often be a reliable measure of material properties. The silver panels in Figure 3.5 are spectralon panels, Spectralon is a Lambertian surface. The apparent brightness of a Lambertian surface to an observer is the same regardless of the observer's angle of view [Koppal, 2014]. By virtue of the spectralon panel providing a nearly uniform reflectance of almost 100% across all wavelengths being imaged, including a spectralon panel in our scene provides a baseline that we can leverage to convert digital number to reflectance via a flat field correction.

The dataset can be split into classes based on days since purchase or based on a qualitative assessment of quality performed by a human. Note that both of these measures are imperfect and induce inter-class and intra-class variability. Days since purchase does not take into account how long a piece of produce may have been sitting on a shelf until it was purchased. However, we have found that this serves as a reasonable proxy for age. The qualitative assessment was inspired by the manner in which many local grocery stores currently assess their produce (i.e., by visual inspection). By using computer vision based techniques, we are assessing the feasibility of automating this process and assessing if finer gradations between age and quality of produce can be determined.

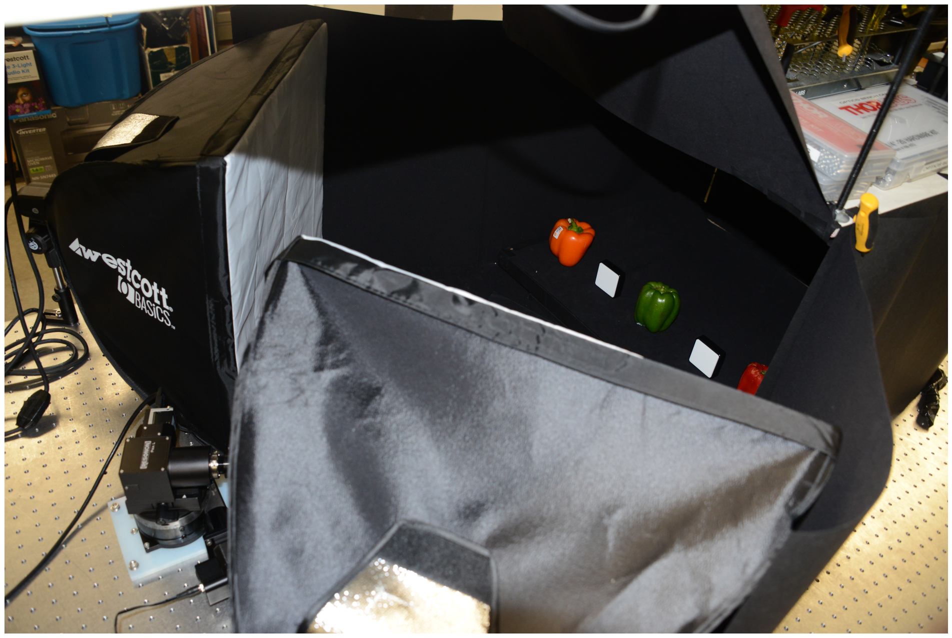


Figure 3.5: The experimental setup used to capture hyper-spectral images of aging produce.

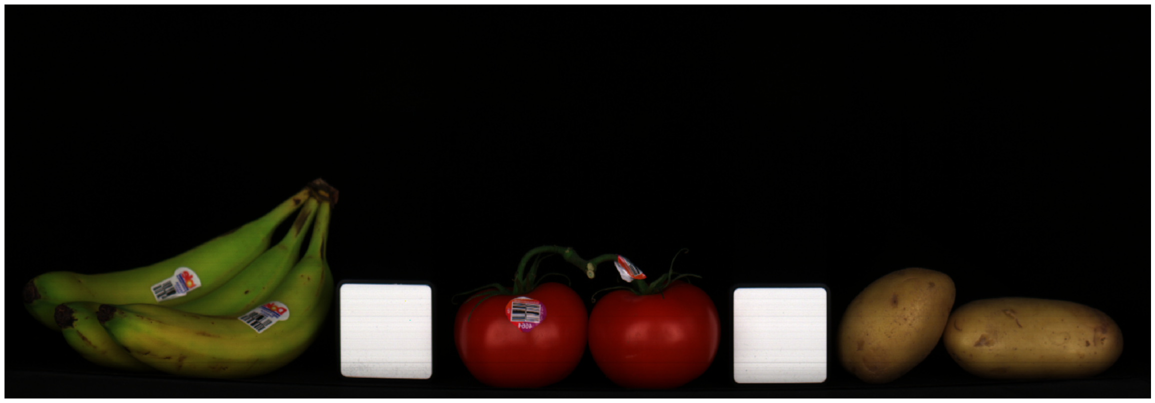


Figure 3.6: An RGB image of bananas, tomatoes, and potatoes, produced from a hyper-spectral image acquired by the Pika L instrument.

The produce and EuroSat datasets provide a diverse set of data on which we can evaluate the performance of convolutional neural network architectures designed to process high-dimensional imagery. The EuroSat dataset is large enough to give us confidence that if a model is performing poorly it is due either to the network architecture or some component of the training process (e.g., hyperparameter tuning), rather than poor results being due to a dataset simply being too small to train a deep neural network effectively. The produce dataset provides a means to validate the performance of any architecture developed on the EuroSat dataset. The produce dataset represents a difficult problem given the inter-class and intra-class variability, variability in class labels, and the small size of the dataset ($\approx$ 100 images per class at the largest).

In Chapter 4, we will present a transfer learning approach using features from the multi-spectral satellite imagery to initialize a network for the produce classification task. This study presents a number of interesting components. Features have been transferred from everyday objects to high altitude imagery but not from high altitude imagery to imagery obtained at close stand-off distances. The multi-spectral bands acquired by the Sentinel instruments are similar to but different from the bands acquired by the Pika L imager. The Sentinel instruments extend into the short wave infrared red range out to 2,190 nm while the Pika L only covers up to 1,000 nm. Also, as the original network is pre-trained on a 10 spectral band dataset, 10 bands must be selected from the hyper-spectral images acquired by the Pika L in some manner.

3.5 Experiments

We investigate how accuracy and convergence rate change using RGB images vs. using multi-spectral images. Additionally we evaluate how accuracy changes with reductions in the size of the training set.

For all experiments, the networks with standard convolutions[1] were trained using stochastic gradient descent with Nesterov momentum[2] set to 0.9 and weight decay (L2 penalty) set to 5×10^{-4}. When using depthwise separable convolutions, we did not use weight decay due to the small number of parameters in the network making regularization less important. The initial learning rate was set to 0.001 and was reduced by 25% every 10 epochs. A batch size of 64 was used, and all experiments were run on a Nvidia GTX 1080TI GPU. These settings were tuned through a series of experiments on our validation set. All results presented are the result of 10 fold cross-validation experiments.

3.5.1 Classification Performance

We evaluated both RGB images and 10 band multi-spectral images from the EuroSat and Produce datasets using both variants of our CNN architecture; the version with standard convolutions and the version with depthwise separable convolutions. When trained with multi-spectral imagery both network variants consistently achieved statistically significant higher classification accuracy (based on a student's t-test) compared to a network trained with RGB images (Table 3.4, 3.5, and 3.6). We expected multi-spectral images to result in greater classification accuracy in theory, given the information that is present in the additional spectral bands that increases separation between land cover classes. However, it was not clear if a small network architecture could process such high-dimensional imagery effectively. The high classification accuracy achieved with the network variant using standard convolutions shows that it is possible to use small and efficient CNN

[1]We use the term standard convolutions to distinguish them from depthwise separable convolutions.

[2]Nesterov momentum is a variant of traditional momentum. The essential difference is the use of a look ahead step that computes the gradient at a future location and the update step uses this information as a correction. See [Sutskever, 2013] Chapter 7 for details.

Table 3.4: EuroSat classification accuracy.

Network	**RGB Accuracy**	**Multi-Spectral Accuracy**
SpectrumNet w/standard convolution	$92.1 \pm 0.9\%$	$96.6 \pm 0.4\%$
SpectrumNet w/depthwise separable convolution	$87.6 \pm 0.8\%$	$95.2 \pm 0.7\%$

architectures to map and extract spectral-spatial features from high dimensional imagery simultaneously, as the standard convolutional filter acts on both the spectral dimension and spatial dimensions of an input. Complex schemes built upon band separation, dimensionality reduction, etc. do not appear to be needed for this type of task.

Incorporating depthwise separable convolutions into our architecture resulted in slightly reduced accuracy on the EuroSat dataset, but slightly higher accuracy on the Produce dataset compared to using standard convolutions. In the case of the EuroSat the difference was small, but still statistically significant, while the difference was not statistically significant for the Produce dataset. However, given the significant reduction in computation requirements when using depthwise separable convolutions, this may be a worthwhile trade-off. The solid performance of depthwise separable convolutions in this domain of high dimensional imagery lends further credence to the Xception hypothesis that spatial and spectral correlations can be mapped separately [Chollet, 2017].

3.5.2 Sample Efficiency

We found that having a small network capable of processing multi-spectral imagery effectively resulted in faster convergence rates and greater sample efficiency

Table 3.5: Classification accuracy when training the networks from scratch on the avocado dataset.

Network	**RGB Accuracy**	**Multi-Spectral Accuracy**
SpectrumNet w/standard convolution	$71.6 \pm 18.4\%$	$92.1 \pm 4.8\%$
SpectrumNet w/depthwise separable convolution	$85.8 \pm 1.0\%$	$93.2 \pm 5.8\%$

Table 3.6: Classification accuracy when training the networks from scratch on the tomato dataset.

Network	**RGB Accuracy**	**Multi-Spectral Accuracy**
SpectrumNet w/standard convolution	$94 \pm 4.9\%$	$97 \pm 4.6\%$
SpectrumNet w/depthwise separable convolution	$96 \pm 4.9\%$	$98 \pm 3.9\%$

during training. As Figure 3.7 shows, a network trained on multi-spectral imagery reaches convergence in $\approx$ 4,000 to 5,000 fewer gradient descent steps compared to using RGB images. This highlights an interesting, and important result. Not only does using multi-spectral imagery result in greater classification accuracy, but a CNN that can work well with multi-spectral imagery is also significantly more sample efficient, as much fewer gradient descent steps are required to reach a given level of accuracy.

