Interferometry Fringe Tracing using Image Segmentation

Jake Lawson

Image segmentation using a modified UNet was applied to interferometry fringe tracing. The machine learning algorithm was capable of tracing complicated interferometry fringe patterns under certain conditions. The model was trained on interferometry data from the Laboratory of Plasma Studies at Cornell.

I. Introduction

The Laboratory of Plasma Studies (LPS), at Cornell University, uses interferometry to collect experimental data. The interferometry images must be fringe traced to extract the experimental data from the images. Fringe tracing is done by tracing the minimums of the interferometry pattern. Currently, fringe tracing is done by a combination of a non-machine learning algorithm and tracing by hand. This algorithm performs well for simple fringe traces but is unable to handle complicated fringe patterns.

Machine learning (ML) has recently been applied to numerous image related tasks previously thought to be difficult to do algorithmically [1]. One example is image segmentation, which is like fringe tracing. Given the success of image segmentation in other image related tasks, it could be applied to interferometry fringe tracing. A more accurate fringe tracing algorithm that can handle complicated interferometry patterns would reduce data analysis times greatly, due to decreasing the amount of tracing that must be done by hand.

II. Theory

A machine learning algorithm is an algorithm that learns how to perform a task better as data is given to it. The data used to train the algorithm is the training dataset, and the data used to test the algorithm is the testing dataset. The goal of a training dataset is to approximate the span of possible data for that experiment or task.

Image segmentation is a field of machine learning, and specifically image processing using ML. An image segmentation algorithm labels different pixels of an image based on what the algorithm "thinks" is present at that pixel. Image segmentation has been used to detect many different objects in images, including people, cars, dogs, and many other different objects. The output of an image segmentation algorithm is a segmentation map. Each of these segmentation maps have a probability at every pixel. This probability is the probability that that pixel contains the object being detected.

III. Network Architecture

The foundation for the ML architecture used is UNet. UNet is a fully convolutional neural network (CNN) originally designed for biomedical image segmentation [12]. A fully convolutional neural network is a CNN that does not have any dense layers, or a layer in which every neuron in that layer is connected to every neuron in the previous layer. CNNs use a convolution operator to connect layers rather than dense layers.

CNNs are used for image processing in ML for multiple reasons. One is the locality enforced by the convolution operator. The locality comes from the kernel size of the convolution operator.

UNet's architecture is made from a contracting path and an expanding path. The goal of the expanding path is to reduce the original image to a feature representation. This is done by sequentially increasing the number of feature channels (initially 3 for RGB images, 1 for greyscale), while downsampling the image. The goal of the contracting path is to create the segmentation map from the feature representation. The contracting path utilizes convolution blocks (groups of sequential convolutions), as well as transposed (or inverse) convolution operators, which upsample the feature representation while decreasing the number of feature channels. Outputs from the contracting path are also given to the expanding path at each stage to remind the network of the original image, which is facilitated via long skip connections.

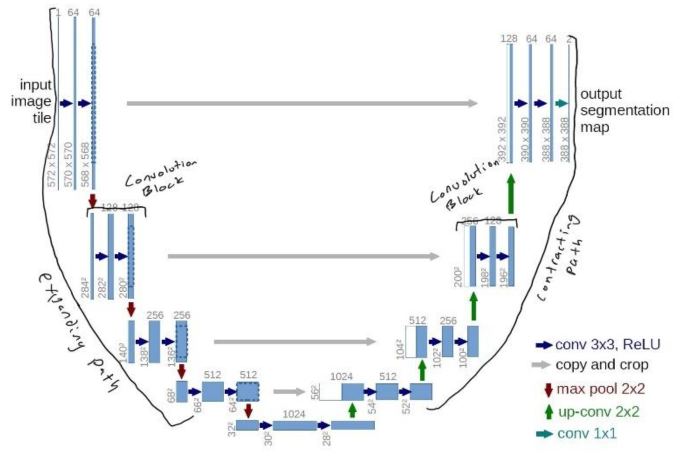


Figure 2. UNet network architecture.

Each path consists of multiple convolution blocks, which sequentially apply convolutions to the input. In the contracting path, a convolution block is applied to the image, followed by a downsampling operation. In the original UNet design, this is a 2x2 max pooling operation, which splits the image into 2x2 sections and only takes the largest value. This downsampling was improved to a 2x2 convolution on the same 2x2 sections described above, which allows for learnable downsampling.

Model Improvements Ideas

Short Skip Connections

Generally, skip connections bring data from one point in the model to a later point. The original UNet implementation already contains long skip connections, connecting the expanding path to the contracting path. Short skip connections were added to the model to connect the beginning and the end of each convolution block. Short skip connections have been shown to increase the trainability of the interior of the model. This allows for more of the model to be effectively used in segmentation.

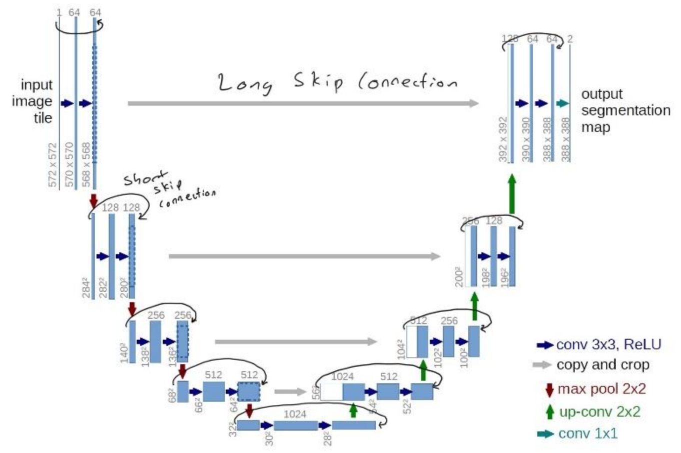


Figure 3. UNet network architecture with short skip connections added.

