Neural Network Application in Hyperspectral and Multilevel Diffractive Lens Imaging

Alexander Hamrick
University of Utah

UUCS-21-004

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

21 April 2021

Abstract

Traditional imaging systems utilize additional optical elements in order generate hyperspectral images or correct optical aberrations. However, with increasing computational power, it is becoming advantageous to simplify our imaging systems and instead rely on postprocessing of the image. Neural networks have been particularly effective in solving such problems since they can approximate any function when given enough data. In this paper, we analyze two neural networks, one of which is novel, on their ability to generate hyperspectral images from image sensor data, and we propose various modifications to an existing neural network pipeline for processing MDL images.

NEURAL NETWORK APPLICATION IN HYPERSPECTRAL AND MULTILEVEL DIFFRACTIVE IMAGING

by

Alexander Hamrick

A Senior Honors Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the

Honors Degree in Bachelor of Science

In

Computer Science

Approved:

[illegible]

Rajesh Menon	Mary Hall
Thesis Faculty Supervisor	Director, School of Computing
Thomas C. Henderson	Sylvia D. Torti, PhD
Honors Faculty Advisor	Dean, Honors College

April 2021

ABSTRACT

Traditional imaging systems utilize additional optical elements in order generate hyperspectral images or correct optical aberrations. However, with increasing computational power, it is becoming advantageous to simplify our imaging systems and instead rely on post-processing of the image. Neural networks have been particularly effective in solving such problems since they can approximate any function when given enough data. In this paper, we analyze two neural networks, one of which is novel, on their ability to generate hyperspectral images from image sensor data, and we propose various modifications to an existing neural network pipeline for processing MDL images.

TABLE OF CONTENTS

ABSTRACT ii

Chapters

1. INTRODUCTION 1

2. HYPERSPECTRAL IMAGING 4

2.1 Background 4
2.2 Related Work 5
2.3 Methods 6
2.3.1 Deep UNET 6
2.3.2 SieveNet 7
2.4 Results 9
2.4.1 32x32 Images 9
2.4.2 64x64 Images 14
2.4.3 Reconstruction Speed 17
2.5 Conclusion 18

3. MULTILEVEL DIFFRACTIVE LENS IMAGING 20

3.1 Background 20
3.2 Related Work 20
3.3 Methods 22
3.4 Results 25
3.5 Conclusion 30

REFERENCES 32

CHAPTER 1

INTRODUCTION

Ever since AlexNet dominated the ImageNet competition in 2012 [1], convolutional neural networks (CNNs) have become the go-to strategy for many image classification and computer vision tasks. It has also been discovered that CNNs perform quite well in a variety of image-to-image regression tasks [2]–[4] as they are able to maintain local spatial coherence while retaining all the pattern recognition of other network architectures. This makes them an excellent choice for many imaging tasks including those related to hyperspectral imaging [5], [6].

Convolutional neural networks (CNNs) work by analyzing small groupings of pixels in a given image and generating a feature map. This feature map is passed to other layers, which can reduce (through pooling) or increase (through further convolutions) the dimensionality of the map. Each of the layers in the network is made up of many nodes, each of which has a weight that determines how much it will manipulate its inputs. This means that the network as a whole simply represents a function, and we can adjust the weights of each node in order to modify that function. The weights of each node are learned via training, during which the network maps input images to an output image or class. At first, the network will perform poorly, since the node weights are arbitrary, but the node weights get trained via an optimizer (like stochastic gradient descent) and backpropagation [7] in order to improve the network's ability to transform input images into the desired output.

The UNET, a type of CNN utilized in both sections of this work, has proven effective for both regression and classification tasks in fields like bioimaging since its conception in 2015 [8]. The UNET differs from other CNN architectures in that it first downscales the image before upscaling it again. It concatenates outputs from the downsampled and upsampled sections in order to gain spatial information on various scales. When diagrammed, as in Figure 1.1, these networks form a U shape, giving them their name.

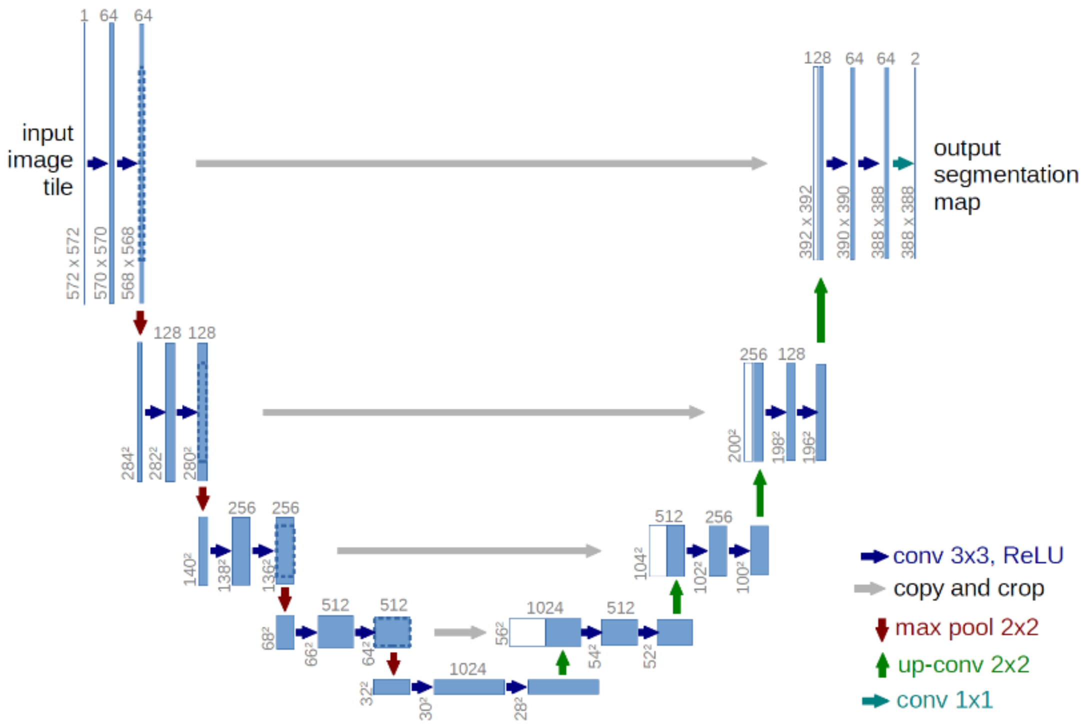

Figure 1.1. A generic UNET architecture (image from [8])

The format for the rest of this paper is as follows: Chapter 2 will discuss two neural networks architectures and their success in generating hyperspectral images from image sensor data; Chapter 3 will analyze a neural network pipeline proposed for reconstructing MDL images, several modifications to this network, and its overall performance on both low-resolution and experimental MDL image sets.

CHAPTER 2

HYPERSPECTRAL IMAGING

2.1 Background

Hyperspectral images measure the same scene across many spectral bands. This allows them to be particularly useful in identifying specific materials and substances. Such images are used to identify cancer [5], oil spills [9] and many other biological, biomedical, geological, and astronomical phenomena [10], [11]. This is because different substances have different spectral signatures, information which can be more effectively utilized when an image contains more spectral information.

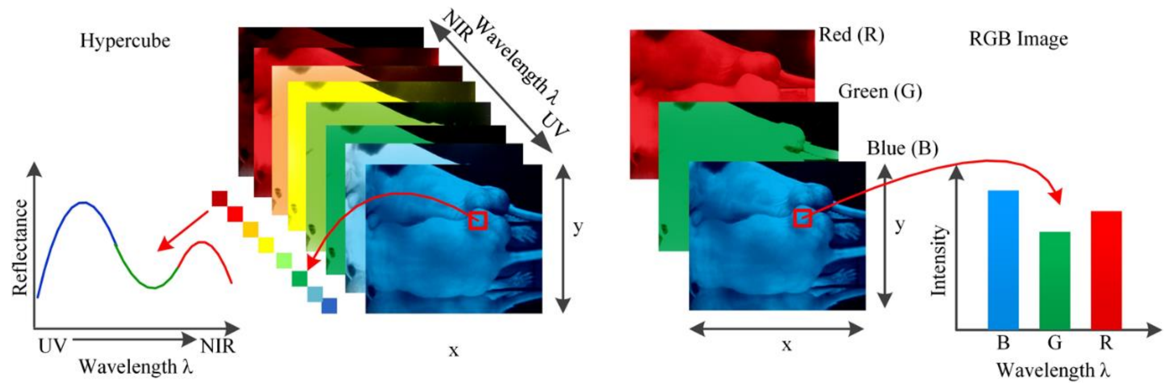

Figure 2.1. Comparing hyperspectral and regular images (image from [12])

It has been shown that images can be generated without optics, using sensor data alone [13], and it is also possible to generate multispectral images computationally when utilizing an added diffractive filter [14]. Past recovery algorithms treated each spectral band (wavelength) as a regularized linear inverse problem, recovering the band after generating a calibration matrix.

