**Ensemble Deep Learning**

Emily Diegel, Rhiannon Hicks, Max Prilutsky, Rachel Swan

Jesse Adams and Margaret Lund, PhD Senior Scientists

Nevada National Security Site

Dr. Mihhail Berezovski, Embry-Riddle Aeronautical University

*Support for the program has been provided by the National Science Foundation (NSF) through REU Award Number*[*DMS - 2050754.*](https://www.nsf.gov/awardsearch/showAward?AWD_ID=2050754)



*Neural networks are an emerging topic in the computer science industry due to their high versatility and efficiency with large datasets. Funded by the National Science Foundation, Embry-Riddle Aeronautical University is partnered with the Nevada National Security Site on the project, Ensemble Deep Learning, through the Research Experience for Undergraduates 2022 summer program. The Nevada National Security Site is seeking deep learning techniques to analyze radiographic images of small-scale nuclear test explosions to ensure that the United States nuclear stockpile remains safe, reliable, and secure. In building a deep learning model, multiple convolutional neural network architectures are developed in parallel and combined to create an ensemble neural network. Neural networks are often referred to as a “black box algorithm” due to the complicated series of weights and biases that make up the model. This algorithm is then able to be analyzed through uncertainty quantification, how certain we are of a prediction. The project’s objective is to determine the comparative differences between the predictive ability of each individual convolutional neural network versus the ensemble neural network. Additionally, we will explore how to use the ensemble model as a method of uncertainty quantification. Adding an ensemble reduced the MSE from 0.003524 to 0.00268 and increased the R2 value from 0.54251 to 0.65247 compared to an individual network.*

**The Nevada National Security Site**

The Nevada National Security Site (NNSS) is a research and development center that performs high hazard testing, waste disposal, threat detection, and stockpile science of nuclear weapons. They provide nuclear and radiological emergency response capabilities and training to the nation. Their primary mission is to ensure that the United States nuclear stockpile remains safe, secure, and reliable. They perform underground classified nuclear experiments, high-tech computer simulations, and detailed engineering analysis. They use gamma rays to produce radiographic images of mid-explosions to analyze the effectiveness and health of the weapons. The analysis of these radiographic images is made possible through deep learning, and the implementation of neural networks.

**Project Scope**

NNSS is working to improve analysis on radiographic images by using neural network models to ensure that the nation's stockpile remains effective and safe. Neural networks provide the opportunity to analyze large amounts of image data and produce a regressive output. They are a predictive algorithm modeled after the neuron connections in the human brain. Using a set of inputs and a set of desired outputs, the network is trained and tested in order to predict the desired outputs. The network trains itself through learning patterns and correlations between data and assigning weights and biases to certain values to produce a label. One problem that occurs when using neural networks is understanding how the network is being trained. This is because neural networks are considered a black-box algorithm, which means that essentially, the network finds patterns that are too complicated for humans to understand, therefore leaving parts of the model unknown to the creator. This algorithm provides no real method to display uncertainty on the predictions provided by the model. Without a measure of uncertainty, there is no telling how dependable predictions are when the actual value is unknown. In industry, it is extremely important to provide a measure of uncertainty alongside a model because these models will be applied to real-world situations and will impact an abundance of lives. For this project, the main goal is to determine an uncertainty quantification method for an ensemble neural network that NNSS can rely upon to use in their own nuclear tests.

The tasks of this project include:

* Analysis of the dataset
* Implementation of convolutional neural network based on a given 2D dataset
* Implementation of ensemble neural network
* Implementation of uncertainty quantification method

**Dataset Description**

The dataset to be used is “Synthesis, optical imaging, and absorption spectroscopy data for 179072 metal oxides” [1]. This dataset includes 179072 samples after cleaning made of 42 total elements in various combinations of 1 to 5 elements. This dataset was provided by the NNSS because it is a similar dataset to what they work with when conducting nuclear testing, but this dataset is unclassified and readily available for us to use.

Input: Images of metal oxides (64 x 64 Pixels, 3 RGB Channels, 179072 Samples)

* Sample images were taken using a commercially available consumer flatbed scanner (EPSON Perfection V600) in reflection configuration at 1200 dpi corresponding to a rate of 2.0 or 0.019 s/sample

Output: Spectra graph (220 Fractional Absorption Coefficients, 179072 Samples)

* Energy range for all spectra is 1.32 eV (left end) to 3.1 eV (right end)
* Discretize into 220 photon energies
* Optical absorption spectra were measured using an on-the-fly scanning UV-Vis dual-sphere spectrometer

Figure 1. Summary of Selected Attributes in the hdf5 [1],

H. *019).*

| **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 | Flatbed Scanner |
| **Spectra** | Fractional Optical Absorbance Spectrum | 0-ca. 0.5 | (220, 180902) | Fractional Absorb. Coefficient | Dual-Sphere Optical Spectrometer |

*This table gives a full description of the inputs and outputs provided in the dataset, and gives insight to what is being measured.*

Figure 2. Comparison of Images of Metal Oxides and their Spectra [1]

A picture containing diagram

Description automatically generated

*Comparison of 100 random images and their associated spectra graphs. Provides insight to the vast number of possible spectra lines that the network will be responsible for predicting.*

In Figure 2’s displayed spectra outputs, the graph uses 220 points to plot the shown spectra. For our purposes this output was simplified by taking every 11th point of the graph. Therefore, only 20 points are plotted on the graph rather than 220. This was done in order to reduce computation time of the neural network. As shown in Figures 3 and 4, the shape of the spectra graph is maintained and therefore the accuracy lost when using only 20 points is insignificant.

Figures 3 and 4. Absorption Spectra for Metal Oxides Comparison

A picture containing shape

Description automatically generated A picture containing icon

Description automatically generated

*A comparison of these graphs shows that even thought the amount of output points is drastically decreased in the right graph, the shape of the spectra graph is maintained when compared to the left graph.*

**Data Analysis**

Before training the neural network, it is necessary to investigate the data and determine correlations and outliers that may affect a predictive model. First, images that had 0 metals were deemed as false inputs. In the total dataset, there were 1803 of these images. Some of the images did contain a seemingly valid oxide image, however without knowing the chemical makeup we could not accurately stratify these. Therefore, all 1803 images were excluded from the dataset before any training or testing was conducted.

Figure 5. Images of Composition 0 for Metals

Chart

Description automatically generatedChart

Description automatically generated

Chart

Description automatically generated

*Three individual sample images and their associated spectra graphs for compositions that are made of 0 metals.*

