A Fast Fourier Transform of the Ionization Fraction of Neutral Hydrogen.

Sigfredo A. Saravia
The Department of Physics and Astronomy,
Brown University, Providence, Rhode Island 02912 USA
sig.saravia616@gmail.com

(Dated: September 19, 2022)

I. INTRODUCTION

As part of the summer 2021 HERA/CHAMP program at the University of Pennsylvania, I was tasked with building and training a convolutional neural network that could make prediction about the 21cm signal coming from the Epoch of Reionization period of the early Universe. The Epoch of Reionization period occurred approximately between 200 Million years and 1 Billion years after the Big Bang. This period in the Universe is particularly interesting because it is when star formation began. The yellow rectangle in the 21cm Light-cone in FIG. 1 highlights the Epoch of Reionization period of interest. It also shows the signal is redshifted. The signal coming from about 200 Million years after the Big Bang has redshifts as high as 16 while the signal from about 1 Billion years is consistent with redshifts of 6.

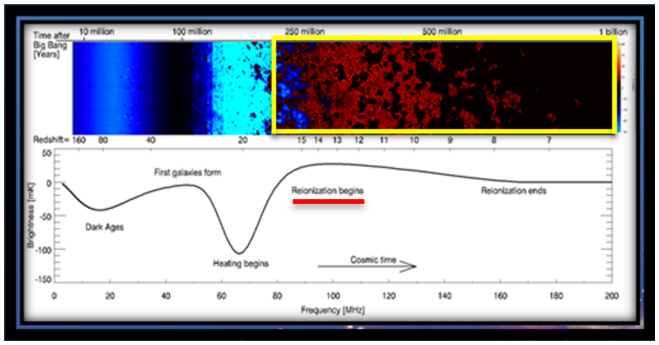

FIG. 1. 21cm Light-cone.

I trained the convolutional neural network using a Toy Model of the ionization fraction of neutral hydrogen. FIG. 2 shows the 128x128 pixel images in the Toy Model. The first images in the figure is a 2D slice of the light-cone in FIG. 1 at a particular redshift. The last images is the ionization fraction of neutral hydrogen with added background noise that is similar to what HERA would encounter as it scans the night sky. The Toy Model consists of 10,000 ionization fraction of neutral hydrogen images. 8,000 of those images were used to train the convolutional neural network and 2,000 images were used to test the convolutional neural network's prediction capabilities. For this particular job the network was trained for a total of 400 epochs. The network performed well with the training and predicting average absolute errors resulting under 1%.

In the summer of 2022, I participated in the Research Experiences for Non-Traditional Undergraduates (RENTU) program at Brown University. I was asked to perform a fast Fourier transform on the Toy Model images of the ionization fraction of neutral hydrogen. A fast Fourier transform coverts the pixel of an image into frequencies called Fourier coefficients. I applied two different filters on the fast Fourier transformed Fourier coefficients to understand which frequencies coefficients could be blocked and still have enough information to accurately recreate the original pixel images. Then, I used the convolutional neural network that I used to train the ionization fraction of neutral hydrogen images to train the filtered fast Fourier transform recreated images. Finally, I compared the results of the average absolute errors for the filtered fast Fourier transform recreated images to the average absolute errors of the ionization fraction of neutral hydrogen.

The goal of this research was to provide insight on which frequency coefficients could be discarded while still having enough data to accurately recreate the original image. Furthermore, this research could give us a glimpse into how the Epoch of Reionization period gave birth to the first stars and galaxies. HERA plans to built a 3D model of the Epoch of Reionization period using 2D ionization fractions of neutral hydrogen images. I began training and testing 2D ionization fractions of neutral hydrogen images using the convolutional neural network that I built in the summer of 2021. I began with a noiseless Toy Model made up of 32x32 pixel images of the ionization fraction of neutral hydrogen. Afterwards, I increased the image size to 128x128 pixel images and I added accurate background noise. Now, I've performed a fast Fourier transform on the 128x128 pixel images and applied filters to the Fourier coefficients of the images. I trained and tested these images using the same convolutional neural network that used to train and test the original ionization fraction of neutral hydrogen images. Finally, I compared the results.

II. A FAST FOURIER TRANSFORM.

A fast Fourier transform coverts the pixel of the image into frequencies called Fourier coefficients. FIG. 3 shows the process of performing a fast Fourier transform on an image of the ionization fraction of neutral hydrogen. First, the pixels of the original image are con-

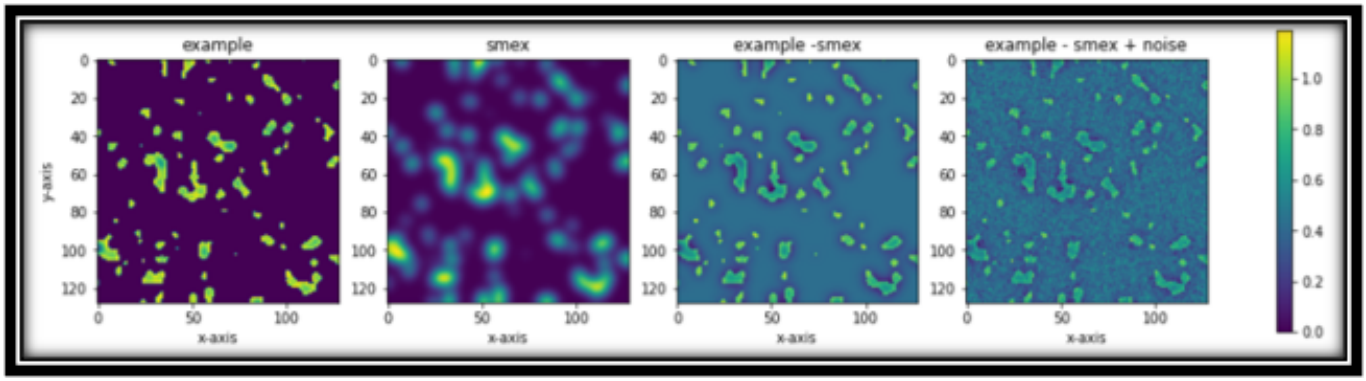

FIG. 2. Toy Model Slice of Ionization Fraction of HI.