These techniques require the long and tedious process of calibration, while still sometimes taking hours to reconstruct a single image. We propose the use of CNNs to generate hyperspectral images from sensor data recovered through a diffractive layer. The ability to do so effectively would have a profound impact on countless fields, making hyperspectral imaging easier and cheaper, while making massive strides in speed over similar strategies [14].

2.2 Related Work

There have been efforts in recent years to utilize neural networks in the generation of multispectral and hyperspectral images. HSCNN was originally proposed as a deep learning solution transform upsampled RGB images into hyperspectral images [15]. HSCNN+ built on top of this by developing two networks which "removed the hand-crafted sampling from HSCNN," and their networks placed first and second in the NTIRE 2018 Spectral Reconstruction Challenge [16]. In 2019, cGANs were proposed to generate a mapping between RGB and multispectral images, although this was done using a small, augmented dataset [17]. The use of GANs in the generation of hyperspectral image generation was further expanded in [18] when Liu *et al*. were able to achieve state-of-the-art results on the ICVL dataset using their own adversarial network.

Clearly neural networks have a place in the generation of hyperspectral images, but other solutions have largely relied on RGB images when generating hyperspectral ones. We propose a solution that takes advantage of an added diffractive filter in order to generate hyperspectral images directly from image sensor data.

2.3 Methods

The data used to train and evaluate our models was all synthetic, and it has been shown that training CNNs on synthetic data can be an effective means for training and supplementing datasets [19], [20]. So, while we are exclusively utilizing synthetic data, our results should indicate if, at least in principle, CNNs can effectively predict hypercubes from sensor data.

Our ground truth images were a set of 5000 real subsamples of hyperspectral images taken from [21]. The synthetic image sensor data was predicted using optical diffraction modeling from the image to the sensor. Our data was split into independent training, validation, and testing datasets. We developed two different CNNs to generate hyperspectral images. They were all built with the Keras library to re-construct 32x32 hyperspectral images with 25 bands (440nm-800nm) from image sensor data of size 72x72. It should be noted that the meaningful dimensions of the input data were 32 x 32 as the rest of the input was zero padding. Our two networks will be referred to as the Deep UNET and the SieveNet, and we discuss their specific implementation details below.

2.3.1 Deep UNET

UNETs have been used in the past with great success in image-to-image regression tasks [4], [8], [22] including as encoders in generative adversarial networks

[3]. They are now a well-established architecture for CNNs. In general, UNETs contain two sections, downsampling and upsampling. These sections are largely symmetrical, and each module is eventually concatenated with its appropriate partner. That is, the i^{th} module, m_i, contains a bridged connection with m_{n-i} if n is the number of total modules in the network. This makes them beneficial for a variety of imaging tasks because they manage to capture both spatial and feature information at a variety of scales. Our Deep UNET follows a similar architecture to that which was posed in the original article [8]. We first crop the input image to obtain dimensions 64x64. The downsampling section consists of a series of convolutional modules. Each module contains two series of: a convolutional layer, followed by a batch normalization, followed by a ReLU activation layer. The module is then finished with a max pooling layer, which decreases the dimensions of the image by a factor of 2. This process was repeated until dimensions of 2x2 with 2048 feature maps was reached and the upsampling section began. The upsampling modules are the same as the downsampling modules, except that they begin with a deconvolutional layer and concatenation with the corresponding sized output from the downsampling section. The upsampling modules also do not contain pooling layers. This repeated until we had dimensions 64x64 with 25 feature maps. These feature maps would ultimately come to represent the 25 bands. Finally, this output was cropped to give dimensions 32x32x25.

2.3.2 SieveNet

The SieveNet is inspired by the ResNet [23] and HighwayNet [24] architectures. It generates three different channels from the input via depthwise convolution with (kernel size, stride) parameters of (8, 1), (4, 2), and (2, 4). Each channel contains a series of convolutional layers (sans padding) and leaky ReLU activation layers (alpha = 0.3). These are ultimately followed by a depthwise convolutional layer (kernel size=original, stride=1). This is added with the intent of helping to align the subnets prior to concatenation. These channels all have size 49x49x512 when concatenated. The concatenated channel undergoes a dropout layer (rate =0.01) and multiple convolutions, each followed by a leaky ReLU layer with decreasing alpha values. The network finishes with a convolutional layer that has an activity regularizer that heavily penalizes negative values. The activity regularizer defines the loss L, shown in equation 2.1, for a given input weight matrix x.

$$L = \left|\left| max\ \{x_i, 0: x_i \in x\} - x \right|\right|_2^2 + mean(|x|) \tag{2.1}$$

The lack of added padding in the convolutional layers allowed the final dimensions to be 32x32x25. Our SieveNet architecture is shown in Figure 2.2.

Both networks were trained in Google Colab using the Adam Optimizer [44]. The loss was defined as a regularized mean square error for the SieveNet while the loss for the Deep UNET was the mean squared error. They were both trained for 64 epochs with a learning rate of 0.0001 before being trained for 64 more epochs with a learning rate of 0.00001 to fine tune the models. Our training dataset consisted of 80% of our original dataset. The networks then utilized 10% of the data for validation during training, and the final 10% was utilized in testing.

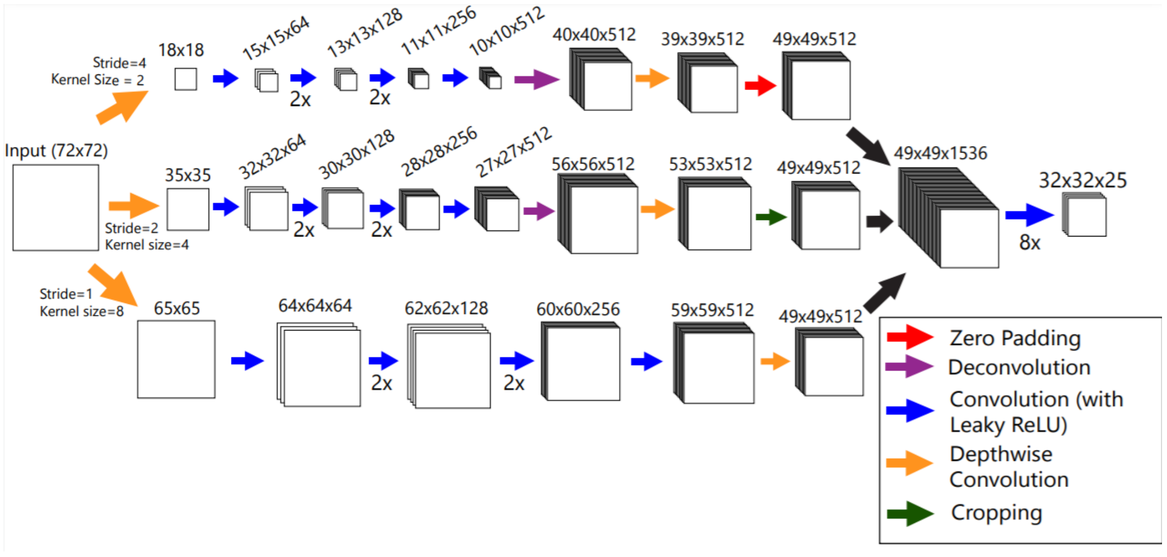

Figure 2.2. The proposed SieveNet architecture.

2.4 Results

2.4.1 32x32 Images

Both of our networks were originally trained and tested using 32x32 images. We will show three testing images to highlight the performance of both of our networks, and we will analyze them in terms of their spatial and spectral performances. For each sample, we show the ground truth RGB image, the predicted RGB image, the spectral plot of the center pixel (marked with an o), and the spectral plot of another randomly selected pixel (marked with an x). We will also discuss the images in terms of their average Structural Similarity Index (SSIM), over the 25 hyperspectral bands. The SSIM for each band was calculated as in [25]. Over all the test images, the average SSIM score for the SieveNet was 0.7483 while the average SSIM score for the UNET was 0.738. We analyze these networks' performance on some interesting samples below.

Figure 2.3 was chosen to be highlighted from the test set because it contains a high degree of complexity for a 32x32 image without containing a lot of complex colors. The ground truth image shows a sun smiling with a rather plain background. Our networks both seemed to reconstruct the general shape of the image, but the reconstructions are out of focus. The details on the target image are far sharper than those in the reconstructions and the color accuracy is passable but certainly not indistinguishable. Notice that the smile largely disappears in the SieveNet and fully disappears in the UNET. Additionally, the UNET produces a noisier, grayer background. Overall, this indicates that these networks both struggle to retain information about fine details. The average SSIM for the SieveNet reconstruction is 0.471 while the average SSIM for the UNET reconstruction is 0.569.