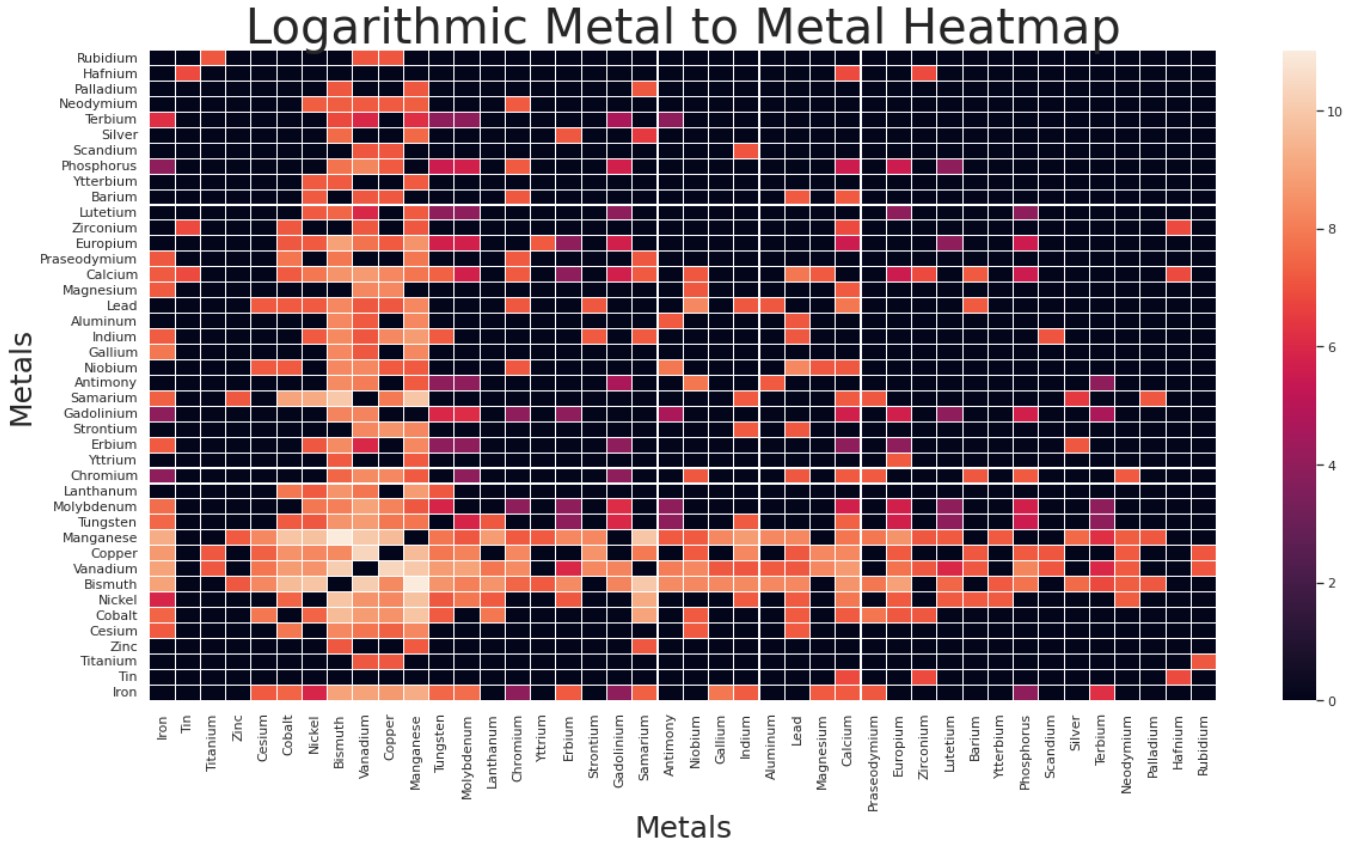
The dataset includes samples with a range of 1-5 metals in various combinations in each sample. Figure 6 shows a full breakdown of the dataset. Out of 179072 total samples, 45.93% contain 4 metal oxides (highest percentage), whereas 1.08% contain 5 metals (lowest percentage). For each sample, compositions vary in concentration and elements included.

Figure 6. Table of Metal Composition Comparison of 179,072 Samples

| Number of Metals in Composition | Total samples out of 179,072 | Percentage |
| --- | --- | --- |
| 1 | 9250 | 5.17% |
| 2 | 19216 | 10.73% |
| 3 | 66427 | 37.10% |
| 4 | 82241 | 45.93% |
| 5 | 1938 | 1.08% |

*Encompassing table numerically describing how often there are a specific number of elements in a combination in a given sample from the entire dataset.*

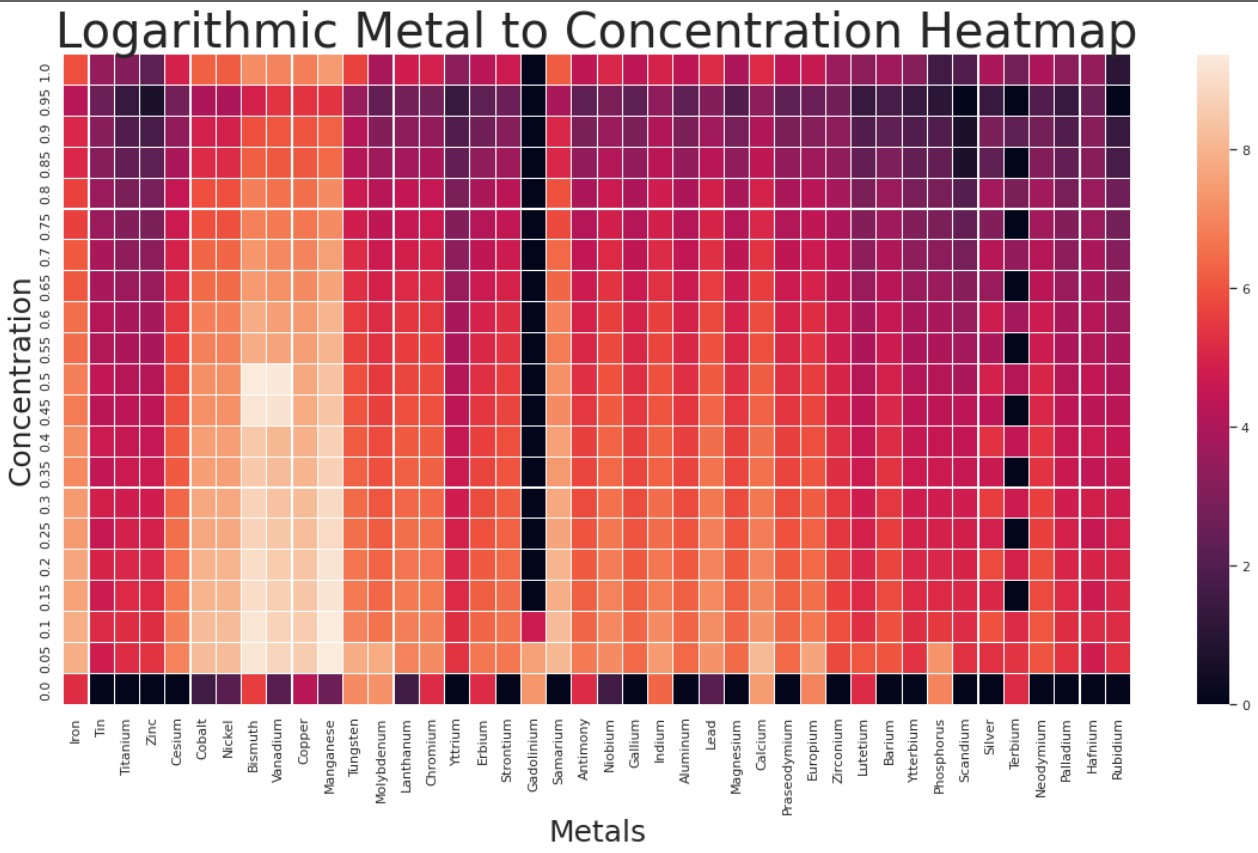
Figure 7. Composition Heatmap of Metal Combinations



*This heatmap is completely symmetrical along the line y = x and provides a visualization of how often specific metals are combined together.*

Figure 7 is a visualization of how many times two metals appear together in the same combination in a sample. Lighter colors represent higher quantities of combinations, while darker colors represent closer to a zero quantity. It is clear that black is the most dominant color on this heatmap, which insinuates that there are a lot of metals that never appear in combination with each other. However, if we look at bismuth and manganese specifically, we see that this is the lightest square on the heatmap and suggests these two elements often appear in combination with each other. Since certain metals appear more in combination, the training dataset needs to be stratified or it will be skewed and unbalanced towards these metals.