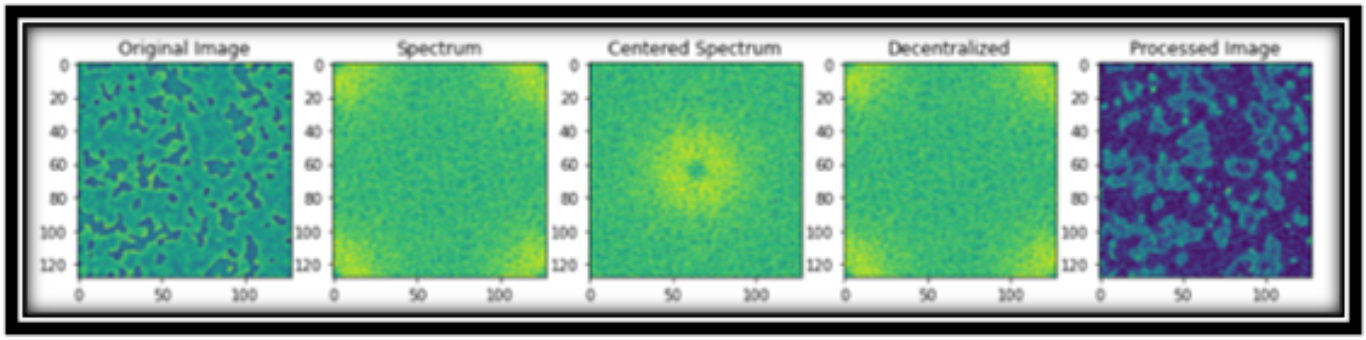

FIG. 3. Fast Fourier Transform.

verted into Fourier coefficients in the spectrum image. The Fourier coefficients spread from lower frequencies in the corner inwards to the center into higher frequencies. Then, I utilized the rotational and translation properties of a Fourier transform to center the zero-frequencies. The frequency coefficients are centered because it makes it easier to apply the different circular filters I used to block certain frequencies. Afterwards the image is filtered, an inverse fast Fourier transform is performed on the filtered frequency coefficients to return them back into their spectrum frequency mode. Finally, the original image is recreated with the information from the filtered frequency coefficients. After this process, I ran the newly filtered fast Fourier transformed 128x128 Toy Model images through the convolutional neural network to test whether the filtered frequency coefficients contain enough information for the network to execute learning and predicting capabilities to within an acceptable error margin.

I investigated two types of filters. The first of the filters was a Low Pass Filter that can be seen in FIG. 4. A Low Pass Filter blocks high frequency Fourier coefficients while allowing low frequency Fourier coefficients to penetrate the filter. More specifically, this is a 20 pixel radius Low Pass Filter. As Equation (1) describes, any Fourier coefficient greater than the radius of the Low Pass Filter is zeroed out and blocked by the filter. The Fourier coefficients that are within the radius penetrate the filter and are used as data to recreate the original image. The recreated image appears to focus on the location of the islands of ionization fraction of neutral hydrogen. This suggests that the filtered frequency coefficients contain data on the concentrated areas of mass.

$$\mathrm{G}(x,y) = \begin{cases} 1 & \text{if } F(x,y) \leq \mathrm{G}_0 \\ 0 & \text{if } F(x,y) \geq \mathrm{G}_0 \end{cases} \tag{1}$$

I applied 5 different Low Pass Filter sizes to the Fourier coefficients of the fast Fourier transform images. Then, I trained and tested the convolutional neural network using the filtered Toy Model images. I began with a 20 Low Pass Filter and increased the filter size by increments of 10 until I reached the 90 Low Pass Filter. The 90 Low Pass Filter has a greater radius than the images therefore it is like not applying a filter at all. I trained and tested these filter Toy Model for 100 epoch each. Moreover, I logged the average absolute errors for each increasing Low Pass Filter in increments of 10 epoch until I reach 100 epoch to study the prediction capabilities as the filters decreased in size. additionally, a Low Pass Filter can help reduce the noise level within the pixels of the images. If the Low Pass Filter images are successfully trained and the convolutional neural network can accurately make prediction then, it is an indication that the most important information is located in the lower frequencies of the Fourier coefficients.

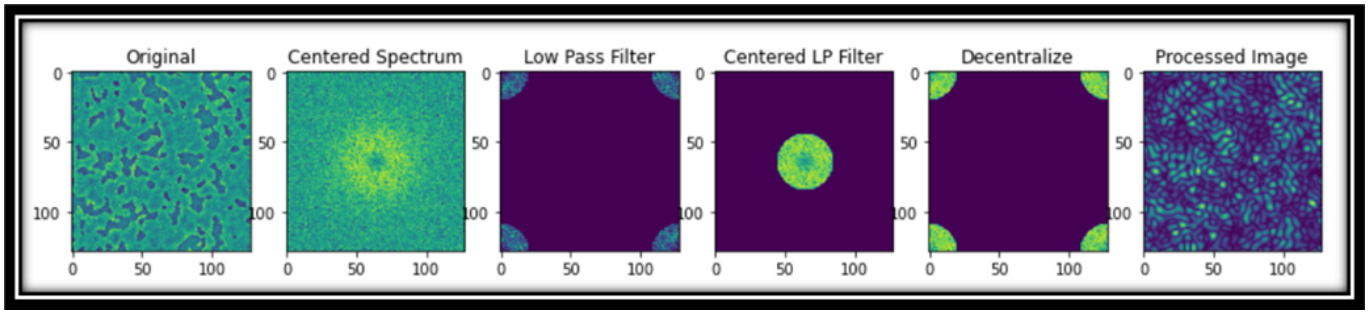

FIG. 4. Low Pass Filter.

