Ensemble Deep Learning: Neural Networks

In connection with the Research Experience for Undergraduates (REU) Program

Support for the program has been provided by the National Science Foundation (NSF) through REU Award Number DMS - 2050754.

Rhiannon Hicks – Program Correspondent
Dr. Mihhail Berezovski – Program Director
Embry-Riddle Aeronautical University

Dr. Jesse Adams and Dr. Margaret Lund – Senior Scientists
Nevada National Security Site

Abstract

Data science is a broad field that encompasses studying and extracting meaning from data, and a subset of this category is machine learning. Machine learning is the tactic of using various techniques to build computer models that are able to learn by themselves using data. A neural network is a type of machine learning that is modeled after how the human brain learns, and takes a given input and predicts an output from this value. Neural networks are extremely applicable in the field of data science, but also come with one large drawback: no method exists for quantifying the amount of error in a neural networks output prediction. This begs the question of what methods we can attempt to implement or create to effectively measure the uncertainty in neural networks so that we can better comprehend and dissect the meaning of their predictions. The Research Experience for Undergraduates program students will attempt to create an ensemble neural network with a valid uncertainty quantification method. Throughout this paper we will explore the complete method that the REU program has attempted in order to combat this problem.

Introduction

Embry-Riddle Aeronautical University's REU program is partnering with the Nevada National Security Site (NNSS) with funding from the National Science Foundation (NSF) in order to develop methods for quantifying and understanding uncertainty in the accuracy of neural network algorithms. The REU students will use publicly available datasets to develop a neural network, and will also create an ensemble with some uncertainty quantification (UQ) approach, such as Monte Carlo (MC) dropout, in order to better comprehend the results of the neural network and the uncertainty factor of their accuracy (Adams & Lund, 2022).

Background

The NNSS, formerly known as the Nevada Test Site (NTS), is a part of the United States Department of Energy (DOE). Their complex is larger than the state of Rhode Island and focuses mainly on Defense Nuclear Nonproliferation and Stewardship Science & Experimentation. Although there are many divisions within the NNSS, REU students will be working directly with the Weapons' Program, whose goal is to ensure the United States' nuclear stockpile remains safe, secure, and effective. In order to do this, they conduct classified nuclear experiments, high-tech computer modeling, and detailed engineering analysis according to their website, (*About the NNSS,* n.d.). At the U1a underground facility high energy tests are conducted so that radiographers can capture high resolution images of the explosions and interpret if things are working the way that they should be. This is where neural networks are widely used. There are thousands of images to sort through and interpret, and there is a lot of post processing to complete to learn anything from the images, so therefore neural networks are used to speed up the process and essentially extract the truth. But when using neural networks to determine if nuclear weapons are working as they should, a high amount of accuracy is needed since the safety of our nation depends on these tests. Without accurate error bars on these neural networks' predictions, detrimental explosions or mishaps could occur because of a slightly off uncertainty value. Therefore, it is necessary to research implementation methods for better quantifying uncertainty levels in neural networks.

Scope of the Problem

Neural networks are computer algorithms modeled after the neuron connections in a human brain and are used to detect relationships and patterns in large sets of data to better understand it. They consist of an input layer, multiple hidden layers, and an output layer. Given some input data a neural network will go through multiple different layers and pathways until it decides on the most likely output value. Another important aspect of neural networks is exactly how they learn to predict a certain output. A given data set will be split into training data and testing data, and a typical split is 70% training and 30% testing. The neural network learns while using the training data, and then attempts to predict the testing data as accurately as possible based on what it has learned. In essence, neural networks can be used as a type of prediction model, but they are also categorized as a 'black box' of information. In other terms, no prediction model has 100 percent accuracy all the time; there is always some error. A widely used example of how neural networks

can have high uncertainty is as follows. Say you have a neural network that takes in images and gives an output of whether they are a cat or a dog, but then you input an image of a zebra. The neural network will still predict this image as either a cat or a dog because it has not been trained to recognize zebras as an output, and therefore has a high uncertainty value and subsequent error bar for that specific image. Learning how to accurately quantify the level of uncertainty in neural networks by accounting for situations like these is what the REU students will be focusing on for the duration of their research.

Uncertainty Quantification (UQ) is the science of using various methods to quantify and reduce the amount of uncertainty in a data set. An ensemble neural network combines multiple similar but individual neural networks to decide on the most likely cumulative output based on each networks individual results. An ensemble is a type of UQ approach because it adds more checks to the neural network to ensure it is as certain as possible about its output value. REU students will attempt to implement an ensemble neural network for their selected dataset alongside another UQ approach to see how low the uncertainty factor in the accuracy can get without overtraining the network. Students will decide on their specific UQ approach individually, and combine individual efforts to develop the best neural network possible.

