2022

Machine learning techniques for reconstruction and segmentation of nanoparticle interferometric signatures

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Thesis

MACHINE LEARNING TECHNIQUES FOR RECONSTRUCTION AND SEGMENTATION OF NANOPARTICLE INTERFEROMETRIC SIGNATURES

by

SHASHANK MANJUNATH

B.Eng., Vanderbilt University, 2018

Submitted in partial fulfillment of the

requirements for the degree of

Master of Science

2022

Approved by

First Reader

M. Selim Ünlü, Ph.D.
Distinguished Professor of Electrical and Computer Engineering
Distinguished Professor of Materials Science and Engineering
Distinguished Professor of Biomedical Engineering

Second Reader

Janusz Konrad, Ph.D.
Professor of Electrical and Computer Engineering

Third Reader

Lei Tian, Ph.D.
Assistant Professor of Electrical and Computer Engineering
Assistant Professor of Biomedical Engineering

Acknowledgments

I would like to thank Professor Selim Ünlü for the opportunity to work with him and his lab on this thesis. Completing this thesis would not have been possible without his support, belief, and expertise. I would additionally like to thank Mete Aslan and Iris Celebi for their helpful discussions and assistance with data collection. Their expertise was invaluable to this project, and this could not have been completed without them. I would furthermore like to thank Professor Konrad and Professor Tian for serving on this thesis committee. Lastly, I would like to thank my parents, my sister, and Lindsay Grizzard for their never-ending support.

MACHINE LEARNING TECHNIQUES FOR RECONSTRUCTION AND SEGMENTATION OF NANOPARTICLE INTERFEROMETRIC SIGNATURES

SHASHANK MANJUNATH

ABSTRACT

Single particle interferometric reflectance imaging sensor (SP-IRIS) allows for label-free biological nanoparticle detection. This imaging technique allows collection of a 3D defocus profile of interferometric signatures of particles on a reflecting surface. However, automated reconstruction of information from SP-IRIS data and identification of particles remains a challenge. In this work, we develop machine learning techniques to perform accurate reconstruction of defocus profiles from undersampled SP-IRIS data, and machine learning techniques to segment particles of interest from background in SP-IRIS data. We test the performance of a direct, single-signal fully connected neural network based reconstruction technique, an untrained non-convolutional network for single-signal reconstruction, and a 2D convolutional neural network reconstruction model for reconstruction of SP-IRIS signal defocus profiles from undersampled data. We then test a UNet based segmentation model to segment particle signals from background signals. Lastly, we propose a novel combined reconstruction and segmentation model which can perform both reconstruction and segmentation on undersampled SP-IRIS data.

Contents

List of Tables

List of Figures

List of Abbreviations

CNN	.....	Convolutional Neural Network
DC	.....	Dice Coefficient
DIP	.....	Deep Image Prior
LPIPS	.....	Learned Perceptual Image Patch Similarity
ML	.....	Machine Learning
MLP	.....	Multilayer Perceptron
MSE	.....	Mean Squared Error
PCA	.....	Principle Component Analysis
PSF	.....	Point Spread Function
ReLU	.....	Rectifying Linear Unit
SNR	.....	Signal-to-Noise Ratio
SP-IRIS	.....	Single Particle Interferometric Reflectance Imaging Sensor
$\mathbb{R}^n$	.....	n-dimensional Real vector space

Chapter 1

Introduction

1.1 Interferometric Imaging

Interferometric imaging allows highly sensitive label-free detection of small biological nanoparticles, assisting in rapid identification of infectious diseases (Avci et al., 2015). However, reconstructing interferometric images with sufficient signal to noise ratio has proven a significant challenge (Yurdakul et al., 2020). Many label-free optical imaging techniques characterize particles based on weak optical scattering from the imaged particles, leading to difficulties distinguishing particles from background imagery. Therefore, there is a significant need to improve the sensitivity of interferometric imaging methods. The essential challenge of this problem is the low signal of data acquired through interferometric imaging. This type of imaging technique produces a 3D defocus intensity stack, with each image individually having low SNR. Each image is taken at a different spatial (z) position, which produces a set of both x and y axis spatial intensities, as well as z axis spatial intensities. We aim to leverage these images to create high-SNR 2D images of the target particle. However, many existing techniques are computationally expensive and require slow, iterative reconstruction algorithms (Yurdakul et al., 2020). Other existing reconstruction techniques require a large number of z position measurements, leading to extended imaging times (Zaraee et al., 2020). An overview of this imaging technique is given in Figure 1·1. Machine learning techniques provide a promising avenue in which to construct performant algorithms for image reconstruction and particle segmentation.

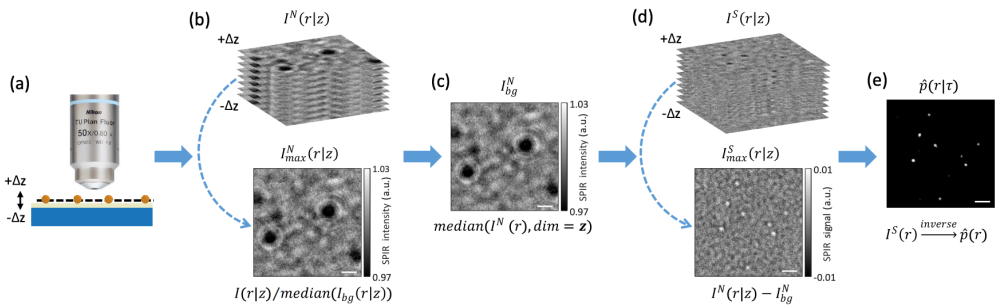


Figure 1·1: Flow chart of computational nanosensing using defocus curves in SP-IRIS (Yurdakul et al., 2020). Image (a) indicates the defocus scan, (b) indicates the defocus stack normalization step, (c) indicates a maximum contrast image, (d) indicates subtracting background from the normalized stack, and (e) indicates the reconstructed single image. We aim to improve steps (d) to (e), and consequently steps (a) to (b), by improving image reconstruction and decreasing the number of z-position images that need to be taken

1.2 Machine Learning

The core of this thesis is development of novel Machine Learning (ML) techniques to improve interferometric image reconstruction and segmentation. We pursue this goal under a supervised learning paradigm, e.g., ground truth high-SNR images will be available to train given machine learning techniques. Supervised learning techniques have provided significant improvements in many classification and regression problems. One notable improvements over the past decade enabled by ML is significant performance gains on the ImageNet dataset (Russakovsky et al., 2015). More relevant to the interferometric image reconstruction task, deep learning techniques have shown extreme promise in many medical imaging reconstruction tasks (Yedder et al., 2020). For our task, as fewer and fewer images are taken in the defocus axis (z axis), we will attempt to reconstruct the image with maximal SNR as compared to the fully-sampled z axis case. We propose to obtain ground truth by collecting a large set of defocus images in a large number of frames, then calculate a single ground truth 2D image. The proposed ground truth data will be shot-noise-limited.

As we subsample the z axis, noise will be introduced from the missing stack images, leading to progressively poorer reconstructions. This subsampled data will be the input to the machine learning algorithm.

In addition to regression alone, we aim to improve downstream particle segmentation algorithms. Deep learning algorithms, in addition to improvements on regression and classification techniques, provide state of the art performance on segmentation problems (Minaee et al., 2020). Existing interferometric image particle segmentation algorithms have relied on traditional computer vision techniques, such as circle detection and peak finding algorithms. However, deep learning techniques provide potential solutions for accurate segmentation models which can robustly detect particles from interferometric images.

1.3 Approach

As discussed previously, in this work we focus on deep learning techniques applied to both interferometric signal reconstruction and segmentation. In this thesis, we investigate existing deep learning techniques developed for signal reconstruction, and apply them to interferometric signal reconstruction techniques. Then, we make a similar investigation for segmentation methods. Lastly, we investigate combining reconstruction and segmentation methods into a single model, resulting in a robust, end-to-end image reconstruction and segmentation system applied to interferometric imaging. Our original data contains 21 z-position samples on a 256x256 grid, creating a 256x256x21 array of data. We input subsampled data to our models, with less than 21 z-position samples included. Typical undersampling levels are 15 samples, 10 samples, and 5 samples. For our reconstruction tasks, we attempt to reconstruct the fully sampled 21 z-position signal for each spatial location (i.e., each 256x256 spatial pixel). For the segmentation task, we attempt to input information from the full 21 z-position sampled data and output a 1-channel segmentation

map of our SP-IRIS data which indicates the position of particles in the image.

Chapter 2

Background and Related Work

2.1 Interferometric Imaging

Single particle interferometric reflectance imaging sensor (SP-IRIS) allows for label-free biological nanoparticle detection. This imaging technique allows collection of a 3D defocus signature of biological nanoparticles particles (Avci et al., 2015). However, automated reconstruction of information from SP-IRIS data and identification of particles remains a challenge. In this work, we develop machine learning techniques to perform accurate reconstruction of defocus profiles from undersampled SP-IRIS data, and machine learning techniques to segment particles of interest from background in SP-IRIS data.