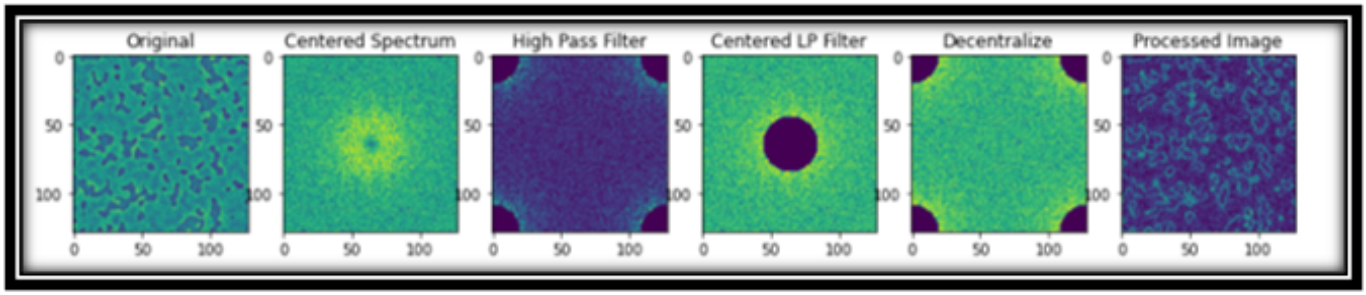

FIG. 5. High Pass Filter.

$$G(x,y) = \begin{cases} 0 & \text{if } F(x,y) \leq G_0 \\ 1 & \text{if } F(x,y) \geq G_0 \end{cases} \tag{2}$$

I also investigated how a High Pass Filter, which can be seen in FIG. 5, could affect the recreation of the original image. Unlike the Low Pass Filter, a High Pass Filter blocks low frequency Fourier coefficients while allowing high frequency Fourier coefficients to pass through the filter. The piece-wise function is Equation (2) shows that the filter blocks all Fourier coefficients within the suggested radius input by zeroing those Fourier coefficients while allowing Fourier coefficients outside that radius to penetrate the filter. For example FIG. 5, demonstrates a 20 pixel radius High Pass Filter. As you can see, the filter blocks most of the centered lower frequency Fourier coefficients resulting in a recreated image that is sharper than the Low Pass Filter image. This suggests that the high frequency Fourier coefficients contain information regarding the outline of the islands in the image while the lower frequency coefficients contain information about the location of the islands in the image. If these types of filter images are successfully trained and tested then, it is an indication that the important information is located in the higher frequency coefficients.

III. CONVOLUTIONAL NEURAL NETWORK.

A convolutional neural network is a machine learning algorithm. The algorithm learns from given data and

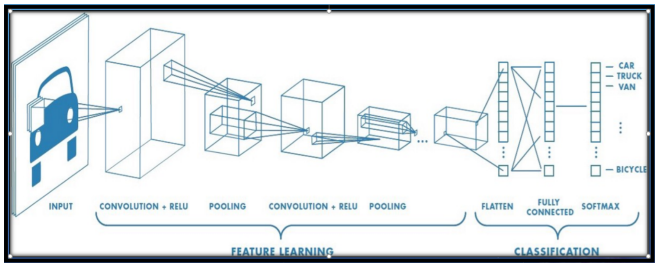

FIG. 6. Convolutional Neural Network.

makes predictions about future unseen data. FIG. 6 is an example of how a convolutional neural network can learn and make predictions. First, a convolutional layers scans the 2D image using a specified *nxn* filter. Then, the network makes use of a MaxPooling command which puts an emphasize on the important features of the image. This process repeats itself multiple times to properly scan the image. After the image is properly scanned, the Flatten Layer takes the image array and converts it into an information vector. This vector passes down the information to the Dense Layers. Dense Layers are fully connected inputs that preform most of the learning process. Finally, the last dense layer makes a prediction based on the information it has learned. In the case of FIG. 6, the car is identified from other types of transportation vehicles.

The convolutional neural network in FIG 7 is the network that I used to train all of the different types of Toy Models I've built. The first convolutional layer starts of with 64 inputs that associate a value to each pixel. The kernel size command is the 3x3 filter used to scan the

pixels of the image. The 'relu' activation function is a piece-wise function that keeps the values between 0 and 1. The input shape dictates the size of the images and for the purpose of this project the images are 128x128 pixel images. Padding 'same' ensures all the pixels are scanned by the filter.

```
# Making a model.
model = Sequential()

# Added Layers.
model.add(Conv2D(64, kernel_size=3, activation="relu", input_shape=(nx, ny, 1), padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dense(32, activation = 'relu'))
model.add(Conv2D(32, kernel_size = 3, activation = 'relu', padding = 'valid'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(16, kernel_size = 3, activation = 'relu', padding = 'valid'))
model.add(Conv2D(8, kernel_size = 3, activation = 'relu', padding ='same'))
model.add(Conv2D(4, kernel_size = 3, activation = 'relu', padding = 'same'))
model.add(Conv2D(2, kernel_size = 1, activation = 'relu', padding = 'same'))
model.add(Flatten())

model.add(Dense(16, activation = 'relu'))
model.add(Dense(8, activation = 'relu'))
model.add(Dense(4, activation = 'relu'))
model.add(Dense(2, activation = 'relu'))

model.add(Dense(1,activation="linear"))
```

FIG. 7. Convolutional Neural Network.