Data Visualization/Description

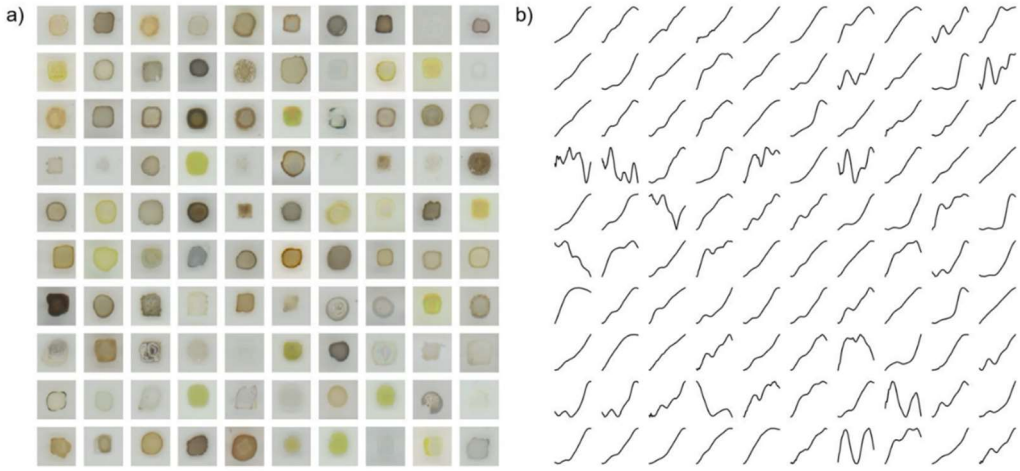
The selected data set is JCAP images and absorption spectra for 179072 metal oxides, and is provided by the US DOE and described by Figure 1, (Gregoire et al., 2018). Inputs are characterized as images and outputs are characterized as spectra. There are 220 output values for each input. Images were taken using a consumer flatbed scanner and scaled to 64 x 64 pixels. The colored imaging in the center is the printed material, whereas the coffee-ring-like structure on the outer edge is due to drying of the printed solutions and you can see these different structures in numerous images within Figure 2. All of the absorption spectra were measured using a dual sphere spectrometer, and represents 220 photon energies between the 1.31 to 3.1 eV energy range. The reported outputs are “fractional optical absorbance, which is the product of the absorptions coefficient and effective material thickness,” (Stein et al., 2019). These are represented by the graphed line data in Figure 2.

Figure 1:

Dataset	Content Description	Data Range	Data Size	Physical Units	Method
Images	Sample images	0–1 for every channel	(64,64,3,180902)	Color values for RGB	platebead scanner
spectra	fractional optical absorbance spectrum	0–ca. 0.5	(220,180902)	fractional absorb. coefficient	dual-sphere optical spectrometer

Description of JCAP Dataset. Note that images are normalized between zero and one, and are routed through three channels (RGB), (Stein et al., 2019).

Figure 2:



Correlation of input images and there corresponding output spectra line graph, (Stein et al., 2019).

Initial Strategy

The first step in our solution is to create a neural network. It was decided that only $1/11^{th}$ of the total output points of the spectra will be used. This simply means that to create the spectra graph every 11^{th} point, for a total of 20 points, will be used instead of 220 points in order to create the line. The reason for this is computational simplicity and to decrease some recurrent oscillations we were experiencing in our trials. In addition, we decided that we would be using 80% training data and 20% testing data. Lastly, in order to quantitatively measure the differences between tweaks to our neural network we created a mean square error graph. Ideally, the mean square error would be equal to zero, which would correspond to the actual data and the predicted value being exactly the same. Essentially, this allows us to argue that the closer to zero the model is, the better it is. Mean square error was chosen because it is one of the most common methods for evaluating prediction models and is easy to comprehend.

In order to expedite creation of the neural network, we found a sample code on Kaggle for a convolutional neural network to use as a starting point. This sample code is shown in Figure 3, (Antounes, 2022). This network is not specific to our data set, so therefore we just used it as a guide to then manipulate and build our own convolutional neural network from it. However, what exactly is the difference between a neural network and a convolutional neural network? A basic neural network has been explained previously in this paper, and a convolutional neural network is just a regular network with at least one convolutional layer included in it. In Figure 3 the convolutional layer is seen as “Conv2D”, and all convolutional layers do is essentially add a filter to the original input to prepare it for the rest of the layers of the network.

Figure 3:

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', inp
ut_shape=(IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(units=1, activation=None)
])
```

Starting code base from Kaggle to build new convolutional neural network off of, (Antounes, 2022).

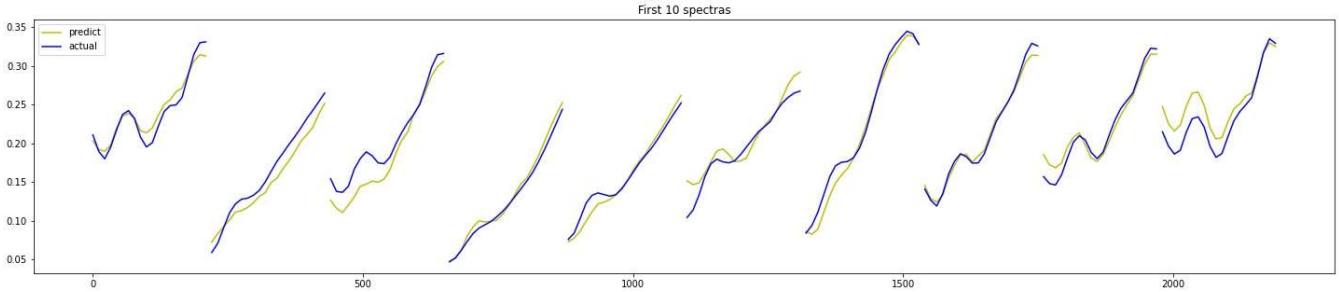
The base code in Figure 3 was altered to create a neural network that is better suited to our data set. For each of the following runs 10,000 samples were used with 50 epochs and a batch size of 32 unless otherwise stated. The sample size is the total number of images, the epochs represent how many passes the entire training dataset goes through, and the batch size represents the number of samples processed before the network is updated.

Initially we used the following layers in our network: Convolutional, Dense, Max Pooling, Convolutional, Dense, Max Pooling, Convolutional, Convolutional, Max Pooling, Flatten, Dense, Dense, Dense. This did not provide a successful model, so after other testing we settled on the following final order of layers: Convolutional, Dense, Max Pooling, Flatten, Dense, Dense, Dense. When running this network, it produced decent correlation between the training and validation data as shown in Figure 4. In addition, a big change between the networks was changing the activation function of the last Dense layer from none to softsign. In simple terms, softsign helps with the smoothing of data, which was needed because we were getting a lot of oscillation in our prediction with the initial model. By implementing softsign our graphs with the new network were much more visually accurate.

In an attempt to increase the accuracy of the network, an ensemble was created with 5 neurons to better improve the correlation, and this is shown in Figure 5. An ensemble is a method used to improve predictive performance in machine learning, and therefore is one type of uncertainty quantification. By using an ensemble in a neural network, instead of all the data being run through one network one time and producing an output, it will go through an ensemble of networks, or essentially multiple neural networks, before producing an output. These neural networks have almost exactly the same architecture, but have slightly different