Batch Normalization

Batch normalization normalizes the data in each feature map. This has been shown to have many beneficial properties for image classification [2] including reducing training time and increasing model generalizability. Currently a form of normalization is present in most all modern image segmentation networks.

Recurrent Operations

Recurrent operation are operations that loop the data through the operator multiple times rather than just a single time. When added to CNNs for image segmentation, they have been shown to improve feature representation for the segmentation task and allow for better performance without increasing the number of model parameters [4], [5].

Attention Gates

Attention gates learn to focus on important structures and regions in the input image and suppress irrelevant regions. The addition of attention gates to a CNN has been shown to increase model sensitivity and prediction accuracy without sacrificing computational efficiency [6].

IV. Dataset

The dataset used is a collection of 105 interferometry images and their corresponding fringe traces produced by the Laboratory of Plasma Studies at Cornell University (LPS). The fringe traces for these images were made by a combination of the previous fringe tracing algorithm and human cleaning and tracing. This dataset contains a mixture of simple and complicated fringe patterns. A simple fringe pattern has a very strong vertical bias in the fringe patterns, with little to no bends or turns in the fringe patterns. They also are similarly spaced out throughout the entire image. More complicated fringe patterns still have a vertical bias but contain many bends and pockets in certain regions of the fringe pattern.

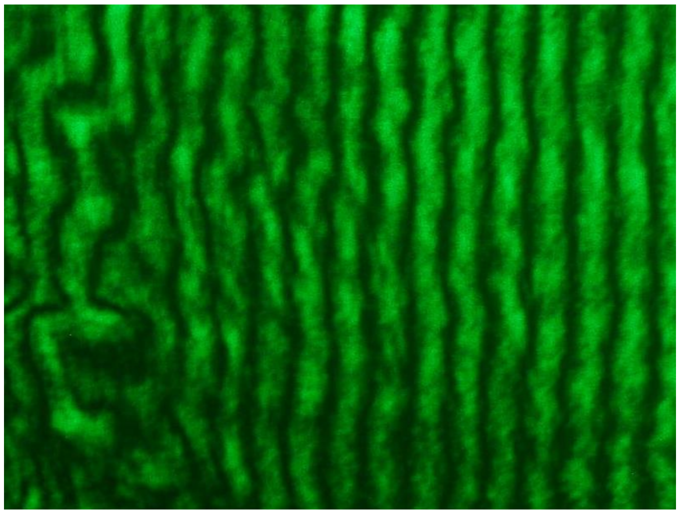


Figure 4. Interferometry image containing a complicated pattern on the left side of the image, and a simpler pattern on the right.

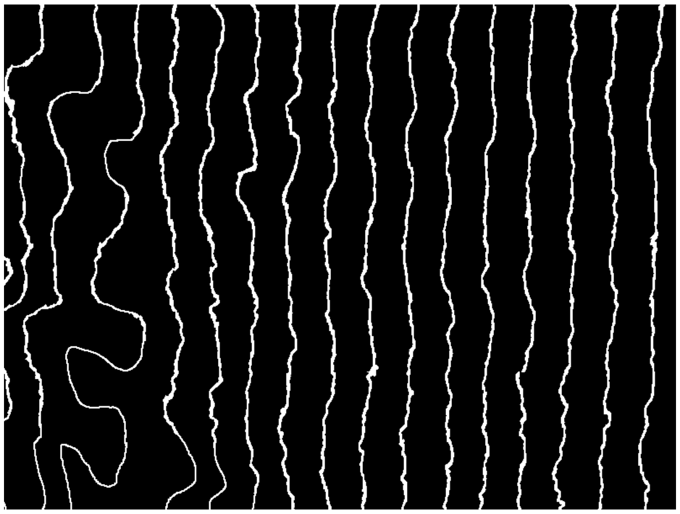


Figure 5. Fringe trace of the interferometry image in figure 3.

Data Preparation

The full interferometry images are too large to be effectively trained on, therefore, they are cut up into smaller images to allow for training. Given a full interferometry image and its fringe trace pair, the unnecessary pixels where no fringe trace is present (typically the top and bottom of each image) are removed. Then the images are cut up into a usable size. The final size of the cut images can be freely chosen, but the image sizes used were 600x400 pixels and 250x200 pixels. An overlap must be added to the images because each time a convolution is applied to them, data on the edge of the image is lost. This overlap was set at 100 pixels on each side for all the cutting, resulting in a final image size of 800x600 and 450x400. The same algorithm is used while cutting the fringe trace images which ensures that the images are cut the exact same way.

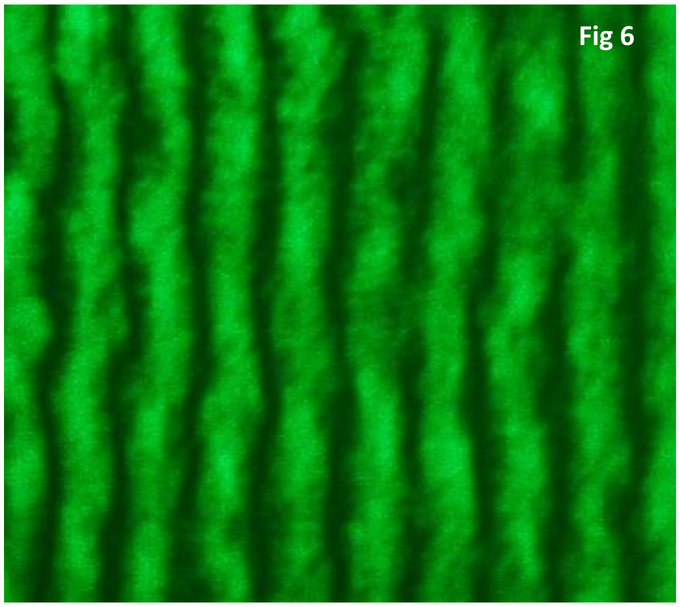
After all the images are cut, they can be used to train the model. The initial dataset contained 105 very large interferometry images and their pairs. After cutting the dataset contained around 2200 800x600 images or 12000 450x400 images.