2.2 Machine Learning

Machine learning (ML) is concerned with the development of algorithms which leverage data to create models which can perform a particular task. Machine learning has found many applications, including email spam filtering, image processing, and dataset dimensionality reduction, among others tasks. While all of these problems fall under the umbrella of machine learning, solutions to these problems can follow different underlying paradigms. These paradigms are supervised and unsupervised learning. Unsupervised learning is concerned with learning information from untagged datasets. These types of algorithms are typically interested in discovering dataset structure or organization, such as clustering algorithms. Algorithms can be fed unstructured data, with no label provided,

and the algorithm is expected to learn about the underlying structure (Hastie et al., 2009). Dimensionality reduction performed by an algorithm such as Principal Component Analysis (PCA) is an example of unsupervised learning (Bishop, 2006). When operating, the PCA algorithm attempts to find a change of variables that can approximate higher dimensional data, all without the use of any labels on the data. Supervised learning, however, is concerned with learning from labeled data.

2.2.1 Supervised learning

Supservised learning is concerned with learning functions that map input data (called *features*), to output labels, based on given input and output pairs (Goodfellow et al., 2016). The key difference between supervised and unsupervised learning is the presence of a labeled output measurement. This measurement can be either a categorical value (such as a binary 1/0 outcome, for example), or an analog measurement (such as an output voltage or signal intensity). In supervised learning, for the classes of algorithms discussed in this thesis, we assume that our dataset is drawn independently and identically distributed (i.i.d) from an underlying data distribution. From our collected dataset, we create two sets of data, the *training* set, and the *testing* set. This provides us a dataset on which to train our algorithm, as well as a dataset of data which we can use to assess our algorithm performance. Note that we do not use any aspect of the testing set to inform the training operation. The testing set is used purely as an assessment of the algorithm's potential performance on unseen data.

2.2.2 Classification and Regression Problems

In this particular application, we are interested in both classification and regression problems. Classification problems are concerned with identifying categorical variables. Example classification problems are email spam detection (classifying whether a given email is spam or not), and identification of handwritten digits (classifying the digit present in an image, which can be any integer value between 0 and 9). Regression problems, on the other

hand, are concerned with prediction of an analog value, such as a voltage or signal amplitude. These problems have fundamental differences in their outcomes. Several classes of traditional machine learning algorithms exist which are tailored to each particular problem type. A commonly used regression algorithm is linear regression, in which we attempt to find a linear model which best fits our data (Hoerl and Kennard, 1970). Similarly, a commonly used classification algorithm is the support vector machine, which attempts to find a hyperplane which can divide data into two classes (Hastie et al., 2009). However, recently deep learning techniques have provided best performance on many classification and regression problems.

2.2.3 Deep Learning Techniques

Deep learning is concerned with the study of multilayer artificial neural networks. Deep neural networks use multilayered artificial neural network layers to extract features from underlying data, in a process called *representation learning*. In particular, a large part of developing performant traditional machine learning algorithms is finding the correct features on which to train an algorithm; choosing poor features can lead to a poor algorithm. SVMs, for example, leverage kernels in order to extract better features from datasets, which can then produce more performant SVM models (Duvenaud, 2014). Deep learning attempts to combine both extraction of features and mapping these features to the desired outcome variable (Goodfellow et al., 2016).

More specifically, the most basic type of deep neural network is the *multilayer perceptron (MLP)*, also known as the fully connected neural network (Goodfellow et al., 2016). These networks consist of multiple fully connected layers of neurons. After each layer, we use a nonlinear activation function. This allows our neural network to be nonlinear. Without the activation function, our model would reduce to a group of matrix multiplications, which is a linear model. In all models used in this thesis, we use the Rectifying Linear Unit (ReLU) activation function (Nwankpa et al., 2018).

$$\text{ReLU}(x) = \max(0, x)$$

where $x \in \mathbb{R}$. We can specify a single fully connected network layer in equations as:

$$f(\mathbf{X}, \mathbf{w}, \mathbf{b}) = \text{ReLU}(\mathbf{X}\mathbf{w} + \mathbf{b})$$

where $\mathbf{X} \in \mathbb{R}^{b \times n}$ is the input data, $\mathbf{w} \in \mathbb{R}^n$ indicates our network weights, and $\mathbf{b} \in \mathbb{R}^b$ is the bias term for our network. We can stack multiple of these layers together to create a deep neural network. At the output layer of our network, we typically do not use a ReLU activation function. For our regression problems, we use a linear activation (i.e., no activation function) on our output layer. For our segmentation problems, we use a sigmoid activation function, given by the following equation:

$$\text{Sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$

for $x \in \mathbb{R}$.

2.2.4 Loss Functions

Now that we have a way to calculate the output of a deep learning algorithm, we need a way to assess algorithm performance. In particular, as we will discuss in 2.2.5, this measurement must be subdifferentiable. For regression, a common loss function is *Mean Squared Error (MSE)*. In particular, given a set of n output variables $y \in \mathbb{R}^n$ and predicted outputs $\hat{y} \in \mathbb{R}^n$, we calculate MSE with the following equation:

$$MSE(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

This allows us to quantify the performance of a regression model. Similarly, for a binary classification task such as the one attempted in this work, we use the *log loss* (also called

binary cross entropy). In particular, given a probability output (i.e., network prediction $\hat{y} \in [0,1]$) and a true label $y \in \{0,1\}$, we calculate the following loss function:

$$\mathcal{L}(x,y) = \frac{1}{n}\sum_{i=1}^{n}[y_i \cdot \log(\hat{y}_i) + (1-y_i) \cdot \log(1-\hat{y}_i)]$$

Typically, the Sigmoid activation function as discussed in 2.2.3 is used to ensure that a given output is a probability value in the range $[0,1]$.

Lastly, in addition to the loss functions given above, we leverage the Learned Perceptual Image Patch Similarity (LPIPS) loss function as part of our regression scheme (Zhang et al., 2018). LPIPS relies on a frozen VGG network trained on the ImageNet dataset to produce a feature vector which can be used to compare the similarity between two images. This technique has been used as part of the loss function in other deep learning regression problems (Kupyn et al., 2019). Here, we use it to enforce that our network outputs standard deviation reconstructions which are reasonably visually similar to the standard deviation reconstruction outputs of the fully sampled signals. LPIPS provides significant advantages over standard L2-based distance calculations, as it is robust to small spatial perturbations (local warping), blurring, and ghosting, which can occur during signal reconstruction.

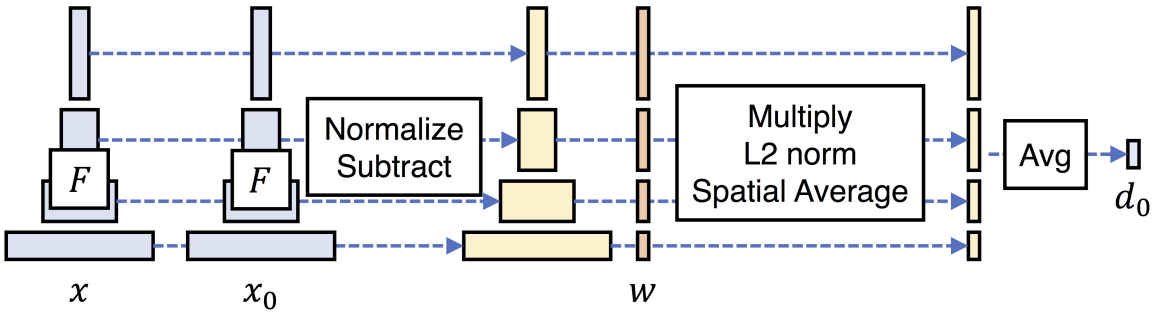


Figure 2·1: Diagram of LPIPS distance computation (Zhang et al., 2018). Given two images x, x_0 and network $\mathcal{F}$, we first compute embeddings, then normalize activations in the channel dimension, then scale each channel by vector w, and calculate the L2 distance between the activations. We then average across the spatial dimension and across all layers.

2.2.5 Backpropagation and Optimization

Once we are able to measure the perfomance of our model using a loss function, we must use that loss function to improve our model weights in order to achieve improved performance. To do this, we use an algorithm called backpropagation (Goodfellow et al., 2016). Backpropagation operates by first calculating the gradient of the loss function with respect to the weights of the model. An optimization algorithm then updates the weights of the model based on these gradients. After many steps of forward propagation of training data, backpropagation of gradients, and optimization, a model can be considered "trained." For our models, we use the Adam optimizer (Kingma and Ba, 2017).

2.2.6 Convolutional Neural Networks

While fully connected neural networks provide a useful way to model many types of data, they are poorly suited to image data. For example, a simple 28×28 image of a handwritten digit as found in the MNIST data set exists in $\mathbb{R}^{784}$ (Lecun et al., 1998). This problem is exacerbated in larger image sizes. Therefore, instead of using individual nodes in a network, each of which corresponds to a single pixel in the input image, we can train convolution kernels (O'Shea and Nash, 2015). These convolution kernels each individually extract different features from our image. Once we have enough convolution layers, we can aggregate the information included in the convolutional layer activations and make predictions such as whether a particle is present in an individual pixel in the case of a classification problem, or a prediction of the signal intensity at a particular z-axis position in the case of a regression problem. In this work, we frequently leverage the UNet architecture (Ronneberger et al., 2015). We show an example 2D UNet architecture in Figure 2·2 (Ronneberger et al., 2015).

