Does Machine Learning Technique Help the Physical Experiments?

Forward and Inverse Problem of the Michelson interferometer

Team member: Siyuan Song (cslogin: ssong26)

- **Introduction**

Michelson interferometer[1] is widely used in modern technology. The interference pattern can give us information about the wavelength of light. Since the data from the experiments is usually a photo, image analysis is very important.

The post-processing of the Michelson interferometer data includes two steps, [1] extract the physical parameters from the pattern; [2] generate the interference pattern based on the given input. In fact, there are some physical phenomena too complicated for people to understand. The machine learning approach provides a physical way for us to describe the world[4].

For the current project, we try to apply CNN in extracting the physical parameters from the images. In addition, we want to apply the dcGAN[2][3] in generating the interference pattern.

- **Methodology**

For the modeling part, we narrow down our problem to study the influence of the wavelength of light. Specifically, the wavelength range is taken to be (300 nm, 700 nm). Since the experimental output is usually center symmetric, we transform our image to the polar coordinate. In other words, the input dimension of the image is (r, θ).

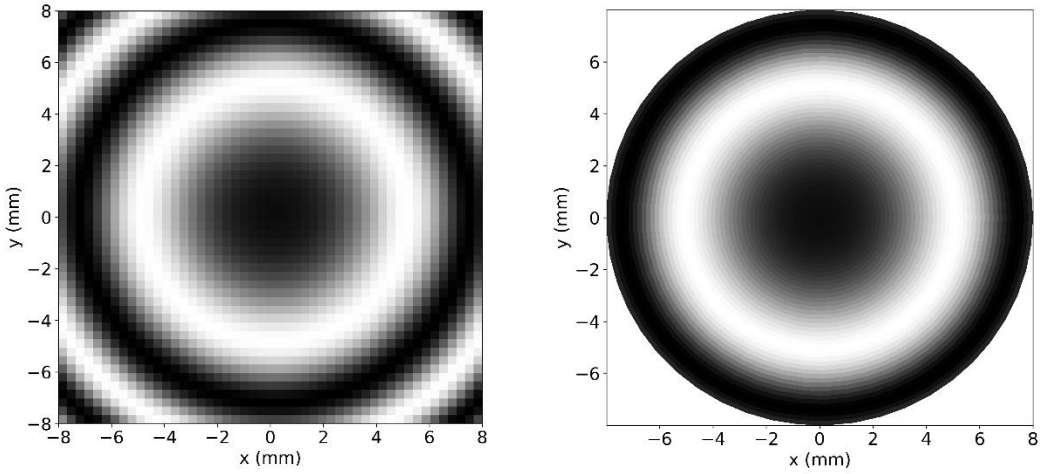

Figure 1 Inference Pattern for the wavelength 700 nm. (left) Input in the Cartesian coordinate (right) Input in the polar coordinate.

As shown in Figure 1, the right image is much clearer than the left image, although both of them have the same dpi. Our results show that the dcGAN with the right input works better than the dcGAN with the left input.

The basic physical parameter for the experimental setup is

TYPE	VALUE	PHYSICAL MEANING
SCREEN_LENGTH	8 mm	The size for the image screen
FOCAL LENGTH	500 mm	The focal length of the lens
RELATIVE DISTANCE	3 mm	The relative distance between two mirrors.

The first task is to extract the wavelength of the light from the image. We use the formula-based raw data to train the CNN network. The size of the image is (50,50). To simplify the expression, we normalized the label (i.e., the wavelength of the laser light) to be in the range of (0,1). The structure of the CNN is shown below.