weighting in each so that multiple different outcomes are provided. This effectively allows the data to be dissected on different paths and then a final output can be decided from these options. For reference, Figure 7 effectively shows how an ensemble is implemented into a neural network. For our ensemble specifically, in order to decide the final output, the geometric average of all the neurons was calculated. The geometric average was used because it supposedly is better fit for data that could have large fluctuations between points, which we were experiencing initially. The formula for geometric mean is $(x_1 * x_2 * x_n)^{\frac{1}{n}}$.

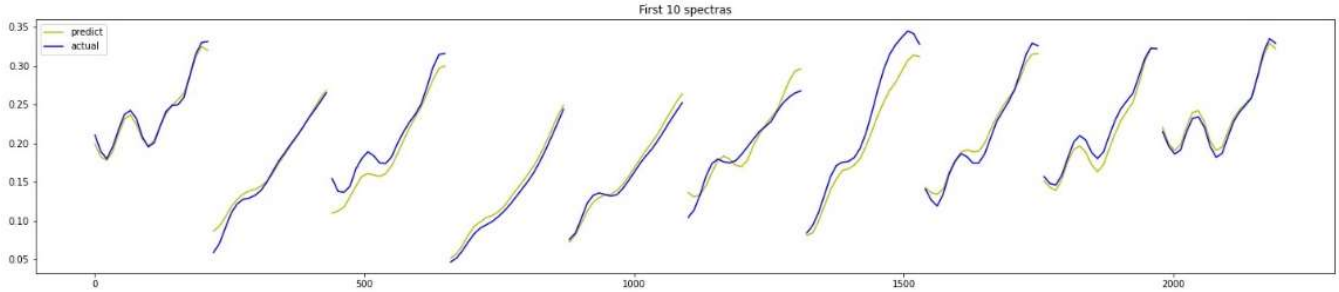
Furthermore, after playing around with the number of epochs and the batch size, it was found that the network had been slightly overtrained at 50 epochs so this was decreased to 40, and the batch size was decreased to 20 in order to train the network more often. This resulted in a slightly more visually accurate display of graphs, shown in Figure 6.

Figure 4:



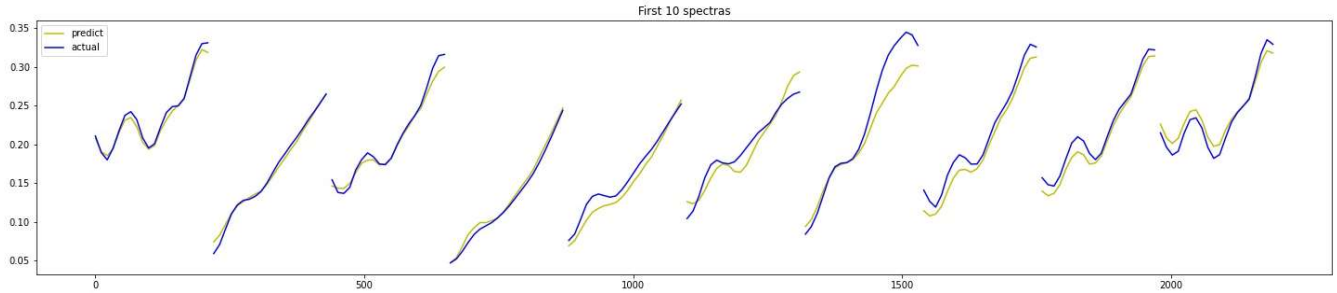
The first 10 spectra graphs with the final neural network architecture, 10,000 images, 50 epochs, a batch size of 32, and 1 neuron (without ensemble).

Figure 5:



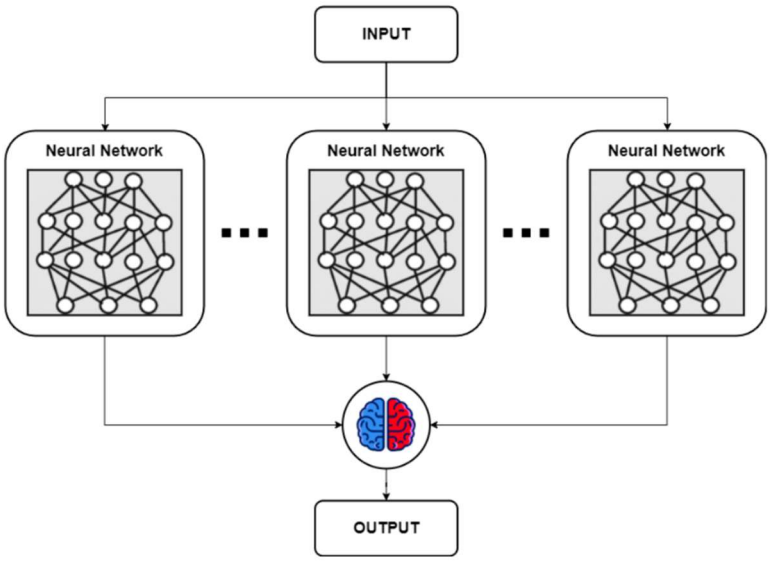
The first 10 spectra graphs with the final neural network architecture, 10,000 images, 50 epochs, a batch size of 32, and 5 neurons (with ensemble).

Figure 6:



The first 10 spectra graphs with the final neural network architecture, 10,000 images, 40 epochs, a batch size of 20, and 5 neurons (with ensemble).

Figure 7:



Visualization of ensemble neural network including three neurons, (Adams & Lund, 2022).

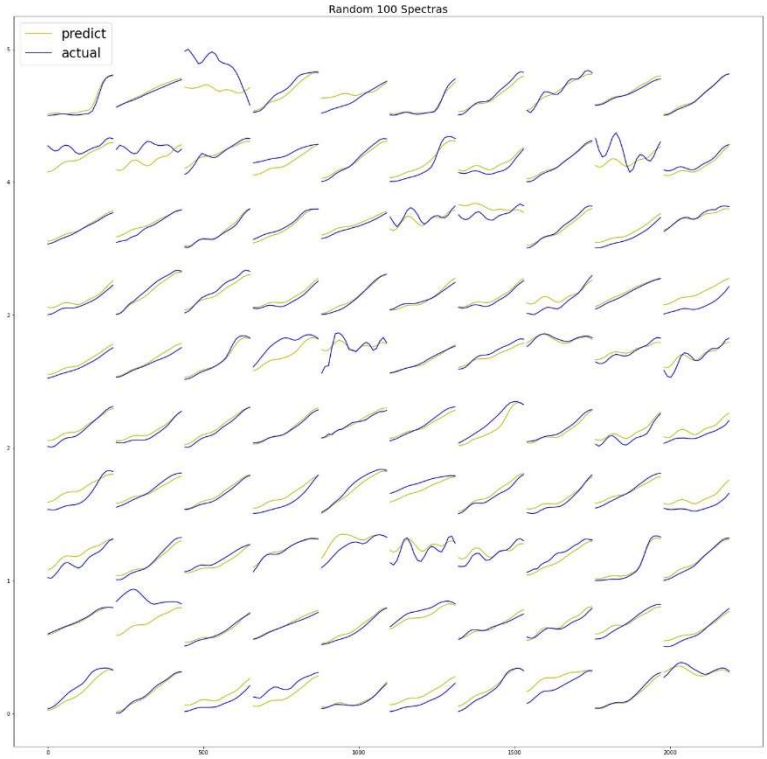
Final Results

Many small factors were changed in order to create our finalized architecture and process before analyzing our UQ approach. First, a Dropout layer at 0.1 was introduced to the architecture after the Max Pooling layer in an effort to obtain better results. Second, early stopping was introduced for the number of epochs. This stops the network from training once a certain mean squared error is reached, and essentially prevents overtraining of the data. This also helps to vary the number of epochs