After the first convolution of the images, I used the MaxPooling command to emphasize the important features of the original images. I wanted to make sure I captures as much information as possible in the first convolution. Then, the convolved images are passed down to the next convolution layer. I used a Dense Layer before the second convolutional layer to help the inputs of the layer to deal with the large set of incoming parameters. The 32 inputs scan the convolved images with a 3x3 filter however, I did not pad the images for this layer. Instead, I ignored the pixels on the edge. This helps with the reduction of the image size. After the convolution, I used the MaxPooling command one more time. The scanning process repeats itself as the inputs decrease and images approach the Flatten Layer. The final 2 input convolution uses a 1x1 filter to ensure the final convolved images are scanned thoroughly to gather as much information as possible before converting the 2D image array into a vector. The final dense layer makes the prediction while also converting the results into an easy to understand linear function.

IV. RESULTS.

I logged the results for 8 different Low Pass Filters. I trained these filtered image sets for 100 Epoch at 107s Epoch each. This resulted in a total training time of 2.97hrs for each training set. I began with a 20 pixel radius Low Pass Filter because a 10 radius Low Pass filter did not collect enough information for the convolutional neural network to train. As the legend in FIG. 8 displays, I started with a 20 pixel radius Low Pass Filter and I increased the radius of the filter in increments of 10 up to the 90 Low Pass Filter. The 20 Low Pass Filter blocks all frequency Fourier coefficients outside of a 20 pixel radius. The first 10 training Epochs resulted in an average absolute error of 0.0980 (9.80%). This is the third highest start for an average absolute training error of a Low Pass Filter. After 100 Epoch, the network committed fewer errors with a 0.0185 (1.85%) average absolute training error. This filtered set trained well but was not the best.

The 90 radius Low Pass Filter was expected to perform well however, it was not the case. It was expected to perform well because the filter's radius is greater than the radius of the images which allows all Fourier coefficients pass through the filter. Instead, the average absolute training error for the first 10 Epochs was the second highest training error at 0.1050 (10.5%). The average absolute training error after 100 Epoch was 0.0312 (3.12%). This was the highest average absolute error after training the network for 100 epoch. A possible reason for this higher error after 100 Epoch could be the result of noise that the convolutional neural network encountered during the training process.

The best performing training image set was the 40 radius Low Pass Filter training set. This set of filtered images has the best starting average absolute training error at 0.0556 (5.56%) for 10 Epoch. I believe this filter performed best because it allowed the right combination of low and high frequency Fourier coefficients to penetrate the filter. After 100 epochs, the 40 radius Low Pass Filter also performed best with an average absolute training error of 0.00118 (0.118%). The training process suggested that the lower frequency Fourier coefficients could poses the important information.

The validation results for the Low Pass Filters in FIG.9represent the prediction capability of the convolutional neural network. The network was presented with 2,000 images it did not see in the training process and it was asked to classify the images. Even though the 20 radius pixel Low Pass Filter performed well in the training process it did not translate into the validation process. The average absolute validation error for 10 Epoch was 0.0999 (9.99%). However, the average absolute validation error for 100 epoch only reach 0.0462 (4.62%). This was the second highest validation error after a 100 Epoch training. The reason for this high error could be a result of blocking to many low frequency Fourier coefficients.

Moreover, the 90 radius Low Pass Filter average absolute validation error began with an error 0.118 (11.8%). After 100 Epoch the average absolute validation error was 0.0501 (5.01%). This was the highest validation error for all the Low Pass Filters. I believe the higher error could be caused by the high frequency Fourier coefficients. The 40 radius Low Pass Filter performed the best during the training process and the validation process. It started with the best average absolute validation error for all the Low Pass Filters at 0.0582 (5.82%) for the first 10 Epochs. The average absolute validation error for 200 epoch was 0.0118 (1.18%). I believe this set of images trained well because it had a good balance of low and high frequency Fourier coefficients.

To test my theory about low frequency Fourier coeffi-

cients I experimented with High Pass Filters. The High Pass Filters blocked frequency Fourier coefficients inside the selected radius area. As the legend in FIG. 10 shows, I trained 7 High Pass Filters. I began with a zero High Pass filter. It allowed all Fourier coefficients to penetrate the filter. Then, I increased the filter sizes up to a 20 radius High Pass Filter. The convolutional neural network became increasingly difficult to train after a 20 radius High Pass Filter. The zero High Pass Filter set of images performed well with the second best starting average absolute training error of 0.1045 (10.45%). The average absolute training error after 100 epoch was 0.0237 (2.37%). The best performing filter was the 5 radius High Pass Filter. It had the best starting average absolute training error at 0.0632 (6.32%) This could be because the center of the fast Fourier transform images have a small amount of high frequencies and blocking them can help the network perform better. After 100 Epoch, this filtered set reach an average absolute training error of 0.0168 (1.68%).