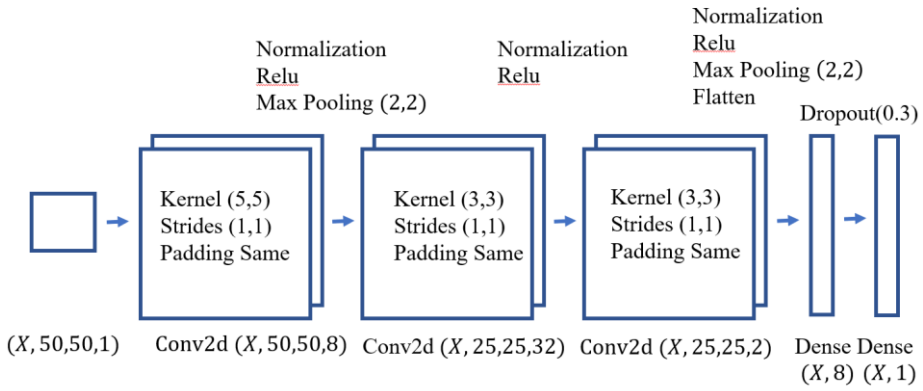

Figure 2 CNN Structure.

The CNN trained for 1000 epochs. Each epoch has 10000 images. The batch size is taken to be 1000. The "Adam" optimizer with the learning rate 0.001 is adopted. The loss function is obtained based on the mean squared error.

The second task is to generate the interference pattern with the random wavelength. We apply the deep convolutional generative adversarial network (i.e. dcGAN) to generate the pattern. The structures for the generator and the discriminator are shown below.

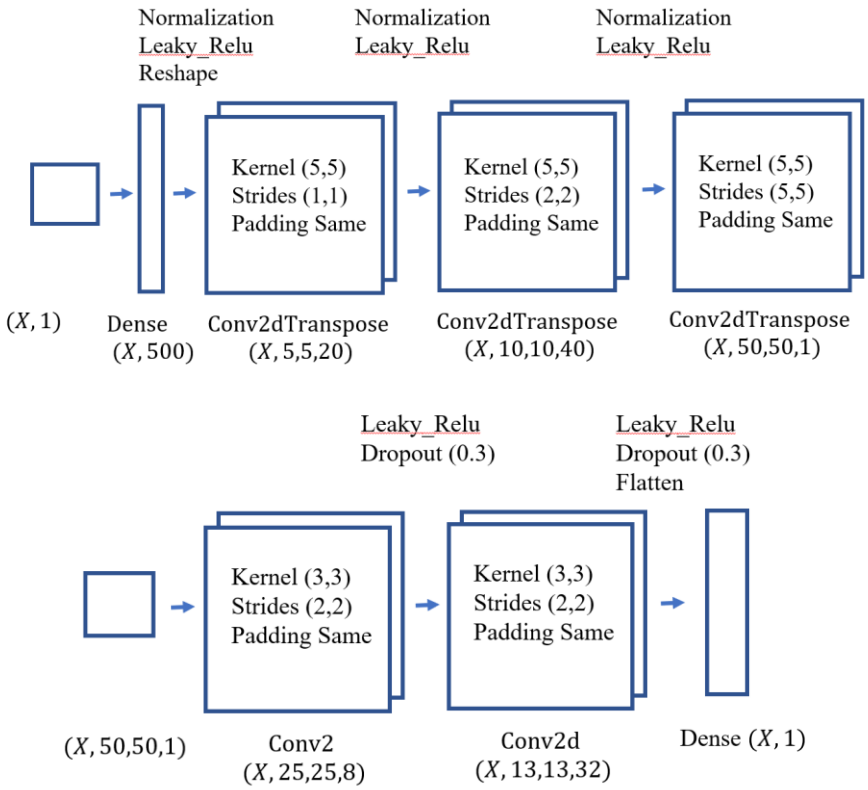

Figure 3 Structure of dcGAN. (Up). Generator (Down). Discriminator

The random input is taken to be one physical parameter in the range of (0,1). The output for the generator is the image of the size (50,50). The discriminator will read the image from the generator and try to identify whether it is the true image or not. The optimizer is taken to be the Adam with the learning rate of 0.0001. The loss function is the cross entropy for the true prediction and false prediction of the discriminator.