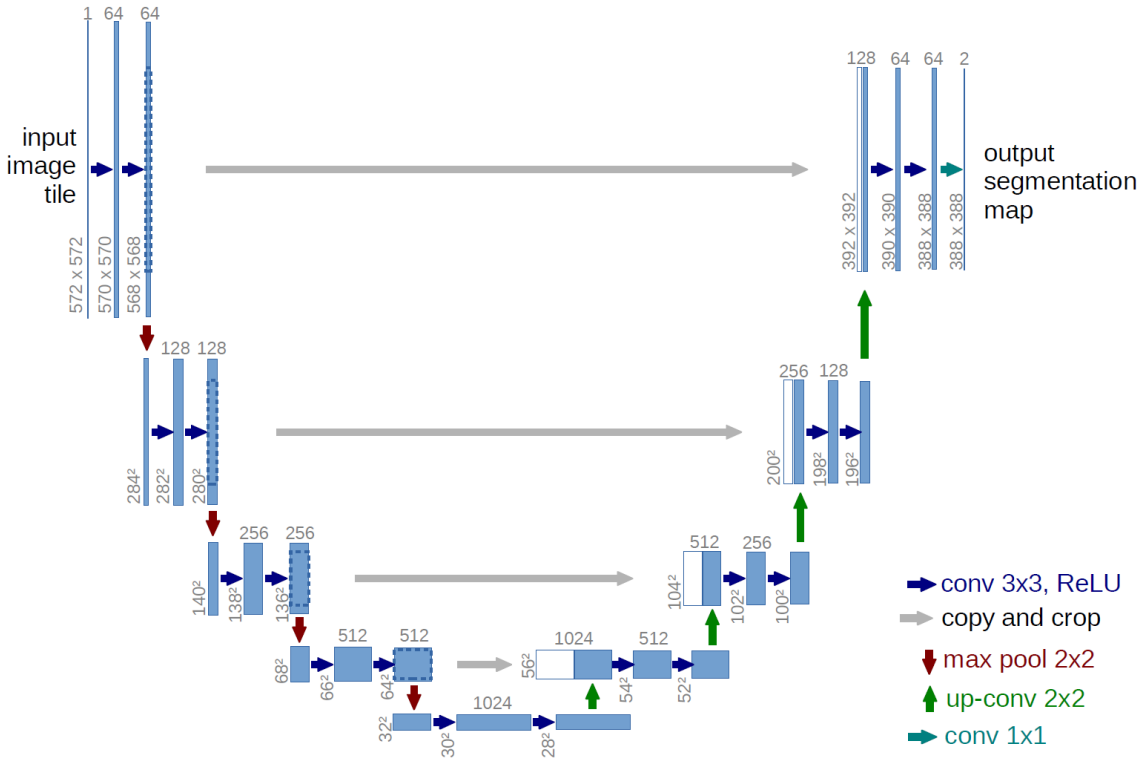


Figure 2·2: General UNet Architecture

2.2.7 Accuracy Metrics

Once we have made predictions using our machine learning model, we must measure their performance. For our regression problems, we use MSE as a measure of performance, as presented in Section 2.2.4. For our classification problems, where we are predicting whether individual pixels contain particle signals or background signals, we use three different metrics. First, we use Dice coefficient to measure the per-pixel accuracy of our models. Dice coefficient is calculated using the following equation:

$$DC(y,\hat{y}) = \frac{2|y \cap \hat{y}|}{|y| + |\hat{y}|}$$

where y is the true segmentation map and $\hat{y}$ is the predicted segmentation map. The operator $|\cdot|$ indicates the cardinality of the segmentation map, which is equivalent to the

area of all particle locations combined.

In addition to measuring the per-pixel accuracy of our model, we want to measure if it accurately identifies individual particles, or if it frequently misses particles. To measure this, we leverage precision and recall. Precision indicates how many of the identified particles are actually particles, while recall indicates how many of the total number of particles the model identified. These metrics are calculated over different particles, rather than over pixels. Precision and recall are calculated with the following functions:

$$\text{Precision} = \frac{TP}{TP+FP}$$
$$\text{Recall} = \frac{TP}{TP+FN}$$

where TP indicates true positives, FP indicates false positives, and FN indicates false negatives. We use a threshold of 0.5 on all network outputs after sigmoid activation to distinguish positive predictions from negative predictions.

2.2.8 Implementation Details

We implement our models in Python using PyTorch 1.10, CUDA 11.3, and CUDNN 8.2 (Paszke et al., 2019). We train and test our models on a server with an i7-2600k CPU, 16GB RAM, and a Quadro RTX 4000 GPU.

Chapter 3

Methods

3.1 Data Collection

We image 80nm gold nanoparticles under a 20× 0.45NA lens. We use an illumination wavelength of 460 nm. The particles are immobilized in water on an SiO2 (60 nm) coated Si chip. When imaging the particles, the axial distance between each step is 1 μm. Two separate data collections were performed, one for the training set and one for the testing set. The training set consists of 31 256×256 image chips, while the test set consists of 39 256×256 image chips. These image chips were then labelled with segmentation maps indicating the presence of particles by hand.

3.2 Signal Reconstruction Model

We first discuss three different ML models that we evaluated on the signal reconstruction task.

3.2.1 Direct Reconstruction

Our first attempt at creating a signal reconstruction model works on a per-pixel basis. We treat each individual pixel on our XY plane as a single signal, and optimize our model to interpolate per-pixel on the XY plane. This yields a model that can accurately interpolate defocus profiles on a per-pixel basis. However, the model is limited in it's visibility of surrounding pixels. Since the point spread function (PSF) of our system is larger than the individual object we are trying to image, some of the object's signal bleeds to surrounding

pixels in the XY plane. However, our single signal reconstruction model has no view of any surrounding pixels; it only accounts for signal intensities in a single XY plane. Therefore, our reconstructions are noisy in the XY-plane, and not realistic to the end viewer. This high-frequency noise leads to issues in downstream image processing algorithms which attempt to identify virus particles from background particles. We call this "Direct Reconstruction." We use a fully connected model with 6 hidden layers, and apply it to each individual signal in our dataset. The fully connected model operates across the z-axis. Each input signal is an n-element vector, where n indicates the sampling level. The output signal is a 21-element vector, since our original data contains 21 z-position samples. We present hyperparameters used for training in Table 3.1.

Table 3.1: Hyperparameters for Direct Reconstruction Model

Hypereparameter	**Value**
Optimizer	Adam
Learning Rate	0.001
Epochs	50

3.2.2 Deep Decoder Reconstruction

The next attempt at creating a signal reconstruction model derives from work done by Heckel et. al. on signal denoising (Heckel and Hand, 2018). This type of model relies on the overparametrization of neural networks, similar to work done by Ulyanov et. al. in the deep image prior (DIP) algorithm (Ulyanov et al., 2020). This technnique relies on the fact that a randomly initialized neural network can be used as a prior for many inverse problems. The core idea is to optimize a network that maps from a randomly initialized input vector to a fully sampled output, then calculate a loss against the known undersampled signal as described in Equation 3.1. In this type of model, we initialize a neural network with random weights. In particular, we optimize the following function:

$$\mathcal{L}(\mathbf{C}) = \|f(G(\mathbf{C})) - \mathbf{x}^\star\|_2^2 \tag{3.1}$$

where G represents our neural network, f represents our undersampling function ,and $\mathbf{C}$ represents our neural network parameters. Our optimized network parameters, $\hat{\mathbf{C}}$, estimates our output image as $\hat{\mathbf{x}} = G(\hat{\mathbf{C}})$. Since we can determine our own undersampling function f, this procedure fits our problem well.

While this technique performs decently, it suffers the same issues as discussed in Section 3.2.1: it does not account for 2D correlations of signals across the XY plane, and furthermore suffers from slow inference times. Since the model requires 128 steps of optimization at each pixel value, inference can take many hours, rendering the model not useful for use on actual interferometric imaging devices. Due to runtime considerations, reconstruction results are averaged over a single 256×256 image chip for the Deep Decoder algorithm. We provide hyperparameters used for training in Table 3.2

Table 3.2: Hyperparameters for Deep Decoder Model

Hyperparameter	**Value**
Optimizer	Adam
Learning Rate	0.001
Epochs	128

3.2.3 UNet Reconstruction

To tackle the issues discussed in Section 3.2.1 and 3.2.2, we propose reconstruction using a convolutional neural network, specifically a UNet based model. For this type of model, we break our image into 256×256 image chips, and train our model to upsample from a $256 \times 256 \times n_1$ to $256 \times 256 \times n_2$, with $n_1 < n_2$. For our models, n_1 is equal to 15, 10, or 5, depending on the undersampling level, and n_2 is equal to 21. Once each individual image chip has been reconstructed, we can then stitch back together the image chips to get a full

image. This technique provides two advantages over the previously discussed techniques. Primarily, using 2D convolutions provides a way to associate data across the XY plane that the techniques previously discussed do not provide. Furthermore, the weight sharing leveraged by UNet, and Convolutional Neural Networks (CNNs) in general, provides significant speedups in reconstruction. We provide hyperparameters used for training in Table 3.3.