On the other hand, the 20 radius High Pass Filter struggled to obtain any useful information to train the convolutional neural network. The average absolute training error was the highest at 0.3231 (32.31%). The network did not perform much better over time. After 100 Epoch, the convolutional neural network committed an average absolute training error of 0.2581 (25.81%). This was the highest error for the entire High Pass Filter training set. As you can see, the convolutional neural network struggled to train as the filter increases in size. This data supports my theory that the most important information on these filtered fast Fourier transformed images are the low frequency Fourier coefficients.

As FIG. 11 shows, the results for the High Pass Filter validation error is very similar to that of the training results for the High Pass Filters. Surprisingly, the best performing High Pass Filter is the 5 radius High Pass Filter. It began with an average absolute error at 0.068 (6.80%) for 10 Epoch. However, at 100 Epoch, the validation was still committing a 0.0248 (2.48%). The 15 High Pass Filter was very chaotic during the training and validation processes while the 20 radius High Pass Filter performed unfavorable. The filter failed to collect enough information from the Fourier coefficients to train well. This translated into the validation process by having errors go from 0.3283 (32.83%) to 0.2683 (2683%) in 100 Epoch. This further re-enforces my theory about low frequency Fourier coefficients. It appears that the low frequency coefficients contain the majority of the important information in these fast Fourier transformed images.

I decided to train a couple of Low Pass Filters for more than 100 Epoch to investigate whether the convolutional neural network's learning and prediction capabilities increased over a longer period of time. First, I ran the 90 radius Low Pass Filter for 200 Epoch. The curve for the 90 radius Low Pass Filter can be seen the right of FIG. 12. It shows that the learning and predicting capabilities are chaotic for the first 100 Epoch but level out after 100 Epoch. The training average absolute error after 200 Epoch was 0.0072 (0.72%). This was a great result but the validation error was 0.0368 (3.68%). This means that while the networked learned from the feature overtime those features did not help to increase the prediction capabilities.

Furthermore, I ran the 40 radius Low Pass Filter image set for 360 Epoch. I wanted to investigate how many Epochs does it take to stabilize the learning and prediction capabilities of the convolutional neural network. The graph on the left of FIG. 12 shows that the network continues to learn but the prediction capabilities plateaus at about 100 Epoch. The average absolute validation error after 360 Epoch was 0.0175 (1.75%) while the average absolute validation error for 100 Epoch was 0.0118 (1.118%). Similar to the 90 radius Low Pass Filter, the network continued to learn overtime however, it did not help to increase the prediction capabilities.

Finally, I compare the results for the ionization fraction of neutral hydrogen Toy Model images with that of the fast Fourier transform filtered images. I trained the ionization fraction of neutral hydrogen images for 500 Epoch which resulted in an average absolute training error of 0.0035 (0.35%). Moreover, the average absolute validation error was 0.0060 (0.60%). The total training time was 14.86hr. On the contrary, I trained the fast Fourier transformed filtered images for 100 Epoch and a total training time of 2.97hr. The results for the average absolute training error of the 40 radius Low Pass Filter was 0.118 (1.18%) and The average absolute validation error was at 0.0158 (1.58%). We are giving up a 1% error for about an 11hr reduction in training time.

V. CONCLUSION

The convolutional neural network trained and predicted better under the Low Pass Filter images than the High Pass Filter images. This data suggests that the important frequencies in the fast Fourier transform images are the lower Fourier coefficient frequencies. Furthermore, the convolutional neural network trained about 11hrs faster on the fast Fourier transform images than it did with the original ionization fraction of neutral hydrogen images. However, this large reduction in training time comes at a cost of about a 1% error increase.