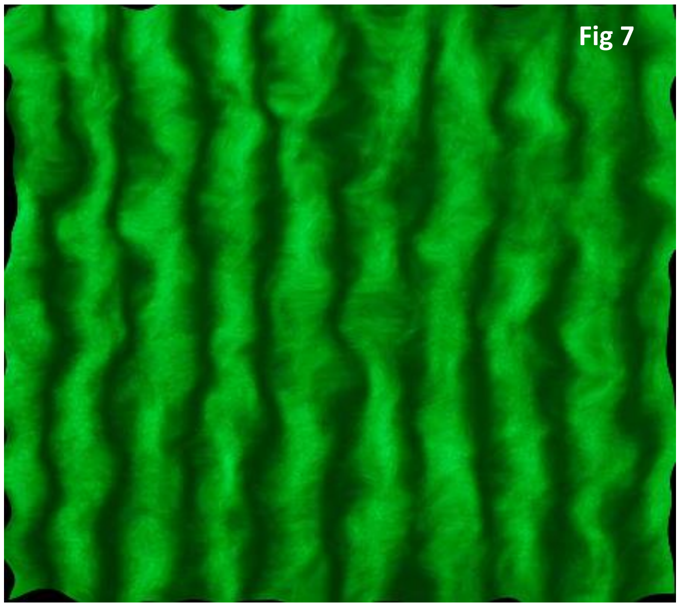
Data Augmentations

Data augmentations have been shown to combat overfitting in image classification and image segmentation tasks [9]. The goal of data augmentations is to better generalize and expand your training dataset by creating new images previously unseen to the model via augmentations. Two main augmentations were applied, image flips, and elastic deformation. These augmentations were applied to training images with a given probability (usually around .05-.2).

For image flips, both horizontal flips and vertical flips were used. These created small changes in the training dataset without fundamentally changing the input image due to the vertical bias of the fringes. The rotations also impart a level of symmetry on the model which should produce less overfitting and better generalizability.

Elastic deformation locally stretches, compresses, and shifts the image in a continuous manner. The outputs can turn simple fringe images into much more complicated fringe images by emulating the bends and pockets of complicated fringe images. Adding elastic deformed images to the training dataset should allow the model to better be able to trace complicated fringe patterns.





Figures 6, 7. Non augmented interferometry image (5) along with the same image after being elastically deformed (6).

V. Training

Model training was done using google colaboratory. This was done because modern machine learning models are greatly sped by using GPUs. The python library PyTorch [11], which is a python library created for deep learning, was used to create the python code for training. The adam optimizer was used as the back propagation optimizer for all models [10].

Loss Functions

ML algorithms are trained by using a loss function, which estimates the error or loss of the model's output when compared to the true output (given fringe trace). This loss is then backpropagated through the model, and the model's parameters are changed slightly to decrease loss.

The loss function I used was a sum of dice loss and binary cross-entropy loss. Both loss functions have been shown to work well for general image segmentation tasks [7]. Binary cross-entropy loss is defined as follows, where y is the true value (0 or 1) and $\hat{y}$ is the model predicted value,

$$L_{BCE}(y,\hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\big(y_i log(\hat{y}_i) + (1-y_i)log(1-\hat{y}_i)\big).$$

Binary cross-entropy loss measures the difference between two probability distributions, and therefore, serves as a good loss function for both image classification and image segmentation. Dice loss is defined as follows,

$$L_{Dice}(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{N} 2y_i\hat{y}_i}{\sum_{i=1}^{N} y_i + \hat{y}_i}.$$

Dice loss basically computes a normalized dot product between the two images and comes from the dice coefficient used in computer vision to calculate the similarity between two images.

A separate loss function approach using a part of a generative adversarial network (GAN) [8] was also tried. GANs are used for generating fake photorealistic images and contain two parts, a generator and discriminator. The goal of the generator is to create fake data from a, typically random, input. The goal of the discriminator is to discriminate between real and fake data. These are trained in an adversarial way, the generator is trying to fool the discriminator, and the discriminator is trying not to get fooled.

For this application, the modified UNet model was used as the generator, and a CNN discriminator was used as a learnable loss function.

VI. Results

Many different variations of the modified UNet model were tested on the above-mentioned dataset. Unfortunately, none of these models were very accurate and consistent in fringe tracing. Overfitting was the primary reason for model testing inaccuracy. Despite the attempts made to combat overfitting, any model that could learn the segmentation task in a short enough period of training time also would overfit the data. Below are the model hyperparameter changes made to reduce training time and combat overfitting.

Initial Kernel Size

For most CNN's, the kernel size of the internal convolution operations is set to three. This is not always the case, however, for the initial convolution kernel size. Therefore, some testing was done to optimize this hyperparameter.