As a further exploration of the sample efficiency of a small CNN applied to high dimensional imagery, we evaluated classification accuracy of our network trained on reduced datasets of both RGB and multi-spectral images. As the size of the training set was reduced, our small CNN architecture was able to maintain high accuracy

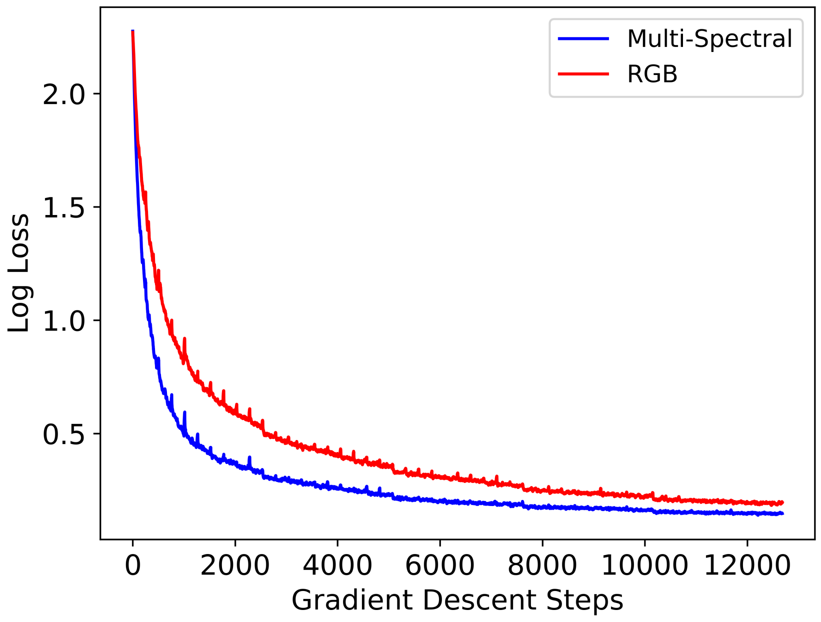


Figure 3.7: Training profile for SpectrumNet with standard convolutions.

($\approx$ 90%) when the training set consisted of 2,500 images. When trained on RGB images, the accuracy of the same network architecture experienced a greater loss in accuracy as the training set was reduced, compared to the network trained on multi-spectral images. For example, at a training set size of 2,500 images, the multi-spectral network experienced a 6.1% reduction in classification accuracy, while the network trained on RGB images had experienced an 11.2% reduction in classification accuracy. This gap widened as the training set was reduced to 1,350 images, (Figure 3.8).

While using depthwise separable convolutions resulted in similar classification accuracy as standard convolutions, convergence was appreciably slower (Figure 3.9). This suggests that decoupling cross-channel correlations and spatial correlations makes it more difficult for the network to learn; although, with sufficient training examples, comparable classification accuracy to standard convolutions can be achieved.

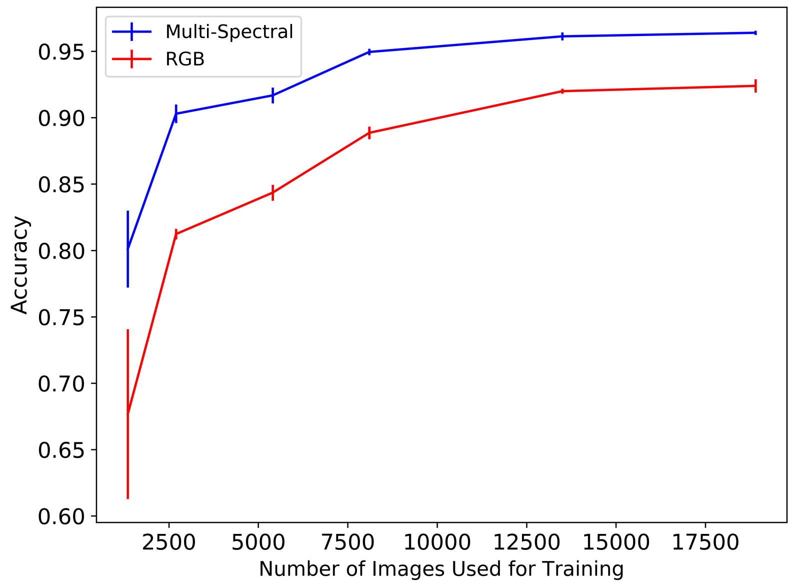


Figure 3.8: Classification test set accuracy as the size of the training set is reduced. Error bars are the standard deviation observed during the cross-validation experiments.

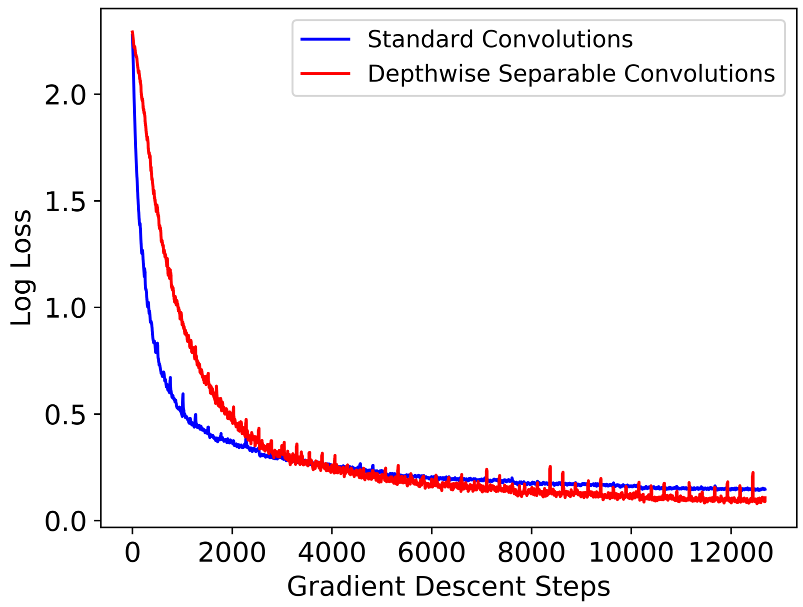


Figure 3.9: Training profile using standard vs depthwise separable convolutions.

3.6 Conclusion

Most previous approaches to dealing with high-dimensional imagery, and the typically small datasets associated with this type of imagery, have focused on transfer learning. Transfer learning with high-dimensional imagery is discussed in the next chapter. These approaches have not utilized the additional spectral information offered by multi-spectral imagery effectively. Other techniques have employed complex multi-stage processes to separate and reduce the dimensionality of the input. In contrast to previous work, we showed that our CNN architectures, which have fewer than one million parameters, can be applied effectively to high dimensional imagery. We also demonstrated that our architecture works well with a small training set, making it particularly useful in working with the small datasets that are common when dealing with multi-spectral imagery.

Depthwise separable convolutions reduced the computational requirements of our network significantly, and we observed only small deterioration in classification accuracy due to the decoupling between spectral and spatial correlations. Although, the difference in accuracy was still statistically significant. However, depthwise separable convolutions proved to be less sample efficient than standard convolutions in our experiments, requiring 500 - 1000 additional gradient descent steps. This suggests that standard convolutions may be the preferred choice when working with small datasets.

Somewhat surprisingly our network architectures achieved high classification accuracy even when trained from scratch on the small produce classification datasets. This likely results from the small parameter counts in our networks which reduce the risk of overfitting, in addition to the observed increase in sample efficiency resulting from training on multi-spectral data.

CHAPTER FOUR

CNN BASED HYPER-SPECTRAL IMAGE CLASSIFICATION WITH AND WITHOUT TRANSFER LEARNING

In this chapter we use hyper-spectral imaging to study the aging characteristics of produce (i.e. fruits, vegetables) in grocery stores, and attempt to make predictions about quality and age based on these measurements.

We can make classifications on a per image basis or a per pixel basis. The focus here will be on making classifications on a per image basis using the convolutional neural network architecture presented in Chapter 3. In addition to capturing spatial patterns that may be useful in age classification, by using multiple pixels a CNN may also be able to make more consistent predictions than a model making predictions on a per pixel basis. Classification could also potentially be done on a patch by patch basis, where each patch is a small group of pixels likely between 48×48 to 64×64 pixels depending on the size of the produce and the stand-off distance between the instrument acquiring the images and the produce itself.

In Chapter 3 we saw that it was possible to train our network architectures from scratch on the small produce dataset and still achieve high classification accuracy. However, the small number of images per class and produce type in the produce dataset makes a transfer learning approach attractive, as boosts in accuracy can often be gained by using a pre-trained network. We believed that this may also be the case here. Given the good performance achieved by our CNN architecture on the EuroSat multi-spectral dataset we hypothesized that the features learned by the network on that dataset would transfer well to the produce classification task. In this chapter we evaluate the performance of the two variants of the SpectrumNet architecture with and without transfer learning; the version with standard convolutions and the version with depthwise separable convolutions.

It was unknown how well features learned from a high altitude overhead image, land cover classification task would transfer to this produce classification task where the images are taken from a much different perspective. Additionally the wavelengths acquired by the multi-spectral instrument on the Sentinel satellites are not exactly the same as those acquired by the Pika L although, they are similar. If transfer learning can work well in this context this could facilitate new machine learning applications in the hyper-spectral and multi-spectral imagery domain where labeled datasets tend to be small and it is difficult to train a deep CNN from scratch.

This work is also motivated by the need to allow grocery stores to make more informed decisions regarding the handling of produce. According to a report from the U.S. Department of Agriculture, 10% of the total food production in the United States was lost as waste at the retail level [Buzby et al., 2014]. This equates to 43 billion pounds of food waste. Depending on shipping conditions, storage conditions, display conditions, etc., produce can age differently, making it difficult for managers to put produce on the shelves at the right time to minimize expenses and food waste from produce that is never purchased by a consumer.

4.1 Related Work

Related work can be divided into work directed towards monitoring bio-chemical processes with hyper/multi-spectral imagery, and work directed towards transfer learning. As the focus of this thesis is on the machine learning methods, we will discuss previous transfer learning related work in more depth, and we will provide a discussion of work directed towards monitoring produce and vegetation to provide proper context and background.

4.1.1 Vegetation/Produce Monitoring with hyper- and multi-spectral imagery

As mentioned in Chapter 2 and demonstrated in Chapter 3, multi-spectral and hyper-spectral imagery provides additional spectral information that is useful for distinguishing between different types of materials. Certain spectral bands have also been shown be sensitive to various bio-chemical processes taking place in vegetation. A number of indices have been developed to provide a measure of these processes including the Normalized Difference Vegetation Index (NDVI), Renormalized Difference Vegetation Index (RDVI), Modified Simple Ratio (MSR), Soil-Adjusted Vegetation Index (SAVI), Soil and Atmospherically Resistant Vegetation Index (SARVI), Triangular Vegetation Index (TVI), and the Modified Chlorophyll Absorption Ratio Index (MCARI), among others [Haboudane et al., 2004].

These index features typically are computed by comparing the ratio of reflectance of different spectral bands. As bio-physical properties of plant life change, the reflectance at different wavelengths is also affected. Comparing how reflectance changes at certain wavelengths can tell us something about the state of the vegetation we are imaging. The normalized difference vegetation index, for example, can provide information about the amount of live green vegetation in a given scene and is computed as,

$$NDVI = \frac{NIR - RED}{NIR + RED}$$

where NIR stands for the reflectance in the near infrared portion of the spectrum, and RED stands for the reflectance in the visible red portion of the spectrum. Photosynthetically active plants tend to reflect wavelengths beyond 700 nm, as this radiation cannot be used to synthesize organic molecules, and rather just heats the plant. The NDVI is designed to quantify this phenomenon. The other mentioned indices are designed similarly to detect changes in moisture concentration, chlorophyll levels, etc.

