**Ensemble Deep Learning**

Emily Diegel

Jesse Adams and Margaret Lund, Ph.D. 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 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. Understanding neural networks is key to making them less of a black-box algorithm and having confidence in them for the future.*

**The Nevada National Security Site**

The Nevada National Security Site 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 the United States stockpile remains safe, secure, and reliable. They perform underground classified nuclear experiments along with 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 create and introduce new deep learning techniques to be used to analyze radiographic images and provide the necessary error bars alongside, to ensure that the nation's stockpile remains effective and safe. Neural networks provide the opportunity to analyze large amounts of image data to produce a regressive output. They are an algorithm modeled after the neurons 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 neural networks pose is the theory behind the black box algorithm. Essentially a neural network creates a black block algorithm because the weights and the biases are unknown in the network, leaving parts of the model unknown by the creator. This algorithm provides no real method to display an error bar on the predictions provided by the model. Without error bars, there is no telling how accurate or dependable these results are. In the industry world, it is extremely important to provide an error bar alongside a model because these models are going to be used on real-world applications and most likely will impact an abundance of lives. For this project, the goal is to determine a method to quantify uncertainty in the ensemble neural network, striving to be able to provide industry with the ability to place a number on validity in predictions made by neural networks.

The tasks of this project include:

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

**Dataset Description:**

Synthesis, optical imaging, and absorption spectroscopy data for 179072 metal oxides [1]

This data set includes 178994 distinct materials samples spans 78 distinct composition spaces, includes 45 elements, and contains more than 80000 unique quinary oxide and 67000 unique quaternary oxide compositions. [2]

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 refection configuration at 1200 dpi corresponding to a rate of 2.0 cm2 s−1 or 0.019 s per sample
* All images were rescaled to 64×64 pixels via the python image library (pillow)

Output: Spectra graph (220 Fractional Absorb 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 2 attributes in the hdf5 container accompanying this manuscript. [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 | Platebead Scanner |
| Spectra | Fractional Optical Absorbance Spectrum | 0-ca. 0.5 | (220, 180902) | Fractional Absorb. Coefficient | Dual-Sphere Optical Spectrometer |

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

A picture containing diagram

Description automatically generated

In the displayed spectra outputs, the graph uses 220 points to plot the shown spectrums. To simply this output, every 11th point was used in the spectra, therefore only 20 points on the graph rather than 220. Minimal accuracy was lost during this preprocessing of the data because the graphs maintained their original shape. This method was used in order to reduce computation time of the the neural network. Rather than predicting 220 points on a graph, it produces 20 points demonstrating practically the same curve.

Figures 3 and 4. Absorption Spectrums for Metal Oxide

A picture containing shape

Description automatically generated A picture containing icon

Description automatically generated

These figures show the minimum difference between only plotting every 11th point or using all 220 points provided in the original dataset. A model can produce 20 outputs accurately faster than 220 points. The curves maintain their shape, allowing this method to be utilized in the neural network.

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.

**Convolutional Neural Networks**

Convolutional neural networks represent the connectivity of the neurons of the human brain. They 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.

Figure 5. Convolutional Neural Network

Text

Description automatically generated

Figure 5 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 64 by 64 (pixels) by 3 (RGB) images with an output of 20 absorption points of a spectra.

**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. [3]

Dense - Computes a weighted average of the input and pass through a non-linear 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 batch size represents the number of samples in a singular batch. Based on the size of the training data, and the batch size, the number of batches can be calculated. The number of epochs represents the number of times each batch is trained. For example, if there are 1000 batches with 100 epochs, each of the 1000 individual batches of data will be trained different times which is different than training all the data in at once in one batch. Batch size allows the network to train smaller samples of data multiple times. This allows the model to learn more patterns in smaller samples of data.

For the implemented convolutional neural network, originally an epoch of 50 was used to train the model. Through further observation, it was shown that 50 epochs produced overtrained models and the graphs produced were unable to recognize patterns in the validation data. The function early stopping was therefore incorporated into our model to prevent overfitting. This function stops the training at the most optimal point where training loss and validation loss begin to increase. With the spectroscopy dataset, for the model to perform well and produce close to the desired results, more less??? epochs were necessary. The batch size for the implemented CNN is 32. Through model testing, the larger the batch size, the more the predictions varied from the label. Lowering the batch size allowed the model to learn more patterns within the inputs and the outputs and assign more accurate weights and biases to produce the output.

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 40,000 images. 80 percent of the images were used for training and 20 percent were used for validation.

* Number of training batches: 1,000 with 32 images in each batch
* Number of validation batches: 250 with 32 images in each batch

Figure 6. Shuffle Split Function Implementation



In order to split our data 80/20, the function Shuffle Split from sk.learn under the model selection packages was implemented in the network. This function randomizes the split of the training and testing data based on the number of splits needed. For instance, if there are multiple networks, the training and testing will have slightly different parts of the data but will still 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, 20,000 testing images were used in making predictions.