Table 3.3: Hyperparameters for UNet Reconstruction Model

Hyperparameter	**Value**
Optimizer	Adam
Learning Rate	0.001
Epochs	64

3.3 Segmentation Model

In addition to reconstruction models, we explore machine learning based techniques which can provide per-pixel segmentation of target particles from reconstructed data. This task inherently requires aggregation of features across the XY plane, as the signal from a single particle can bleed across pixel intensities. Therefore, a convolutional neural network is best suited to this task. We leverage a UNet based technique similar to that discussed in Section 3.2.3. Similar to the reconstruction approach, we use 256×256 image chips from our data. However, for our segmentation model, we map from three channels to one channel (a binary output mask). In the first channel of our input image, we place the per-pixel standard deviation of our data. In the second channel, we place the per-pixel maximum intensity, and in the last channel we place the per-pixel minimum intensity. Since particle signals have high standard deviation, they inherently include low intensity and high intensity locations across the z-axis. Using the standard deviation along with the maximum and minimum values allows the model to maximize input information. While mapping directly from the full reconstruction image of n_2 channels is an attractive option, it leads to decreased reconstruction performance as compared to performing mapping from the three-channel

image described above to the segmentation map. We provide hyperparameters used for training in Table 3.4.

Table 3.4: Hyperparameters for Segmentation Model

Hyperparameter	Value
Optimizer	Adam
Learning Rate	0.0001
Epochs	128

3.4 Combined Training of Reconstruction and Classification

Lastly, we explore combining training of our reconstruction and classification model. While accurate reconstruction is a key part of this research, we also wish to improve the downstream segmentation and ability to identify particles in our data. To this end, we propose a combined reconstruction and classification model. This model is made up of two UNet models, with a combined reconstruction and segmentation loss function. We formulate our loss function using both a reconstruction and segmentation loss. Our reconstruction loss is made up of a per-pixel loss in the form of Mean Squared Error and a content loss in the form of the LPIPS loss. While the MSE loss compares per-signal reconstructions, the LPIPS loss takes in the standard deviation reconstruction of the output of the reconstruction model, and compares it to the standard deviation reconstruction of the fully sampled data. Our segmentation loss is the Binary Cross-Entropy loss. We additionally include scaling parameters which allow for balancing of each portion of our loss function. Overall, our loss takes the form:

$$\mathcal{L}_{\text{combined}} = \lambda_{\text{reconstruction}}(\lambda_{\text{pixel}}\mathcal{L}_{\text{pixel}} + \lambda_{\text{content}}\mathcal{L}_{\text{content}}) + \lambda_{\text{segmentation}}\mathcal{L}_{\text{segmentation}}$$

For our implementation, we use the hyperparameters given in Table 3.5 in all experiments.

Table 3.5: Hyperparameters for Combined Model

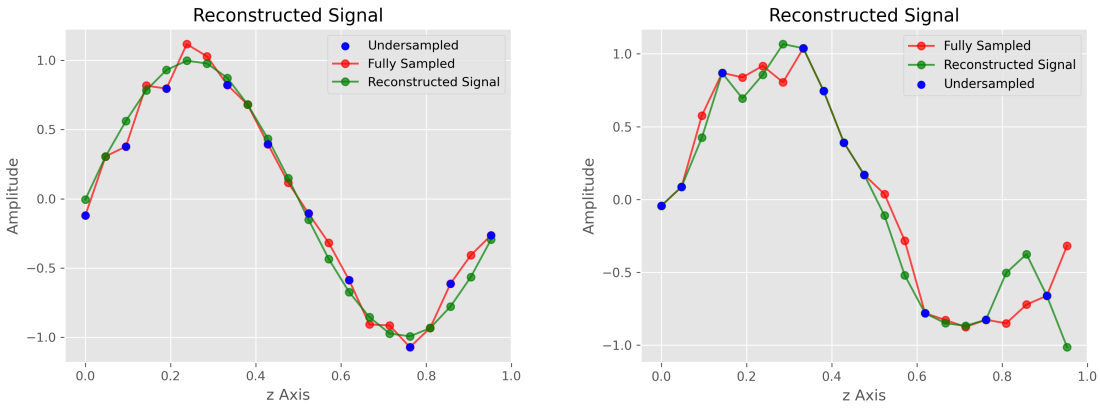
Hyperparameter	**Value**
$\lambda_{\text{reconstruction}}$	1.0
$\lambda_{\text{segmentation}}$	1.0
λ_{content}	0.006
λ_{content}	0.5
Optimizer	Adam
Learning Rate	0.001
Epochs	256

Chapter 4

Results

4.1 Synthetic Data Reconstruction

First, we present results from synthetic data reconstruction using our reconstruction techniques discussed in Section 3.2. To generate synthetic data, we take a sine wave sampled with 21 samples over 1 period, and undersample it with even spacing to 10 samples. We then add random Gaussian noise with a mean of 0 and a variance equivalent to 10% of the amplitude of our sine wave in order to introduce noise to our problem. We present sample reconstructions for our direct neural network reconstruction technique and our deep decoder reconstruction techniques below.



(a) Direct Reconstruction **(b)** Deep Decoder Reconstruction

Figure 4·1: Example Signal Reconstructions of Synthetic Data

These two images illustrate some of the fundamental differences between the two algorithms. While our direct reconstruction tends to interpolate unknown information with

a low-frequency interpolation, the deep decoder technique tends to use a higher frequency interpolation. This is supported by recent findings in literature (Rahaman et al., 2019). Since our synthetic data is inherently low frequency in nature, our direct neural network reconstruction works best for this task, as indicated in Table 4.1. In this table, we indicate the average MSE of each algorithm on 1024 test signals. A lower MSE value is better.

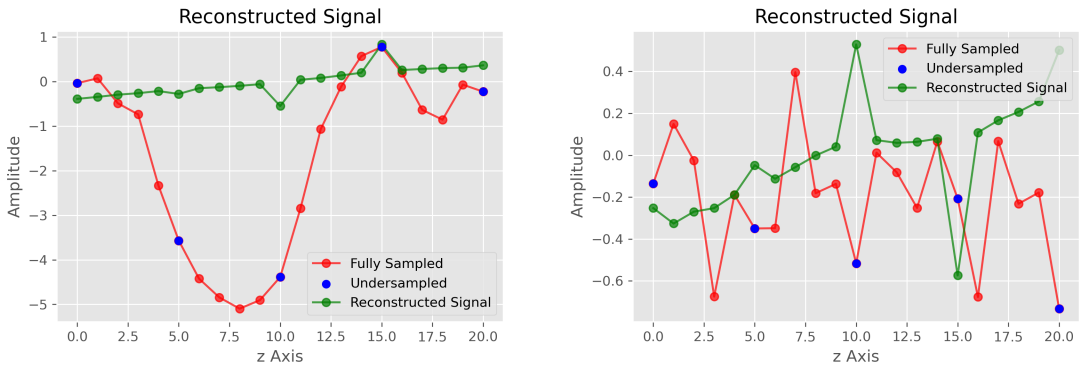
Table 4.1: Comparison of Synthetic Signal Test Set Mean Squared Error

	Mean Squared Error
Direct Reconstruction	0.0101
Deep Decoder	0.1387

4.2 Real Data

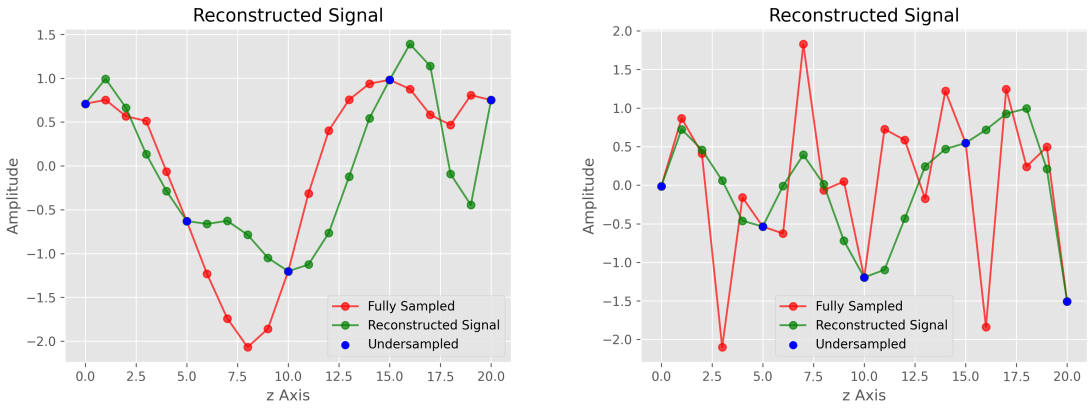
4.2.1 Real Data Reconstruction

Now, we present results from reconstruction using real interferometric data on each of our reconstruction techniques. We provide a 256×256 image chip of the standard deviation reconstruction of our data, as well as sample reconstructions over z-position from the dataset for a pixel containing noise and a pixel containing a particle. For our data shown here, we sample 5 positions from a full grid of 21 positions.