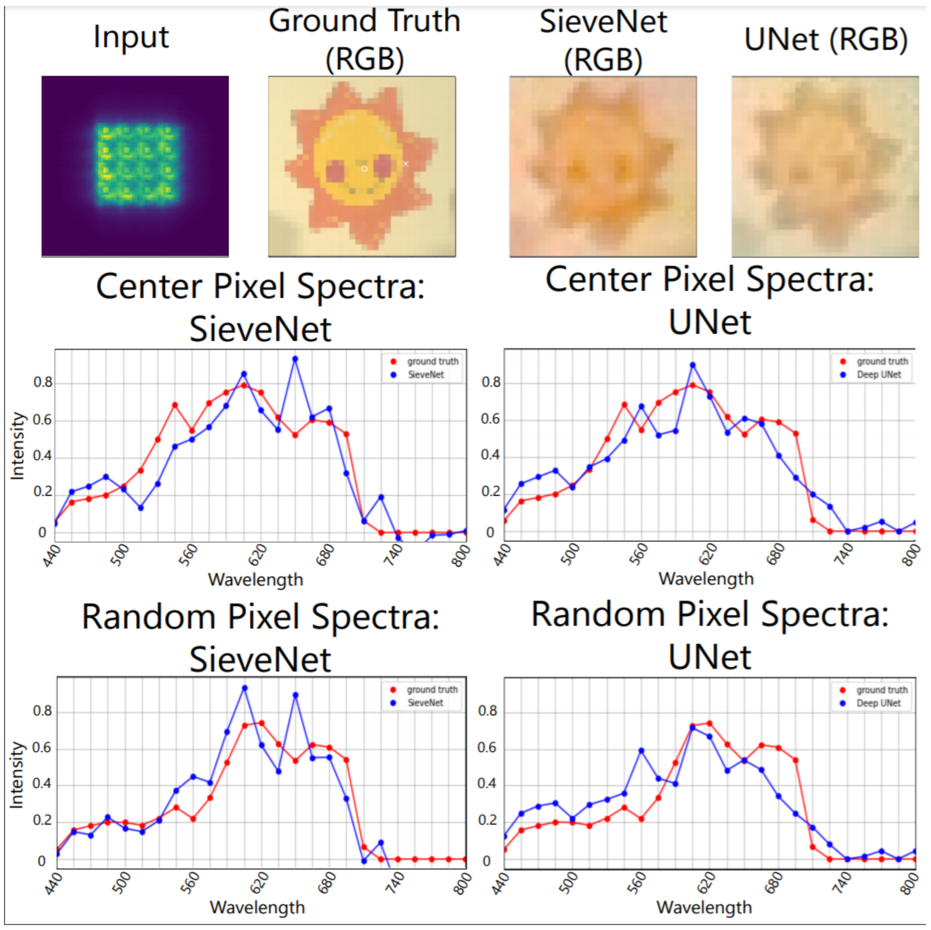

Figure 2.3

Both networks produced promising spectral results for both the center pixel and the randomly chosen pixel (located in the shadow at the bottom right). The general shape is preserved in   all four plots, although the UNET seems to follow the curve a little closer. Overall, Figure 2.3 shows promise in the continued development of these networks, but highlights the need for improved fine detail retention.

Figure 2.4 shows an image which was chosen due to its sharp edges and plain colors. It makes sense that the networks would be able to perform better on this simpler image, and the SSIM results reflect that. The SieveNet had an average SSIM score of 0.637, while the UNET had an average SSIM score of 0.753. Clearly, both of the networks are able to capture the general structure of the image, but the lines in the reconstructions are considerably less distinct than in the ground truth image. Similarly, the colors surrounding the main panel are almost indistinguishable from each other in the UNET, and barely distinguishable in the SieveNet. Both networks had poor color accuracy, even for the main red color, producing instead washed out pink and brown versions.

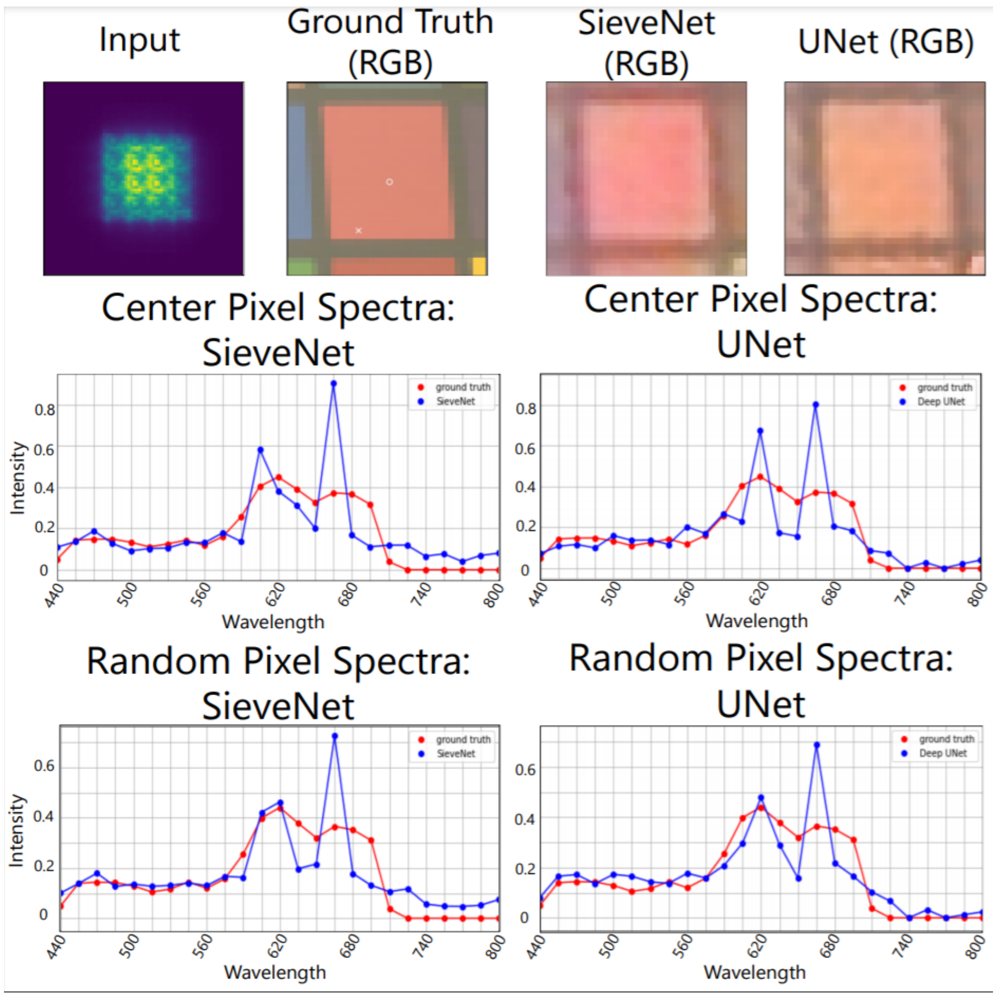

Figure 2.4

As for the spectral profiles, we can see that both networks had two separate spikes in the middling wavelengths for both of the tested pixels. This may indicate that the networks are experiencing similar problems when reconstructing the images, which could be due to shared architectural bottlenecks or an insufficient training dataset.

Figure 2.5 shows an image that was chosen due to its clear, fine features in the center. We immediately see the same story as with the other samples in terms of general shape. Both networks produced a very recognizable image with similarly sized features. Again, The Deep UNET seems to be noisier while the SieveNet has worse color accuracy. The SSIM scores were 0.64 for the SieveNet and 0.756 for the UNET.