between each individual network in the ensemble model. Third, two randomizations were introduced with the same goal of preventing overtraining and creating a better overall network. The first randomization is the random binning of importing the data. Before the data would be randomized and then a certain number of images that were in a group would be imported and analyzed. Now, the images are sorted on a spectrum from light to dark and random binning chooses buckets of 50 images from random portions of the entire dataset. The second form of randomization is the random shuffling of data when it is being trained. Before the training data was seen in the same order between the networks, now the same dataset is shuffled before moving onto the next network. Both of these randomizations are helping to prevent overtraining in our network. Lastly, the number of networks was increased from 5 to 10 in order to produce more variation and hopefully improve our overall networks performance.

Figure 8:

Random 100 spectras of 1 network on 40,000 images without random binning.

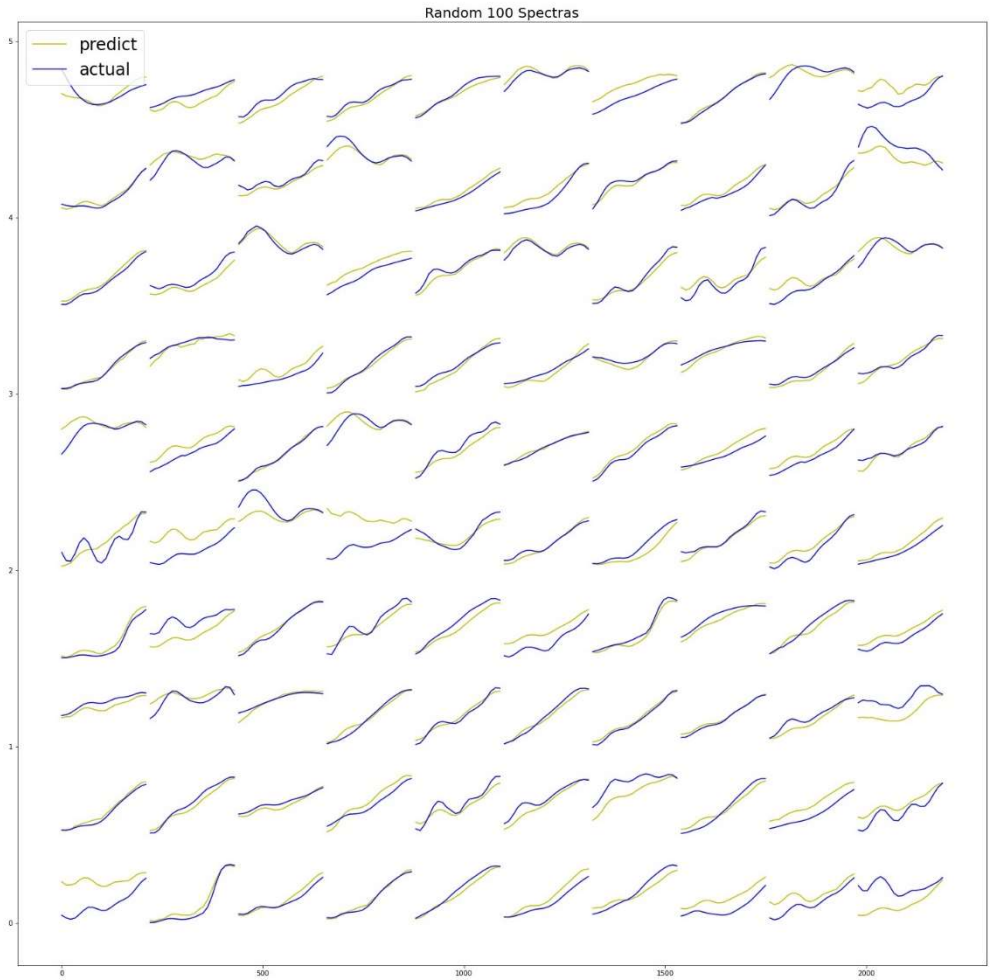
Figure 9:



Random 100 spectras of 1 network on 40,000 images with random binning.

When comparing Figures 8 and 9 it is apparent that Figure 8 has a lot of actual spectra lines (blue) that look the same, whereas Figure 9 has a much larger variation between actual spectras. This is because by using random binning we get a much more accurate representation of the entire dataset, and it is less likely that our model will become overtrained.

Figure 10:



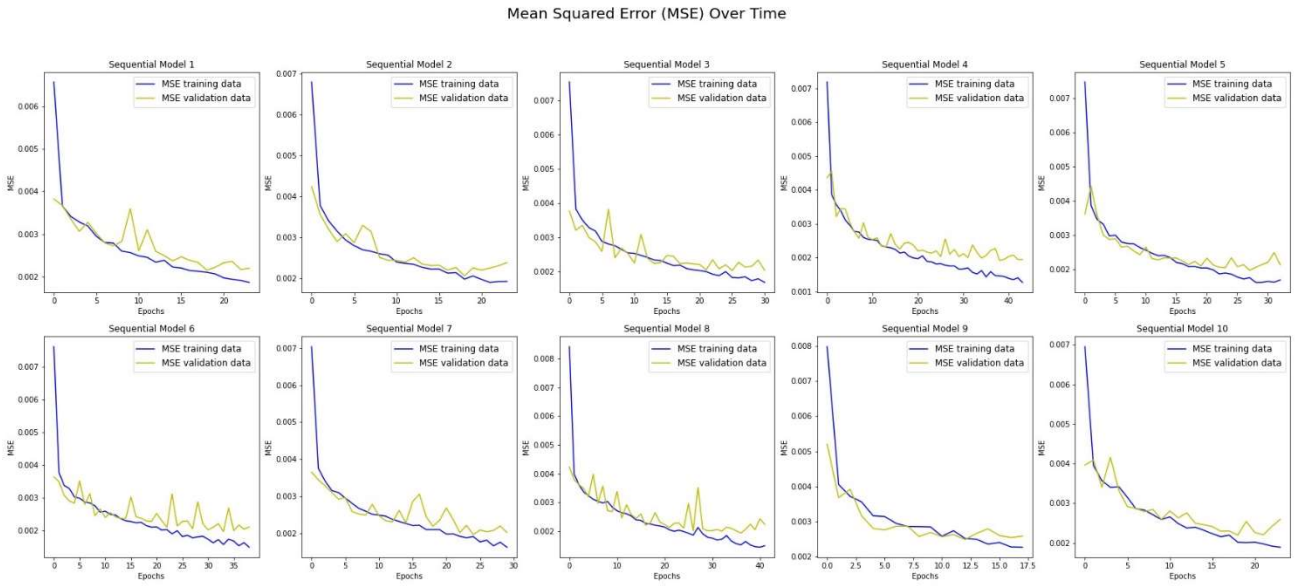
Random 100 spectras of 10 networks on 40,000 images with random binning.

By comparing Figures 9 and 10 the effect that an ensemble has on the network can be seen. Although it is hard to see visually the ensemble produces predicted spectra that are more accurate than when only using a single network. The ensemble introduces variance between network predictions and then takes the average. Time constraints only allowed us to build an ensemble that averages all the network predictions equally, but a way to improve upon this would be to weight networks that had a lower mean squared error more significantly than networks with a high mean squared error.