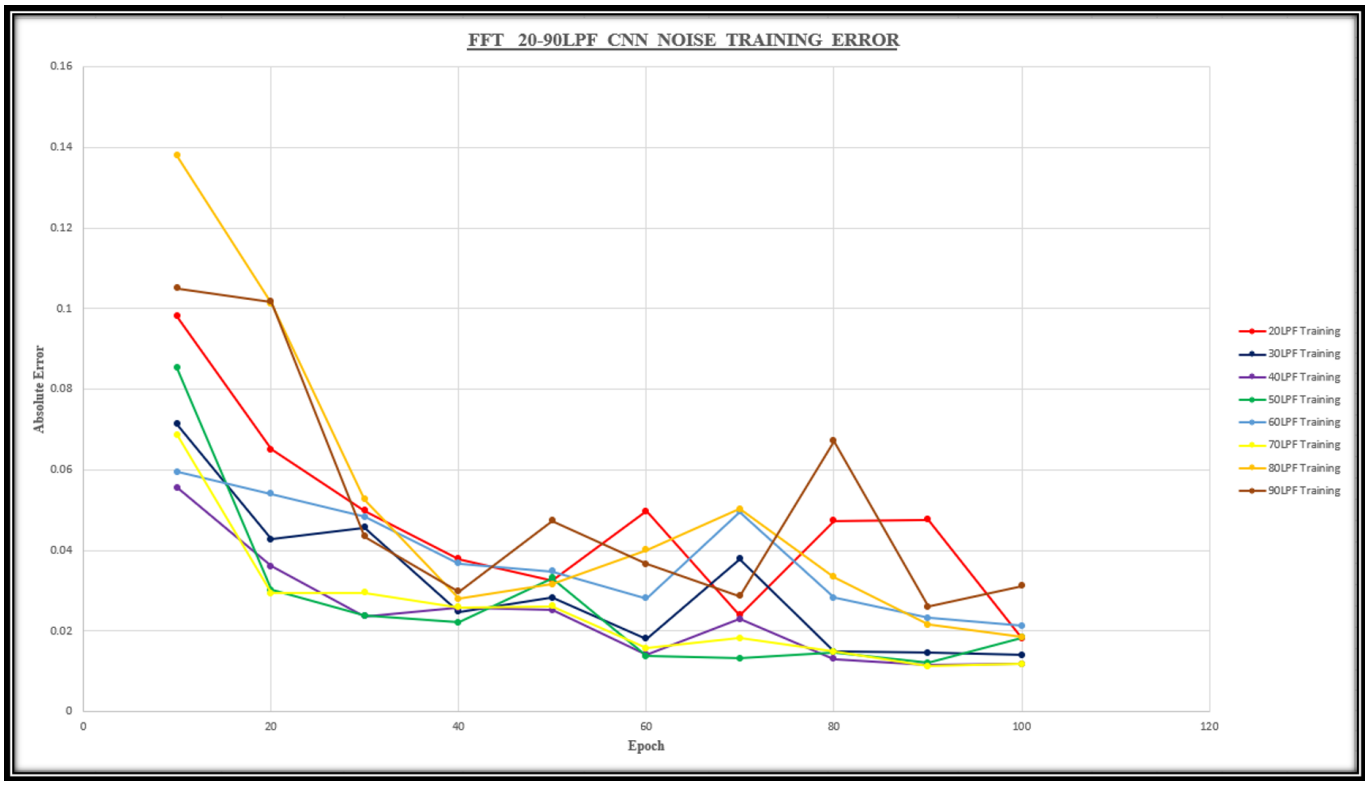

FIG. 8. 20-90 Low Pass Filter Training Errors.

h!

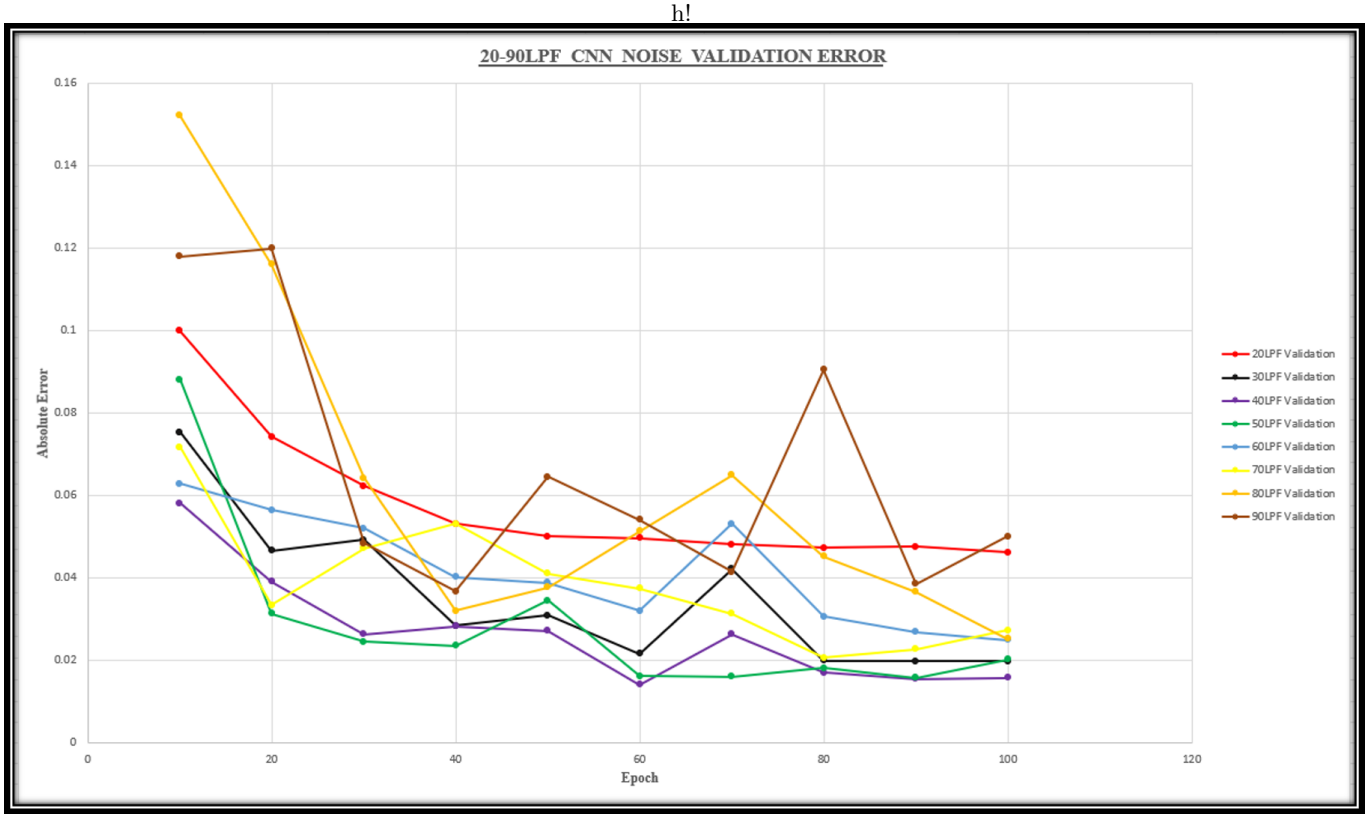

FIG. 9. 20-90 Low Pass Filter Validation Errors.