4.1.2 Transfer Learning

Transfer learning is a rather broad sub-field of machine learning focused on developing techniques to transfer the knowledge learned by a model on one task to a new model that is trained on a separate but to some degree related task. The models in our case are convolutional neural networks. [Yosinski et al., 2014] found that initializing any number of layers of a new network with weights learned from a pre-trained network can provide generalization benefits that remain even after fine-tuning the new network on a different dataset.

The work directed towards transfer learning on multi-spectral image related tasks has focused on transferring from neural networks trained on RGB datasets to multi-spectral problems, as there are very few publicly available multi-spectral datasets with ground truth labels. In almost all cases the RGB dataset used for pre-training was ImageNet. [Penatti et al., 2015] studied the generalization of features learned from everyday objects to the remote sensing domain. They worked primarily with the RGB UCMerced dataset that includes images partitioned into 21 land use classes (e.g. freeway, residential, runway). They also used the Brazilian Coffee Scenes dataset which consists of aerial images with green, red, and near-infrared spectral bands. They found that features learned by a convolutional neural network on ImageNet generalized fairly well to aerial scenes although, the Brazilian Coffee Scenes dataset proved more challenging then the UCMerced dataset. [Penatti et al., 2015] hypothesized this difference in classification accuracy was due to the UCMerced dataset consisting of classes with objects like buildings, cars, and airplanes, which are more similar to the objects found in ImageNet classes, whereas the Brazilian Coffee Scenes dataset has more homogeneous textures that are more difficult to separate between classes. The produce classification task considered here is more similar to the Brazilian Coffee Scenes dataset than the UCMerced dataset in that the different classes all contain images of the same type of produce, which generally have a high degree of homogeneity in terms of shape and size.

In [Xie et al., 2015, Castelluccio et al., 2015], the aforementioned transfer learning technique was explored using RGB imagery. These transfer learning approaches worked surprisingly well, given that the datasets the networks were trained on included classes of objects from entirely different perspectives than those obtained from high altitude overhead images.

Relatively little work has been devoted to applying CNNs to high dimensional multi-spectral imagery. In [Helber et al., 2017] multi-spectral images from the Sentinel satellite constellation were analyzed, using a transfer learning approach. The authors used a Google LeNet and a ResNet-50 network, pretrained on ImageNet, to perform land cover classification. These networks were fine tuned to produce land cover classifications by replacing their final fully connected layers. The authors manually chose 3 band combinations as inputs to the network, since the architecture of the pretrained model could not be modified to accept more than three spectral bands.

[Perez et al., 2017] presented one of the only alternative approaches we have seen to transfer features learned by a network trained on RGB images to multi-spectral data, rather than simply selecting three bands from the multi-spectral or hyper-spectral data and discarding the rest. Specifically, they used a network pre-trained on ImageNet, then modified the input layer to accept additional spectral bands by setting the weights for the RGB bands to the pre-trained ImageNet weights, and then setting the weights for the non-RGB bands to the mean of the three RGB weights at the same position in the same filter. They were attempting to predict poverty metrics for underdeveloped areas using satellite imagery from the Landsat constellation.

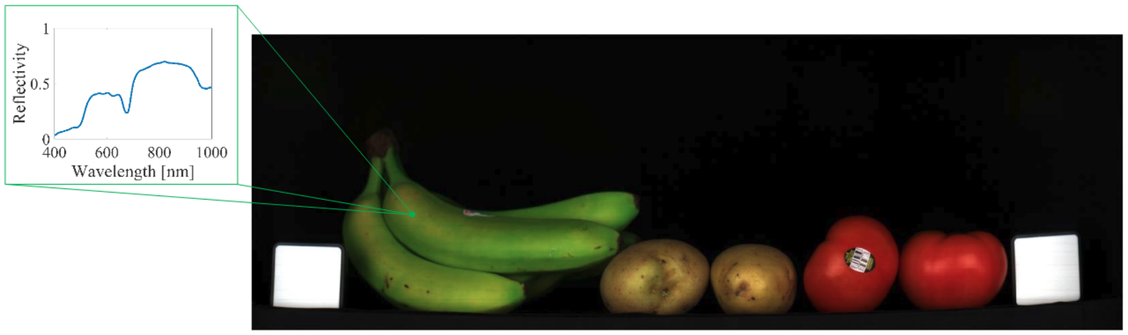
From a network mechanics point of view, the transfer learning approach used by [Perez et al., 2017] allows a network to accept additional spectral inputs while still initializing a majority of the network with pre-trained weights. However, multiple researchers have shown that the initial layers in a neural network produce the most general features [Yosinski et al., 2014, Zeiler and Fergus, 2014]. With the approach adopted by [Perez et al., 2017], spectral features from bands other than red, green, and blue still had to be learned from scratch.

In our work, we pre-train our networks on multi-spectral data and transfer the learned network weights to a network trained on a subset of bands from a hyper-spectral image. We then make comparisons to transfer learning from networks trained on RGB images as well as compare the performance of our networks when trained from scratch.

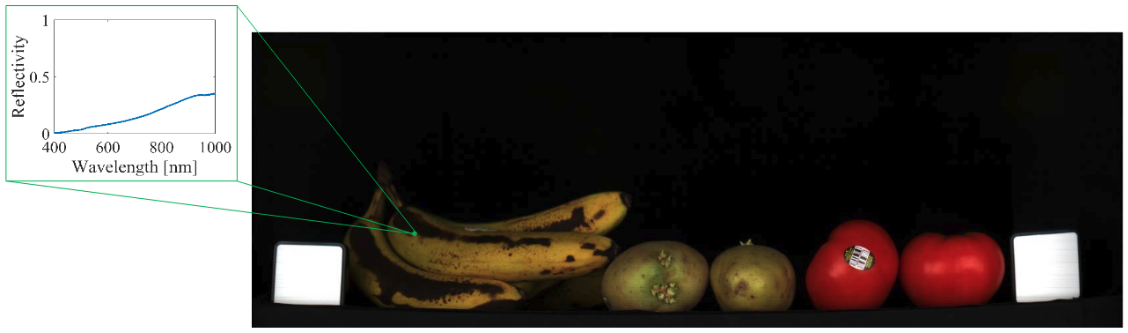
4.2 Spectral Characteristics of Aging Produce

In order to understand the type of data and processes we are working with, we will first examine visually how reflectance changes for different types of produce at different points in an aging process. Recall that when a hyper-spectral image is captured, it produces a data cube with the x and y spatial dimensions and the λ wavelength dimension. Now imagine choosing a single pixel from the plane defined by the x and y spatial dimensions and plotting the reflectance at each wavelength for that single pixel. This produces a reflectance curve, or spectrum, for that pixel. Multiple reflectance curves may be referred to in the plural as spectra.

In order for reflectance information to be useful in predicting the age or quality of produce, the reflectance curves for multiple pixels belonging to a piece of produce must exhibit changes over time. These changes may not be apparent immediately to a human trying to distill patterns from thousands of reflectance spectra (though in some



(a) Reflectance spectrum for a single banana pixel early on in the aging process.



(b) Reflectance spectrum for a single banana pixel at a late stage in the aging process.

Figure 4.1: Comparison of spectra on an aging banana. Image source: [Logan et al., 2018]

cases they are), but may be learned with enough examples by a machine learning or statistical model. Figure 4.1 demonstrates how dramatically reflectance information can change over the course of the aging process of a piece of produce. While fresh and old bananas show quite dramatic changes over several days, day to day changes are more subtle, and different types of produce exhibit different small changes in their spectra that are hard to identify by simply observing plots of reflectance over time. This necessitates the need for a machine learning or statistical model.

Looking at the aging characteristics of bananas over the course of several days, we observe extensive change in the reflectance characteristics of bananas in Figure 4.2. Bananas have proven to exhibit particularly large changes in their reflectance

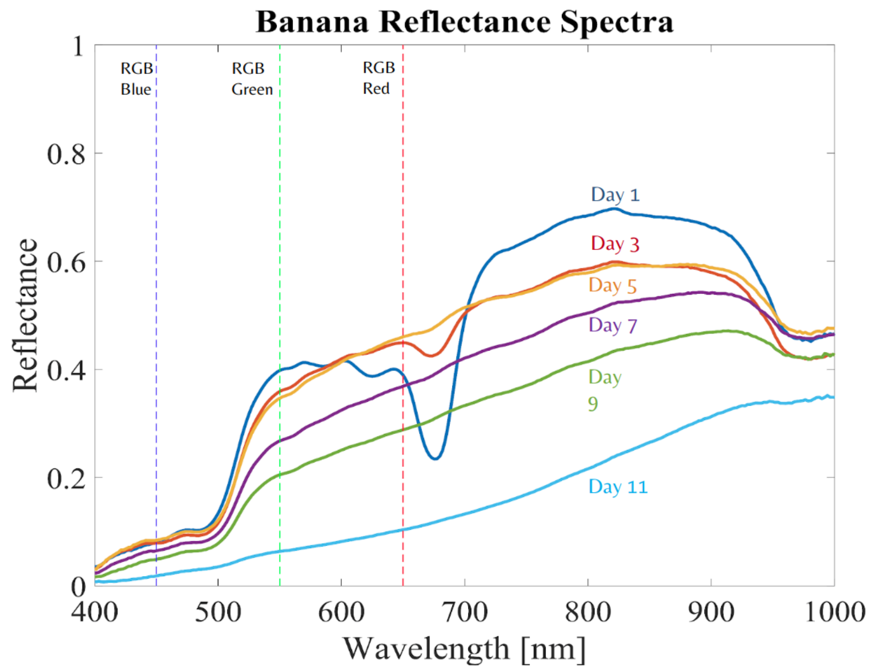


Figure 4.2: Changes in the reflectance spectra as a banana ages.

spectra as they age. Other types of produce do not exhibit such distinct differences from day to day. However, we do observe changes in reflectance characteristics over time for a variety of produce types, so classifying the age of produce based on spectral characteristics may be possible. As another example, Figure 4.3 and Figure 4.4 show the reflectance spectra for a tomato and an avocado over the course of several days

4.3 Experiments

For all experiments, the networks with standard convolutions were trained using stochastic gradient descent with Nesterov momentum set to 0.9 and weight decay set to 5×10^{-4}. When using depthwise separable convolutions, we did not use weight decay due to the small number of parameters in the network. The initial learning rate was set to 0.001 and was reduced by 25% every 10 epochs. A batch size of 4 was used, and all experiments were run on a Nvidia GTX 1080TI GPU. All results

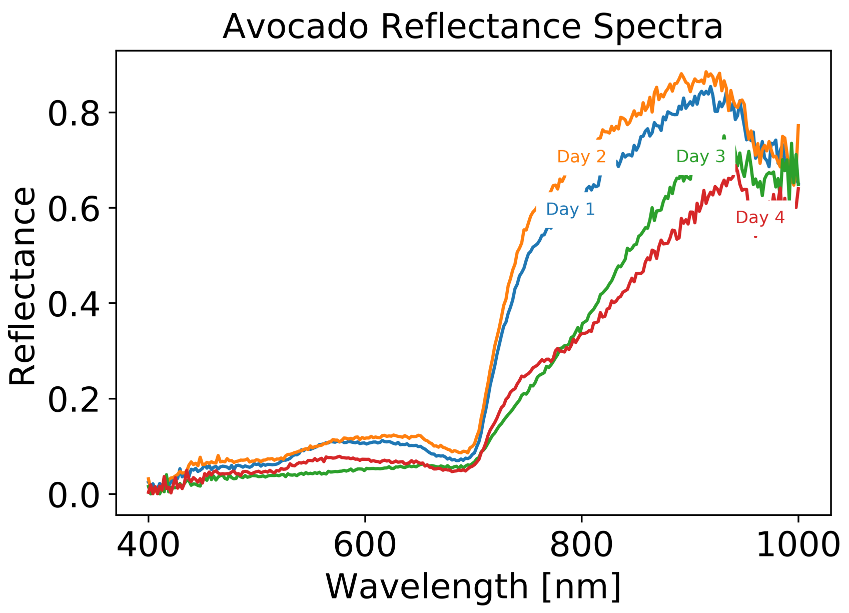


Figure 4.3: Changes in the reflectance from day to day as an avocado ages.