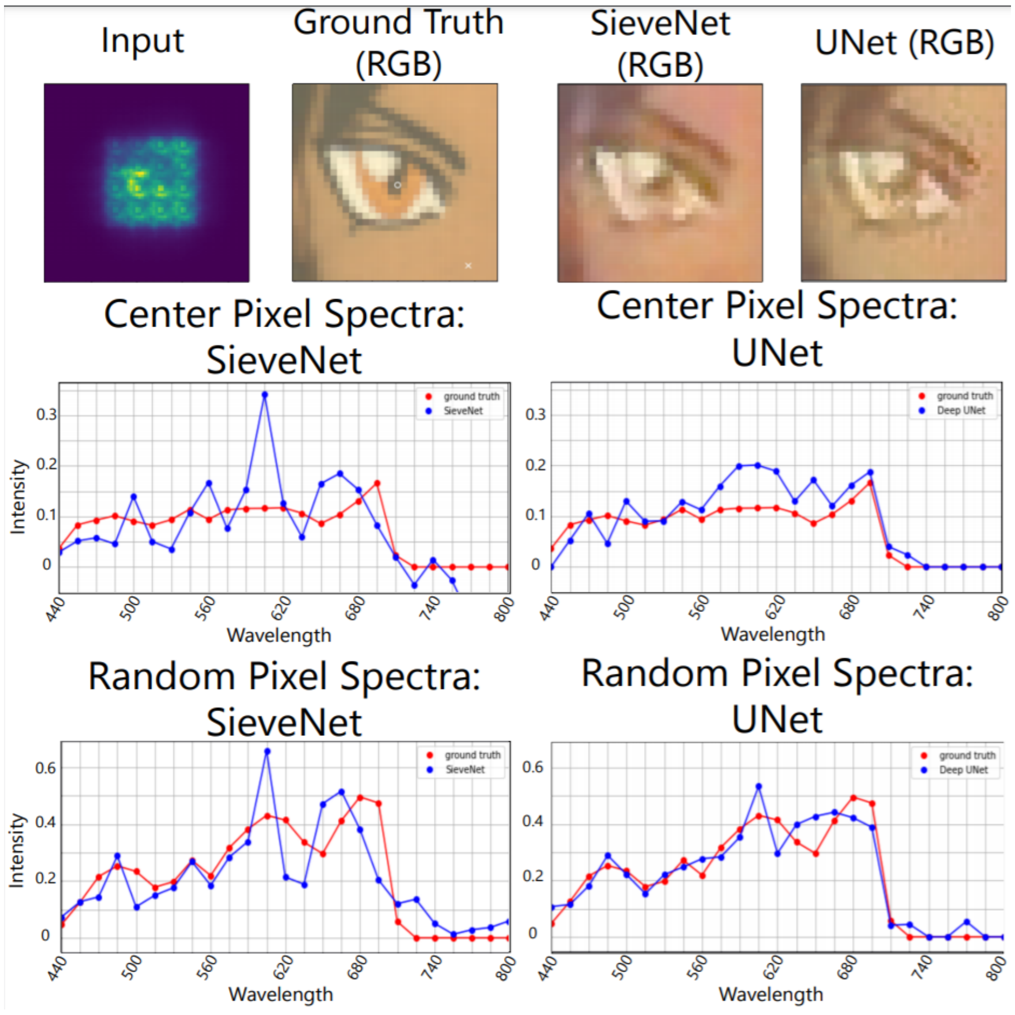

Figure 2.5

The spectral profiles of this image indicate that the SieveNet seems to repeatedly predict spikes in intensity in the middling wavelengths. The UNET, on the other hand, seems to have a more representative spectral plot for both the center and random pixels. Additionally, both networks had poor reconstructions of the center pixel, as it is in between 3 edges of contrasting colors, which again highlights the case where these networks struggle the most.

Overall, these samples were reflective of our networks' performances on other samples. Our results seem to indicate that both networks do well in reconstructing large shapes but struggle to maintain the fine detail of the hyperspectral images. The UNET also produces overly noisy images while the SieveNet seems to give most images a red tint.

2.4.2 64x64 Images

Our networks were originally trained to reconstruct 32x32 images, but given the SieveNet's flexibility in input sizes (simply requires an input image with zero padding of 40x40), we decided to train it on 64x64 images as well. We started the network with the initial weights from after the 32x32 training, which acted as preliminary training before fine tuning the network on 64x64 images. This made it so that we only had to train the network for an additional 64 epochs before obtaining solid results. Our 64x64 image training set generated the synthetic data the same way, as subsamples of larger hyperspectral images, but it also included zero matrices as well as linear combinations of spatially constant samples. These additions were made in order to teach the network the linearity of the problem. Overall, the SieveNet had an average SSIM score of 0.8282 on the testing images, a significant improvement over the 32x32 hyperspectral images. Below we highlight the performance on some of the more difficult images.

Figure 2.6 shows the reconstruction of a higher resolution version of the image from Figure 2.3 shown in our 32x32 section, and it highlights how well the network scales to higher resolutions. Spatially, the RGB version of this image looks really good, and this is likely due both due to the improved training dataset as well as the increased resolution, since that masks the errors in reconstructing fine details that are more apparent at lower resolutions. The average SSIM across the 25 bands of this hyperspectral image was 0.617 which is far better than the 0.471 from the 32x32 version. The spectral plots for the two chosen pixels are also quite encouraging.

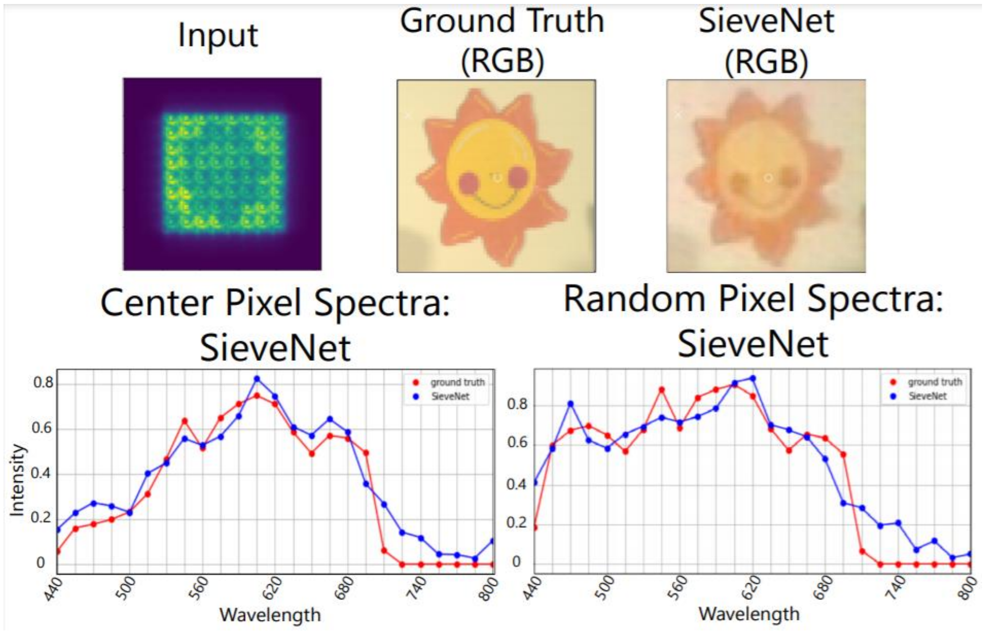

Figure 2.6

Figure 2.7 shows a similar story, again with a higher resolution version of one of the images highlighted earlier in Figure 2.5. As shown in the RGB version, this image looks far better than its 32x32 counterpart. This is backed up by the fact that its average SSIM is 0.764 as opposed to 0.640 in the 32x32 image. Again, the center pixel of this image shows poor spectral results, as the pixel borders a high-contrast edge, but the random pixel from the doll’s forehead shows that the network does well in reconstructing pixels from low contrast areas.

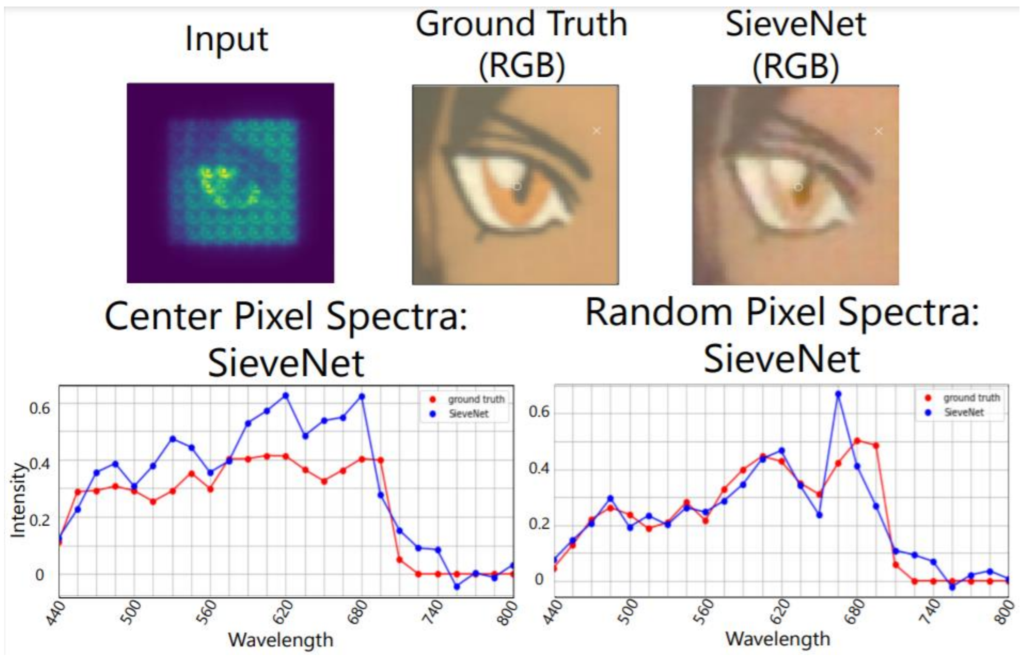

Figure 2.7

Figure 2.8 is a new, higher resolution image with lots of contrast, but the SieveNet still reconstructs the image very well. The RGB image is, again, very recognizable, and the large features all appear spatially accurate. It does appear that the network struggled to define the whiskers as these were only a few pixels in width. This once again highlights the room for improvement in reconstructing fine details. The average SSIM for this image was 0.722, which is on par with the previous shown reconstructions. Additionally, both of the plotted pixels were located near contrasting edges, and both show decent results.