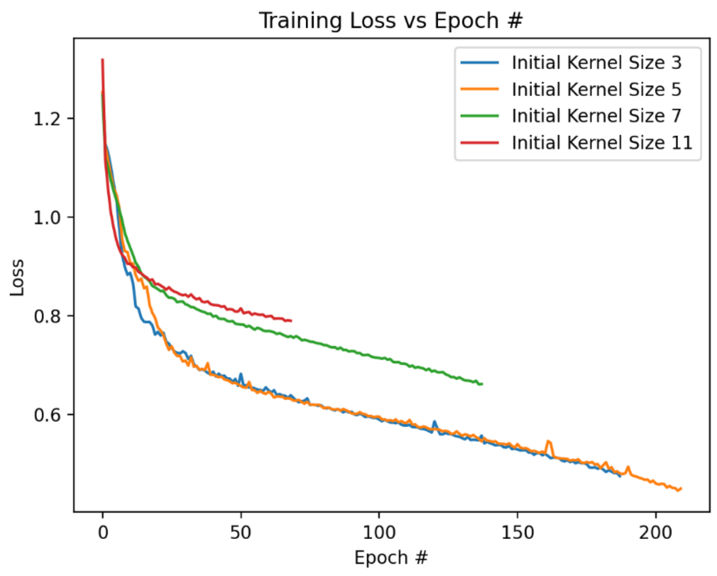


Figure 8. Training loss plots for each of the shown kernel sizes. From this plot, an initial kernel size of 3 or 5 is optimal.

Given the results in figure 8, the initial kernel size was set to 3 for the rest of the models.

Batch Normalization and Short Skip Connections

As mentioned earlier, batch normalization has been shown to have many beneficial properties for image classification and is present in modern image segmentation models. Short skip connections allow for more of the model to be used by spreading out parameter updates throughout the model during training.

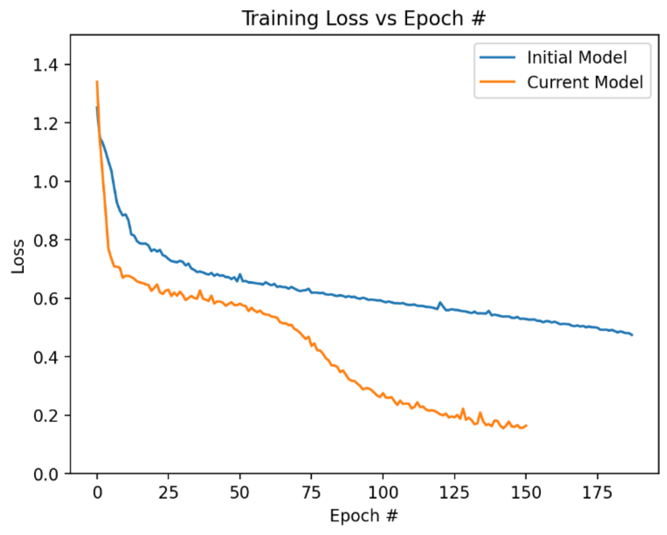


Figure 9. Training loss plots of initial model (no batch normalization or short skip connections) and model with batch normalization and short skip connections.

The initial model would have taken far too long to train to an acceptable performance level assuming training did not speed up or slow down for epochs after 200. Training most likely would have slowed down as training continued, so the initial model was not capable of performing the task adequately. The addition of batch normalization and short skip connections were very beneficial to the model's performance in training. The drop in loss of the current model at around epoch 75 is most likely associated with overfitting.

Recurrent Operations

As mentioned earlier, recurrent operations loop the data through the operation multiple times rather than just once. They have been shown to create better feature representation and therefore better performance without increasing model parameter size.

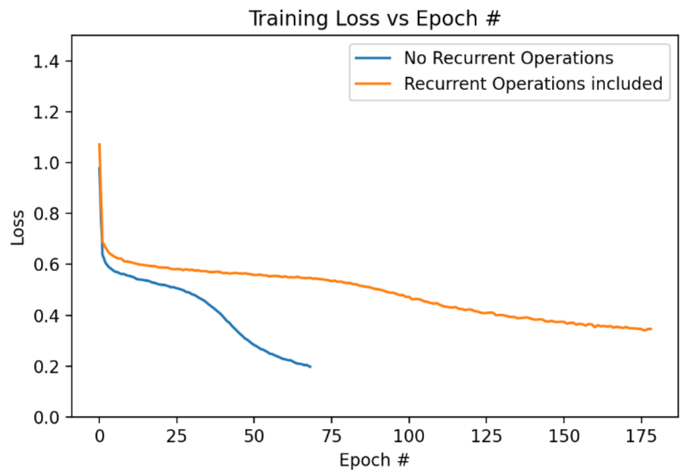


Figure 10. Training loss of model with and without recurrent operations present.

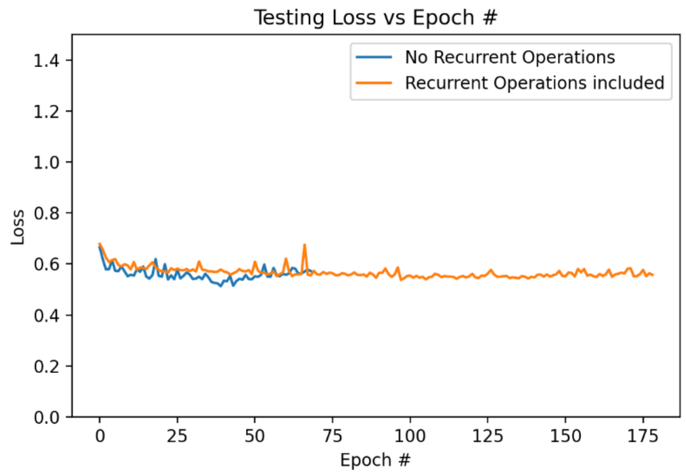


Figure 11. Testing loss of model with and without recurrent operations present.