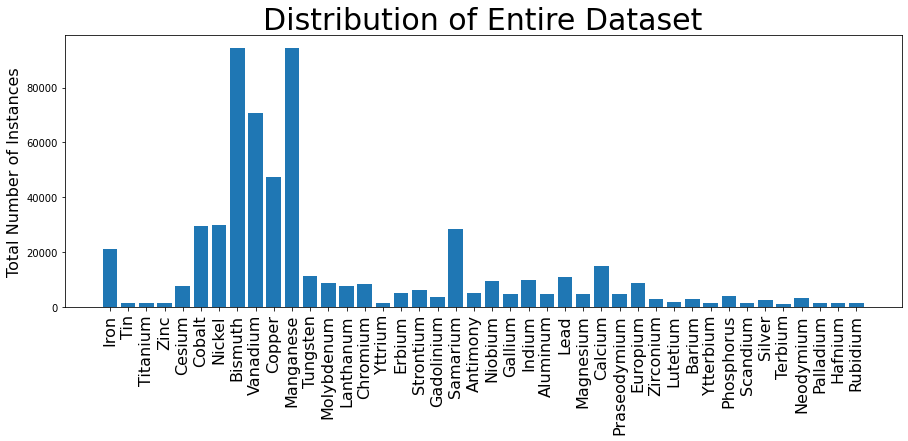
Figure 8. Concentration Heatmap



*Visualization of variance of typical concentration amounts between metals over the entire dataset.*

Figure 8 displays how often each metal appears in certain concentration amounts over the entire dataset. Lighter colors represent a higher quantity of a metal appearing at a ceratin concentration, meanwhile darker colors represent a lower quantity of a metal appearing at a certain concentration. Typically when looking at this datasets columns there are lighter colors at the lower concentrations down to 0.05, and there are darker colors at the higher concentration values. The 0.0 block represents where the concentration is less than 0.05 and it is apparent that there is a mix of elements who do and do not appear at this level of concentration. This provides insight that most metals are only represented in a sample at a very low concentration when they are in combinations. It is also shown that the columns for bismuth and manganese are much lighter overall than other columns in this heatmap, showing they appear more in the dataset than other elements, no matter what concentration they appear at.

Figure 9. Bar Graph of Distribution of Metals



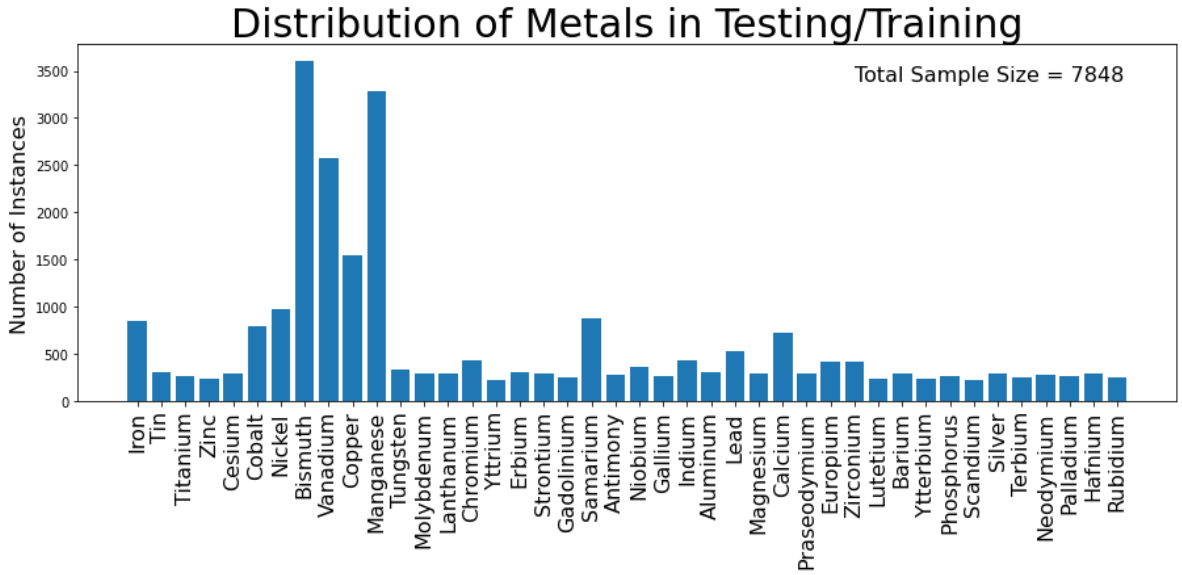
*Representation of number of instances each element appears over the entire ~180000 dataset.*

This bar graph represents the number of times each metal appears in the dataset, regardless of what concentration it appears with. Bismuth and manganese appear the most in the dataset at roughly 90,000 instances, followed by vanadium at approximately 70,000 instances. Most of the metals appear much less than 20,000 times in the dataset with the smallest at 1077. From this graph it can be determined that the conclusions from Figure 7 and Figure 8 regarding bismuth and manganese are fortified by this graph. Since bismuth and manganese are the most prominent elements in the entire dataset, by basic probability it makes sense that they occur the most in combination with each other, and that they have a higher number of instances regardless of their concentration. Because most of the other elements are not highly represented, bismuth and manganese stick out since by simple probability they are more likely to be in a randomly selected sample from the dataset.

**Stratified Sampling**

Because of limitations due to available RAM and our computer capabilities, our research was not able to be conducted on the entire dataset and therefore we had to take a subset size of the entire dataset. This leads to an important limitation on our model since data is very unevenly distributed. Since certain metals appear more in combination, the training dataset needs to be stratified or it will be skewed and unbalanced towards these metals. A stratified random sample of the data must be taken instead of a completely randomized sample to ensure the model is being trained on each element. Additionally, we can only take a certain subset size of our whole data as too big a sample will lead to overtraining on manganese, bismuth, and other highly represented elements.

When training a neural network, it is important to have a balanced dataset with outputs being represented equally. If a training dataset is unbalanced, the model will be better at predicting outputs that have a larger representation in the dataset because these outputs were seen more during training. Without a balanced dataset, the model will be biased towards certain outputs. To create a balanced dataset, the same number of each element should be represented equally. Therefore, we created an algorithm that ensured every element in the dataset was represented at least 300 times. The algorithm also incentivized elements to not exceed 900 instances, but because of how often bismuth, manganese, and some other elements occur in combinations, their number of instances had to increase in order to maintain the 300 minimum value. As shown in Figure 10, this creates a more balanced and stratified dataset that can be used to train our neural network.

Figure 10. Bar Graph of Stratified Dataset

*Representation of number of instances each element appears over the stratified dataset of 7848 images.*

**Convolutional Neural Networks**

Convolutional neural networks (CNN) produce outputs from image-based data through convolutional, dense, max pooling, and flatten layers. The images of the metal oxides are the input values of the network, and the output values are the values of the spectra. This network produces a regressive output of 20 points on the spectra. The underlying data relationships are unknown between RGB and absorption. Machine learning provides the ability to exploit hidden information in high spatial-resolution images to establish a relationship in RGB and absorption.