Overall, the SieveNet performed at least as well on the 64x64 images as it did on the original 32x32 images. This gives us confidence that our networks can produce promising results when given a comprehensive dataset that helps the models learn the linearity of the problem. Additionally, further improvements to the network architectures will certainly lead to even better results.

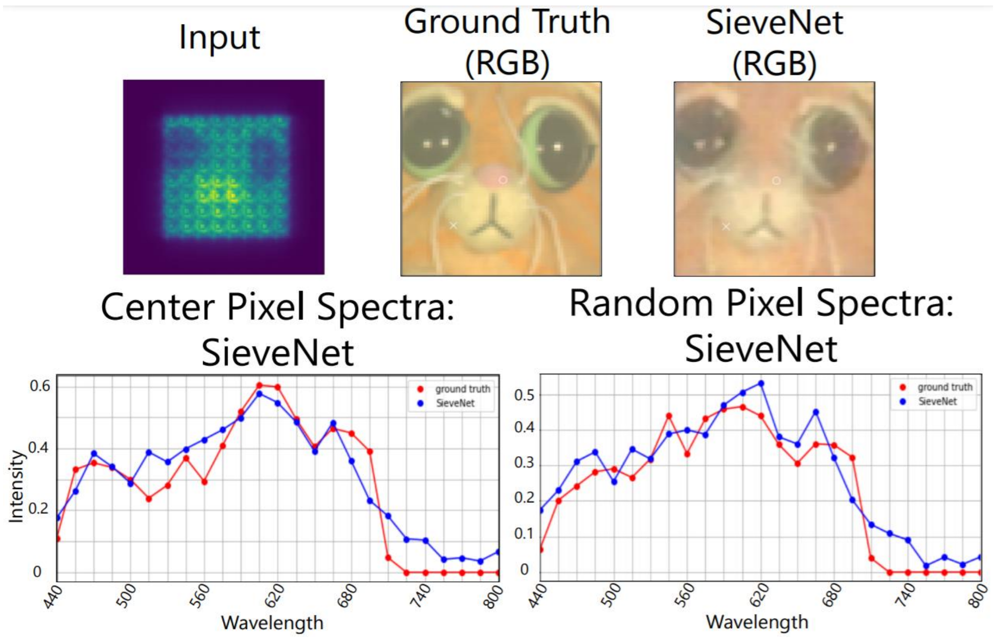

Figure 2.8

2.4.3 Reconstruction Speed

While it is true that CNNs can take days to train, the calibration phase required to obtain the calibration matrices for regression requires a comparable amount of time. Where these techniques differ tremendously is in their reconstruction time. Using a strategy similar to that proposed in [14], it took well over 4 minutes to generate a 320x320x25 hyperspectral image. On the other hand, our SieveNet generated the same hyperspectral image in less than 3 seconds. See Fig. 9 in Appendix A for a full table of our experiment. Our results show that the speedup of CNNs over the former strategy can be over 100 times (Figure 2.9), and makes real-time, computational hyperspectral imaging a possibility. The reconstruction of most images will be almost instant on adequate hardware, as this comparison was done on the same CPU (Intel Xeon 24 Core 2.66Ghz, with 128 GB RAM) for both methods to compare them as fairly as possible.

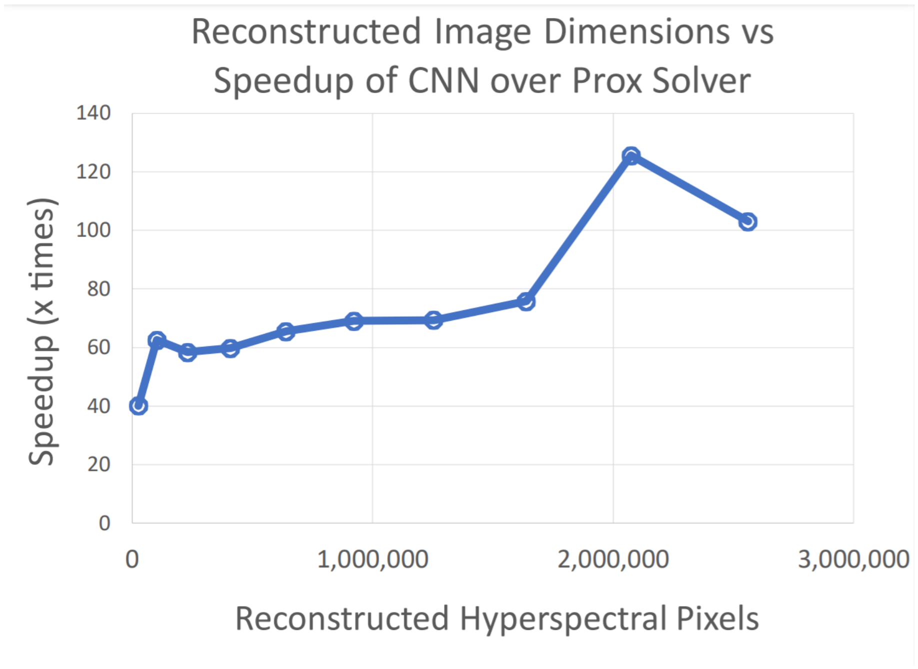

Figure 2.9 The speedup of SieveNet over linear algebra-based solvers

2.5 Conclusion

We studied the performance of neural networks in generating hyperspectral images via synthetic sensor data. We proposed two general CNN architectures to achieve this task, and we demonstrated that these networks could produce very promising results, often producing reasonably accurate pixel spectra. Our models struggled most to generate fine details, but they demonstrated that convolutional neural networks certainly have a space in the field of hyperspectral image construction. It is possible that future models could see significant improvements upon utilizing denoising techniques [26]. Similarly, architectural tweaks could greatly improve our networks’ efficiency [27] and accuracy while additional data preprocessing could help the networks more effectively predict the appropriate hypercube. We believe our results warrant further research into the use of CNNs in constructing images via sensor data and the use of experimental data for training models in future studies.

CHAPTER 3

MULTILEVEL DIFFRACTIVE LENS IMAGING

3.1 Background

Refractive lenses are the traditional means for focusing light within imaging mechanisms, however other types of lenses can also be used. In recent years, diffractive lenses have gained popularity due to their lighter and thinner form [28]. Multilevel diffractive lenses in particular are useful for their relative ease of production and high image quality when compared with other flat lenses [28]-[30]. However, a problem with a majority of these strategies is that these imaging systems experience significant aberrations (particularly chromatic) which result in a distorted image [28], [31]. In order to eliminate these distortions without additional optical elements, we necessarily need to apply various forms of post-processing to the images. In this chapter, we will analyze the end-to-end, image processing pipeline proposed in [29] with various modifications, on our own experimental datasets using two different diffractive lenses.

3.2 Related Work

Numerous works have been proposed to allow high quality imaging with only a single lens element, as opposed to the numerous optical elements included in most modern imaging systems. In 2011, a non-stationary deconvolutional method was proposed for removing optical aberrations [32]. Following this, another system was proposed to more efficiently solve for the point spread functions associated with particular optical systems via a convex cross-channel term [33]. Works have also shown that such deconvolutions can be performed blind, or without any calibration step [34], [35].

Deep learning has also seen massive advancements in image processing. The SRCNN [36] allowed for the generation of high quality, super-resolution images. This was followed by the VDSR [37] as well as the DRCN [38] and DRRN [39] networks, each of which draws on the skip connections from ResNet [23]. Other popular networks for image-to-image tasks include the VGG-16 [40], Inception Net [41], ResNet, and UNET [8] architectures. Recently, though, the proposal of generative adversarial networks (GANs) [42], and more specifically conditional generative adversarial networks (cGANs) [43], has allowed for a whole new model of image reconstruction. In this chapter we will utilize one such cGAN, Pix2Pix [3] which performs well in a variety of image-to-image tasks. This Pix2Pix architecture is utilized in [29] by their MDL image processing pipeline.

The Pix2Pix network provides a generic framework for image-to-image translation. This is because it contains a generator network which modifies some input image, and a discriminator network, which determines whether a given input is real or fake (with fake meaning generated by the generator network). The loss function for the generator is then tied to its ability to pass off its images to the discriminator as real. This creates a cycle whereby the discriminator and generator each improve as they find new ways to discern fakes and new ways to make the image look more realistic.