Now that we have the progression of our final network, the real research begins: quantifying how well our prediction model operates. In order to do this, we looked at our data from a few different angles. First, we verified that the network was working correctly by investigating the mean squared error of each individual network. Next, a linear regression scatterplot model was created to determine an R^2 value and visualize if there were certain places in the spectra where the prediction model did not work well. Then, we created a confidence interval around the final prediction and plotted the actual spectra on the same graph. Lastly, and most importantly, we created a function that computes the accuracy of our neural network. This test calculates the percentage of how often the predicted points did fall inside of the confidence interval for each spectra that was tested. All of these error quantifications were conducted on the model represented by Figure 10.

Mean Squared Error

Figure 11:

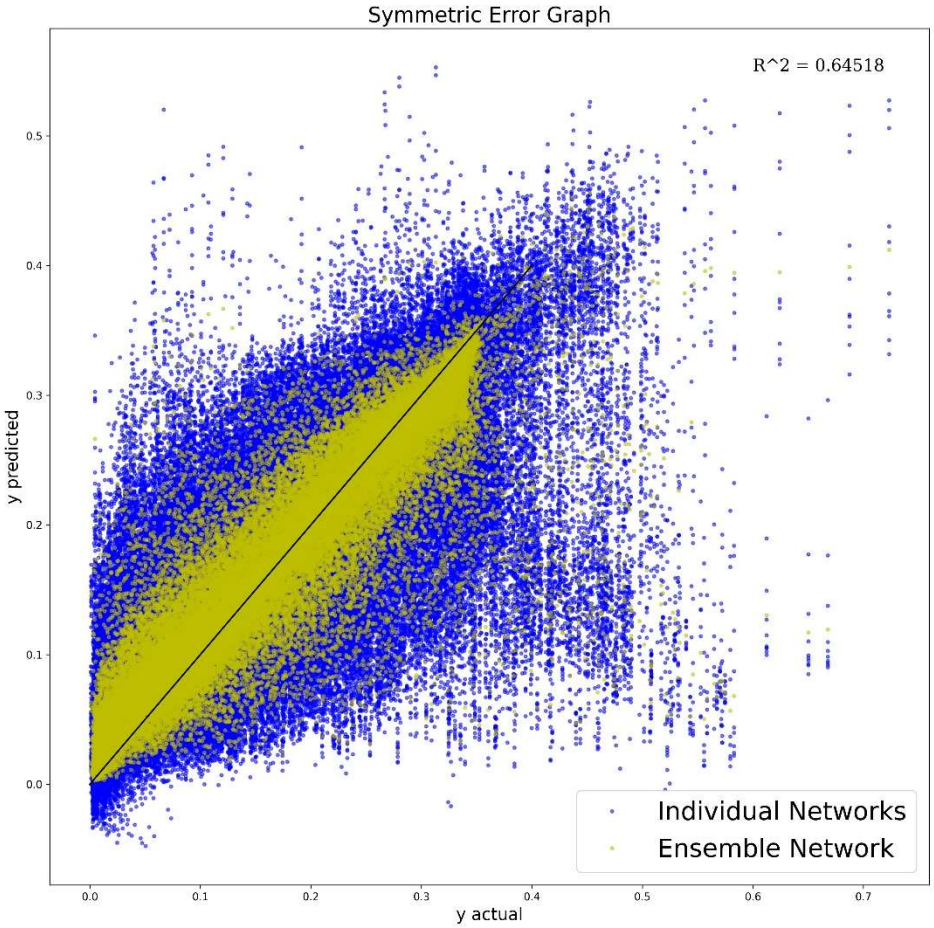


Visual representation of each individual networks mean squared error progression. Displays the number of epochs that each individual network was trained on.

Figure 11 shows the mean squared error over the number of epochs for each individual network. It is clear that all of the networks stop training at a mean square error value that is less than 0.003, and most are closer to 0.001. These numbers are extremely close to zero, which should mean that the predicted spectra were close to the actual spectra for each model when analyzing the training data only. When looking at the individual networks it is obvious that each were trained in a different way because the error fluctuates differently in each. A drawback of this model is that it does not give an idea of how well the prediction model works when using the validation data, which is what truly matters.

Linear Regression Model

Figure 12:



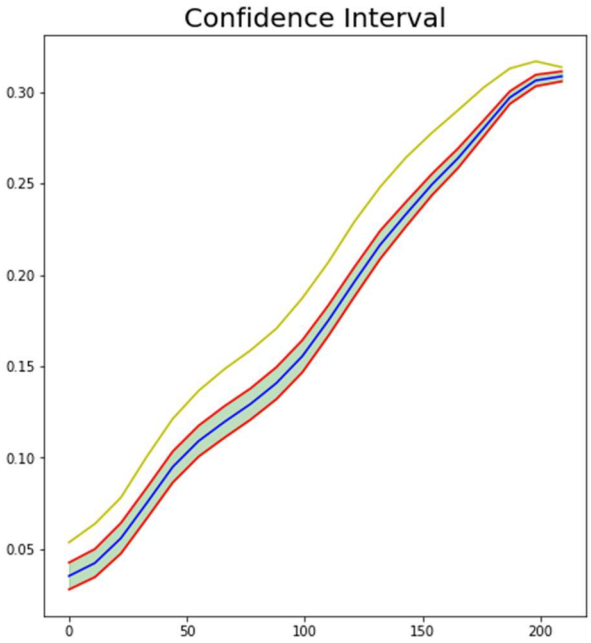
Linear regression scatterplot visualization of each individual networks predictions (blue) versus the final averaged ensemble prediction (yellow). The line y = x is drawn to show where plotted points would fall if the predicted value perfectly equals the actual value.