The third task is to combine the generator in the dcGAN model with the CNN model. The generator reads an input x and generate an image. The CNN reads the generated image and gives a label x'. The loss function is defined based on the mean squared error of the difference between the input x and the output x'. The Adam optimizer with the learning rate of 0.0001 is adopted.

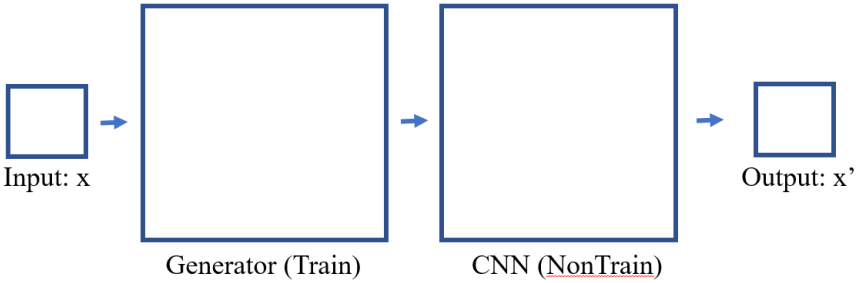

Figure 4 Structure of generator/CNN.

- **Results**

All the training is performed on my personal PC with the GPU acceleration (GTX 1060).

The CNN trained for about 900 seconds. The mean squared loss as a function of the time is given below

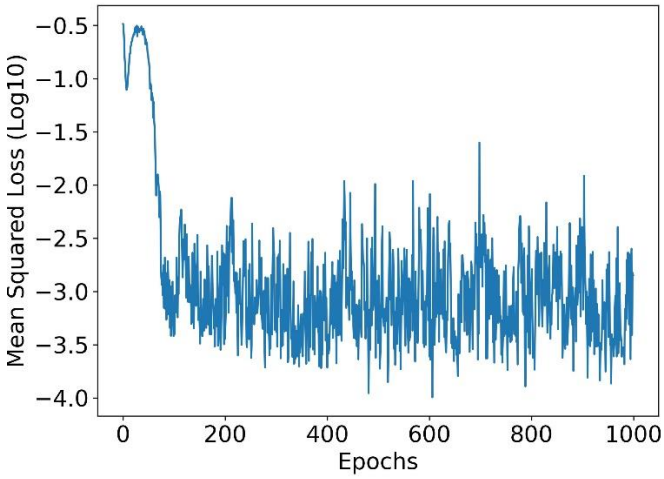

Figure 5 CNN: Mean squared error as a function of the epochs.

The mean squared loss has a strange reversal at the initial 40 epochs. Then the loss decreases significantly for the next 60 epochs. After 100 epochs, the loss suffers oscillation. As a comparison, the CNN model with the input in the Cartesian coordinate has the more stable loss function after 100 epochs.

The dcGAN trained for about 1200 seconds. To show the influence of the coordinate of the input date, we give the 16 output images for both the Cartesian coordinate and the polar coordinate.

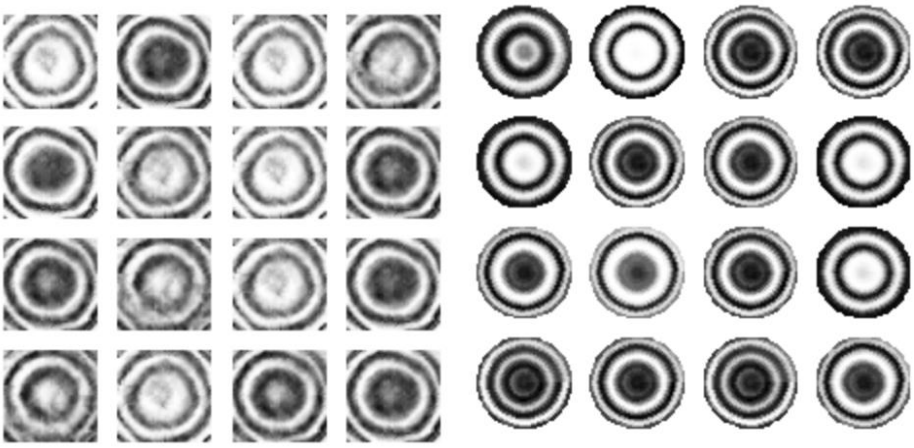

Figure 6 dcGAN generated image after training after 1000 epochs. (left) dcGAN trained with input in the Cartesian coordinate. (right) dcGAN trained with the input in the polar coordinate.