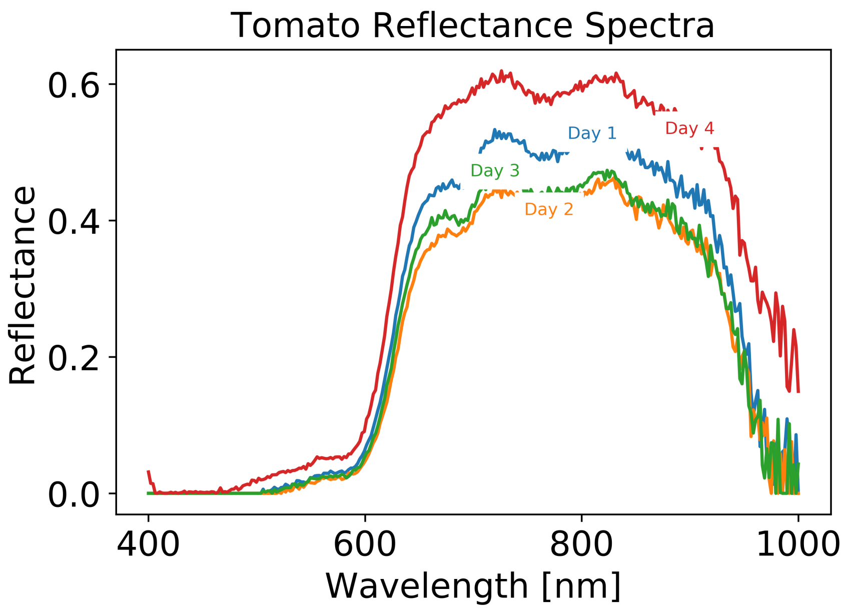


Figure 4.4: Changes in the reflectance from day to day as a tomato ages.

presented are the result of 10 fold cross-validation experiments.

4.3.1 Produce Classification via Transfer Learning

We use the produce dataset described in chapter 3. We focus on an avocado and tomato classification task, with the fruits being partitioned into two classes. The first class consists of fresh produce, while the second class consists of produce that has been identified as no longer fresh. A third party assessed each piece of produce on each day of data collection to assign each piece of produce a class label. The classes in the avocado dataset were nearly balanced with 101 images of fresh avocados and 96 images of non-fresh avocados. The tomato dataset was more imbalanced with 40 fresh tomatoes and 60 non-fresh tomatoes. Figure 4.5 and Figure 4.6 show examples of images of avocados and tomatoes used in these experiments. They are viewed in false color[1] at 700 nm. Although we have data from other types of produce, we limit our analysis to tomatoes and avocados for two main reasons.

- Some types of produce like bananas come in bunches. Our data does not contain images with single bananas. Since individual pieces of produce have been observed to age at different rates, attempting to classify multiple bananas at once could confuse our analysis.

- The avocado and tomato datasets are fairly well balanced between fresh and non-fresh classes while also consisting of an appreciable number of images so we can get more reliable estimates of classification accuracy and standard deviation. When dealing with a very small dataset, as is the case with some of the other types of produce we have gathered data for, training a deep CNN would not be feasible. Even if it was feasible, even a single misclassification could produce

[1]False color refers to color rendering methods used to display images in color which were recorded in the visible or non-visible portion of the spectrum.

large drops in accuracy making it difficult to evaluate the true capability of the model.

The pre-trained networks were trained on the EuroSat multi-spectral dataset, also detailed in Chapter 3. As the pre-networks were trained on EuroSat with ten spectral bands, ten spectral bands were selected from the hyper-spectral images acquired by the Pika L instrument, for further fine tuning of the pre-trained networks on the produce dataset. The spectral bands were selected using a newly developed method that relies on a genetic algorithm based approach to feature selection [Walton et al., 2019].

For comparison we also pre-train our network architectures on the RGB bands from the EuroSat dataset, and further fine tune the pre-trained networks on a selection of wavelengths from the RGB region of the hyper-spectral produce images. We used the wavelengths 641 nm (red), 551 nm (green), and 460 nm (blue). The RGB wavelengths chosen were chosen to be similar to the RGB wavelengths used in EuroSat satellite, but they were chosen deliberately to not be exactly the same. Since the multi-spectral wavelengths from the EuroSat dataset also do not match exactly with the wavelengths used in the produce dataset it would be an unfair comparison to evaluate transfer learning efficacy on RGB images using identical wavelengths in the source and target domains.

We conduct two distinct experiments for both the tomato dataset and avocado dataset. We first perform transfer learning from the EuroSat dataset, and during training of the pre-trained network on the target (produce) domain we allow the gradient of the error to backpropagate into the pre-trained weights so that they are updated along with the weights in the randomly initialized final convolutional layer. In the second experiment we freeze the pre-trained weights so that the gradient of the error is used only to update the weights in the randomly initialized final convolutional layer. The pre-trained weights are not updated at all. We split our analysis into these two experiments to assess how similar the source domain (EuroSat) and the target domain (produce) are. If the domains are very similar we expect freezing the pre-trained weights should result in the highest performance since these parameters should also be relevant to the produce dataset and they have been trained on a large multi-spectral dataset which should improve generalization on unseen test data. On the other hand if there are key differences between the source domain and target domain we hypothesize that allowing the gradient of the error to backpropagate into the pre-trained weights will lead to the highest classification accuracy as the pre-trained weights in addition to the randomly initialized weights in the final convolutional layer will be fine-tuned for the specific task at hand. We also repeat the results from training SpectrumNet from scratch on the produce dataset first shown in Chapter 3 for ease of comparison to the transfer learning results.

4.3.2 Transfer Learning Efficacy

[Penatti et al., 2015] demonstrated the benefit of transferring from ImageNet, which consists of different classes of objects photographed at relatively close range, to high altitude overhead satellite imagery. The benefit was not as pronounced when transferring to an image classification task that was not object centric such as the Brazilian Coffee Scenes dataset that organizes images into coffee or non-coffee crop, land cover types.

Similar differences exist between the source and target tasks in our case. However, we are transferring in the opposite direction, from high altitude, overhead, multi-spectral satellite imagery to more object centric multi-spectral imagery captured at close range. We first discuss the experimental results with regard to the SpectrumNet variant using depthwise separable convolutions. Given the differences between the

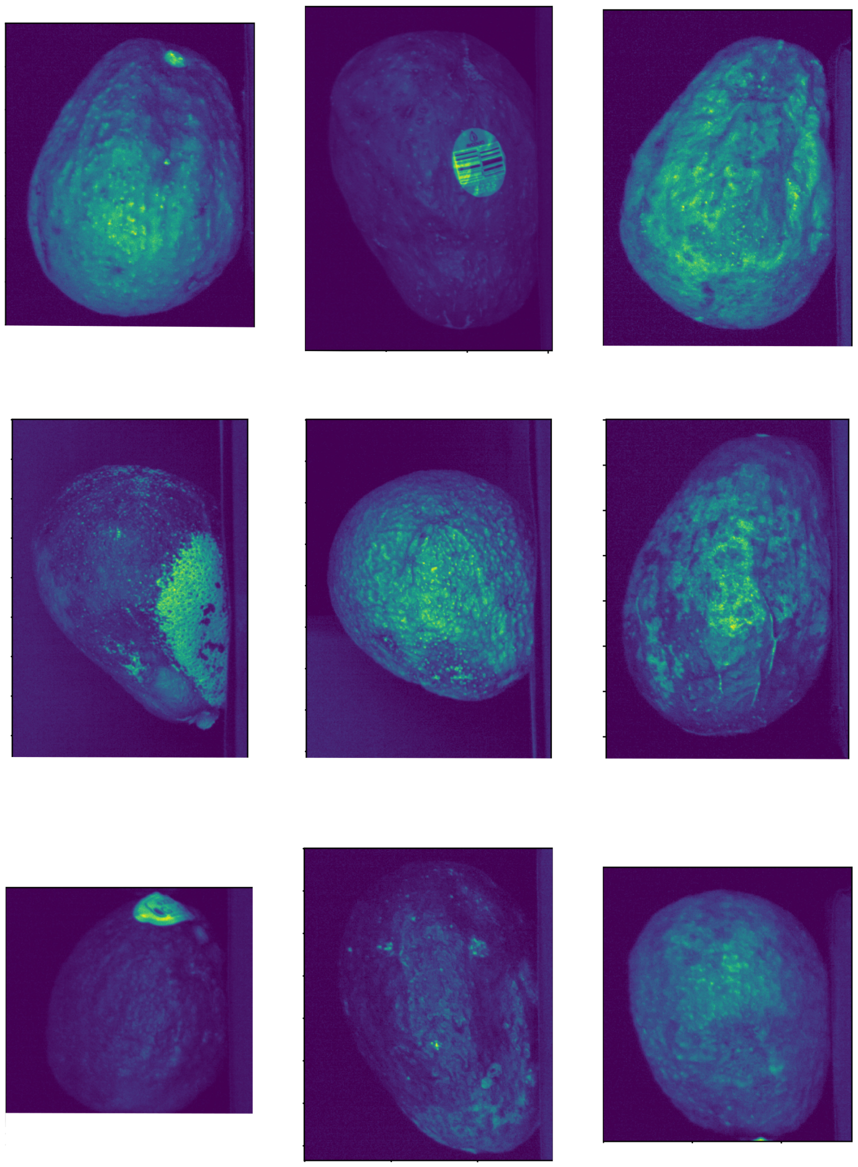


Figure 4.5: A collection of the avocados used in this study viewed in false color at 700 nm.