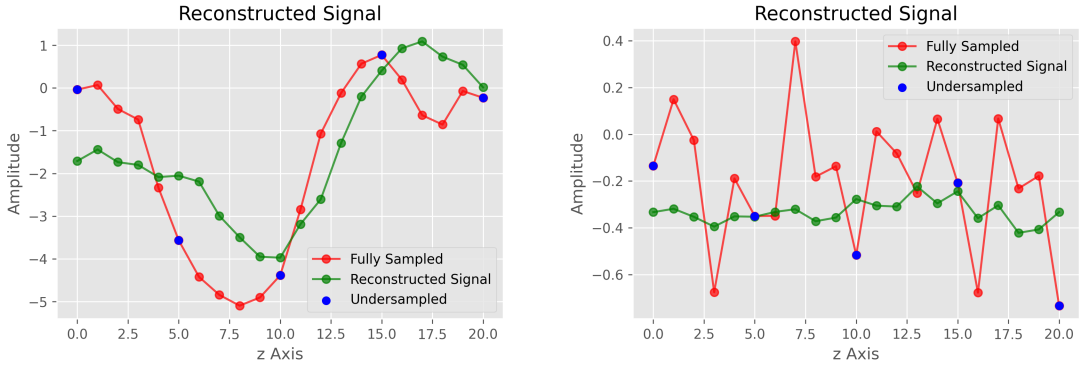
(a) Particle Data **(b)** Noise Data

Figure 4.2: Example Signal Reconstructions of Real Data using Direct Reconstruction



(a) Particle Data **(b)** Noise Data

Figure 4·3: Example Signal Reconstructions of Real Data using Deep Decoder



(a) Particle Data **(b)** Noise Data

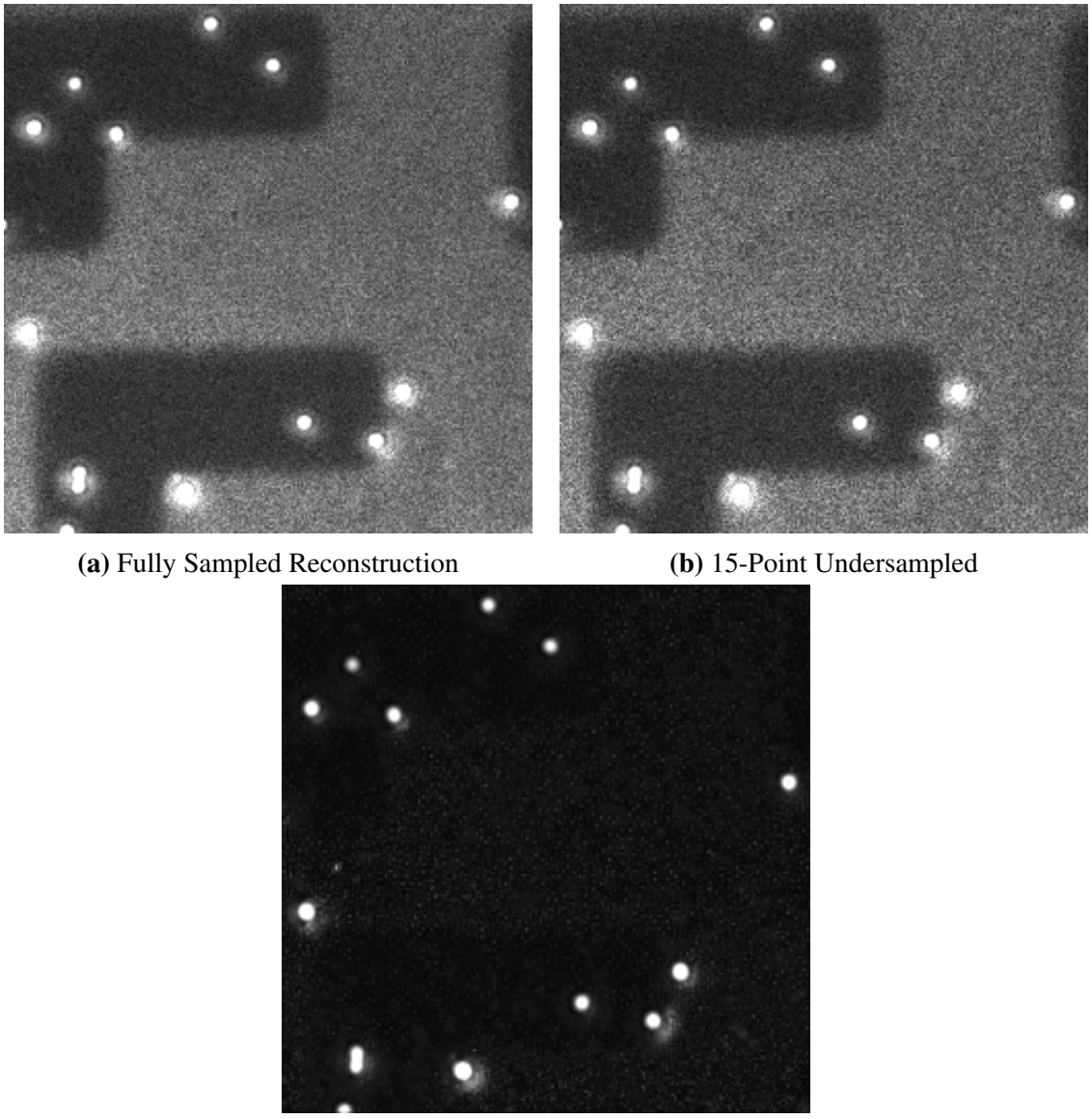
Figure 4·4: Example Signal Reconstructions of Real Data using UNet

These figures indicates that our UNet model performs best on the reconstruction task. We provide aggregate test set Mean Squared Errors in 4.2 support this assertion.

Table 4.2: Comparison of Real Data Test Set Mean Squared Error For Various Undersampling Levels

Mean Squared Error	15 Samples	10 Samples	5 Samples
Direct NN	0.851	0.854	0.862
Deep Decoder	0.801	0.891	1.035
UNet	0.028	0.028	0.039

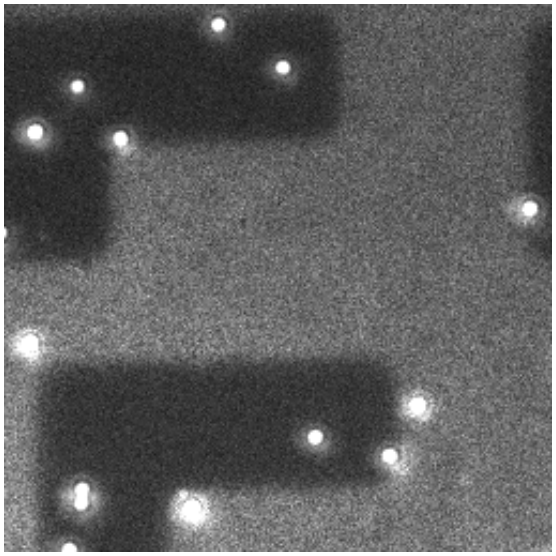
From these results, it is clear that our UNet model performs best on the reconstruction task. We provide example standard deviation reconstructions of true and reconstructed images in the Figure 4·5, Figure 4·6, and Figure 4·7. All images shown have a minimum displayed value of 0 and a maximum displayed value of 0.5 to normalize contrast between the images.



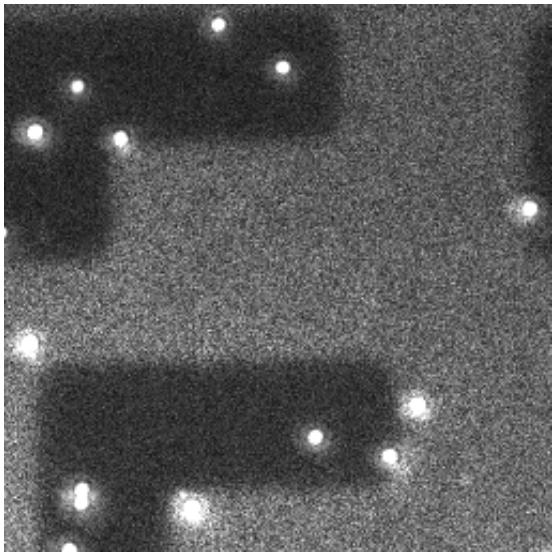
(a) Fully Sampled Reconstruction **(b)** 15-Point Undersampled

(c) UNet Reconstruction

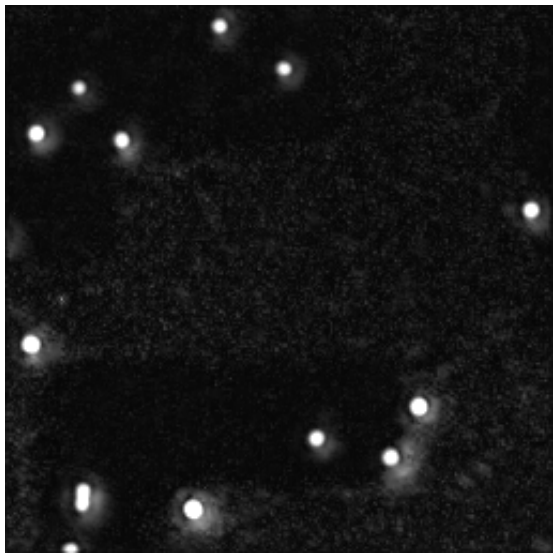
Figure 4·5: Example Signal Reconstructions of Real Data using UNet for 15-point Undersampling