Figure 11. Convolutional Neural Network

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 62, 62, 64) 1792

dense (Dense) (None, 62, 62, 128) 8320

max\_pooling2d (MaxPooling2D) (None, 31, 31, 128) 0

dropout (Dropout) (None, 31, 31, 128) 0

flatten (Flatten) (None, 123008) 0

dense\_1 (Dense) (None, 128) 15745152

dense\_2 (Dense) (None, 64) 8256

dense\_3 (Dense) (None, 20) 1300

=================================================================

Total params: 15,764,820

Trainable params: 15,764,820

Non-trainable params: 0

*Description of single convolutional neural network model architecture.*

Figure 11 shows the architecture of the convolutional network that was implemented for the spectroscopy dataset. In total there were 8 layers in the sequential model. The input shape for our model is a 64 by 64 (pixels) by 3 (RGB) image with an output of 20 absorption points of a spectrum. Below is a complete description of the individual layers in the model.

**Layer Summary**

Conv2D Layer - Applies a filter to an area of the image and a dot product is calculated between the input pixels with the filter. The dot product is fed through an output array. The filter shifts then by a stride to every kernel in the image to reduce the size of the overall image matrix [2].

Dense - Computes a weighted average of the input and passes through a nonlinear function also known as an activation function.

Max Pooling2D - Calculates the maximum value of each patch in each of the feature maps. Highlights the most important feature of the patch through pooling down into a smaller matrix.

Dropout – Randomly sets neuron values to zero in deep layers to prevent overfitting.

Flatten - Converts the data into a 1D array for inputting into the next layer.

**Training**

The final steps before training and testing our network model is determining the number of epochs and the batch size. The batch size represents the number of samples evaluated in an epoch before weights and biases are adjusted. The number of epochs represents the number of times the training set is passed thr ough the network. Both values must be properly adjusted to avoid overfitting to the training set.

For out implemented convolutional neural network it was decided to incorporate early stopping into our model to prevent overfitting. This function stops training the network at the most optimal point where the difference between training loss and validation loss begins to increase. The maximum number of epochs is capped at 50, since through testing it was determined this is the about the maximum value that can be achieved before severe overtraining of the dataset.

The decided batch size for the implemented CNN is 32. Through model testing, larger batch sizes resulted in worse models as the weights weren’t being updated enough. Lowering the batch size below 32 caused the network to take increasingly longer to train since the network was being updated so often, and it was more difficult for patterns to be detected. When using a batch size equal to 32, this allowed the model to learn patterns within the inputs and the outputs and assign more accurate weights and biases to produce the output while not being too computationally taxing.

As the model is trained, it is tested in parallel to evaluate if the network is training properly. The subset size used to train this model was 7,848 images, as shown in Figure 10 of the stratified dataset. 80% of the images were used for training and 20% were used for validation.

To split our data 80/20, the function Shuffle Split from sk.learn was implemented in the network. This function randomizes the spread of the training and testing data based on the number of splits needed. For instance, in an ensemble, the training and testing will be from various parts of the data and will have overlap which will be seen through ensemble methods further throughout this report. Regarding a singular network, the order of which the data is split is random. This prevents overtraining of certain classes of the data. Variety of training data is key to creating an accurate model and this split allows for variety in training samples if the dataset is ordered for a certain class or regressive output.

Testing data is used to ensure that the network can predict values from images that it has not seen before. The images that are used to train the data should not be the same to test if the network is able to function as intended. The purpose of testing data is to introduce new inputs to the network to then have the network produce the output that is unknown by the model. In our model 2,000 random testing images were used in making predictions.

**Results: Single Neural Network**

Figure 12. Single Network Prediction Example

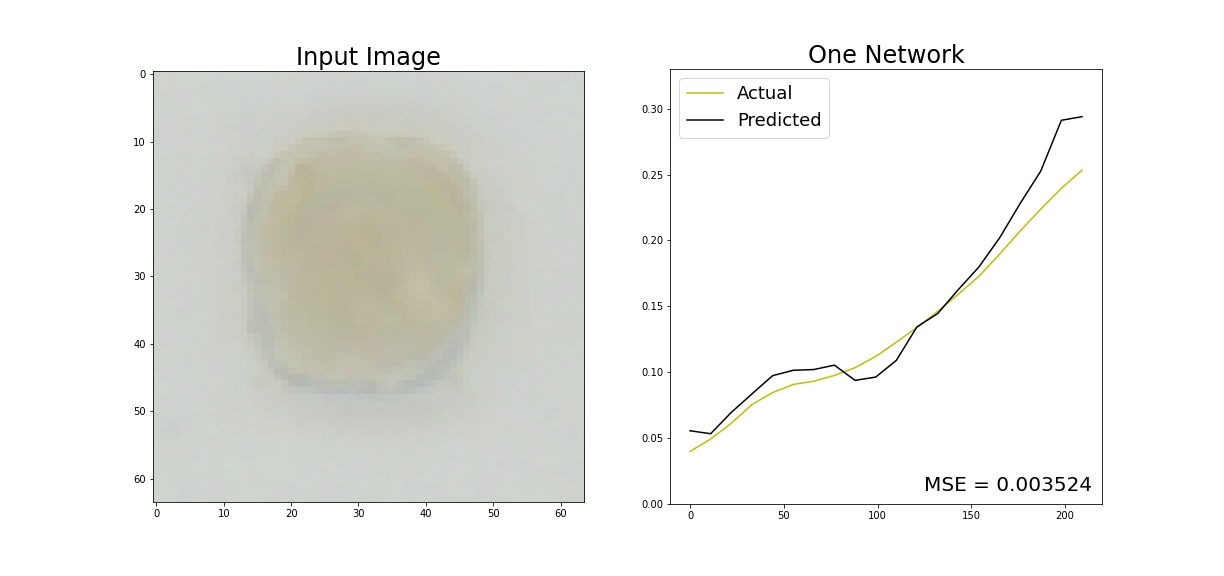
*Singular example of one neural network prediction with comparison to actual image spectra and actual spectra graph.*

Figure 12 depicts a specific example of what our model is doing. The image on the left gives an example of a spectra image, and the graph on the right compares the corresponding actual spectra graph and single neural network prediction. Looking at this graph it is clear there are some clear oscillations and jagged points in the prediction. These features are likely due to the fact that we are operating with a very small subset of data, and did not investigate what metals made up this sample. Overall this would be considered a good prediction since it is so close to the actual prediction, follows the same general shape, and has a very low mean squared error (MSE) value.

In order to get an encompassing picture of how the network is performing over ~8000 samples, a histogram was created and this is shown in Figure 13. This graph provides a representation of the distance error between the actual spectra line graph and the actual line graph, and shows how often specific values of error occur. Visually, it is clear that Figure 13 has a bell shaped distribution curve, which is expected. This reinforces the idea that our single neural network model is making good predictions. There are higher number of instances for smaller error values. In fact, most of our predictions fall between +/- 0.1, and since the highest error value that could possibly be obtained is 1, this allows us to draw the conclusion that our network model is accurately making predictions within a 10% error window for over and under approximations. Something else to notice is that the MSE value for the entire single neural network is 0.003524. Since the ideal value for this metric is zero, this also supports the claim that our network is making mostly accurate predictions in terms of the distance between actual and predicted spectra graphs.

Figure 13 helps us understand how well our model is predicting, but to further verify our claim that our model is working well we created a scatter plot that is visualized by Figure 14.