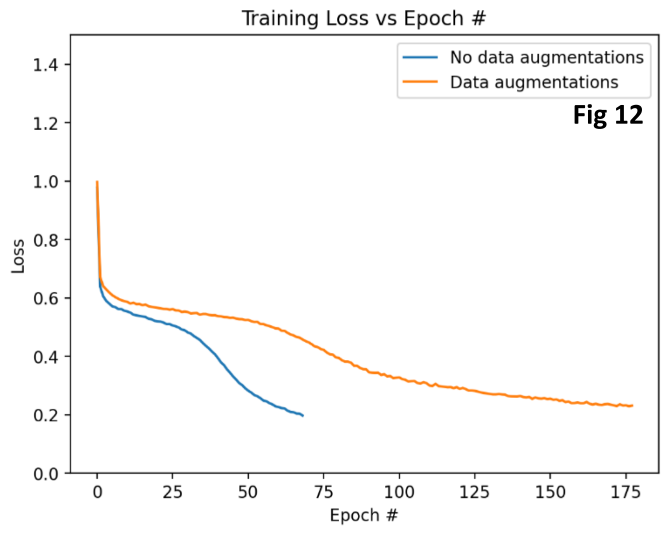
Given the above loss plots, recurrent/residual operations did not seem to be beneficial to the model performance. While they did help to combat overfitting, they did that by worsening training performance, not improving testing performance. Because of this, they were not used in later models. One main problem in testing residual/recurrent operations was the combinatorial possibilities of where to position them in the model. There is not a standard way of inserting them into a model. Given a different positioning than what I tested, they could be beneficial, but this seems rather unlikely given the negative impacts they had on training.

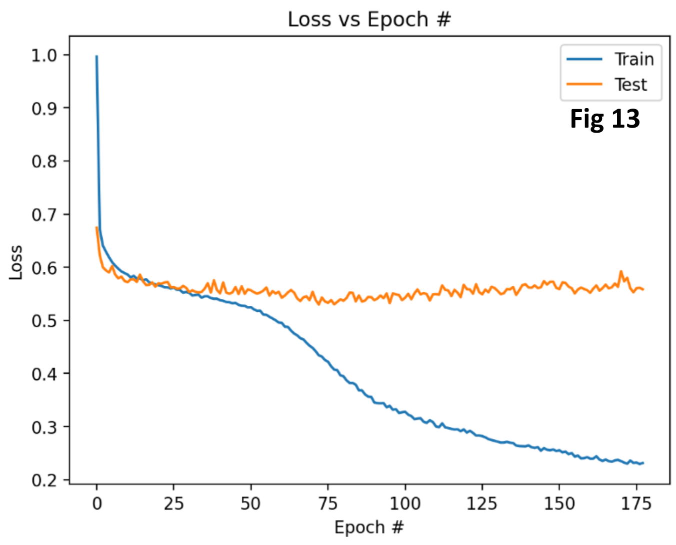
Attention Gates

Attention gates have been shown to learn to focus on important structures and suppress irrelevant structures in image segmentation. Due to some training and testing data being overwritten, I do not have plots for the performance of attention gates. Regardless, they did not work. Training loss went up to around 2 (normal loss after first epoch is around 1.1) and did not decrease much during training. Similar to residual/recurrent operations, there are multiple different ways to add attention gates to CNNs, and more specifically UNet. Therefore, it is possible that other ways of adding attention gates into UNet would produce better results, but this is unlikely. The way of adding them used seemed to be the standard and most common way.

Data Augmentations

Two types of data augmentations were used to counter overfitting of the model, image flips and elastic deformation. The severity of the elastic deformation used is controlled by two parameters, one that controls how much pixels move, and the other controls how continuous this transformation is. The severity parameter was set with a normal distribution, and the continuity parameter was set to a constant relatively high value.





Figures 12, 13. The top plot is the training loss plot for a model trained with and without data augmentations (11). The bottom plot is training and testing loss for model trained using data augmentations (12).

From the figures above, data augmentations were able to slow overfitting. This, however, was not enough to eliminate overfitting. Other data augmentations could also be applied, like image brightness and contrast augmentations due to the variability of brightness and contrast in the actual dataset.

GAN Loss Function

Generative adversarial networks can create fake photorealistic images. This is in part due to the discriminator used in a GAN. The discriminator discriminates between real and fake created images. Due to this capability, a discriminator was used as a learnable loss function. To properly train the discriminator to be used as a loss function, the model was trained using dice and binary cross entropy loss for 100 epochs. During these 100 epochs, the discriminator was also trained with the model output being labeled fake, and the true fringe trace being labeled real. After 100 epochs, the models training was governed by the discriminator output rather than dice and binary cross entropy loss.

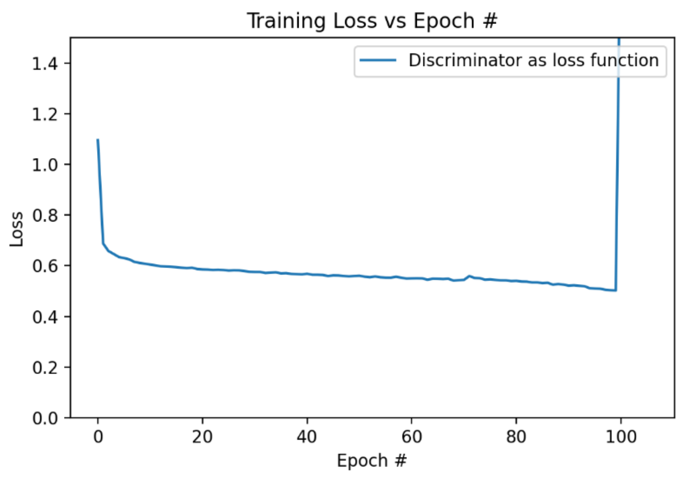


Figure 14. Training loss plot using a discriminator as a learnable loss function. Dice and binary cross-entropy loss were used for training until epoch 100. Then the trained discriminator was used.