(a) Fully Sampled Reconstruction

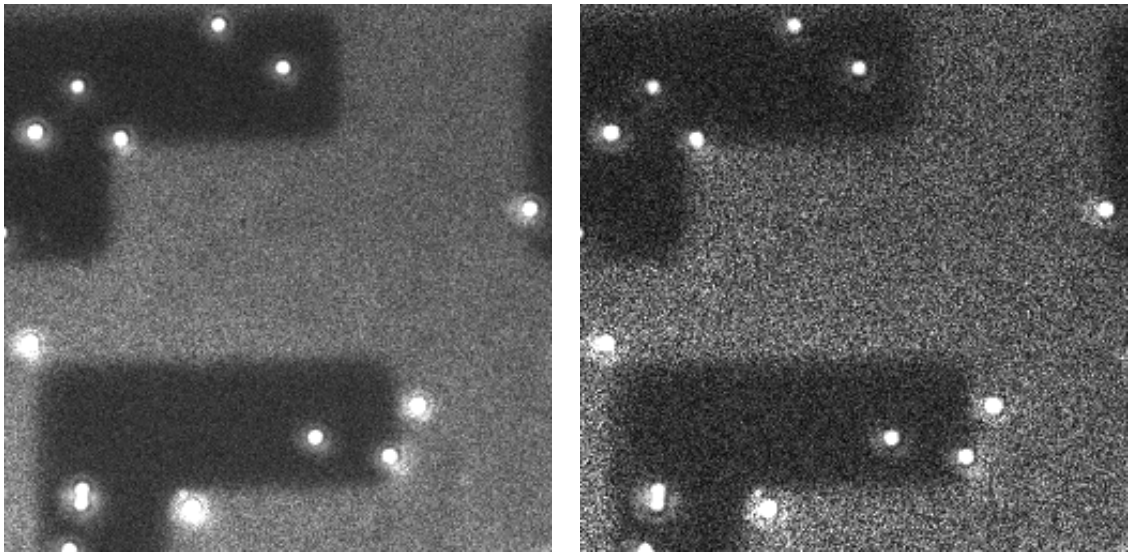


(b) 10-Point Undersampled

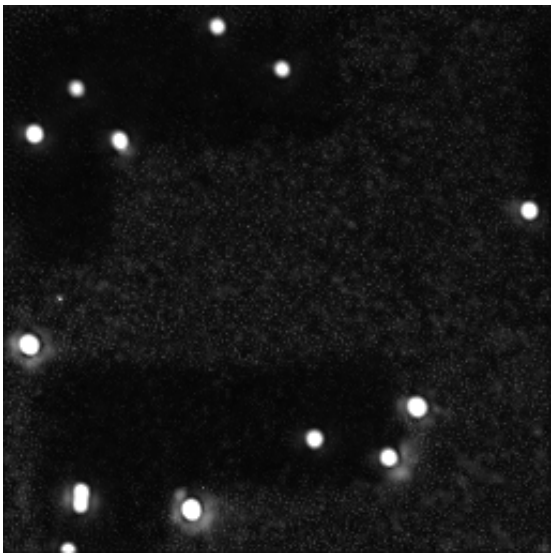


(c) UNet Reconstruction

Figure 4·6: Example Signal Reconstructions of Real Data using UNet for 10-point Undersampling



(a) Fully Sampled Reconstruction **(b)** 5-Point Undersampled



(c) UNet Reconstruction

Figure 4·7: Example Signal Reconstructions of Real Data using UNet for 5-point Undersampling

4.2.2 Real Data Segmentation

Now, we present results for real data segmentation alone. For this task, we test three models, all of which follow the same UNet architecture. To preprocess our data, we use the per-pixel standard deviation, maximum value, and minimum value as inputs to our seg-

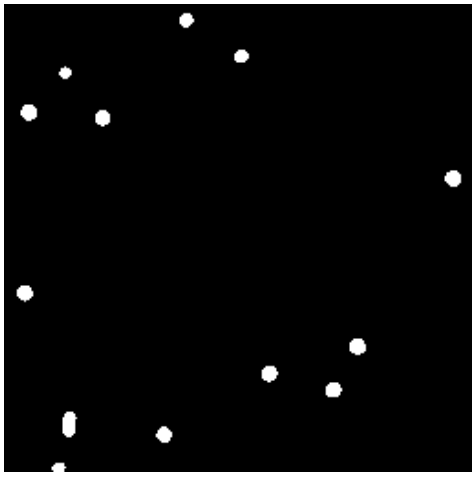
mentation model. This creates a surrogate 3-channel image which allows maximum model performance.

We measure model performance with three metrics: Dice coefficient (DC), Precision, and recall. Precision and recall are calculated per-particle, not per-pixel. Note that for these results, we use the true, fully sampled image as input to the model. For a first experiment, we train the model on the fully sampled image, and undersampled images. We recalculate the standard deviation, maximum, and minimum value for each undersampling case. We present results in Table 4.3.

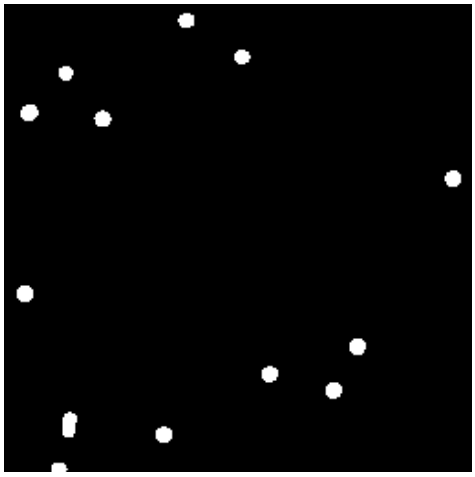
Table 4.3: Segmentation Model Performance Metrics

	Dice Coefficient	**Precision**	**Recall**
Fully Sampled	0.7017	0.9378	0.9691
UNet 15 Samples	0.7273	0.9258	0.9593
UNet 10 Samples	0.7084	0.9548	0.9591
UNet 5 Samples	0.6769	0.9220	0.9551

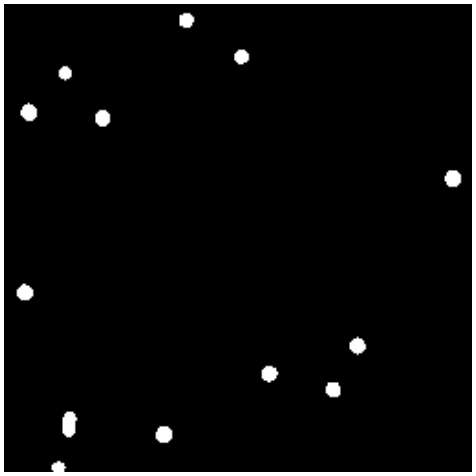
From these results, we see that precision and recall degrade as less samples are provided, while the Dice coefficient generally degrades as less samples are provided, with the exception of the fully sampled case. We provide sample segmentation images below for a single image chip in the test set for various levels of undersampling in Figure 4·8.



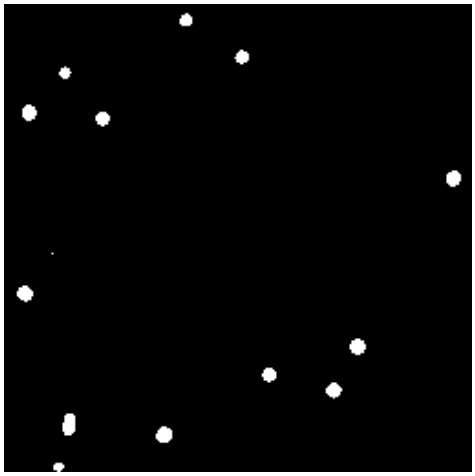
(a) Fully Sampled Data Segmentation



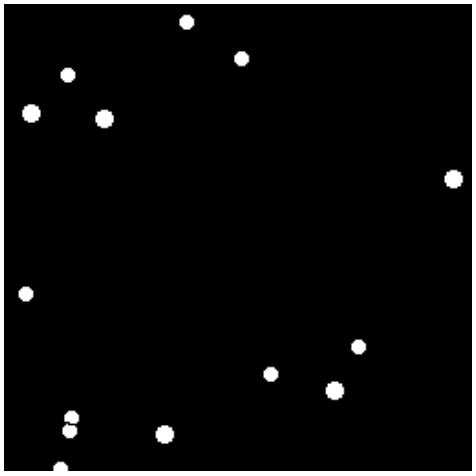
(b) 15-Point Segmentation



(c) 10-Point Segmentation



(d) 5-Point Segmentation



(e) Segmentation Label

Figure 4·8: Example Segmentation Maps of Real Data using UNet

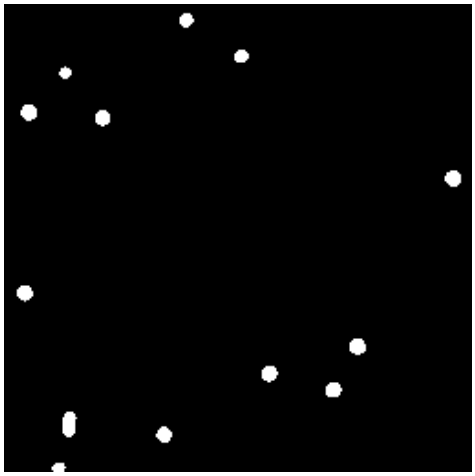
4.2.3 Real Data Combined Training

Now, we consider combined training of segmentation and reconstruction models. We first run our undersampled data through our reconstruction model, and calculate our reconstruction loss. We then calculate our surrogate 3-channel image, and run this image through our segmentation model, creating an output segmentation map. We present results in Table 4.4. The “Fully Sampled" column presents just the segmentation model run on the fully sampled data, as presented previously in Table 4.3. We additionally include the UNet alone segmentation metrics for comparison.