This graph represents the distribution of all predicted points on the spectra graphs. It is obvious that the ensemble does increase the accuracy of the predicted points since the yellow cloud is much more condensed around y = x than the blue cloud of points. Since in a perfect model the R^2 value would be equal to 1.00 (where all the predictions were 100% accurate), Figure 12 shows that a decent R^2 value of 0.64518 was obtained, but it is clear that there are some limitations to our models ability to predict accurately.

This model is working with the validation data which gives us a better insight to our model, however the R^2 value is not the best measurement to use when trying to understand accuracy. For example, the prediction model could have produced predicted values that are all close together and near y=x, which would produce a high R^2, but these values could all be outside the desired confidence interval, and therefore very inaccurate. This model does not effectively compare the predicted spectra to the actual spectra.

Confidence Interval

Figure 13:

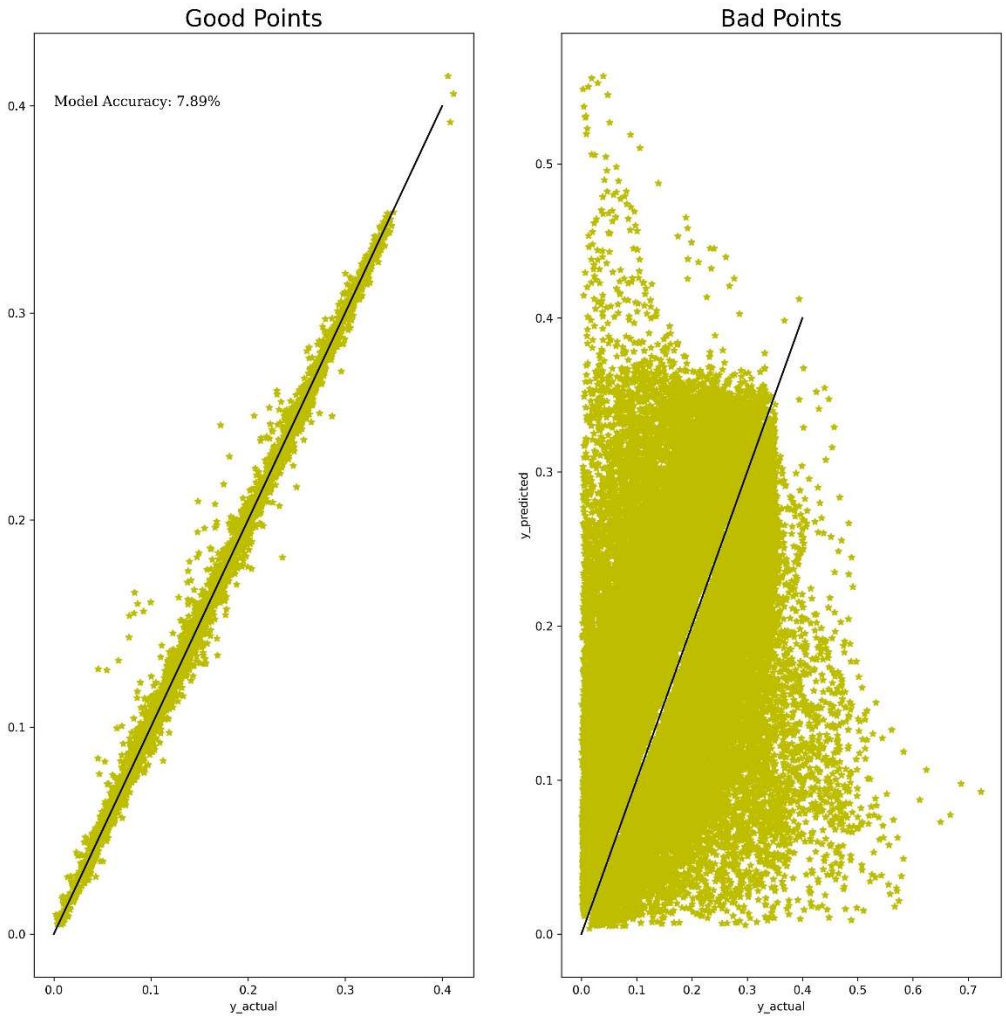


Visualization of a confidence interval. The actual spectra line is given in yellow, the predicted spectra line is given in blue, the red lines indicate the upper and lower bounds of a 95% confidence interval, and the green area fills in the said confidence interval.

A 95% confidence interval was defined in order to apply the accuracy prediction model, but is also a good visualization of how our network is failing. As we can see in Figure 13, the confidence interval around the predicted spectra is a very small area. This essentially means that each of the individual networks was not varied enough in their predictions. Each networks prediction was close to one another, so the entire network has a small area with a lot of confidence in its prediction, but we can see that the actual spectra does not fall anywhere inside of the confidence interval. This suggests that our prediction model is not one that should be relied upon as of right now. Figure 13 verifies that our prediction model is extremely confident but very inaccurate in its prediction, and this is further verified by the accuracy prediction model. The confidence interval was calculated using the formula $CI = mean \pm 1.96 \frac{standard\ deviation}{\sqrt{sample\ size}}$, where CI is the confidence interval and 1.96 corresponds to 95%.