Figure 13. Single Network Histogram Error

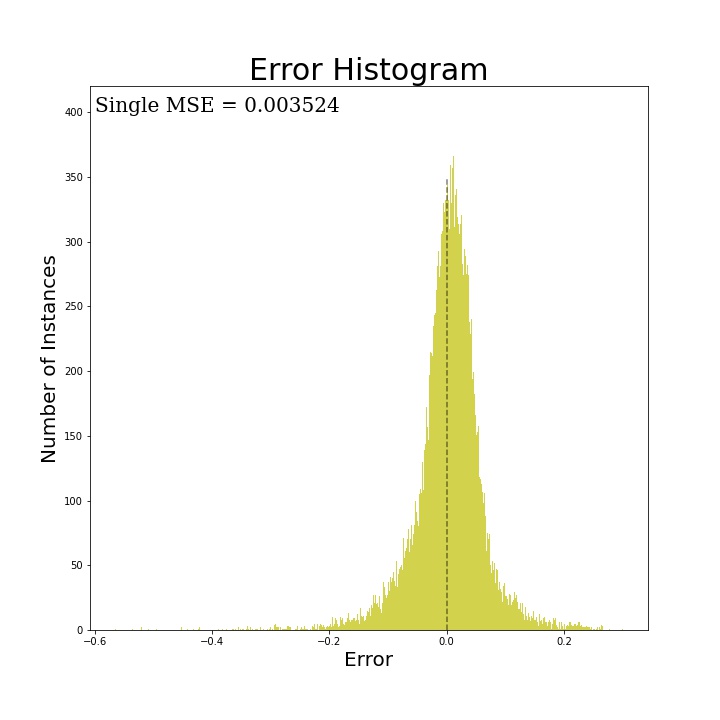
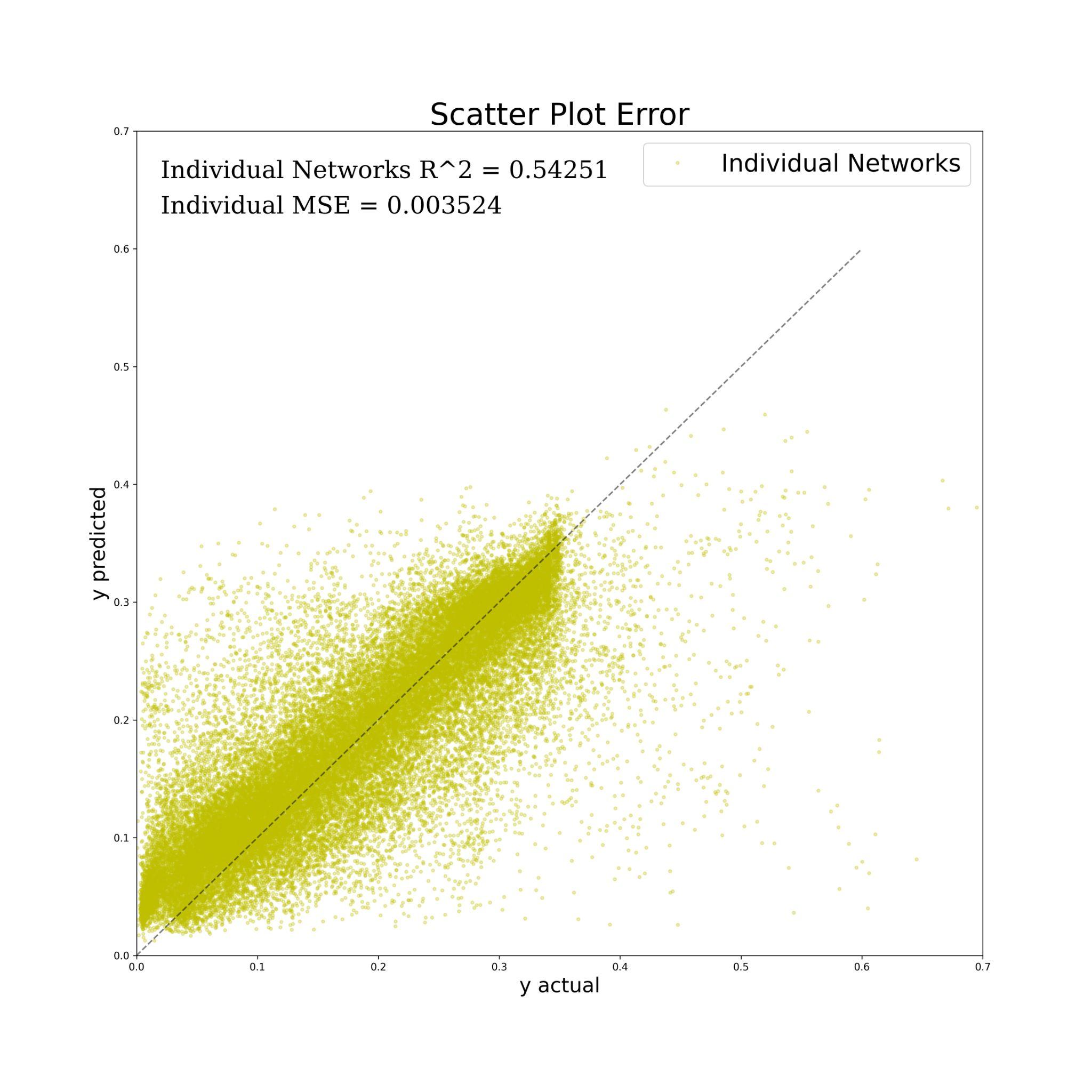
*Histogram distance error visualization for single neural network predictions with overall MSE value.*

Figure 14. Single Network Scatter Plot Error

*Scatter plot error visualization for single neural network predictions with R^2 value.*

Figure 14 depicts a scatter plot with every prediction from the single neural network. The predictions are arranged so that the line y = x (which is plotted in black) represents the location for plotted points that the predicted spectra exactly matches the actual spectra graph value. Therefore this line is the desired location for all plotted points, and it is obvious that most of our predictions are clustered around this line. The distance error is represented in the y direction, so therefore points above y = x are overapproximations and points under this line are underapproximations.

The purpose of this graph is to view the results in a different way. From this graph we can see that at the beginning of the line y = x in the lower left corner there are a lot of overapproximations, however, when looking at the upper right portion of the cluster we can see that there are more underapproximations at that point. Another benefit of this graph is that since we have a specific line that ideally the data would be clustered around, we can fit an R^2 value to our predictions. As shown in Figure 14 the R^2 value for the single neural network is 0.54251. Since an ideal R^2 value is 1.00, this suggests that while our predictions are generally good, there are definitely some consistent outliers in out data that if improved upon would decrease the uncertainty in our network.

**Ensemble Neural Networks**

Figure 15. Diagram of Ensemble Neural Network [3]

Diagram

Description automatically generated

*Diagram of how an ensemble neural network works. Each individual network receives the same input put produces their own output which is later combined for a final prediction.*

Ensemble neural networks are a type of uncertainty quantification method because they use multiple models of the same network to make slight variations of the same prediction, and therefore increase how certain a neural network is about its prediction. An ensemble is simply multiple of the same neural networks that each take the same input, train slightly differently, and each network produces their own output which is then combined for a final output. A diagram giving a visual aid of how an ensemble works is provided in Figure 15.