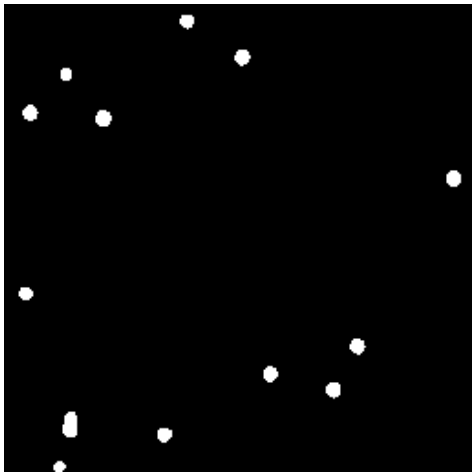
Table 4.4: Combined Model Segmentation Performance Metrics

	Dice Coefficient	**Precision**	**Recall**
Fully Sampled	0.7017	0.9378	0.9591
UNet 15 Samples	0.7273	0.9258	0.9593
Combined 15 Samples	0.6927	0.9765	0.9327
UNet 10 Samples	0.7084	0.9548	0.9591
Combined 10 Samples	0.6921	0.9812	0.9414
UNet 5 Samples	0.6769	0.9220	0.9551
Combined 5 Samples	0.6994	0.9677	0.9417

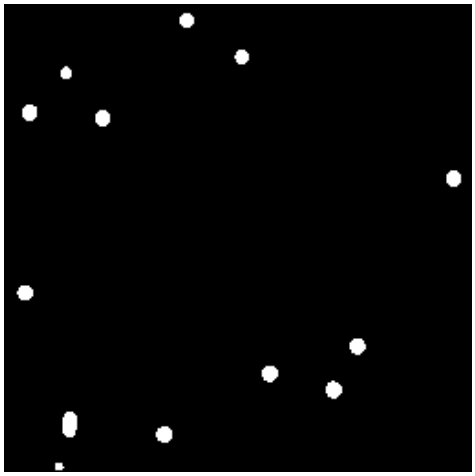
From these results, we find that the combined model suffers a small decrease in Dice coefficient as compared to the segmentation model alone in the 15 and 10 sample case, but actually outperforms the undersampled data alone in Dice coefficient for the 5 sample case. Furthermore, the combined model provides improved precision over corresponding segmentation alone models, at a small cost to recall. We provide example segmentation maps in Figure 4·9



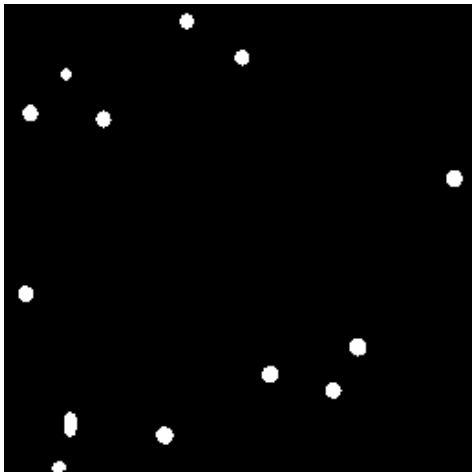
(a) Fully Sampled Data Segmentation



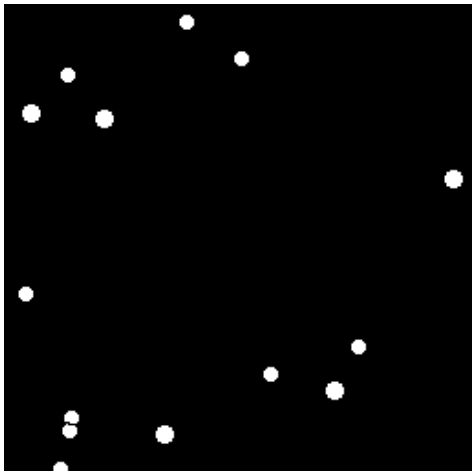
(b) 15-Point Combined Segmentation



(c) 10-Point Combined Segmentation



(d) 5-Point Combined Segmentation



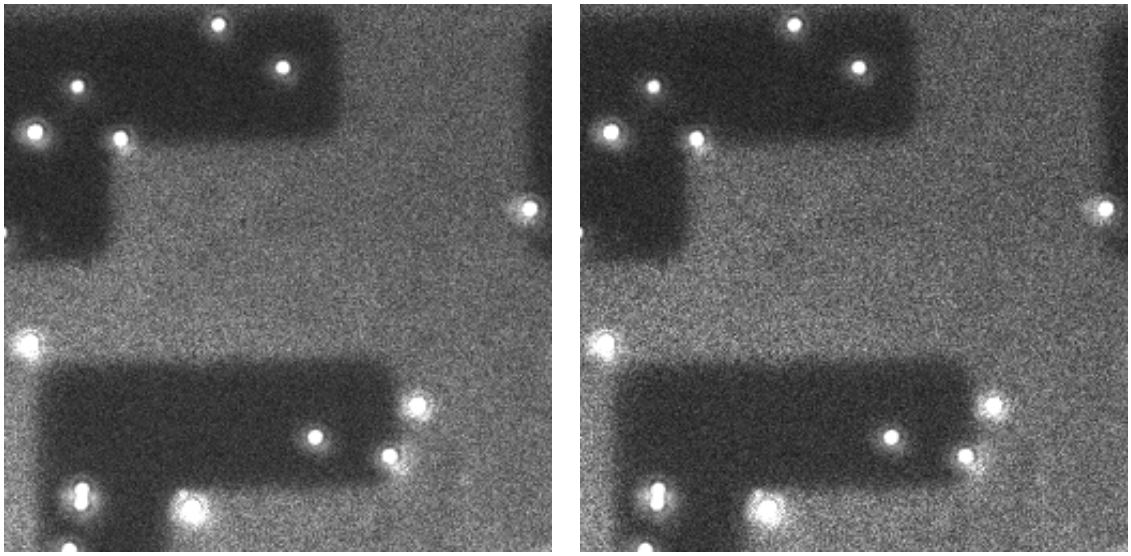
(e) Segmentation Label

Figure 4·9: Example Segmentation Maps of Real Data using Combined Models

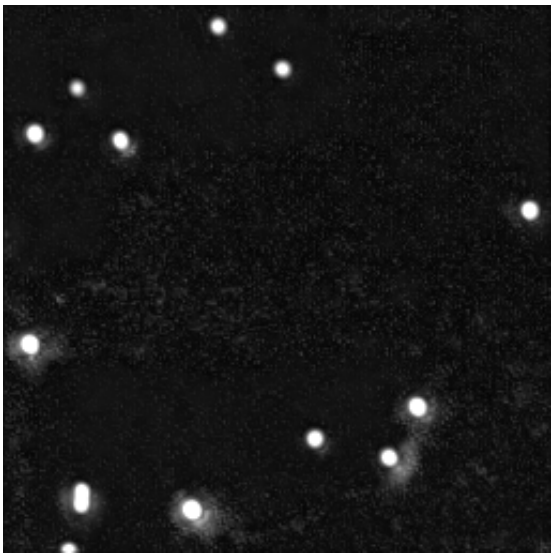
In addition to our segmentation results, we also present our reconstruction results for our combined model in Table 4.5, Figure 4·10, Figure 4·11, and Figure 4·12.

Table 4.5: Comparison of Real Data Test Set MSE for Various Undersampling Levels of UNet and Combined Models

	15 Samples	10 Samples	5 Samples
UNet	0.0276	0.0282	0.0387
Combined Model	0.0244	0.0256	0.02486

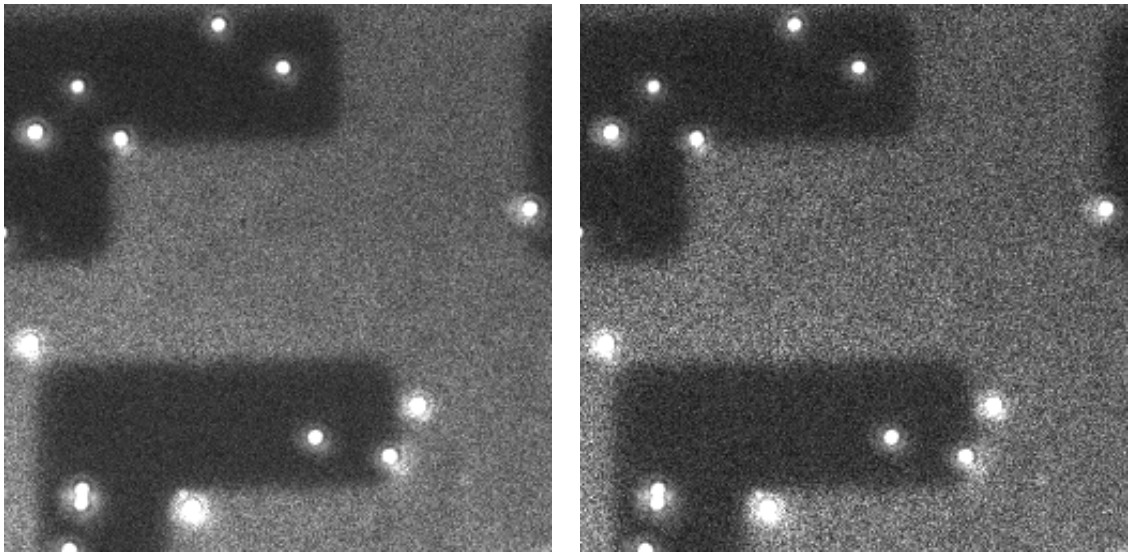


(a) Fully Sampled Reconstruction **(b)** 15-Point Undersampled

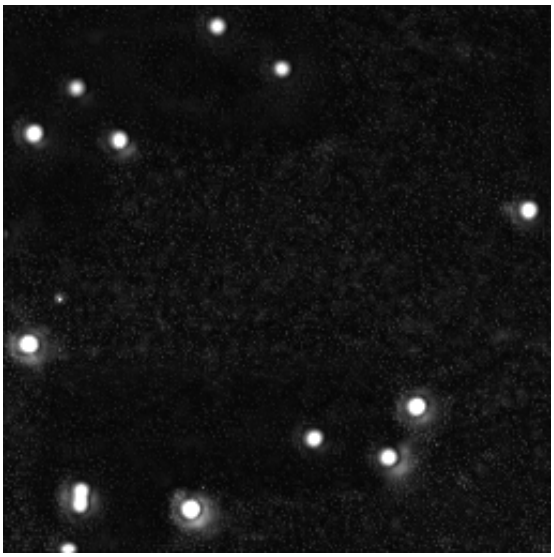


(c) Combined Model Reconstruction

Figure 4·10: Example Signal Reconstructions of Real Data using Combined Model for 15-point Undersampling