Given the results in figure 14, the learnable loss function did not work. The loss immediately rose to well above the starting loss of the model, and the segmentation outputs were completely blank. Discriminators are notoriously hard to use due to the adversarial way they are trained, which complicates training a model for this task too much.

Image Sizes

Two image sizes were used to train models, as described above. The two sizes are 800x600 and 450x400. Of the two sizes, 800x600 seemed to have better results. This is due in part to each image containing more data, and therefore making it harder for the model to overfit. The larger images also allowed for a smaller batch size while training, which tends to generalize better.

Best Model Results

The best model was applied to new images not previously seen by the model in either the training or testing set. This model performed adequately on both complicated and simple fringe traces when the images were bright. For dull images, the model performed poorly, but this can be rectified by adjusting the brightness of the dull images.

Fig 15

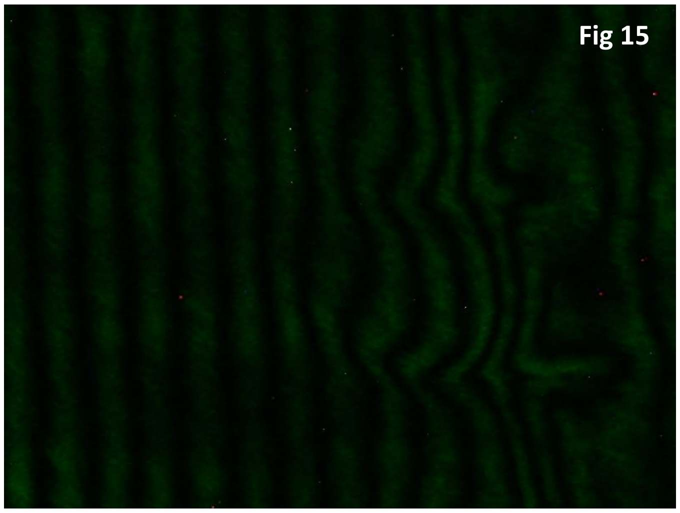


Fig 16

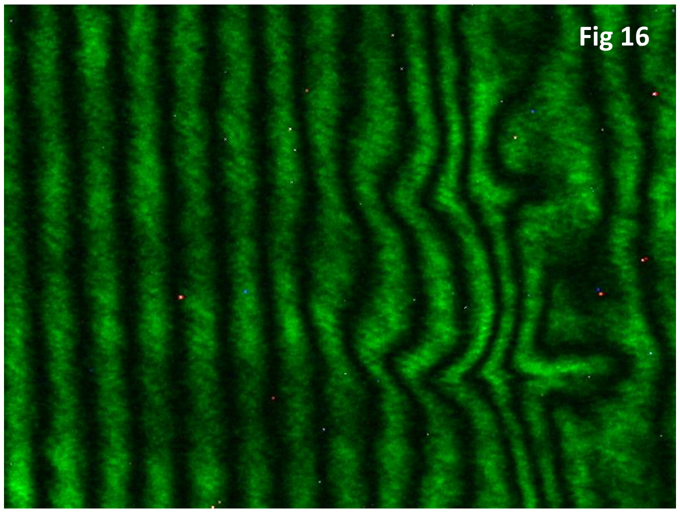


Fig 17

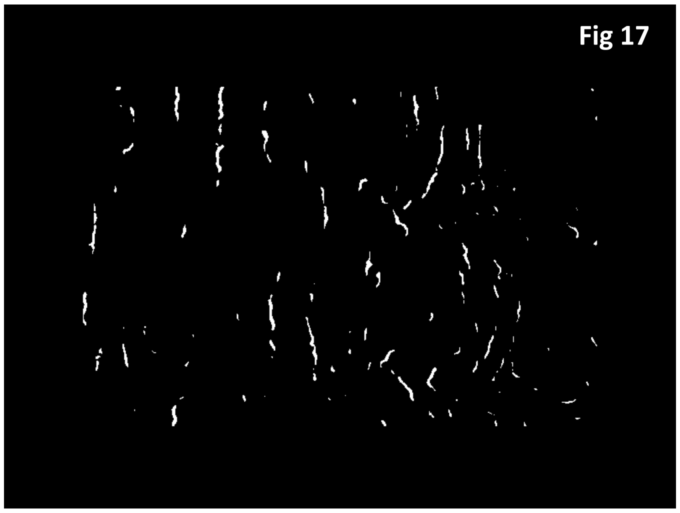
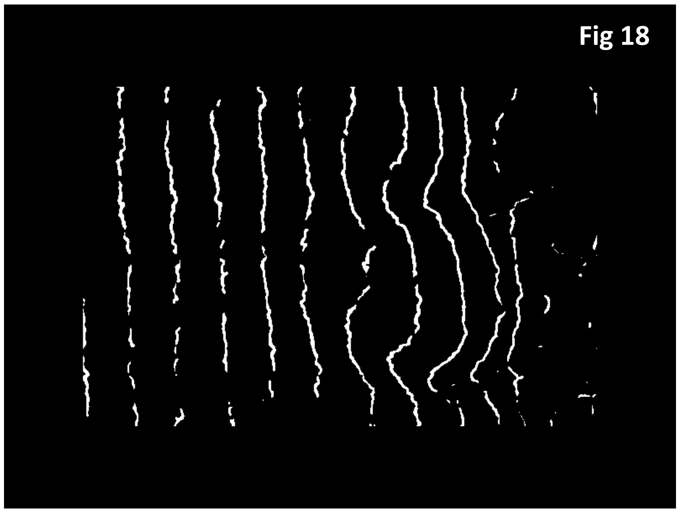


Fig 18



Figures 15, 16, 17, 18. 15 is the original interferometry image. 16 is the same as 15 but with the image brightened. 17 is the model output of the original image. 18 is the model output of the brightened image.