**Implemented Ensemble Neural Network**

For our ensemble, using the architecture from Figure 11, ten individual networks were trained in parallel. Then, all ten individual network predictions were averaged in order to create a final ensemble network prediction. This method of combining predictions is something that could have been improved upon with more time. Despite having the same architecture the networks do produce different results. When initialized each of the networks has randomized weights and due to a learning rate of 0.001 being implemented, these weights can only change so much. Additionally, as mentioned earlier, the package shuffle split was used to split the data 80/20. The individual networks train on similar data with approximately an 85% overlap between any two individual networks. The effect of this can be seen in Figure 16 below.

Figure 16 demonstrates how the MSE changes for each of the 10 networks in the ensemble. This loss function was used to optimize the model during the training. The yellow line represents the validation data, and the blue line represents the training data. Each network was trained using early stopping, altering the number of epochs used to train each model. As the number of epochs increases, the MSE decreases and this is because weights and biases are adjusted according to the value of MSE. The smaller the MSE, the better the overall network is performing.

Figure 16. Mean Squared Error of Ensemble Over Time

*MSE representation of ensemble. Visual representation of how each individual network is being trained differently.*Graphical user interface

Description automatically generated

**Results: Ensemble Neural Network**

Figure 17. Ensemble Network Prediction Example

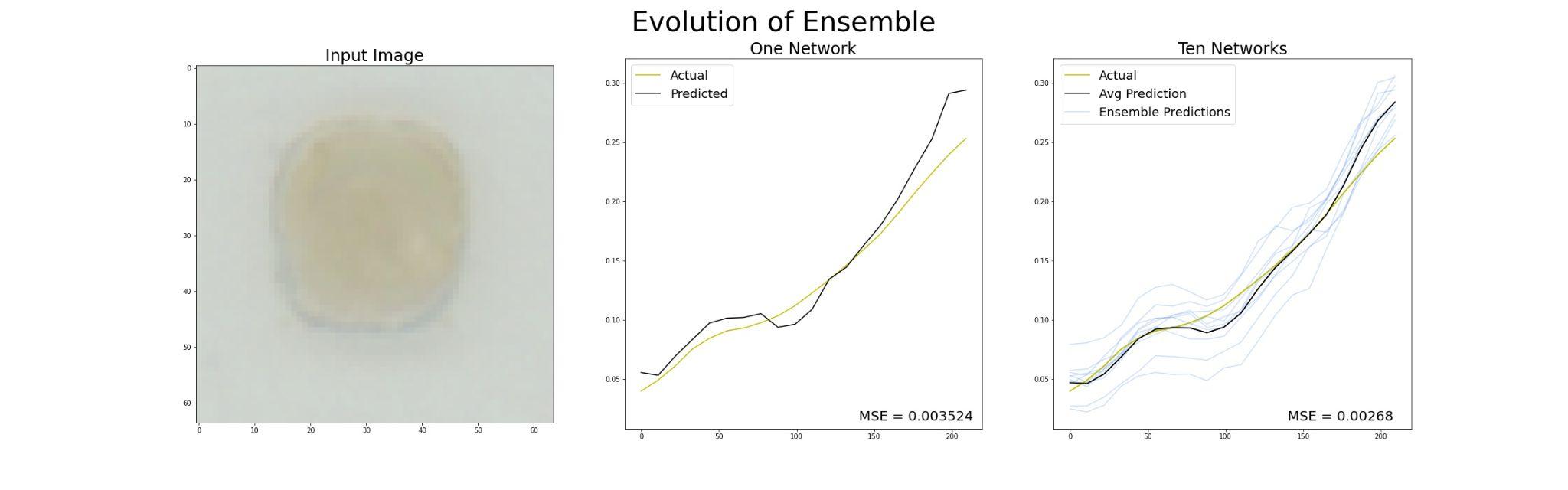
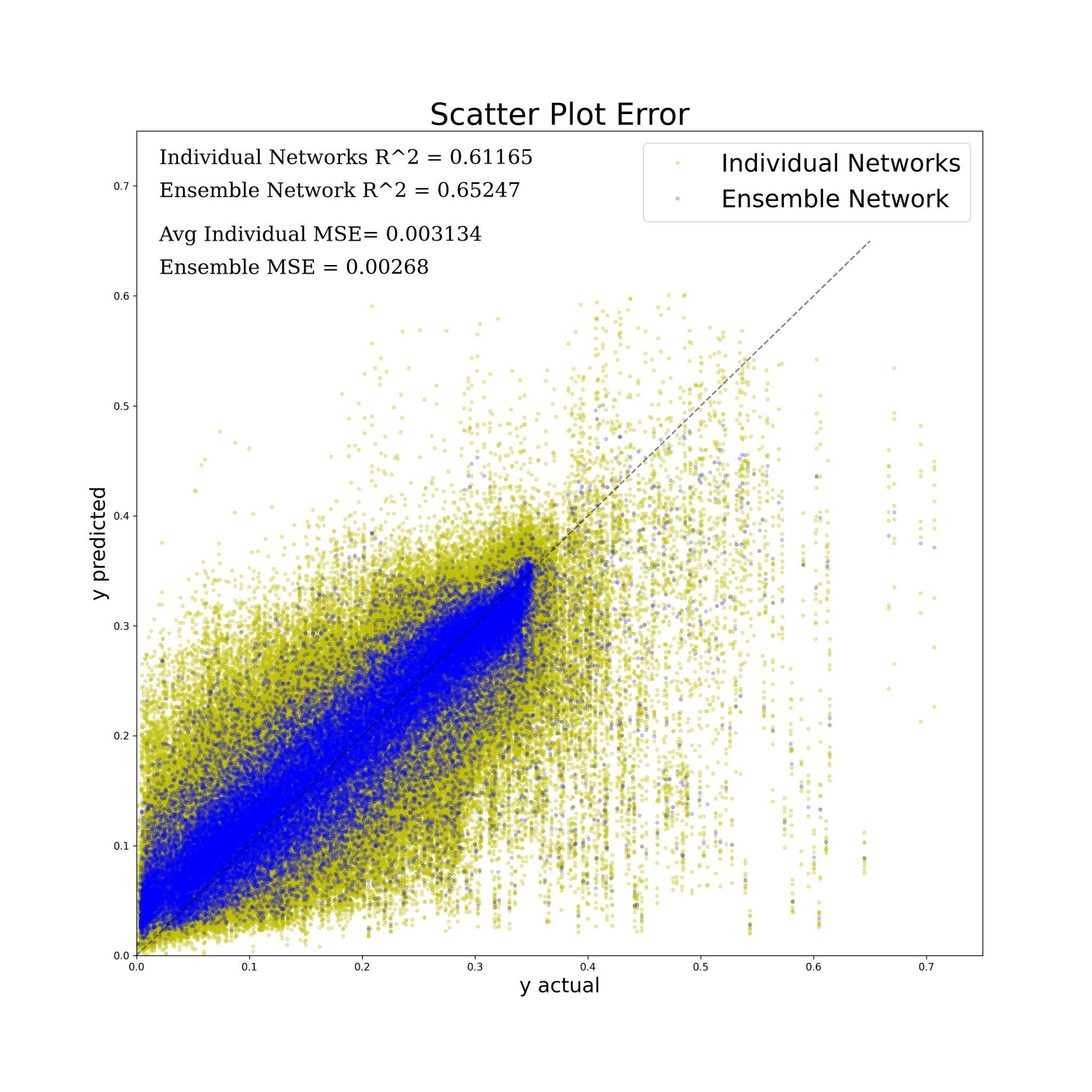
*Comparison of 10 individual network predictions, the averaged ensemble prediction, and ground truth spectra for a singular spectra image.* 

Figure 17 shows a sample metal oxide from the dataset on the left, the prediction of a our singular network in the center for reference, and the ensemble predictions and finalized prediction on the right. There are 10 predictions from the individual neural networks, which are averaged in the ensemble to produce a single output. From this figure alone we can see that the ensemble has already improved our model. First, the averaged predicted spectra is much smoother than any of the single network predictions. Second, it is clear that the MSE of the ensemble prediction model is even lower than the single network’s MSE, which was already low. This fortifies that our prediction model is giving accurate graphs.