3.3 Methods

In order to process our MDL images, we utilized the technique proposed in [29] with minor modifications. This process consists of an end-to-end neural network pipeline. The first element of the pipeline is a pix2pix conditional GAN, built with a UNET as the generator and a PatchGAN network as the discriminator. This is the same as the network proposed in [3]. The structure of our network is shown in Figure 3.1. We utilize the Adam optimizer [44] with a learning rate of 0.0002 and momentum parameters of 0.5 and 0.999. The second element of the network pipeline consists of a Deep Recursive Residual Network (DRRN), as proposed in [39]. Figure 3.2 shows the general architecture of a DRRN, consisting of concatenated recursive convolutional blocks. Our DRRN was built using 9 recursive units and one recursive block, as described in [29]. We used the Adam optimizer for this network as well with a learning rate of 0.0001. In order to train our pipeline, we would first train the GAN for 100 epochs, before feeding its color corrected images into the DRRN as the input images and training the DRRN for 100 epochs. We show results for training the DRRN patch-wise on the images as well as image-wise.

Our experimental datasets were generated by two different lenses, L1 and L2. Both of these lenses were fabricated via direct laser write grayscale lithography [45], and L2 is the same as L3 in [46]. Details about L1 and L2 are shown in Table 3.1.

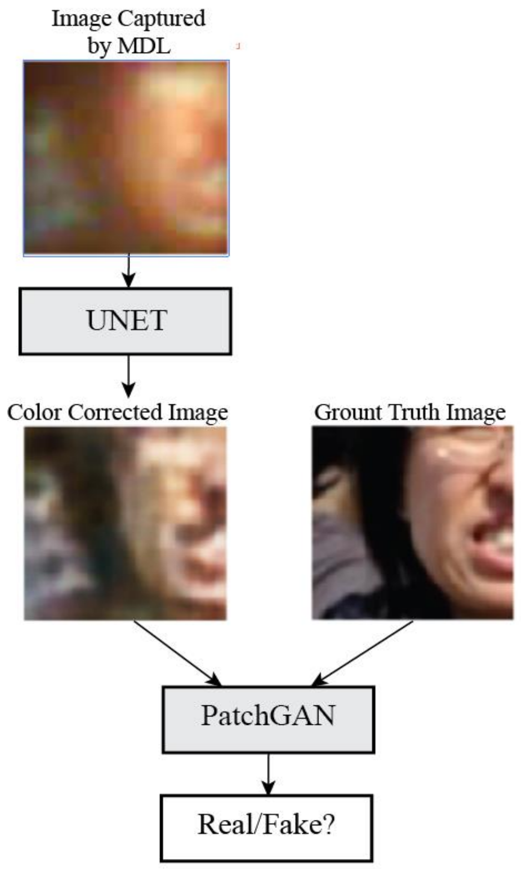

Figure 3.1 Our Pix2Pix GAN Architecture

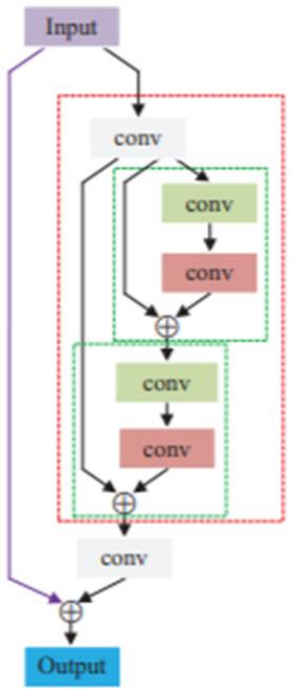

Figure 3.2 Architecture of a DRRN. The red box represents a recursive block, while the green boxes denote recursive units. (image from [39])

Flat Lens	Diameter (mm)	Focal Length (mm)	F#	Operating Wavelengths (nm)
L1	0.15	1	6.67	450-850
L2	4	45	11.25	450-750

Table 3.1 A description of the flat lenses used for gathering images

Our analysis of this neural network pipeline consists of three parts: comparing L1 and L2, adding preprocessing techniques, and evaluating various DRRN loss functions and architectures. In order to do all of these, we needed to make our own datasets. All of our ground truth images were 64x64 subsamples of images taken from [47], and we used both lenses to capture the ground truth images from a display (Omen model B0711T4RDF). For the purposes of lens comparisons, we created two datasets (one with each lens) of size 784, with testing datasets of size 156. Using these datasets, we compared the performance of L1 and L2. Then, we created a dataset with 13268 training images and 3317 testing images taken with L2. We used this dataset to test the use of preprocessing techniques. The preprocessing included image sharpening, median filtering, and white balancing done in MATLAB using the “imsharpen,” “medfilt2,” and “wbalance” functions. We also used this expanded dataset to test modifications to the network pipeline. When modifying the pipeline, we only modified the DRRN, and we modified it in three ways: adjusting the architecture to give it 16 recursive units and 3 recursive blocks; modifying the loss function to include Mean Absolute Error, Mean Squared Error, the cross-channel loss proposed in [29], and SSIM (calculated as 1-SSIM); and training it both patch-wise (via 8x8 patches) and image-wise. We analyze our image processing results via SSIM and Peak Signal-to-Noise Ratio (PSNR).

Finally, after determining the optimal lens and pipeline, we will show some experimental images and their reconstructions via the pipeline, which we hope will demonstrate the ability for such a pipeline to work in a realistic setting, as opposed to working on a given set of images captured from a display.

3.4 Results

When comparing L1 and L2, we find that L2 generally has better color accuracy and clearer captured images. This, in turn, resulted in better network performance, with the L1 processed images having an SSIM of 0.355 and PSNR of 16.33 while L2’s processed images had and SSIM of .385 and a PSNR of 16.53. Figure 3.3 shows some sample images taken by these two lenses. After performing this experiment, we decided to use L2 for the remainder of our study, due to its higher quality in terms of captured and processed images.

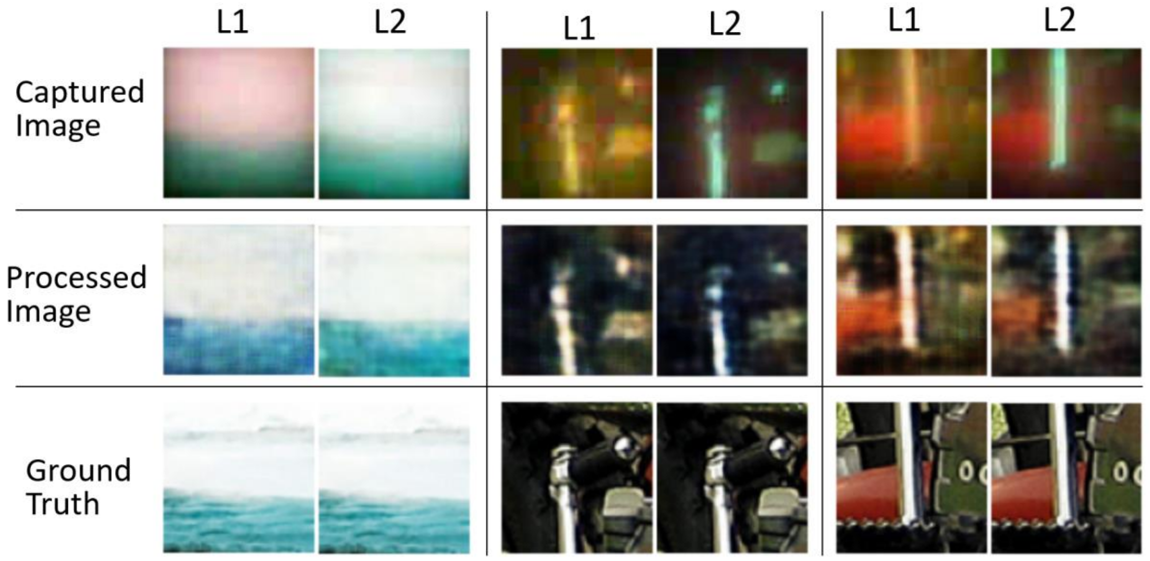

Figure 3.3 A comparison between L1 and L2

Next, with our increased dataset size, we compared our network pipeline with and without MATLAB image preprocessing. For the preprocessed images, the SSIM score was 0.4290 while the PSNR score was 16.43. On the other hand, the unprocessed images had an SSIM score of 0.4647 and a PSNR score of 18.09. This indicated to us that the preprocessing was likely taking away some of the valuable information that the network was using to map our MDL images to the ground truth. Figure 3.4 shows a comparison between a few preprocessed and unprocessed images, and the middle image shows a major difference between the two, as the shingles of the MATLAB preprocessed roof seem more blurred together than those in the unprocessed image when we compare the results of the network pipeline. However, the main differences between the two sets of images appears to be a shift in hue, although it doesn't appear that this shift in the preprocessed images really made the network output more similar to the ground truth.