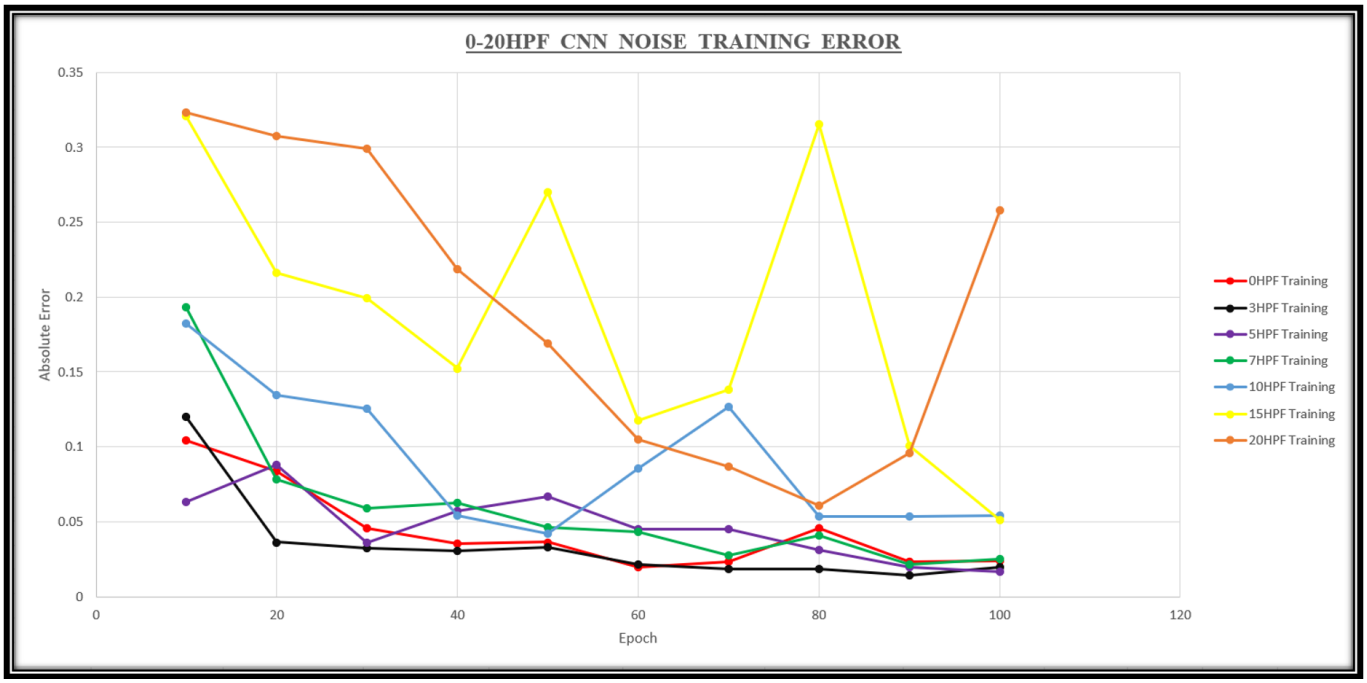

FIG. 10. 0-20 High Pass Filter Training Errors.

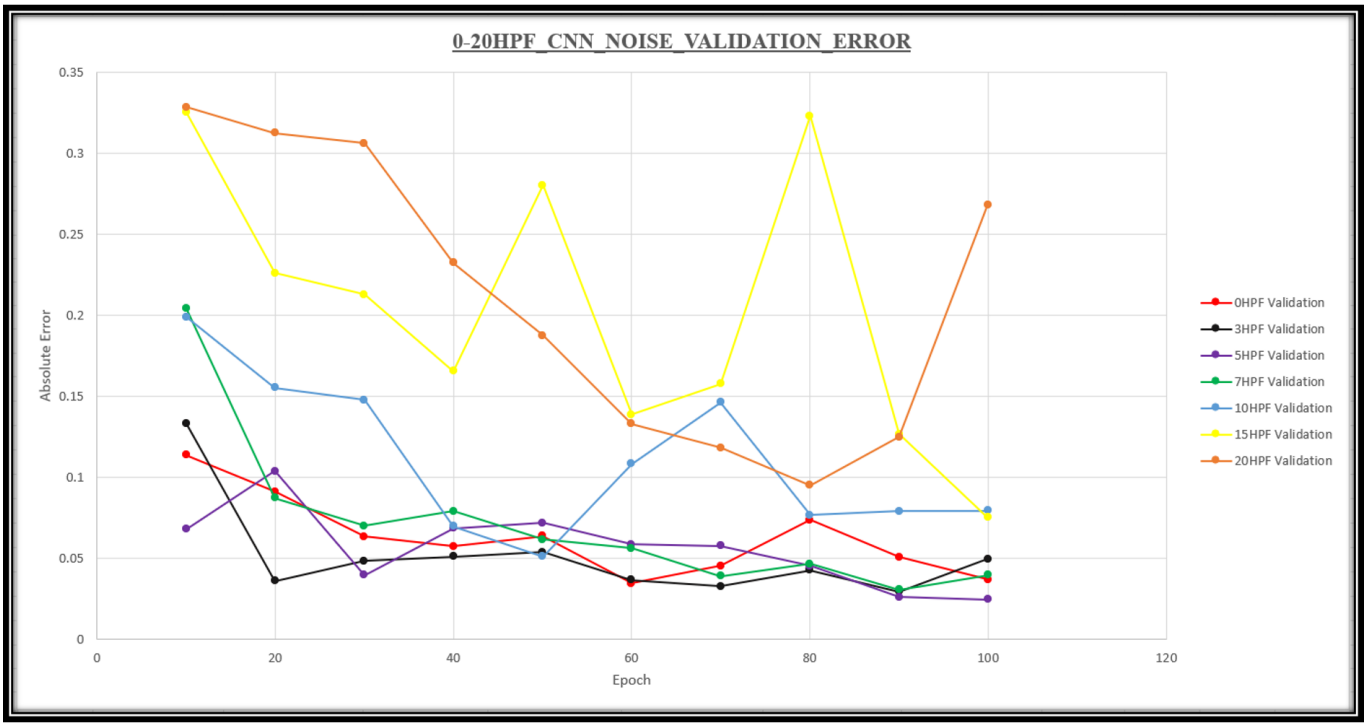

FIG. 11. 0-20 High Pass Filter Validation Errors.