With the ensemble made it is important to find a way to evaluate how well the model performed. Since the output is regression and not categorical, this method is not immediately apparent. To start, we recreated the scatter plot error graph that we made for the single neural network in Figure 14 to include all of the individual predictions as well as the final ensemble output. This is shown in Figure 18. The characteristics of how to read the graph are the exact same as in Figure 14.

In Figure 18 the yellow points represent every prediction of all 10 individual neural networks. For this reason the R^2 value of the individual networks increased to 0.61165, which is the average of all the yellow points. In addition, the MSE of the individual networks decreased to 0.003134, again due to to averaging all of the individual networks. On this graph the averaged prediction from the ensemble is plotted in blue, and it is visually apparent that the blue cloud is more clustered around the line y = x than the yellow cloud. This is expected since the ensemble prediction is the average of all 10 individual predictions. Also, the R^2 value of the ensemble is 0.65247, which is higher than the individual networks’ R^2, and the ensemble MSE is 0.00268, which is lower than the individual networks’ MSE. Based on how these numbers changed we are able to conclude that using an ensemble improve the accuracy of our overall predictions by lowering the amount of uncertainty in the model.

Figure 18. Symmetric Error Graph of ENN vs NN

*Scatter plot error visualization comparison of individual network predictions and the averaged ensemble neural network’s final prediction with associated R^2 and MSE values.* 

**Uncertainty Quantification**

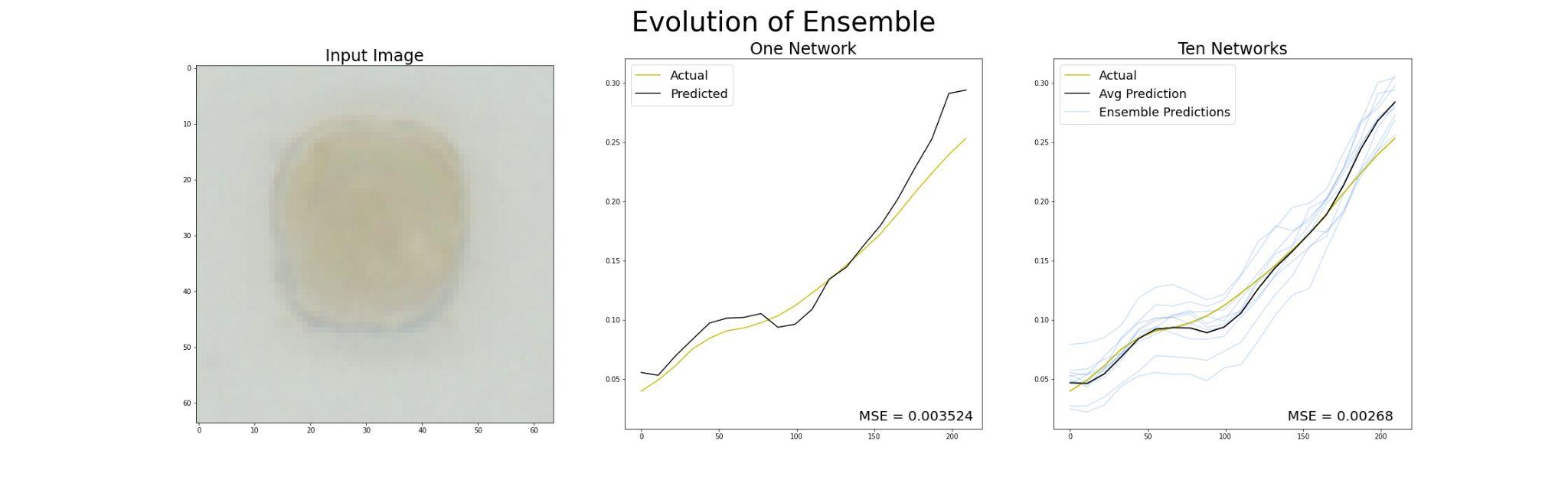
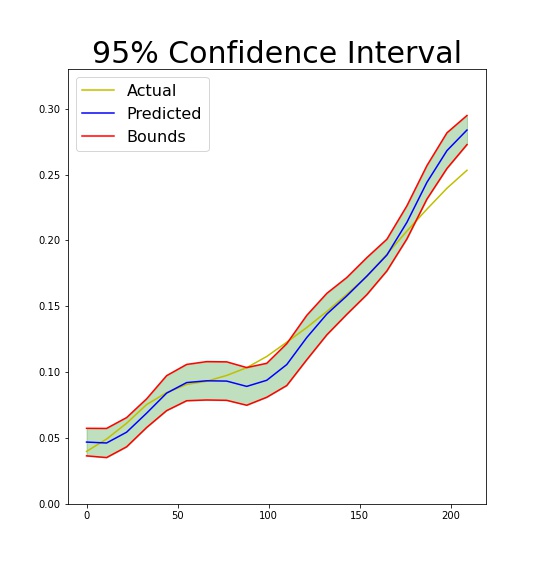
A problem of neural networks is that they are often referred to as a “black box algorithm” due to the unknown weights and biases assigned to variables. There is a lack of reliability, robustness, transparency, and interpretability when these prediction models are used. Networks are searching for patterns to relate the inputs to the outputs and produce the optimal model for the given dataset, yet, these models do not produce or measure the error bars associated alongside predictions. So, if a prediction is run through the model, the user is unsure of how confident the model is pertaining to that prediction. Without a measure of confidence, the prediction is less trustworthy to a real-world problem. Thus, it is essential that a model outputs an uncertainty alongside a prediction for it to be applied to a problem.

There are two types of uncertainty in UQ: Aleatoric uncertainty and Epistemic uncertainty. Aleatoric uncertainty is uncertainty in the data generating process and cannot be reduced even with more training data. Often datasets used to train the models have a certain amount of noise present. This model is then trained on the training data alongside the noise and adjusts the weights according to the data. This creates a default in the model due to the slightly adjusted weights in which consider the noise. Predictions become less certain due to the training data of the model. By contrast, Epistemic uncertainty is the ignorance of the predictive model due to the lack of training data [3]. Neural networks require enough data to build an effective model. The more training data it is fed, the more relationships that the network can learn and implement into the structure.