The dcGAN trained with the input in the polar coordinate has much more smooth output than the dcGAN trained with the input in the Cartesian coordinate. It means that for the input with the center symmetry, the CNN network can better capture the feature of image in the polar coordinate.

The combination of the generator and the CNN is much simpler. The generator trained for 100 epochs in about 200 seconds.

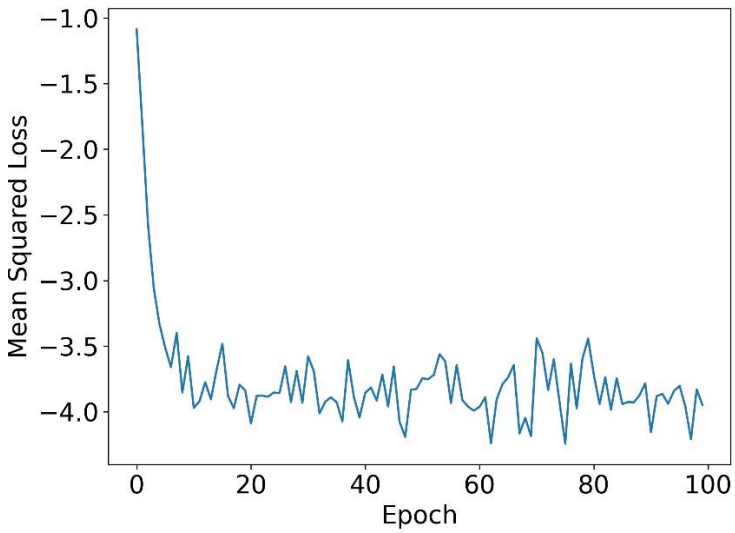

Figure 7 Mean squared loss for the generator/CNN model.

The mean squared error decays so fast for the first 10 epochs. Then, the loss will approach a constant value around 10^{-4}. The testing shows the generator works better than before. According to our imagination, it should generate the accurate image for any given input. However, the result is terrible.

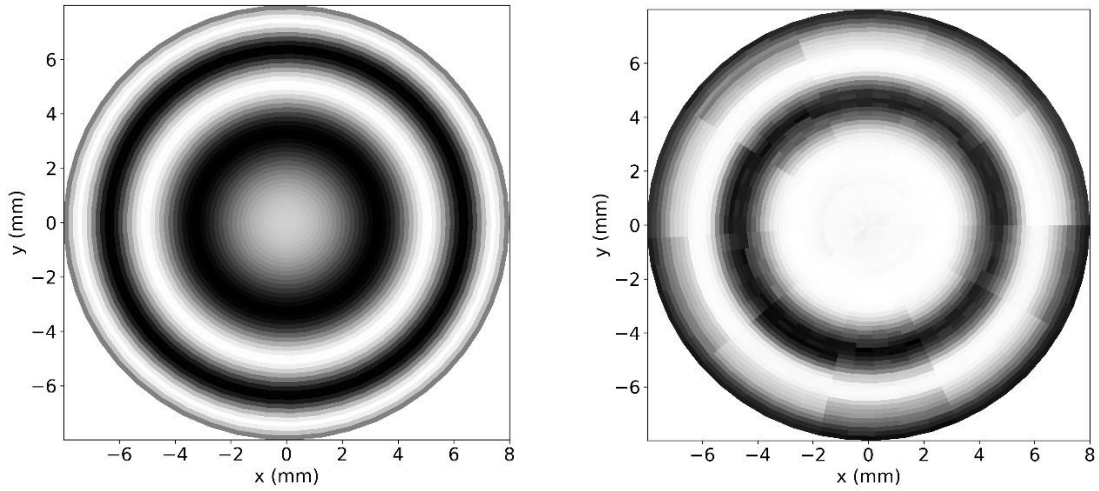

Figure 8 Interference pattern for 350 nm. (left) real image. (right) fake image from the generator (CNN reads 352.8 nm).