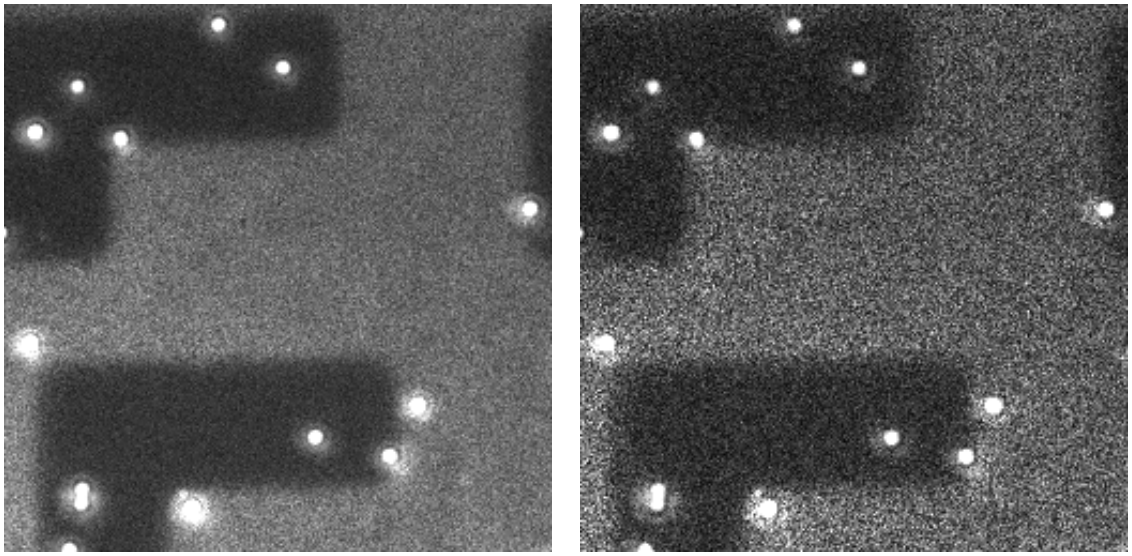


(a) Fully Sampled Reconstruction **(b)** 10-Point Undersampled

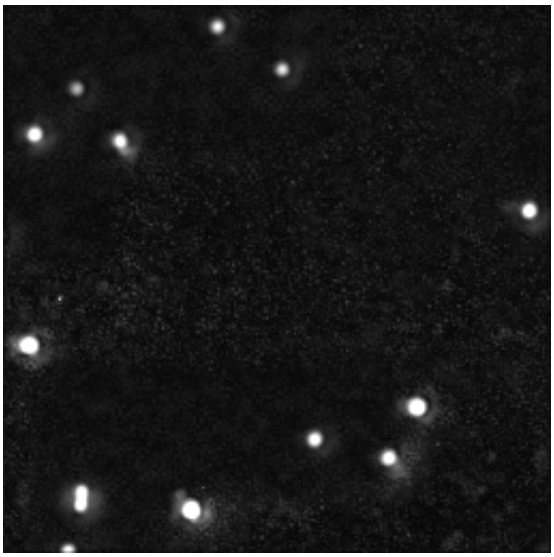


(c) Combined Model Reconstruction

Figure 4·11: Example Signal Reconstructions of Real Data using Combined Model for 10-point Undersampling



(a) Fully Sampled Reconstruction **(b)** 5-Point Undersampled



(c) Combined Model Reconstruction

Figure 4·12: Example Signal Reconstructions of Real Data using Combined Model for 5-point Undersampling

4.2.4 Runtime

Lastly, we consider the runtimes of each algorithm. In order to provide sufficient temporal fidelity for the SP-IRIS instrument, the algorithm must run in less than 60 seconds on a 25 Megapixel (MP) image. We present runtime results in 4.6.

Table 4.6: Runtimes of Reconstruction and Segmentation Models

	Runtime (signals/second)	Runtime on 25 MP image
Direct Reconst	15196	27 minutes
Deep Decoder	1.14	7916 hours
UNet Reconst	1260257	19.83 seconds
Segmentation	2945188	8.49 seconds
Combined	1170473	21.36 seconds

From this table, it is clear that the UNet reconstruction model, segmentation model and the combined model are the only models fast enough to be useful for the SP-IRIS system.

Chapter 5

Conclusions

5.1 Summary of the thesis

In this work, we implement a novel combined classification and reconstruction deep learning model for SP-IRIS defocus images, and compare its performance against individual existing deep learning models. The results of this have implications on SP-IRIS imaging. Primarily, due to the ability of deep neural networks to consistently reconstruct and segment highly undersampled data, fewer images can be taken using SP-IRIS while still maintaining sufficient particle detection. This has implications on imaging throughput, as the technique can be performed more quickly with less z-positions sampled. Additionally, we can increase the SNR of individual sampling locations while maintaining imaging speeds. For example, using SP-IRIS we can image 5 z-positions, but take 4 images at each position. We then take the average of the set of images at each position, leading to a less noisy set of output measurements. Lastly, we have automated the particle identification task, and mitigated inconsistencies between different labellers by providing an automated technique which can consistently segment biological nanoparticles. Our best performing algorithms all run in 22 seconds, allowing us to take automated particle counts in 22 second intervals. Future work includes testing of this model on target instrument computational hardware and data, and development of an adversarial training module for the algorithm.

References

Avci, O., Ünlü, N. L., Özkumur, A. Y., and Ünlü, M. S. (2015). Interferometric Reflectance Imaging Sensor (IRIS)–A Platform Technology for Multiplexed Diagnostics and Digital Detection. *Sensors (Basel, Switzerland)*, 15(7):17649–17665.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Information science and statistics. Springer, New York.

Duvenaud, D. (2014). *Automatic Model Construction with Gaussian Processes*. PhD thesis, Computational and Biological Learning Laboratory, University of Cambridge.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer, 2 edition.

Heckel, R. and Hand, P. (2018). Deep decoder: Concise image representations from untrained non-convolutional networks. *CoRR*, abs/1810.03982.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67.

Kingma, D. P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*. arXiv: 1412.6980.

Kupyn, O., Martyniuk, T., Wu, J., and Wang, Z. (2019). DeblurGAN-v2: Deblurring (Orders-of-Magnitude) Faster and Better. *arXiv:1908.03826 [cs]*. arXiv: 1908.03826.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., and Terzopoulos, D. (2020). Image Segmentation Using Deep Learning: A Survey. *arXiv:2001.05566 [cs]*. arXiv: 2001.05566.

Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. (2018). Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *arXiv:1811.03378 [cs]*. arXiv: 1811.03378.

O'Shea, K. and Nash, R. (2015). An Introduction to Convolutional Neural Networks. *arXiv:1511.08458 [cs]*. arXiv: 1511.08458.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F. A., Bengio, Y., and Courville, A. (2019). On the Spectral Bias of Neural Networks. *arXiv:1806.08734 [cs, stat]*. arXiv: 1806.08734.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv:1505.04597 [cs]*. arXiv: 1505.04597.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *arXiv:1409.0575 [cs]*. arXiv: 1409.0575.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2020). Deep Image Prior. *International Journal of Computer Vision*, 128(7):1867–1888. arXiv: 1711.10925.

Yedder, H. B., Cardoen, B., and Hamarneh, G. (2020). Deep Learning for Biomedical Image Reconstruction: A Survey. *arXiv:2002.12351 [cs, eess]*. arXiv: 2002.12351.

Yurdakul, C., Ünlü, M. S., and Ünlü, M. S. (2020). Computational nanosensing from defocus in single particle interferometric reflectance microscopy. *Optics Letters*, 45(23):6546–6549. Publisher: Optical Society of America.

Zaraee, N., kanik, F. E., Bhuiya, A. M., Gong, E. S., Geib, M. T., Lortlar Ünlü, N., Ozkumur, A. Y., Dupuis, J. R., and Ünlü, M. S. (2020). Highly sensitive and label-free digital detection of whole cell E. coli with Interferometric Reflectance Imaging. *Biosensors and Bioelectronics*, 162:112258.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. *arXiv:1801.03924 [cs]*. arXiv: 1801.03924.

CURRICULUM VITAE