**Implemented Uncertainty Quantification**

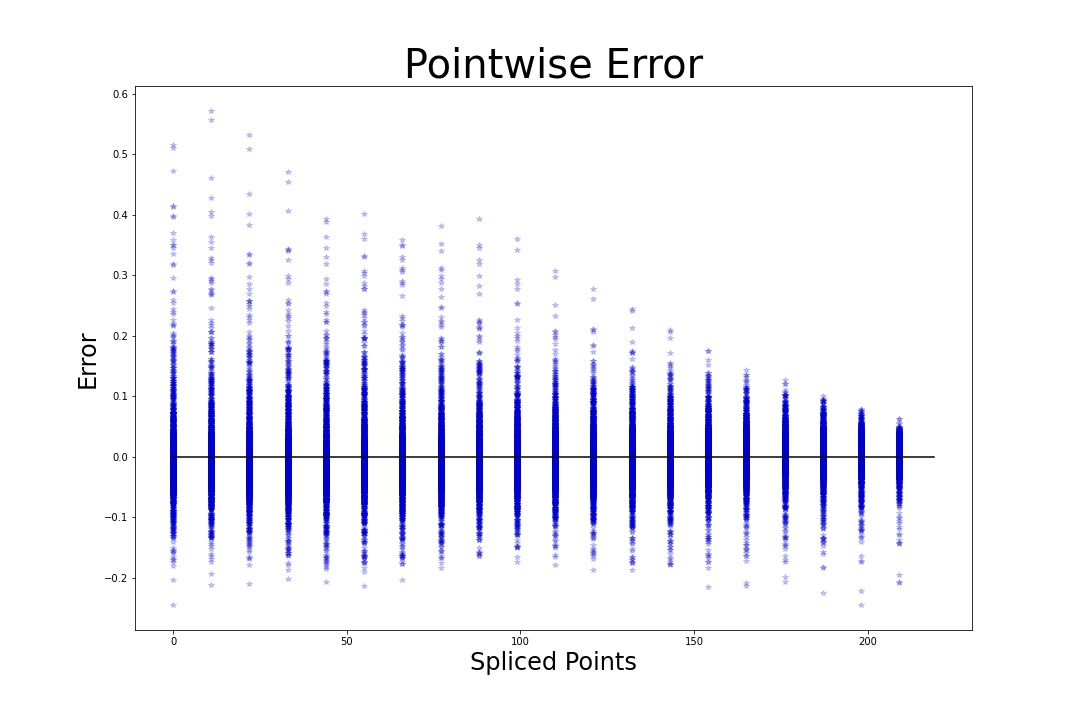
Along with providing better predictions, ensemble networks themselves are a method of Uncertainty Quantification. Since multiple models have contributed to the final output it is important to investigate the distribution of the individual networks predictions. To accomplish this we made a 95% confidence interval around the 20 points predicted for each spectra. This can be seen in Figure 19 and 20 below.

Figure 19. Ensemble Predictions Figure 20. 95% Confidence Interval

*Comparison of individual network predictions (left) with changing area of a confidence interval (right).*

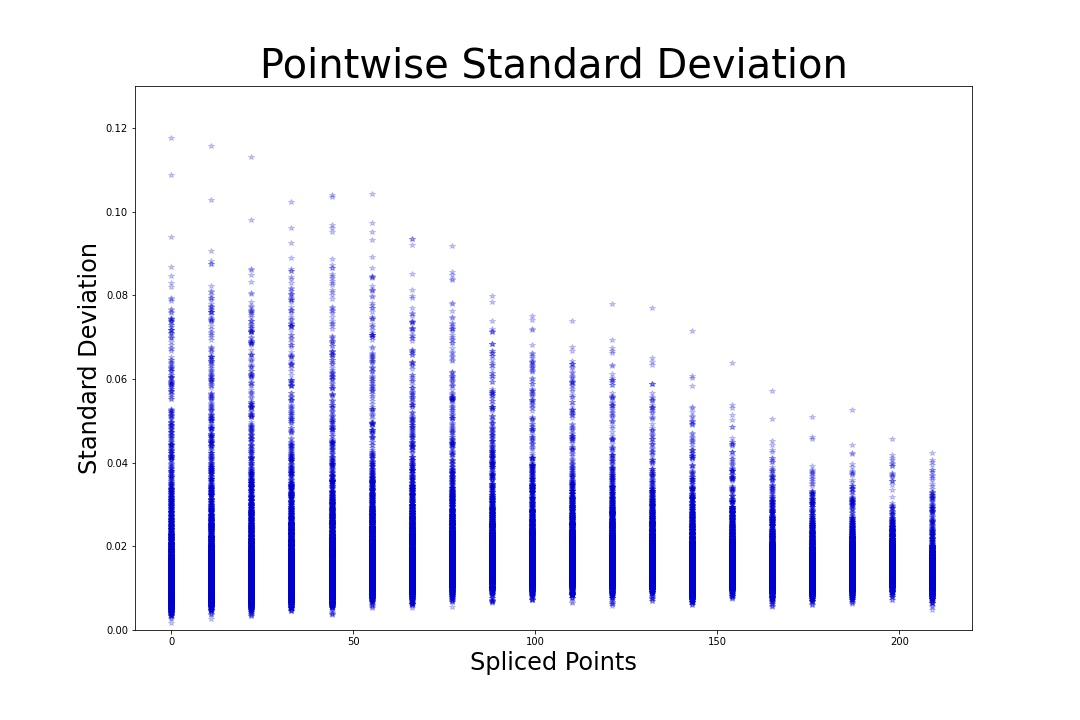
Above we can see that the spread of the ensemble predictions in Figure 19 match the changing area of the confidence interval in Figure 20. By creating these confidence intervals we can see that the later points of the spectra are closely distributed since the area is relatively small. Conversely, the earlier points are more widely distributed since the area is relatively large. Now, let’s look at this idea across the entire dataset.

In Figure 21 and Figure 22 below we separate the data by each of the 20 points of the spectra to see the relationship between error and uncertainty. Standard deviation is measured in Figure 22 as it is the sole factor in determining the size of the confidence interval.

Figure 21. Point-Wise Error of Ensemble

*Displays the point-wise error in each of the 20 points in each individual prediction in the ensemble network.*

Figure 22. Point-Wise Standard Deviation of Ensemble

*Displays the point-wise standard deviation in each of the 20 points in each individual prediction in the ensemble network.* 

In Figure 21 we see that the error starts out widely distributed but becomes closer as we get to the later points. This type of behavior is also seen in Figure 22. This shows that the error of the network is seemingly in proportion with the uncertainty. This is important as the error plot can only be made when the true value is known. However, if the true value is known then we have no need for a neural network. The plot of the standard deviations can be made even if the actual value is unknown as it is made from the network outputs. This will help us determine a standard deviation cutoff for which predictions to trust and which ones to remove.

**Conclusion**

In conclusion, our singular networks were able to make spectra graph predictions

consistently within 10% error to the actual spectra. Additionally, adding an ensemble reduced the mean squared error of the model while increasing the R² value of the predictions with respect to the actual spectra. Finally, we were able to use uncertainty quantification techniques such as confidence intervals and pointwise error graphs to determine the robustness of our model.

**References**

[1] H. S. Stein, E. Soedarmadji, P. F. Newhouse, Dan Guevarra, and J. M. Gregoire,

“Synthesis, optical imaging, and absorption spectroscopy data for 179072 metal oxides,” *Scientific Data*, vol. 6, no. 1, Mar. 2019, doi: 10.1038/s41597-019-0019-4.

[2] By: IBM Cloud Education, “What are convolutional neural networks?,” IBM. [Online]. Available: https://www.ibm.com/cloud/learn/convolutional-neural-networks. [Accessed: 09-Jun-2022].

[3] M. Cerliani, “Neural Networks Ensemble,” *Medium*, Jul. 01, 2020. https://towardsdatascience.com/neural-networks-ensemble-33f33bea7df3 (accessed Jun. 01, 2022).