As shown in Figure 8, the image from the generator is different from the image of the physical law. However, the CNN will still extract 352.835144 nm from the fake image. It means that the unphysical image cheats the CNN to give the "correct" wavelength.

- **Challenges**

In other words, the CNN shows "good" results for both the true images and the fake images. Without the discriminator, we cannot tell whether the "reasonable" CNN output is from a successful experiment or from an illusion. In the real experiments, we could get a lot of noise. We usually don't know whether the experiment is successful or not. By putting our experimental data, we can definitely get some output. Do these outputs have physical meaning? Can we directly use them?

- **Reflection**
- How do you feel your project ultimately turned out? How did you do relative to your base/target/stretch goals?

This project is a benchmark test for my future experiments. The CNN can extract the parameter from the experimental results. The dcGAN can generate the experimental results without any physical details.

The generator can provide unreasonable images which may cheat the CNN. The CNN has too many trainable_parameters that multiple different inputs may give the same output. This property is bad for some scientific studies.

- Did your model work out the way you expected it to?

Yes, partially it works. The CNN model can extract the information and the dcGAN can generate artificial images. However, the CNN model cannot tell which is a physical input and which is a fake input.

- How did your approach change over time? What kind of pivots did you make, if any?

At the beginning, I was working on the "Digital Image Correlation". However, the random speckle is too difficult to deal with. Then I was working on the image-analysis of the interference pattern.

The CNN works well all the time. However, the dcGAN worked so badly at the beginning. The output image from the generator is not symmetric! With the consideration of the input image, I transformed the input data from Cartesian coordinate to the polar coordinate. Then, the dcGAN works much better.

I think we need to find out the major symmetry of our data before we do any machine learning. By adding a reorganization layer, we can significantly reduce the number of trainable parameters in our model and accelerate the learning process.

- Would you have done differently if you could do your project over again?

Yes, I will do it differently. I want to add a discriminator to my CNN model that the CNN model can tell whether or not the input is physical.

- What do you think you can further improve on if you had more time?

I will think about the re-organization layer. Currently, I use the re-organization layer to transform the Cartesian coordinate to the polar coordinate. However, most of the other images have more complicated structure. This "re-organization process" should be done by the machine learning itself, not by my personal experience.

In addition, I will develop a discriminator for my CNN model. My CNN model should know its input.

- What are your biggest takeaways from this project/what did you learn?

The data preprocessing is very important. We have to think about the structures of the input data and preprocess it to the well-organized format for the model to train.

In addition, we cannot use CNN blindly. We should develop extra discriminators help us to decide whether the input data is physical or not[5].

- **Acknowledge:**

We would like to thank Professor Chen Sun and TA Jacob Yu for their support guidance throughout the course of this project.

- **Reference:**

[1] EW, M.A.M., 1887. The relative motion of the Earth and the luminiferous aether. The American Journal of Science. Third Series, 34, pp.333-345.

[2] https://www.tensorflow.org/tutorials/generative/dcgan

[3] Radford, A., Metz, L. and Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

[4] Casert, C., Mills, K., Vieijra, T., Ryckebusch, J. and Tamblyn, I., 2020. Optical lattice experiments at unobserved conditions and scales through generative adversarial deep learning. arXiv preprint arXiv:2002.07055

[5] De Bézenac, E., Pajot, A. and Gallinari, P., 2019. Deep learning for physical processes: Incorporating prior scientific knowledge. Journal of Statistical Mechanics: Theory and Experiment, 2019(12), p.124009.