Fig 19

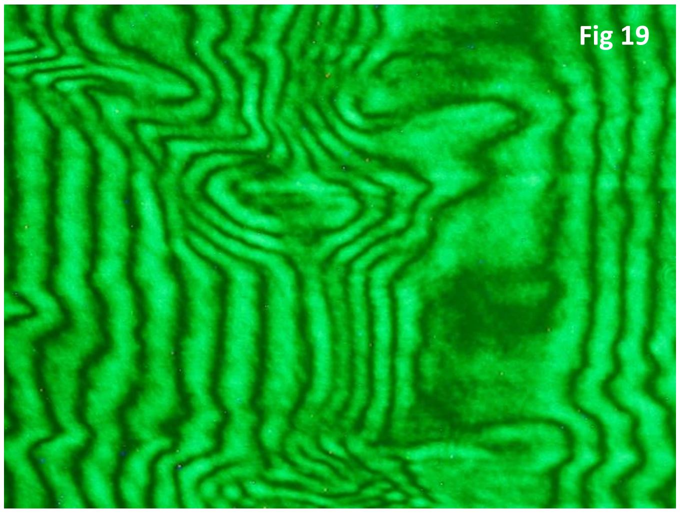
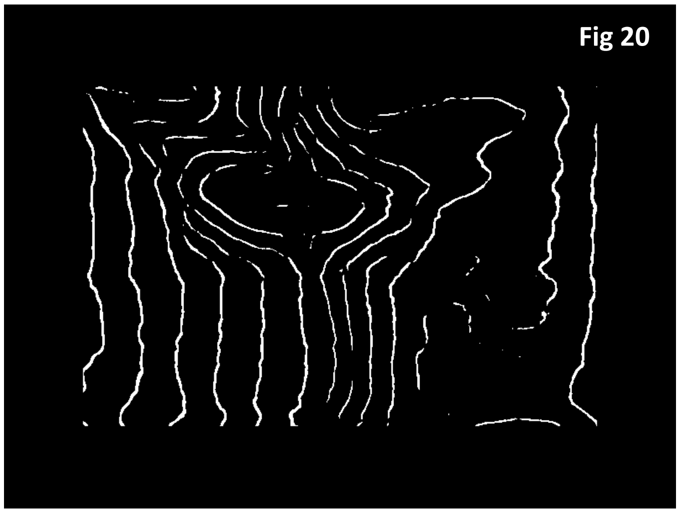
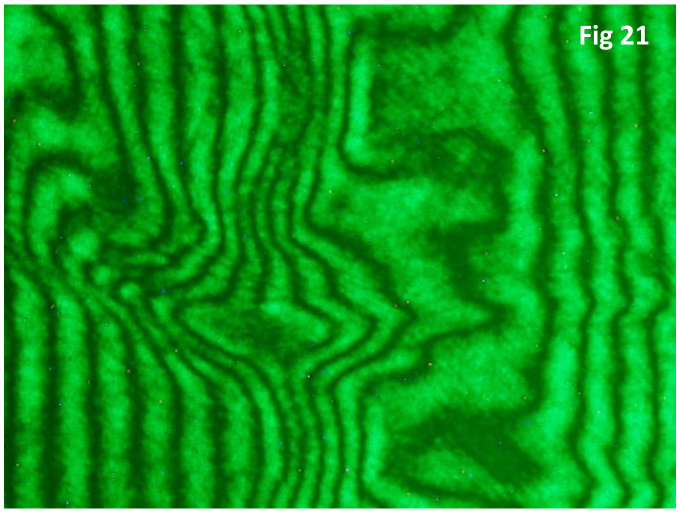
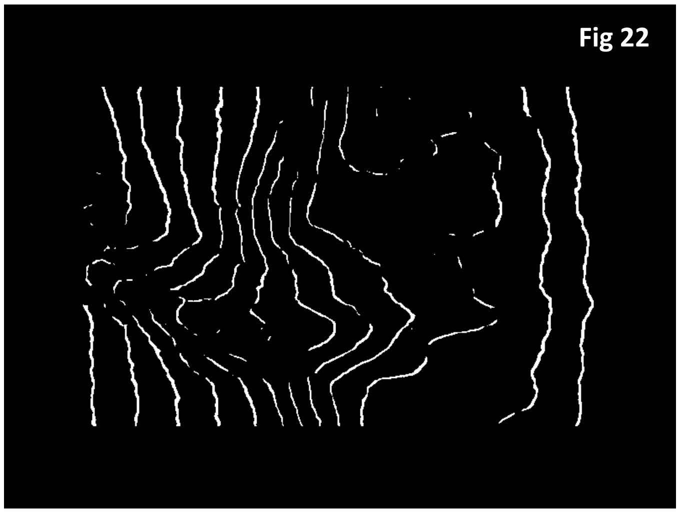