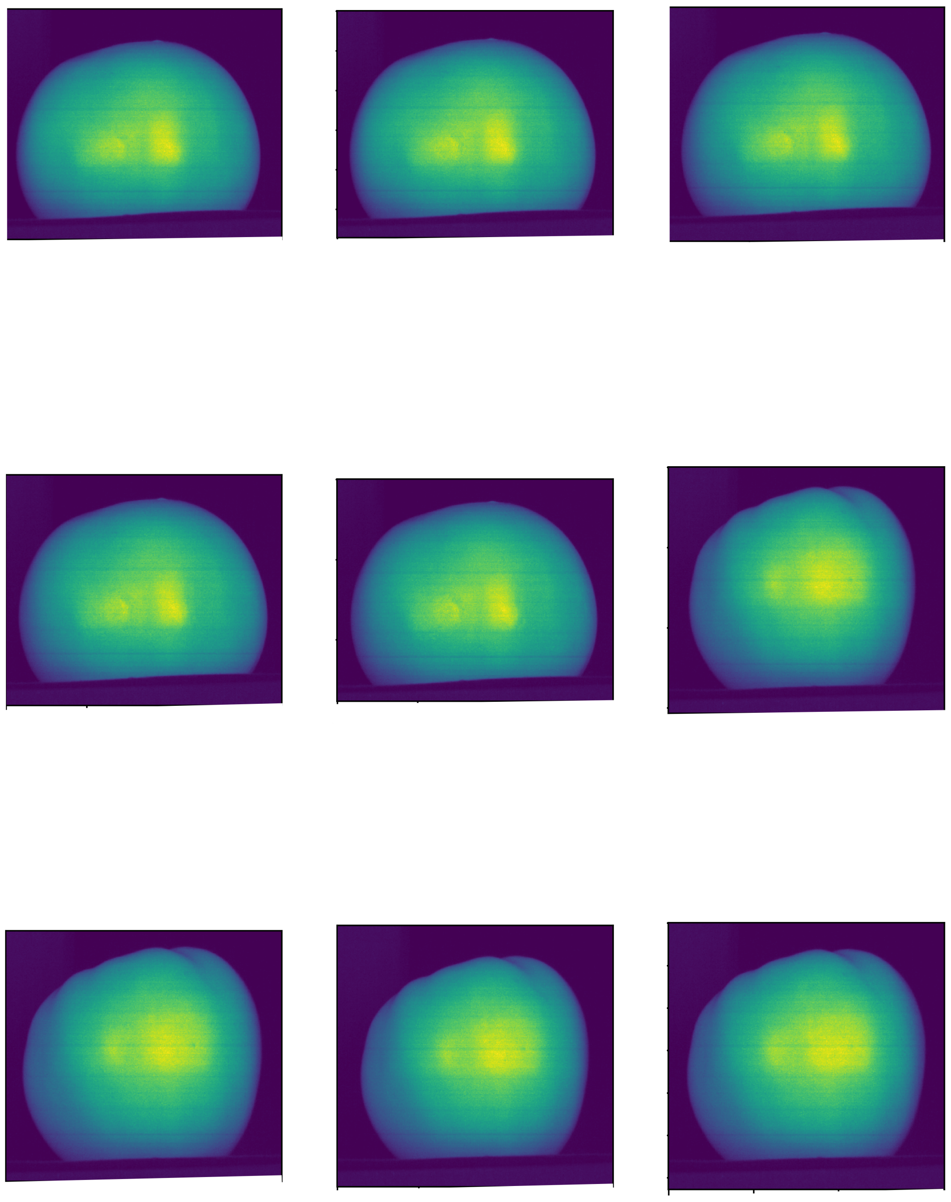


Figure 4.6: A collection of the tomatoes used in this study viewed in false color at 700 nm.

source and target domain transfer learning from multi-spectral satellite imagery still resulted in increased accuracy on the produce evaluation task compared to training from scratch when we allowed the gradient of the error to backpropagate into the pre-trained weights; see Tables 4.2 and 4.5. See Tables 4.1 and 4.4 for the classification accuracy achieved when training from scratch. When freezing the pre-trained weights accuracy was diminished by a statistically significant margin compared to transfer learning with non-frozen weights and training from scratch, see Tables 4.3 and 4.6. We believe this suggests that similar *spectral* rather than spatial features are being successfully transferred from the satellite imagery dataset to our produce classification task. Our reasoning is that the high level spatial features of the satellite imagery and the produce dataset our very different so transferring high level spatial features between the two domains is difficult, but since we still achieve improved accuracy vs training from scratch there must be useful features being transferred and the only information present in the source domain besides spatial features are the learned spectral features. This is a significant result regarding the application of deep neural networks to the multi-spectral and hyper-spectral imaging domain. These experiments show that it is possible to successfully transfer from abundant multi-spectral satellite imagery (though labeled data is generally more scarce) to a multi-spectral task with small training set sizes (100-200 images in our case).

An unresolved issue is the deterioration in performance observed during transfer learning vs training from scratch when using standard convolutions. We observed significant decreases in accuracy and increased variability in classification accuracy when attempting transfer learning using standard convolutions. Using ideas from the MobileNet architecture [Howard et al., 2017], we introduced batch normalization layers following every depthwise separable convolution layer in one version of the SpectrumNet architecture. The main differences between the two variants of the

Table 4.1: Classification accuracy when training the networks from scratch on the avocado dataset.

Network	**RGB Accuracy**	**Multi-Spectral Accuracy**
SpectrumNet w/standard convolution	$71.6 \pm 18.4\%$	$92.1 \pm 4.8\%$
SpectrumNet w/depthwise separable convolution	$85.8 \pm 1.0\%$	$93.2 \pm 5.8\%$

SpectrumNet architecture are the use of standard convolutions vs. depthwise separable convolutions and the introduction of batch normalization layers into the spectral modules of the SpectrumNet variant that used depthwise separable convolutions. There is no reason why transfer learning using depthwise separable convolutions would be more effective or easier than using standard convolutions, in fact standard convolutions are more commonly used in CNNs. This leads us to suspect that the difference in transfer learning efficacy that we observed between the network architecture using standard convolutions and the architecture using depthwise separable convolutions is due to the introduction of the batch normalization layers. We believe the reason that batch normalization may help transfer learning is due to differences in the distributions of data in the spectral bands from satellite imagery and our produce data. By normalizing the outputs of intermediate layers in the network, batch normalization may alleviate the issue with differing distributions, resulting in more effective transfer learning.

4.3.3 Sample Efficiency

Recall from Chapter 3 that a network trained on multi-spectral data reached higher levels of accuracy with fewer training iterations compared to a network trained on RGB images, suggesting that a multi-spectral network was more sample efficient.

Table 4.2: Transfer learning classification accuracy on the avocado dataset when allowing the gradient of the error to backpropagate into pre-trained weights.

Network	**RGB Accuracy**	**Multi-Spectral Accuracy**
SpectrumNet w/standard convolution	$72.1 \pm 14.3\%$	$62.6 \pm 16.7\%$
SpectrumNet w/depthwise separable convolution	$93.1 \pm 3.2\%$	$96.3 \pm 3.4\%$

Table 4.3: Transfer learning classification accuracy on the avocado dataset when freezing the pre-trained weights.

Network	**RGB Accuracy**	**Multi-Spectral Accuracy**
SpectrumNet w/standard convolution	$64.2 \pm 15.7\%$	$62.1 \pm 14.5\%$
SpectrumNet w/depthwise separable convolution	$75.8 \pm 8.9\%$	$83.2 \pm 6.9\%$

Table 4.4: Classification accuracy when training the networks from scratch on the tomato dataset.

Network	**RGB Accuracy**	**Multi-Spectral Accuracy**
SpectrumNet w/standard convolution	$94 \pm 4.9\%$	$97 \pm 4.6\%$
SpectrumNet w/depthwise separable convolution	$96 \pm 4.9\%$	$98 \pm 3.9\%$

We observe the same effect here. Given the same number of gradient descent steps, a network trained from scratch on multi-spectral imagery consistently achieved higher accuracy when compared to a network trained from scratch on RGB data. This effect was more pronounced in the avocado dataset than the tomato dataset, but the tomato

Table 4.5: Transfer learning classification accuracy on the tomato dataset when allowing the gradient of the error to backpropagate into pre-trained weights.

Network	**RGB Accuracy**	**Multi-Spectral Accuracy**
SpectrumNet w/standard convolution	$42.0 \pm 6.0\%$	$47.0 \pm 11.0\%$
SpectrumNet w/depthwise separable convolution	$96 \pm 4.9\%$	$99 \pm 2.9\%$

Table 4.6: Transfer learning classification accuracy on the tomato dataset when freezing the pre-trained weights.

Network	**RGB Accuracy**	**Multi-Spectral Accuracy**
SpectrumNet w/standard convolution	$44.0 \pm 8.0\%$	$49.0 \pm 8.0\%$
SpectrumNet w/depthwise separable convolution	$75 \pm 9.2\%$	$71 \pm 8.3\%$

dataset was much smaller, making it more difficult to draw firm conclusions. We believe the case for greater sample efficiency of multi-spectral networks is strengthened by this effect being observed across two different tasks; land cover classification using satellite imagery, and produce evaluation at close stand-off distances. Given our small dataset sizes (≈ 100 images for the tomato dataset, and ≈ 200 images for the avocado dataset), the classification accuracy achieved by both versions of the SpectrumNet architecture when trained from scratch on multi-spectral imagery is impressive, with the worst result being 92.1% accuracy.

4.4 Conclusion

For many tasks, datasets are small compared to massive labeled image repositories such as ImageNet or CIFAR-10, on which deep neural network architectures are typically trained and evaluated. For tasks of typical interest in the computer vision community, we are usually working with RGB images so networks pre-trained on ImageNet provide a good starting point for fine-tuning on a new but related task. Additionally, researchers have shown that networks pre-trained on ImageNet can still provide benefits to tasks that are quite distant from the source task [Penatti et al., 2015, Perez et al., 2017].

However, it is not straight forward to employ a transfer learning approach when we are attempting to classify high dimensional multi-spectral or hyper-spectral imagery. Additional spectral bands beyond the RGB domain can have different distributions and detect different features than what can be found in ImageNet. For example spectral features of different types of materials or different biological processes are not found in the current ImageNet repository.

This motivated our research into the possibility of transfer learning from a large dataset in the multi-spectral domain - the EuroSat dataset from the Sentinel satellite constellation - to a much smaller but also multi-spectral, produce quality classification task. These experiments also served as an additional test of the convolutional neural network architectures we developed in Chapter 3.

Our cross-validation experiments showed that transfer learning from multi-spectral satellite imagery to a task also consisting of multi-spectral data is a viable approach to training a deep neural network on a specialized domain with a small amount of labeled data. Given the large amount of available multi-spectral satellite imagery (e.g. Sentinel, LandSat), unsupervised pre-training on this data for other multi-spectral or hyper-spectral classification tasks that also have small training sets could be particularly useful.

Also significant was the high accuracy obtained by our neural network architectures when trained from scratch on the produce classification tasks. Multi-spectral data resulted in better classification accuracy versus using only RGB wavelengths. We believe that the small number of parameters in our networks, originally inspired by network architectures for embedded systems, make our network architectures particularly suited for small datasets although, this conclusion merits additional experiments on other small datasets.