Accuracy Prediction Model

Figure 14:



Visualization of accuracy prediction model. Left: shows 'Good Points' that fall within the 95% confidence interval. Right: shows 'Bad Points' that fall outside of the 95% confidence interval.

Figure 14 is a visual representation of the accuracy model and shows linear regressions with the points inside and outside of the 95% confidence interval, respectively. However, the most important part of this model is the percentage on the Good Points graph. This number, 7.89%, represents the accuracy of the entire prediction model. Ideally this number would be close to 100%, which it is obviously not. So although our prediction model is very unreliable, we were able to identify that it is unreliable, which is a very necessary step for neural networks in general.

This value was calculated by using the confidence interval on each spectra. The percentage represents the number of points that fall inside of the 95% confidence interval divided by the total number of points.

References

Adams, J., & Lund, M. (2022). ERAU REU Project Proposal: Ensemble Deep Learning. Nevada National Security Site.

Antounes. (2022, January 5). *Convolutional Neural Network for Image Regression*. Kaggle. Retrieved May 30, 2022, from https://www.kaggle.com/code/matthieubritoantunes/convolutional-neural-network-for-image-regression

Gregoire, J. M., Stein, H. S., Soedarmadji, E., Newhouse, P. F., & Guevarra, D. (2018, October 29). *JCAP images and absorption spectra for 179072 Metal oxides*. Office of Scientific and Technical Information. Retrieved May 25, 2022, from https://www.osti.gov/dataexplorer/biblio/dataset/1479489

Mission Support and Test Services. (n.d.). *About the NNSS*. Nevada National Security Site. Retrieved May 25, 2022, from https://www.nnss.gov/pages/about.html

Stein, H. S., Soedarmadji, E., Newhouse, P. F., Guevarra, D., & Gregoire, J. M. (2019, March 27). *Synthesis, optical imaging, and absorption spectroscopy data for 179072 metal oxides*. Nature News. Retrieved May 25, 2022, from https://www.nature.com/articles/s41597-019-0019-4