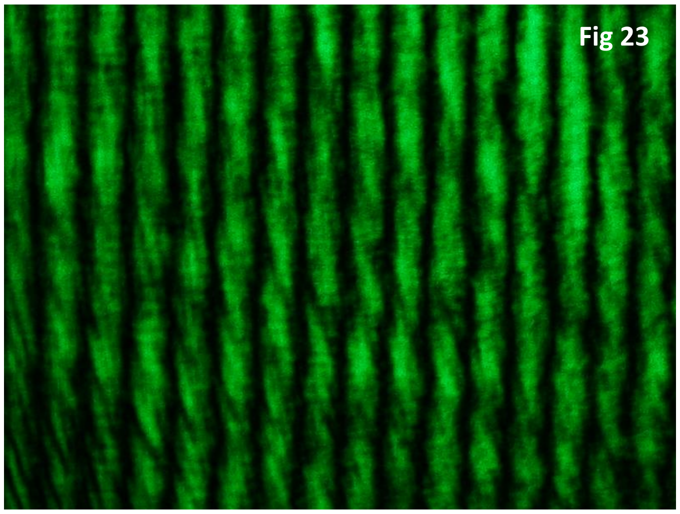
Fig 20







Figures 19, 20, 21, and 22. Interferometry images and model outputs for previously unseen complicated interferometry images.



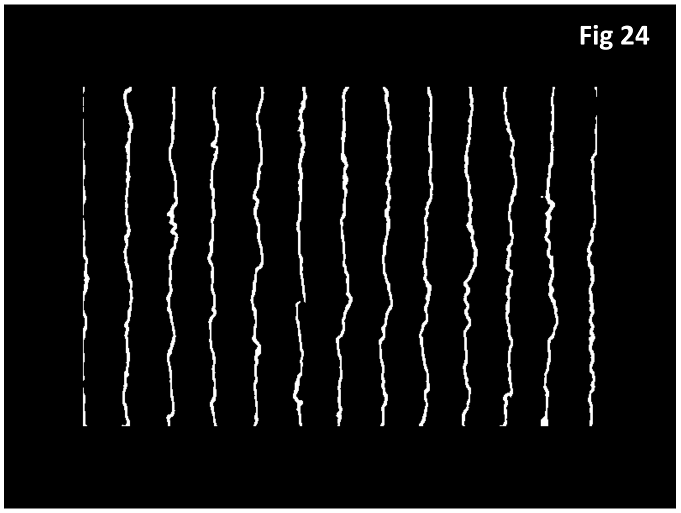


Figure 23, 24. Interferometry image and model output for previously unseen simple interferometry image.

The results indicate that a ML based interferometry fringe tracing algorithm could be used to help reduce human tracing times for complicated fringe traces in some images, but the outputs would still need to be corrected and finished by human tracing. The results of the model vary greatly depending on brightness, which can be partially corrected by brightening the image.

VII. Conclusion and Future Work

At LPS, a fast and accurate method for interferometry fringe tracing would be very valuable because it would reduce interferometry data analysis time, and greatly reduce the amount of human tracing required. ML image segmentation algorithms have shown promising results in similar image related tasks as interferometry fringe tracing; therefore, a ML based interferometry fringe tracing algorithm was developed. This algorithm performed adequately on given complicated interferometry fringe images, and well on simple interferometry fringe images. Ultimately, whether to use this algorithm to assist in interferometry image analysis is up to the person performing the analysis, but this algorithm could serve as a good tool for speeding up said analysis.

This algorithm would most likely benefit from a further increase in the dataset size. The current dataset size is 105 image pairs. There are very few limitations for training on a larger dataset other than the increased training time, scaling up the dataset could serve as a very easy way of boosting results. The brightening of images before training could also be a simple way to improve results. Since this model did not perform well on dull images after training, it does not make sense to keep them in the dataset as is. Brightening these images could also serve to increase the size of the dataset because the model would be presented with more orientations of fringe patterns in a way that better enables the model to learn from it.

VIII. References

[1] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz and D. Terzopoulos, "Image Segmentation Using Deep Learning: A Survey," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 7, pp. 3523-3542, 1 July 2022, doi: 10.1109/TPAMI.2021.3059968

[2] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift arXiv:1502.03167v3 [cs.LG]

[3] The Importance of Skip Connections in Biomedical Image Segmentation arXiv:1608.04117v2 [cs.CV]

[4] Alom MZ, Yakopcic C, Hasan M, Taha TM, Asari VK. Recurrent residual U-Net for medical image segmentation. J Med Imaging (Bellingham). 2019 Jan;6(1):014006. doi: 10.1117/1.JMI.6.1.014006. Epub 2019 Mar 27. PMID: 30944843; PMCID: PMC6435980.

[5] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[6] Attention U-Net: Learning Where to Look for the Pancreas arXiv:1804.03999v3 [cs.CV]

[7] A survey of loss functions for semantic segmentation arXiv:2006.14822v4 [eess.IV]

[8] Generative Adversarial Networks arXiv:1406.2661v1 [stat.ML]

[9] Image Data Augmentation for Deep Learning: A Survey arXiv:2204.08610v1 [cs.CV]

[10] Adam: A Method for Stochastic Optimization arXiv:1412.6980v9 [cs.LG]

[11] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32 (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[12] U-Net: Convolutional Networks for Biomedical Image Segmentation arXiv:1505.04597v1 [cs.CV]

[13] Zuo, C., Qian, J., Feng, S. et al. Deep learning in optical metrology: a review. Light Sci Appl 11, 39 (2022). https://doi.org/10.1038/s41377-022-00714-x