CHAPTER FIVE

UNCERTAINTY QUANTIFICATION

The probabilities given by a softmax layer at the output of a neural network are often wrongly construed as the confidence a model has in its output. It is entirely possible for a model to have a high softmax output and high uncertainty in that output. Principled uncertainty estimates[1] can provide crucial information, allowing us to handle uncertain network outputs and be more confident when deploying deep learning models. As an example, on a classification task we could forward images that a model is highly uncertain about to a human for review.

Recent work by Gal and Ghahramani, has presented dropout as a Bayesian approximation of Gaussian processes. Dropout was originally proposed in [Hinton et al., 2012]. The concept of dropout is straightforward; we simply randomly drop nodes (and their connections) from a network during training. This is equivalent to imposing a Bernoulli distribution over a set of random variables (the network weights) as each weight is chosen to be dropped with probability equal to zero or one. Dropout was originally proposed as a regularization method for large network with millions of parameters, as the number of parameters in the network in question is artificially "thinned" throughout training. At test time when we wish to perform inference on unseen inputs, no weights are dropped from the network. Instead the outputs of all of the thinned networks created during training are effectively averaged by scaling the network weights to be smaller.

Dropout has since received a few different interpretations as to its effect and role in a neural network. We refer the reader to [Gal and Ghahramani, 2016] for

[1][Gal and Ghahramani, 2016] use the term principled in the sense that the uncertainty estimates we will obtain from our networks basically approximate those of a Gaussian process.

a derivation showing that the dropout objective is mathematically equivalent to an approximation of a probabilistic deep Gaussian process. The basic idea is that both dropout and Gaussian processes place distributions over random models or functions. They showed that model uncertainty can be obtained from any network architecture that uses dropout. Following the results presented in [Gal and Ghahramani, 2016], our approximate predictive distribution is given for a new input point $\mathbf{x}^*$ by,

$$q(\mathbf{y}^*|\mathbf{x}^*) = \int p(\mathbf{y}^*|\mathbf{x}^*, \omega) q(\omega) d\omega$$

where $\mathbf{y}^* \in \mathbb{R}^D$, D is the number of classes, and $\omega = \{\mathbf{W}_i\}_{i=1}^L$ is the set of random variables (weights) in a network with L layers. The distribution $q(\omega)$ is approximating the distribution $p(\omega|\mathbf{X}, \mathbf{Y})$ which we cannot evaluate analytically, hence the need for the approximating distribution $q(\omega)$. Here, we assume we have a dataset composed of N inputs, $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ with matching labels $\{\mathbf{y}_1, ..., \mathbf{y}_N\}$. We denote the dataset inputs as $\mathbf{X} \in \mathbb{R}^{N \times D}$ and the labels $\mathbf{Y} \in \mathbb{R}^{N \times C}$, with D being the dimensionality of the inputs and C the number of possible classes.

We can estimate the mean and variance of the approximate predictive distribution empirically by sampling T sets of vectors of weights from the Bernoulli distribution introduced over the weights by dropout, $\{\mathbf{z}_1^t, ..., \mathbf{z}_L^t\}_{t=1}^T$, with $\mathbf{z}_i^t = [z_{i,j}^t]_{j=1}^{K_i}$, and K_i defining the dimension of the weight matrix for a particular layer, giving a set of weight realizations, $\{\mathbf{W}_i^t, ..., \mathbf{W}_L^t\}_{t=1}^T$. The approximation to $q(\mathbf{y}^*|\mathbf{x}^*)$ then is,

$$q(\mathbf{y}^*|\mathbf{x}^*) \approx \frac{1}{T} \sum_{t=1}^{T} p(\mathbf{y}^*|\mathbf{x}^*, \omega_t)$$

with $\omega_t \sim q(\omega)$.

The above procedure has been shown to provide a Monte Carlo estimate of the predictive mean and predictive uncertainty (variance of the network outputs), and it is referred to as Monte Carlo dropout in the original derivation. In practice, we perform T stochastic forward passes through the model and calculate the mean and variance of the output distribution to obtain an estimate of uncertainty. This technique gives us an assessment of epistemic uncertainty, which is uncertainty in the model of a process potentially arising from things like limited data or inaccurate measurements.

With this technique in hand and a network architecture that is capable of processing the full spectral-spatial input from high dimensional imagery, we can evaluate how the uncertainty of deep neural networks depends on the information contained in spectral channels, which to our knowledge, has not been explored previously.

5.1 Related Work

Development of methods for quantifying uncertainty in neural networks has generally not received as much attention as development of methods to improve the accuracy of neural networks. However, a review of recent literature shows that there seems to be an increasing trend of research directed towards uncertainty quantification methods in neural networks. This includes research into Bayesian versions of neural networks, estimates of uncertainty using ensemble based methods, and research into casting existing popular mechanisms in neural networks (i.e., dropout) as a way to extract uncertainty information from a neural network.

A Bayesian network models a distribution over the weights in a neural network, rather than maintaining a single value. While this idea is theoretically sound it can prove difficult in practice due to excessive computational cost. In [Graves, 2011] an approximate variational method was introduced for inference in an effort to reduce computational difficulties, but success was still limited.

A combination of an ensemble based and Bayesian based uncertainty quantification method was explored in [Pearce et al., 2018]. They regularise parameters about values drawn from a prior distribution. This regularizes, or “anchors” each neural network in the ensemble about a draw from a prior distribution.

The method we use in the experiments here, Monte Carlo dropout, is based on an analysis in [Gal and Ghahramani, 2016] showing that dropout approximates a Gaussian process, allowing estimates of variance to be created through multiple forward passes of the network, with a new dropout mask applied to the network weights between each forward pass.

In our experiments we use the Monte Carlo dropout method to explore how the uncertainty of the predictions made by a deep neural network architecture changes with the introduction of additional spectral channels. We have not found any related work on this topic in either the machine learning or remote sensing communities.

5.2 Experiments

We assessed network uncertainty characteristics for RGB images and multi-spectral images using Monte Carlo dropout, with 50 stochastic forward passes of our network. We examine how uncertainty changes with additional spectral bands by comparing uncertainty in the output of a network trained on RGB images vs. a network trained on multi-spectral images. We look at two different cases:

1. The network trained on RGB imagery and the network trained on multi-spectral imagery both make a correct prediction.

2. The network trained on RGB imagery makes an incorrect prediction while the network trained on multi-spectral imagery makes a correct prediction.

By studying the case when both the RGB image and the multi-spectral image lead to a correct classification we can determine if the additional spectral information present in the multi-spectral image is still buying us something, namely lower uncertainty so that if we were actually using one of these networks in the wild we could have higher confidence in the output of a model trained on multi-spectral imagery. We can use the case when a RGB image leads to an incorrect prediction and a multi-spectral image leads to a correct prediction to assess how much additional separation between classes is introduced by additional spectral information and how this impacts the uncertainty of the model.

5.2.1 EuroSat Dataset

Figure 5.1 shows an example of the distributions of the network output obtained for a single image using just the RGB bands of the image, compared to using 10 spectral bands from the image. The histograms are produced by plotting the results of the 50 stochastic forward passes of our model on both the RGB image and the multi-spectral image. In this case the image was correctly classified both when using RGB bands and 10 spectral bands. Although a correct classification was achieved using just RGB bands in this particular case, the multi-spectral image results in lower variance in network predictions, indicating that the additional spectral bands reduce the uncertainty in our network's predictions. By reducing the uncertainty in the network's predictions, the additional spectral information can increase our confidence that this network is producing correct predictions. This is valuable knowledge if we actually want to deploy our models for real world use cases.

It is also interesting to observe the case where an image is classified incorrectly using RGB bands but classified correctly when using the full multi-spectral image. Figure 5.2 illustrates this situation. In this case, class 0 was the correct class, and class

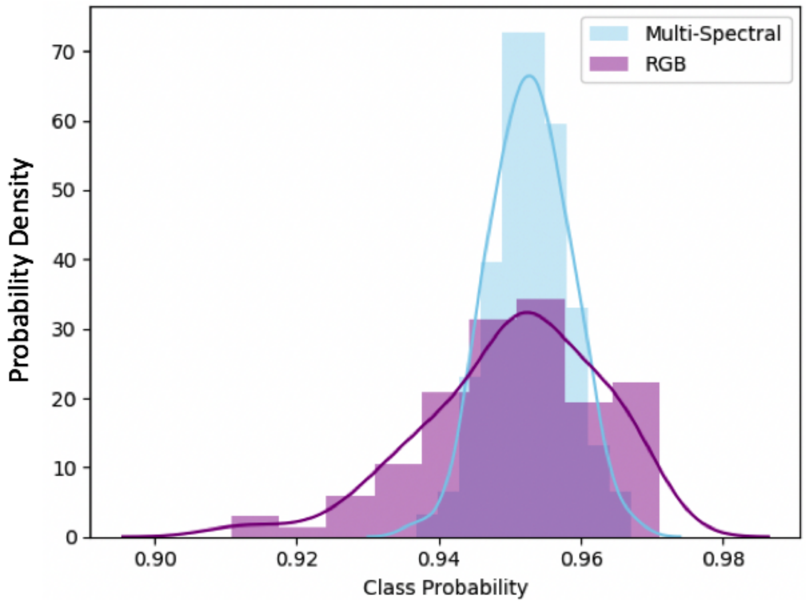


Figure 5.1: Network uncertainty for an image that was classified correctly both when using RGB bands and multi-spectral bands.

3 was the incorrect class predicted using just the RGB bands. When using the multi-spectral image, the separation between classes is large, and the variance associated with each class is small. In contrast, when using the RGB bands, the variance associated with each class is wide, to the point that the two different class distributions are overlapping. This reveals the high uncertainty associated with only using the RGB bands to make a classification decision in this case. On the rare occasions when an image class is predicted correctly using RGB and predicted incorrectly with multi-spectral inputs, typically high variance is exhibited using both band combinations. This analysis shows us that in addition to reducing the uncertainty of a deep neural network, the additional spectral information also often increases class separation making the prediction task easier. This makes sense given the higher classification accuracy we observed when using multi-spectral data in Chapter 3.