During testing, the dataset was completely randomized to avoid clusters of similar data that would overfit the model to a specific type of composition of metal oxide. Also, a bucketing technique was implemented in two of the models. Bucketing takes groups of 50 images in different parts of the overall data based on the subset size. Instead of taking the first 40,000 images, it takes 40,000 in groups of 50 in randomized sections of the overall dataset. This is necessary due to the lack of GPU RAM in order to process the whole dataset.

For model 4 Gaussian noise was implemented into the model. Gaussian noise is a layer in the network that introduces noise through the application of standard deviation to the inputs of that layer. It prevents overfitting and functions as a regularization layer in training but has no affect when using the model to make predictions.

**Loss Function**

Loss functions are used to optimize the model in order to minimize the loss function. The goal is to achieve zero loss with any loss function, yet that is not probable for certain models. This function allows the model to define the best weights for the given data. The loss function is directly related to the activation function used in the output layer of the convolutional neural network. The output layer is a choice about the framing of the prediction problem, and the choice of the loss function is a way to calculate the error for a given framing problem.[4]

Loss functions are applied after a batch has been trained on one epoch. Based on the calculation of the loss function, the network will evaluate and adjust the weights applied to data to minimize the loss function. Then the network will train another batch and adjust the weights according to the loss function.

Diagram

Description automatically generated

Figure 6. Loss Function Diagram [4]

The loss function used in the implemented convolutional neural network is mean squared error.

Figure 7. Equation of mean squared error

MSE is used for regression outputs such that the values are continuous. It calculates the square difference between the true and predicted values. Ideally, zero MSE is desired, concluding that there is no loss. MSE is calculated throughout the training process and used to evaluate how well the network is training.

Figure 6. Table of 4 Neural Network Models Trained Differently with the Same Architecture

|  | Model 1 | Model 2 | Model 3 | Model 4 |
| --- | --- | --- | --- | --- |
| #Training Images | 178994 | 40000 | 40000 | 40000 |
| Batch Size | 32 | 32 | 32 | 32 |
| Epochs | Early Stop | Early Stop | Early Stop | Early Stop |
| Loss Function | MSE | MSE | MSE | MSE |
| Bucket Technique | No | No | Yes | Yes |
| Gaussian Noise | No | No | No | No |

**Model 1 Results- to come(will justify further!!!!!!!!!) \*\*\*\*\*will update\*\*\*\***

**Model 2 Results**Chart, line chart, histogram

Description automatically generatedChart, line chart

Description automatically generated

**R^** Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

**Ensemble Neural Networks**

Ensemble neural network combine the decisions of multiple neural networks in parallel to produce a single output for the ensemble network. Outputs may be combined using different methods including taking the average or using a neural network to combine the outputs. Ensemble networks provide the opportunity to use different sections of the dataset in different networks. Each individual neural network in the model can have a slightly different architecture and different training data to output varying results instead of using one singular output from a neural network. This concept can be seen below in figure 4.

Figure 10. Diagram of Ensemble Neural Network [5]

Diagram

Description automatically generated

The brain displayed on figure 10 is where the outputs are combined using a chosen method to produce a singular desired output. If a neural network is used to combine the results of the individual models, the outputs of the individual models would then become in the inputs of the neural network. The outputs used to train the final neural network would be the original outputs from the given dataset. The final neural network would be strictly regressive because the inputs and the outputs are numerical. Only dense layers will be used in this network to produce the desired output.

**Implemented Ensemble Neural Network**

For the implemented ensemble neural network, ten identical convolutional neural networks were trained in parallel. Each was trained on 90,000 of the same images. The architecture of the individual networks include: one convolutional layer was utilized, one max pooling layer, one dropout layer, one flatten layer, and a total of four dense layers. Each individual model in the ensemble neural network initializes differently even with the same architecture and training data therefore results in slightly varied predictions.

Training: Loss function:

* 80 percent training Mean squared error
* 20 percent validation Metrics:
* Epochs = early stopping Mean squared error
* Batch size = 25

Shuffle Split

Figure 11. Loss Function for ENN Figure 12. Loss Function Metric for ENN

Chart, histogram

Description automatically generatedChart, diagram, histogram

Description automatically generated with medium confidence

Figure 11 demonstrates the MSLE over the 5 neural networks in the ensemble. This loss function was used to optimize the model during the training whereas the metric, MAPE, was measured during the training but had no affect on the optimization of the network. The training and validation data was tracked for both metrics. The yellow representing the validation data and the blue representing the training data. The loss functions decrease as the network is trained because weights and biases are adjusted according to the value of MSLE. The smaller the MSLE, the better the network is preforming.