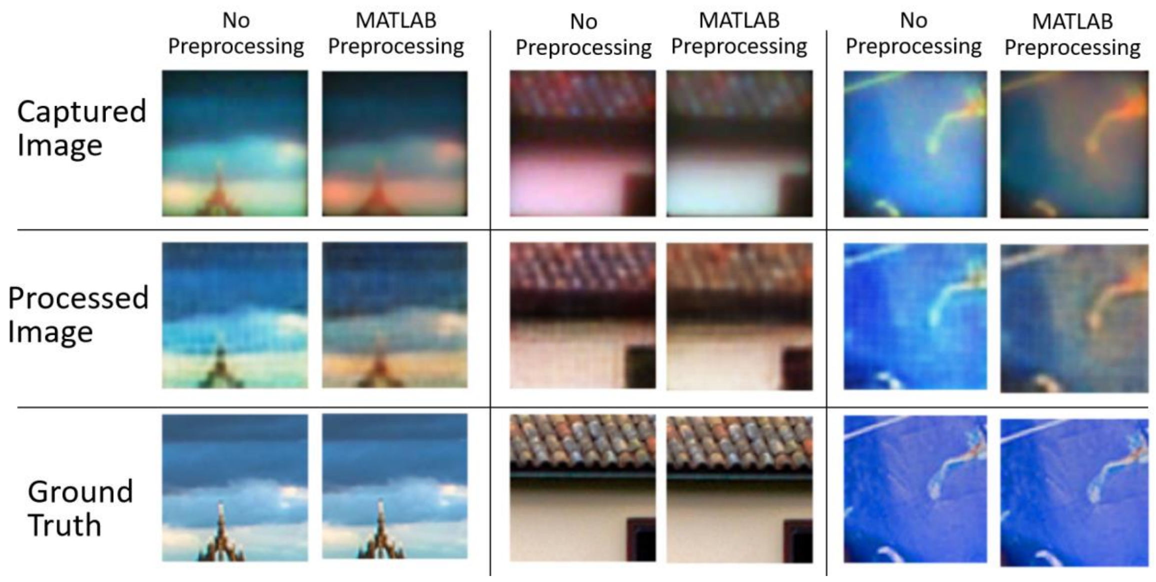

Figure 3.4 A comparison between images with and without MATLAB preprocessing

Finally, the results from our tests regarding the different pipeline configurations are shown in Table 3.2 and a comparison of the various reconstructions of an image are shown in Figure 3.5. Our results a show that none of the DRRN configurations provided any benefit to the mean SSIM or PSNR scores over the GAN itself. Additionally, none of the reconstructions in Figure 3.5 look significantly improved or even different from the GAN reconstruction. Finally, there was no significant difference between the patch-wise and image-wise DRRN reconstructions, although this is not particularly surprising given that none of the DRRN configurations significantly altered their input image.

Pipeline	DRRN Loss Function	SSIM	PSNR
Captured Images	N/A	0.38498	16.1958
GAN	N/A	0.47863	18.2106
GAN + Patch-wise DRRN	MAE	0.44152	17.9516
GAN + Patch-wise DRRN	MSE	0.43522	17.6346
GAN + Patch-wise DRRN	Cross Channel	0.44823	17.7244
GAN + Patch-wise DRRN	SSIM	0.47336	17.6189
GAN + Image-wise DRRN	MAE	0.4647	18.0881
GAN + Image-wise DRRN	MSE	0.44466	17.7889
GAN + Image-wise DRRN	Cross Channel	0.45113	17.8148
GAN + Image-wise DRRN	SSIM	0.46658	18.2095
GAN + Image-wise DRRN with 16 units	MAE	0.45077	17.8131
GAN + Image-wise DRRN with 3 blocks	MAE	0.44623	18.0693
GAN + Image-wise DRRN with 3 blocks, 16 units	MAE	0.44696	17.9866

Table 3.2 The mean PSNR and SSIM scores for each variation of the pipeline

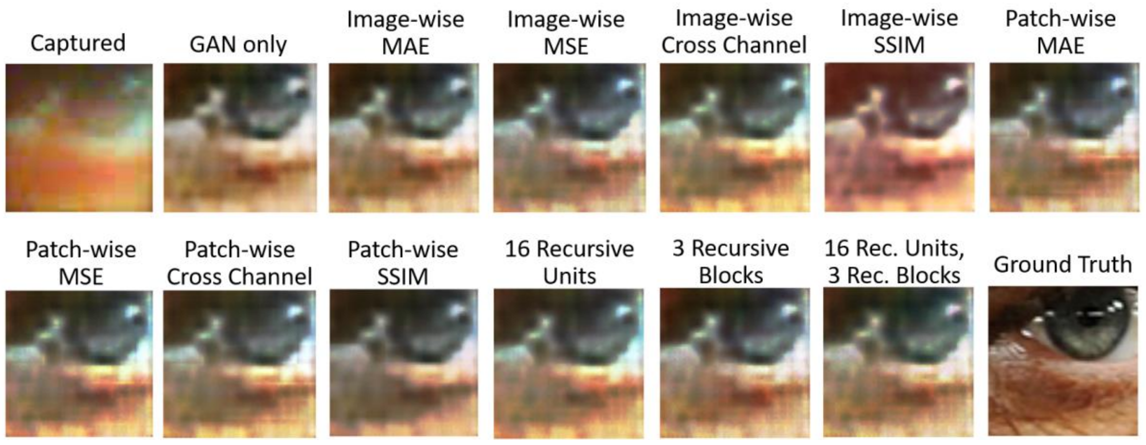

Figure 3.5 A comparison of the different pipeline reconstructions. The descriptions of the images refer to the DRRN that was used in the pipeline.

In pursuance of our goal to determine a pipeline for general MDL use, we attempted to reconstruct various experimental images taken with L2. These images were not captured off of a display like the training images were, so we would expect a slightly worse performance by the pipeline when reconstructing them. An example of two of these reconstructions are shown in Figure 3.6. Unfortunately, our pipeline appears to perform poorly when reconstructing experimental images, as it simply serves to distort the images further.

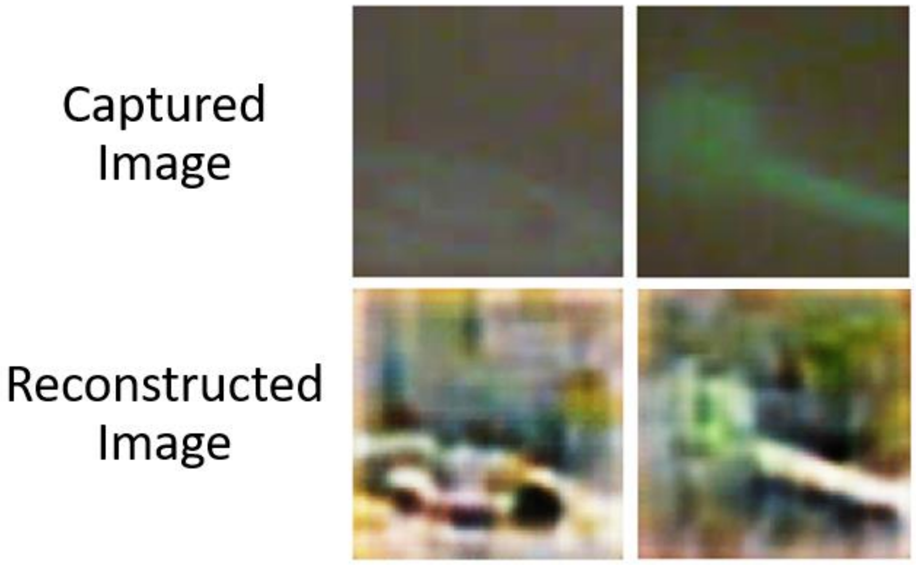

Figure 3.6. Example of experimental images (not captured from display) and their reconstructions. A black and white circular logo is pictured on the left, and a headphone is featured on the right.

3.5 Conclusion

We studied an end-to-end neural network pipeline with numerous variations and found that, despite marginal success in reconstructing testing images, such a network performs poorly on our experimental data. Our tests also indicate that the DRRN does not significantly impact the results of the output, and in most cases the DRRN actually lowered both the SSIM and PSNR scores of the original GAN output. Similarly, training the DRRN patch-wise did not significantly affect the performance when compared to image-wise training.

This suggests that for such a network to be successful, it may need to be trained on experimental images, capturing the same scenes with both a traditional imaging system as well as an MDL. Alternatively, there may be other modifications that can be made to the network to ensure that, despite being captured from a display, the image translations will still translate well to more general images.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks,” Communications of the ACM, vol. 60, no. 6, pp. 84–90, 2017.