We also compiled class wide statistics on the distributions of our network's

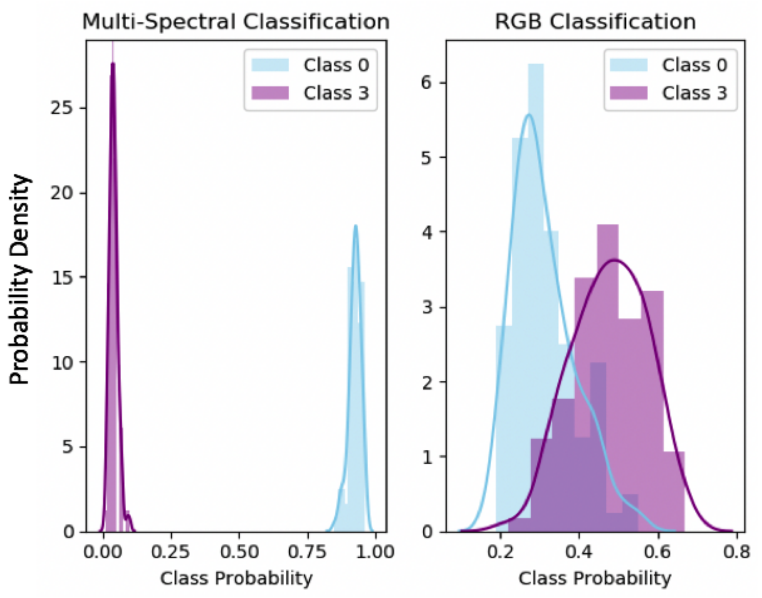


Figure 5.2: Network uncertainty for an image that was incorrectly classified using RGB bands, but was correctly classified when using 10 multi-spectral bands. In this case class 0 is the correct class, and class 3 is the incorrect class predicted when using the RGB image.

outputs for each land cover class using RGB images and multi-spectral images. Figure 5.3 shows a kernel density estimate of these distributions. Across all classes Figure 5.3 shows the substantial reduction in uncertainty that can be obtained when we have a network that works well with high dimensional imagery.

5.2.2 Produce Dataset

We also studied the uncertainty characteristics of a deep neural network when applied to our produce dataset. We observed the same general trends that we saw when analyzing uncertainty related to the EuroSat dataset. Typically a multi-spectral network exhibited smaller variance in its outputs compared to a network trained on RGB imagery although, there are cases where both the RGB network and the multi-spectral network both show high uncertainty. Figures 5.4 and 5.5 show network prediction distributions on an avocado and a tomato image that were both classified

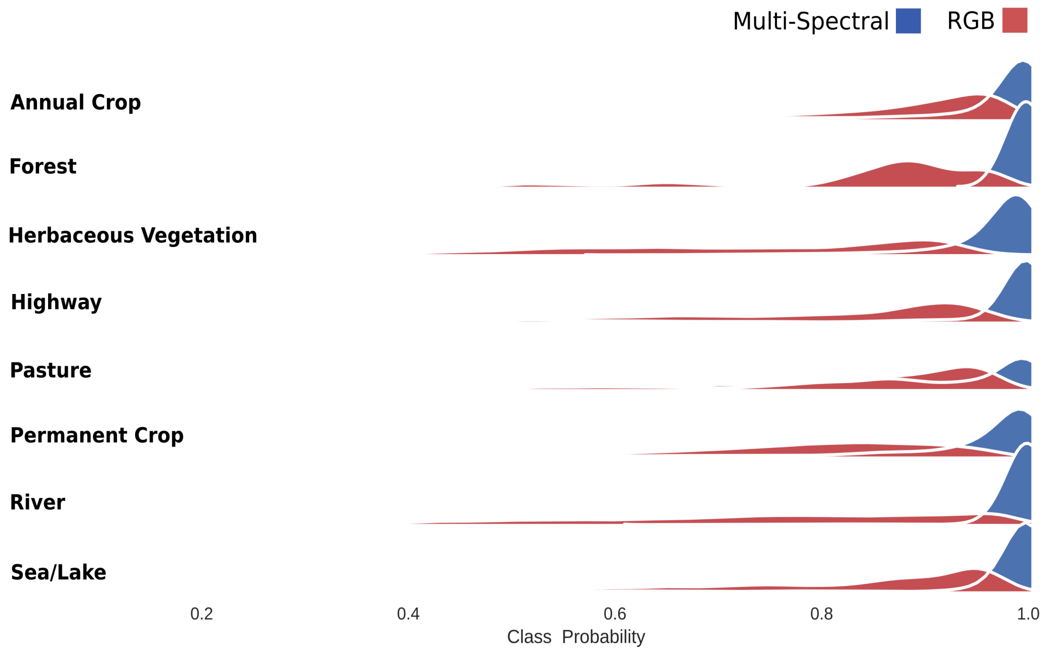


Figure 5.3: Network uncertainty displayed as kernel density estimates using RGB vs. multi-spectral images partitioned by land cover class.

correctly.

In Figure 5.6 we can see a case where an avocado was correctly classified as fresh by the multi-spectral network, but incorrectly classified when only using RGB wavelengths. In this case class 0 was the correct class and class 1 was the incorrect class. We can again see the wide separation in classes and low variance when making predictions with the multi-spectral network, while the RGB network has much wider variance.

These results are quite encouraging, showing that multi-spectral imagery provides benefits beyond higher classification accuracy. A network capable of processing multi-spectral imagery effectively can in many cases increase class separation and decrease the uncertainty in the predictions that our deep neural networks are making.

Figure 5.7 shows the avocado that was correctly classified with the multi-spectral

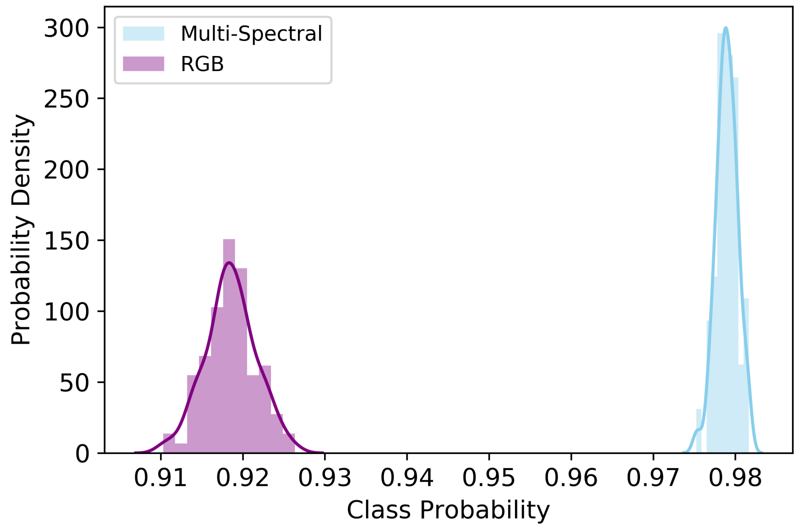


Figure 5.4: Network uncertainty on an avocado classification task.

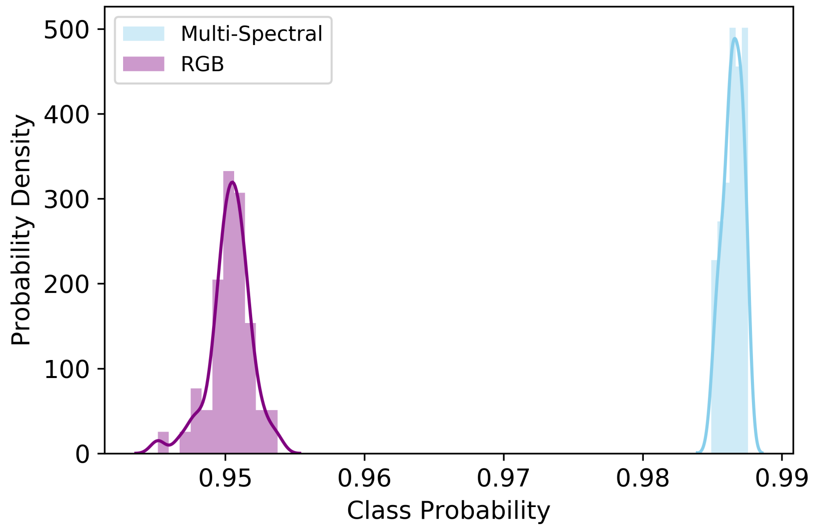


Figure 5.5: Network uncertainty on a tomato classification task.

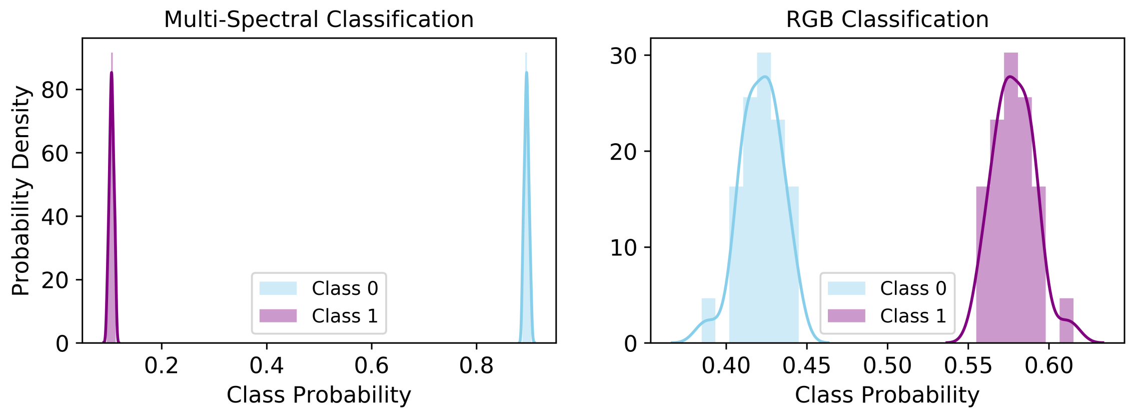


Figure 5.6: In this case the multi-spectral network produced a correct classification (class 0) while the RGB network produced an incorrect classification (class 1).

network and incorrectly classified with the RGB network, this image was used to produce the distributions shown in Figure 5.6. Notice there are no easily identifiable visual features, which creates difficulties for produce managers when trying to sort and organize produce. Our multi-spectral convolutional neural network was able to exploit patterns in the spectral content of the image across multiple wavelengths to classify this piece of produce correctly.

5.3 Conclusion

Criticism of neural networks is often directed at their interpretability (or lack thereof) and the difficulty in determining when we should not trust the outputs a network is giving us. The Monte Carlo dropout method we used here provides a means to assess the uncertainty in the predictions a neural network is making. When a network has high uncertainty we should be wary of trusting the predictions it is making. We leveraged this technique to explore how additional spectral bands in the input fed to a deep neural network impact the uncertainty characteristics of the network. This analysis showed that not only do additional spectral bands often

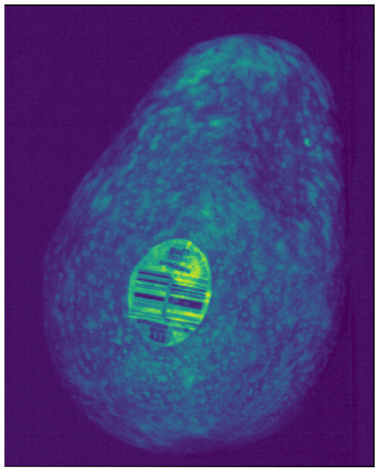


Figure 5.7: An avocado that was correctly classified by a multi-spectral network, but incorrectly classified by a network trained on RGB images. Shown in false color at 700 nm.

increase the classification accuracy of a network, uncertainty in network predictions is usually reduced significantly. This means if we are deploying a model for an application in the real world we can be more confident in the predictions a deep neural network is making when using multi-spectral or hyper-spectral data vs using RGB images.