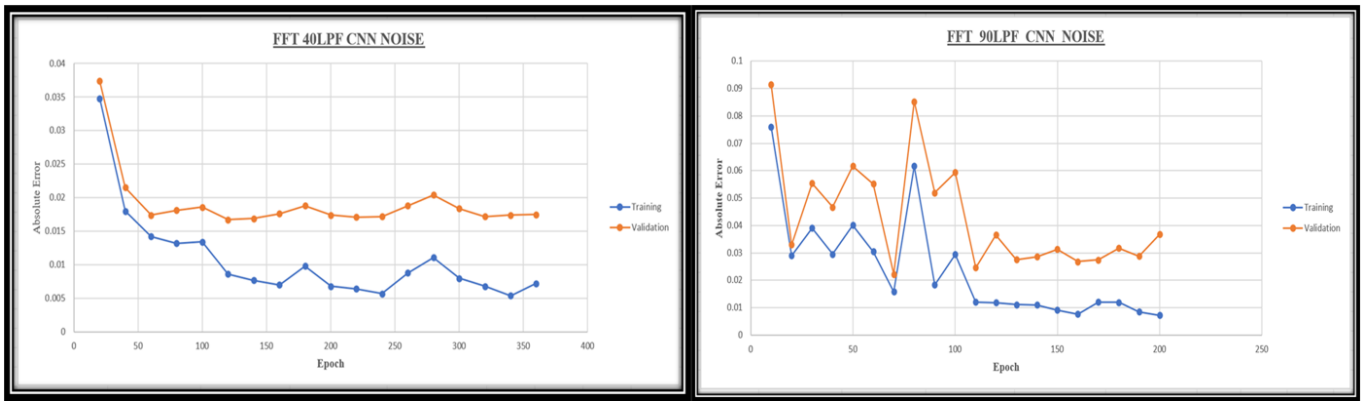

FIG. 12. Low Pass Filters Greater Than 100 Epoch Results.

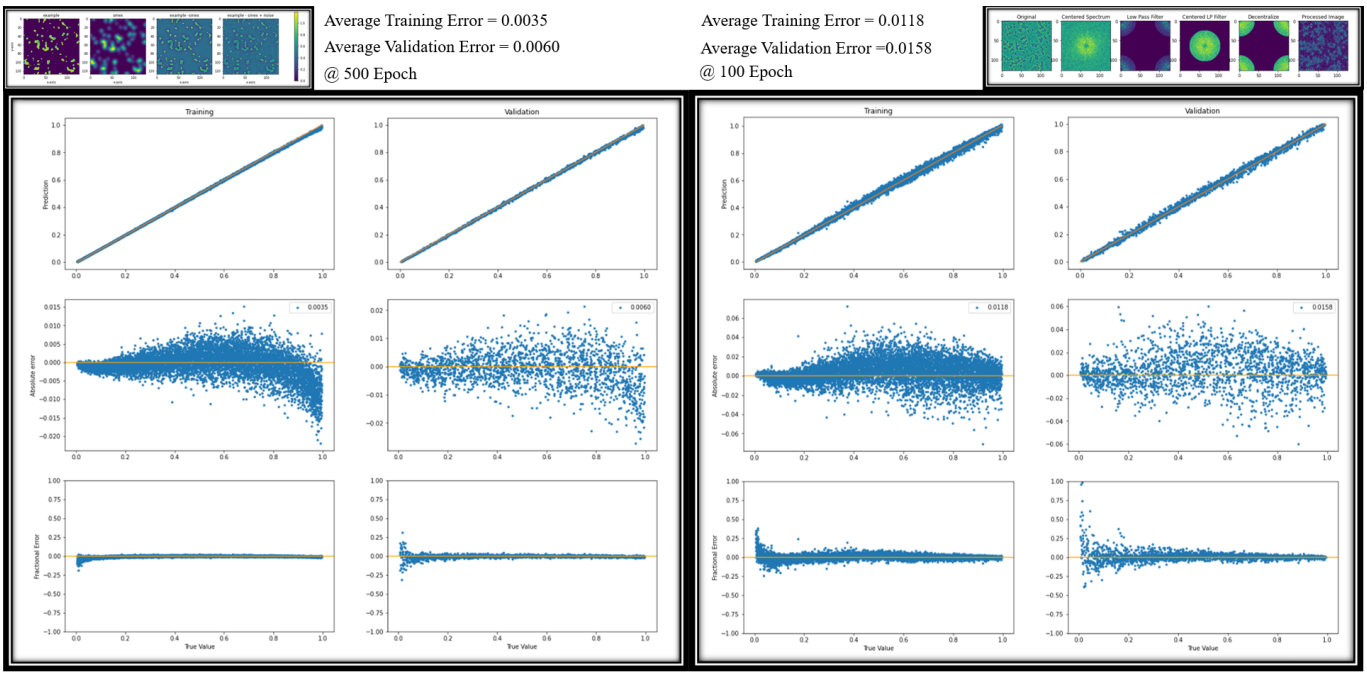

FIG. 13. CNN Result Comparison.