[2] S. Mahendran, H. Ali, and R. Vidal, “3d pose regression using convolutional neural networks,” in The IEEE International Conference on Computer Vision (ICCV) Workshops, Oct 2017.

[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1125–1134.

[4] W. Yao, Z. Zeng, C. Lian, and H. Tang, “Pixel-wise regression using UNET and its application on pansharpening,” Neurocomputing, vol. 312, pp. 364–371, 2018.

[5] M. Halicek, G. Lu, J. V. Little, X. Wang, M. Patel, C. C. Griffith, M. W.El-Deiry, A. Y. Chen, and B. Fei, “Deep convolutional neural networks for classifying head and neck cancer using hyperspectral imaging,” Journal of Biomedical Optics, vol. 22, no. 6, pp. 060 503–060 503, 2017.

[6] H. Kanayama, T. Ma, S. Tsuchikawa, and T. Inagaki, “Cognitive spectroscopy for wood species identification: near infrared hyperspectral imaging combined with convolutional neural networks,” Analyst, vol.144, pp. 6438–6446, 2019. [Online]. Available: http://dx.doi.org/10.1039/C9AN01180C

[7] David E. Rumelhart; James L. McClelland, "Learning Internal Representations by Error Propagation," in Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations, MITP, 1987, pp.318-362

[8] O. Ronneberger, P. Fischer, and T. Brox, “UNET: Convolutional networks for biomedical image segmentation,” in International Conference on Medical image computing and computer-assisted intervention. Springer,2015, pp. 234–241.

[9] M. S. Alam and P. Sidike, “Trends in oil spill detection via hyperspectral imaging,” in2012 7th International Conference on Electrical and Computer Engineering. IEEE, 2012, pp. 858–862.

[10] A. A. Gowen, Y. Feng, E. Gaston, and V. Valdramidis, “Recent applications of hyperspectral imaging in microbiology,” Talanta, vol. 137, pp.43–54, 2015.

[11] R. Arablouei and F. de Hoog, “Hyperspectral image recovery via hybrid regularization,” Ieee Transactions On Image Processing, vol. 26, no.12, pp. 5649–5663, 2016.

[12] Deep Learning Applications for Hyperspectral Imaging: A Systematic Review - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Hyperspectral-and-RGB-Image-Components-18_fig1_339580294 [accessed 5 Apr, 2021]

[13] G. Kim, K. Isaacson, R. Palmer, and R. Menon, “Lensless photography with only an image sensor,” Appl. Opt., vol. 56, no. 23, pp. 6450–6456, Aug 2017. [Online]. Available: http://ao.osa.org/abstract.cfm?URI=ao-56-23-6450

[14] P. Wang and R. Menon, “Computational multispectral video imaging[invited],” J. Opt. Soc. Am. A, vol. 35, no. 1, pp. 189–199, Jan 2018. [Online]. Available: http://josaa.osa.org/abstract.cfm?URI=josaa-35-1-189

[15] Z. Xiong, Z. Shi, H. Li, L. Wang, D. Liu and F. Wu, "HSCNN: CNN-Based Hyperspectral Image Recovery from Spectrally Undersampled Projections," *2017*

IEEE International Conference on Computer Vision Workshops (ICCVW), Venice, Italy, 2017, pp. 518-525, doi: 10.1109/ICCVW.2017.68.

[16] Z. Shi, C. Chen, Z. Xiong, D. Liu and F. Wu, "HSCNN+: Advanced CNN-based hyperspectral recovery from RGB images", *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, pp. 939-947, Jun. 2018.

[17] K. G. Lore, K. K. Reddy, M. Giering and E. A. Bernal, "Generative adversarial networks for spectral super-resolution and bidirectional RGB-To-Multispectral mapping", *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, pp. 926-933, Jun. 2019.

[18] X. Liu, A. Gherbi, Z. Wei, W. Li and M. Cheriet, "Multispectral Image Reconstruction From Color Images Using Enhanced Variational Autoencoder and Generative Adversarial Network," in *IEEE Access*, vol. 9, pp. 1666-1679, 2021, doi: 10.1109/ACCESS.2020.3047074.

[19] H. Aksoy and A. Dahamsheh, "Markov chain-incorporated and synthetic data-supported conditional artificial neural network models for forecasting monthly precipitation in arid regions," Journal of Hydrology, vol.562, pp. 758–779, 2018.

[20] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil,T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2018.

[21] Hyperspectral Images Database. [Online]. Available: https://sites.google.com/site/hyperspectralcolorimaging/dataset

[22] Oktay O., Schlemper J., Le Folgoc L., Lee M.C.H., Heinrich M., Misawa K., Mori K., McDonagh S., Hammerla N.Y., Kainz B., et al. Attention UNET: Learning Where to Look for the Pancreas 2018.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2016-. IEEE, 2016, pp. 770–778.

[24] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," arXiv preprint arXiv:1505.00387, 2015.

[25] Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861.

[26] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W.Hsieh, "Cspnet: A new backbone that can enhance learning capability of cnn," ArXiv, vol. abs/1911.11929, 2019.

[27] Kunihiko Fukushima (2007) Neocognitron. Scholarpedia, 2(1):1717.

[28] Sourangsu Banerji, Monjurul Meem, Apratim Majumder, Fernando Guevara Vasquez, Berardi Sensale-Rodriguez, and Rajesh Menon, "Imaging with flat optics: metalenses or diffractive lenses?," Optica 6, 805-810 (2019)

[29] A. Nikonorov *et al*., "Deep Learning-Based Imaging using Single-Lens and Multi-Aperture Diffractive Optical Systems," *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW),* Seoul, Korea (South), 2019, pp. 3969-3977, doi: 10.1109/ICCVW.2019.00491.

[30] Banerji, S., Cooke, J. & Sensale-Rodriguez, B. Impact of fabrication errors and refractive index on multilevel diffractive lens performance. *Sci Rep* **10,** 14608 (2020). https://doi.org/10.1038/s41598-020-71480-2

[31] Dennis W. Prather, David Pustai, and Shouyuan Shi, "Performance of multilevel diffractive lenses as a function of f-number," Appl. Opt. 40, 207-210 (2001)

[32] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Scholkopf. ¨ Non-stationary correction of optical aberrations. In Computer Vision (ICCV), 2011 IEEE International Conference on, pages 659–666. IEEE, 2011.

[33] F. Heide, M. Rouf, M. B. Hullin, B. Labitzke, W. Heidrich, and A. Kolb. High-quality computational imaging through simple lenses. ACM Transactions on Graphics (TOG), 32(5):149, 2013.

[34] Schuler C.J., Hirsch M., Harmeling S., Schölkopf B. (2012) Blind Correction of Optical Aberrations. In: Fitzgibbon A., Lazebnik S., Perona P., Sato Y., Schmid C. (eds) Computer Vision – ECCV 2012. ECCV 2012. Lecture Notes in Computer Science, vol 7574. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-33712-3_14

[35] T. Yue, J. Suo, J. Wang, X. Cao, and Q. Dai. Blind optical aberration correction by exploring geometric and visual priors. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[36] C. Dong, C. C. Loy, K. He, and X. Tang. Image superresolution using deep convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(2):295–307, 2015

[37] J. Kim, J. K. Lee and K. M. Lee, "Accurate Image Super-Resolution Using Very

Deep Convolutional Networks," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 1646-1654, doi: 10.1109/CVPR.2016.182.

[38] J. Kim, J. K. Lee and K. M. Lee, "Deeply-Recursive Convolutional Network for Image Super-Resolution," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 1637-1645, doi: 10.1109/CVPR.2016.181.

[39] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2017):3147-3155, 2017.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in International Conference on Learning Representations (ICLR), 2015.

[41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, and A. Rabinovich, "Going deeper with convolutions," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[42] Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron C., and Bengio, Yoshua. Generative adversarial nets. NIPS, 2014.

[43] M. Mirza and S. Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784, 2014

[44] D. P. Kingma, and J. Ba. Adam: A method for stochastic optimization. arXiv, arXiv:1412.6980, 2014

[45] McKenna, Curt, Kevin Walsh, Mark Crain, and Joseph Lake. "Maskless Direct Write Grayscale Lithography for MEMS Applications." In Micro/Nano Symposium (UGIM), 2010 18th Biennial University/Government/Industry, pp. 1-4. IEEE, 2010.

[46] Monjurul Meem, Apratim Majumder, and Rajesh Menon, "Free-form broadband flat lenses for visible imaging," OSA Continuum 4, 491-497 (2021)

[47] Diffractive Images Database. [Online]. Available: https://onedrive.live.com/?authkey=%21AI8JGURqxd%5F3Zw4&id=DE021C4AD0700624%214784&cid=DE021C4AD0700624