CHAPTER SIX

CONCLUSION

Increasing availability of multi- and hyper-spectral imagery, and the valuable information that can be gleaned from such imagery motivates the development of techniques that can automate the analysis process. Multi-spectral and hyper-spectral imagery is valuable for agricultural applications, disaster recovery operations, and environmental monitoring, in addition to other applications.

We developed small and efficient convolutional neural network architectures and demonstrated the efficacy of these architectures on high-dimensional imagery, in addition to studying multi-spectral transfer learning approaches, and the uncertainty quantification of deep neural networks when applied to high-dimensional imagery.

6.1 Summary

In this work we explored the adaptation of small (in terms of number of parameters) convolutional neural networks to work well with high-dimensional imagery. The vast majority of work regarding convolutional neural networks has been performed on large, labeled, RGB image datasets.

We hypothesized that a network with relatively few parameters could work well with high-dimensional imagery, reducing the computation associated with such large input images, while simultaneously reducing the risk of overfitting to the smaller labeled datasets that are common with mult- and hyper-spectral imagery.

We demonstrated that our hypothesis was correct and that previous approaches to dealing with high dimensional imagery were often needlessly complex, adding several band separation and dimensionality reduction stages, or the previous approaches threw away valuable spectral information when transferring features from an RGB domain to a multi- or hyper-spectral domain.

We then showed that a transfer learning approach from multi-spectral satellite data to a small multi-spectral produce classification task ($\approx$ 100 images/class) could be very effective. Often multi-spectral information is discarded due to small training sets and the lack of pre-trained networks for multi-spectral imagery. This study showed that it was possible to transfer features learned from high-altitude overhead multi-spectral images to a significantly different, but also multi-spectral image classification problem and achieve good results.

Further, we applied Monte Carlo dropout to quantify uncertainty in our deep neural networks and demonstrated that a network that works well with high dimensional imagery also significantly reduces the uncertainty associated with its predictions, compared to a network trained solely on RGB images. This was a previously unknown benefit to making predictions using additional spectral bands during classification tasks and should further motivate the development of models specifically for high-dimensional imagery.

6.2 Contributions

This thesis made a number of contributions to the application of small and efficient neural network architectures to high-dimensional imagery.

- In this work we adapted recent developments in compressed, efficient convolutional networks (e.g., MobileNets [Howard et al., 2017] and SqueezeNets [Iandola et al., 2016]) to create a small (in terms of number of parameters) and computationally efficient convolutional neural network, SpectrumNet, capable of being trained to process the full spectral-spatial input space from multi-spectral imagery.

- We showed that it is possible to train such a network end-to-end on multi-spectral imagery, using no data augmentation, that achieves greater classification accuracy and higher sample efficiency than an identical network trained on RGB images. Additionally we did not need complex band separation or dimensionality reduction techniques.

- We extended our analysis to a Bayesian version of the SpectrumNet architecture in question, using Monte Carlo dropout [Gal and Ghahramani, 2016], and showed that with a multi-spectral network, epistemic uncertainty in the model is significantly reduced compared to an identical network using 3-channel RGB images. This allows us to be more confident in the predictions a multi-spectral network is making. This is the first analysis that we are aware of that studies how additional spectral channels impact uncertainty characteristics of deep neural networks.

- We demonstrated the application of our CNN architecture to a difficult produce (e.g. fruits, vegetables) quality classification task. We showed the effectiveness of the SpectrumNet architecture on this task in addition to demonstrating that it is possible to automate the classification of produce into different levels of quality or freshness.

- Finally, we showed the efficacy of transfer learning from multi-spectral satellite imagery to the produce task, as well as demonstrated the capability of our networks to train effectively on the small number of images present in the produce dataset.

6.3 Future Work

Given how little high-dimensional imagery in conjunction with deep learning has been explored relative to the RGB domain there are several interesting areas for future work. Layerwise relevance propagation is a technique to propagate the predictions made at the output of a network back to the inputs of the network such that the input pixels that contributed most significantly to the output classification are highlighted. This technique could be extended to multi-spectral and hyper-spectral imagery to assess relevance according to spectral features in addition to the spatial features that are highlighted with the current implementation of the technique. We also suspect that layerwise relevance propagation could be combined with Monte Carlo dropout to produce layerwise relevance distributions, in order to highlight the uncertainty associated with the features which are identified as highly relevant.

Recent work has presented what appears to be an effective technique for visualizing the loss landscape of deep neural networks [Li et al., 2018]. The loss landscape or surface corresponds to the loss function to be optimized that must be navigated by gradient descent. Studying the loss surface of a network trained on high-dimensional imagery vs. RGB images could reveal insights into the types of architectural components that work well when dealing with imagery that has a large number of spectral bands. It would also be interesting to study how the uncertainty of a deep neural network corresponds to the convexity or smoothness of the loss surface. Perhaps there is no correspondence, or perhaps a smoother more convex surface results in lower uncertainty.

Also, given the small size of the SpectrumNet architecture it has the potential to be deployed to some form of embedded platform for high-dimensional image processing. With no need for large compute resources this could enable a small sensor and processing package capable of being easily deployed in a variety of environments.

Michael Baylor. Planet labs targets a search engine of the world, 2018. URL `https://www.nasaspaceflight.com/2018/01/planet-labs-targets-search-engine-world`.

Deepak Bose, Michael J Wright, and Grant E Palmer. Uncertainty analysis of laminar aeroheating predictions for mars entries. *Journal of Thermophysics and Heat Transfer*, 20(4):652–662, 2006.

Jean C Buzby, Hodan Farah-Wells, and Jeffrey Hyman. The estimated amount, value, and calories of postharvest food losses at the retail and consumer levels in the united states. *USDA-ERS Economic Information Bulletin*, (121), 2014.

Rich Caruana. Learning many related tasks at the same time with backpropagation. In *Advances in neural information processing systems*, pages 657–664, 1995.

Marco Castelluccio, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva. Land use classification in remote sensing images by convolutional neural networks. *arXiv preprint arXiv:1508.00092*, 2015.

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

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

François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint*, pages 1610–02357, 2017.

Roger N Clark, Trude VV King, Matthew Klejwa, Gregg A Swayze, and Norma Vergo. High spectral resolution reflectance spectroscopy of minerals. *Journal of Geophysical Research: Solid Earth*, 95(B8):12653–12680, 1990.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Conference on Computer Vision and Pattern Recognition*, 2009.

Descartes Labs. Advancing the science of corn forecasting, 2016. URL `https://medium.com/descarteslabs-team/advancing-the-science-of-corn-forecasting-350603e3c57f`.

European Space Agency. Sentinel online, 2018. URL `https://sentinel.esa.int/web/sentinel/home`.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International conference on machine learning*, pages 1050–1059, 2016.

Alexander FH Goetz. Three decades of hyperspectral remote sensing of the earth: A personal view. *Remote Sensing of Environment*, 113:S5–S16, 2009.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Megandhren Govender, K Chetty, and Hartley Bulcock. A review of hyperspectral remote sensing and its application in vegetation and water resource studies. *Water Sa*, 33(2), 2007.

Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

Sascha Grusche. Basic slit spectroscope reveals three-dimensional scenes through diagonal slices of hyperspectral cubes. *Applied optics*, 53(20):4594–4603, 2014.

Driss Haboudane, John R Miller, Elizabeth Pattey, Pablo J Zarco-Tejada, and Ian B Strachan. Hyperspectral vegetation indices and novel algorithms for predicting green lai of crop canopies: Modeling and validation in the context of precision agriculture. *Remote sensing of environment*, 90(3):337–352, 2004.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, pages 97–109, 1970.

Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5353–5360, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *arXiv preprint arXiv:1709.00029*, 2017.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Conference on Computer Vision and Pattern Recognition*, volume 1, page 3, 2017.

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

George Klir and Bo Yuan. *Fuzzy sets and fuzzy logic*, volume 4. Prentice hall New Jersey, 1995.

Sanjeev J Koppal. Lambertian reflectance. In *Computer Vision*, pages 441–443. Springer, 2014.

Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7), 2010.

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

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6391–6401, 2018.

Haida Liang. Advances in multispectral and hyperspectral imaging for archaeology and art conservation. *Applied Physics A*, 106(2):309–323, 2012.

Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

Riley Logan, Byran Scherrer, Jacob Senecal, Neil Walton, Amy Peerlinck, John Sheppard, and Joe Shaw. Using hyperspectral imaging with machine learning to monitor grocery store produce. In *Optical Science and Engineering Conference*. Montana State University, 2018.

Guolan Lu and Baowei Fei. Medical hyperspectral imaging: a review. *Journal of biomedical optics*, 19(1), 2014.

Holly M Miller, Natalie R Sexton, Lynne Koontz, John Loomis, Stephen R Koontz, and Caroline Hermans. *The users, uses, and value of Landsat and other moderate-resolution satellite imagery in the United States-Executive report*. US Department of the Interior, Geological Survey, 2011.

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

Tony O'Hagan. Dicing with the unknown. *Significance*, 1(3):132–133, 2004.

Tim Pearce, Mohamed Zaki, Alexandra Brintrup, and Andy Neel. Uncertainty in neural networks: Bayesian ensembling. *arXiv preprint arXiv:1810.05546*, 2018.

Otávio AB Penatti, Keiller Nogueira, and Jefersson A dos Santos. Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 44–51, 2015.

Anthony Perez, Christopher Yeh, George Azzari, Marshall Burke, David Lobell, and Stefano Ermon. Poverty prediction with public landsat 7 satellite imagery and machine learning. *arXiv preprint arXiv:1711.03654*, 2017.

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

Anirban Santara, Kaustubh Mani, Pranoot Hatwar, Ankit Singh, Ankur Garg, Kirti Padia, and Pabitra Mitra. Bass net: band-adaptive spectral-spatial feature learning neural network for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(9):5293–5301, 2017.

Viktor Slavkovikj, Steven Verstockt, Wesley De Neve, Sofie Van Hoecke, and Rik Van de Walle. Hyperspectral image classification with convolutional neural networks. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1159–1162, 2015.

Ilya Sutskever. *Training recurrent neural networks*. University of Toronto Toronto, Ontario, Canada, 2013.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

Neil Walton, John Sheppard, and Joseph Shaw. Using a genetic algorithm with histogram-based feature selection in hyperspectral image classification. In *The Genetic and Evolutionary Computation Conference*, 2019.

Michael Xie, Neal Jean, Marshall Burke, David Lobell, and Stefano Ermon. Transfer learning from deep features for remote sensing and poverty mapping. *arXiv preprint arXiv:1510.00098*, 2015.

Dongbin Xiu and George Em Karniadakis. Modeling uncertainty in flow simulations via generalized polynomial chaos. *Journal of computational physics*, 187(1):137–167, 2003.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.

Wenzhi Zhao and Shihong Du. Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach. *IEEE Transactions on Geoscience and Remote Sensing*, 54(8):4544–4554, 2016.