\*\*\*WILL PROABALY CHANGE\*\*\*

Chart, line chart

Description automatically generatedA picture containing text, person

Description automatically generated

\*\*\*\*graph to show accuracy\*\*\*

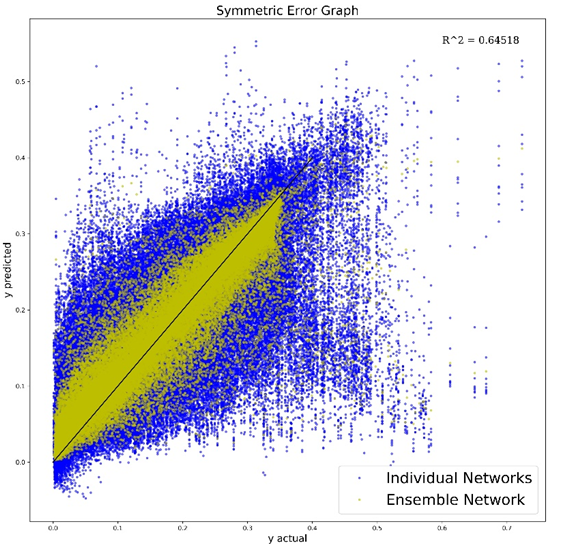
\*\*\*show how ENN is better than NN\*\*\*

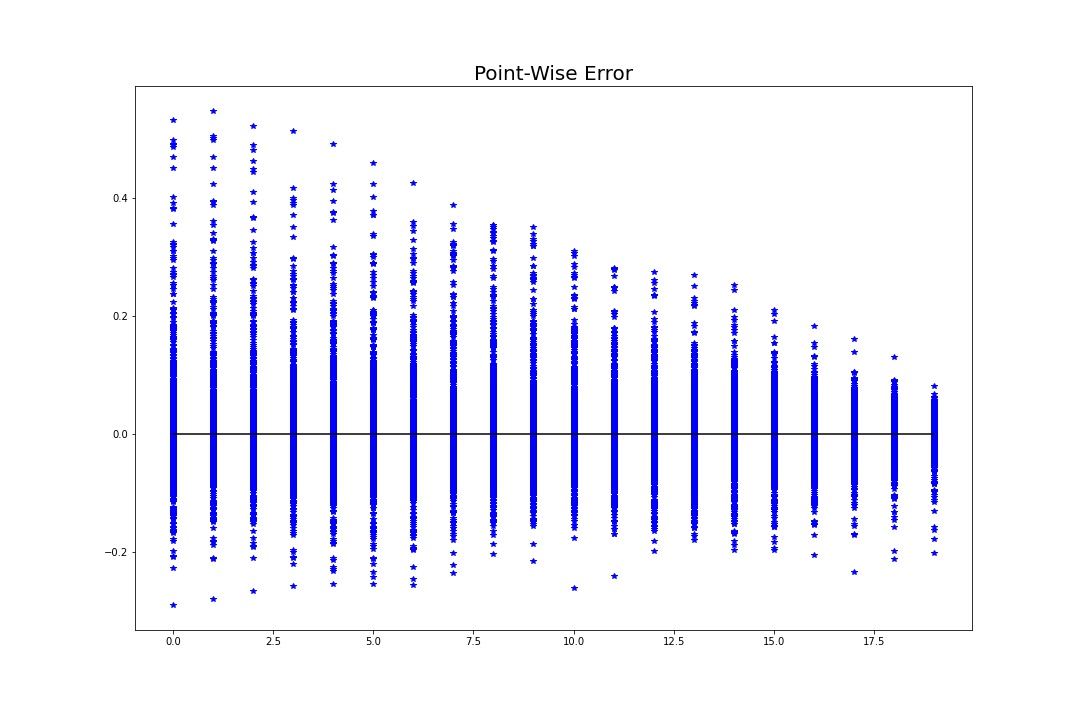
**Results**

**With the ensemble made it is important to find a way to evaluate how good the model was. Since the output is regressional and not categorical this method is not immediately apparent. We decided to compare the actual values against the predicted values of the network. This means that**

**points that are vertically closer to y = x are more accurate. Points above y=x are overestimated and points below are underestimated. The drawback to this method is that we lose the structure of keeping the spectra together. This graph doesn’t tell if one or more points in each spectra is consistently wrong. Thus, we also created the point-wise error graph. This shows the error for every spectra at each of the points. For this graph we want the points to be along y=0. Since the graphs are generally increasing, we expect that the error will increase as we move to the right. However, many of them have a spike in the beginning or middle and then flatten out to the right. We have not figured out the cause for this yet.**

**Figure # Symmetric Error Graph**

****

****

**Uncertainty Quantification**

Neural networks pose a very important issue due to the design of the models. 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 due the nature of their architecture and training process. The network is essentially searching for patterns to relate the inputs to the outputs and produce the optimal model for the given dataset. One key aspect that these models do not produce, or measure is the error bars associated alongside predictions. For instance, 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 unapplicable to a real-world problem. It is essential that a model outputs an error bar alongside a prediction for it to be applied to a problem.

The uncertainty quantification is how confident a network is about its output and how much it knows about the input given. There are two types of uncertainty: 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 adjust 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. Epistemic uncertainty is the ignorance of the predictive model due to the lack of training data. [5] Neural networks require a sufficient amount of data in order to build an effective model that can make confident predictions. The more training data it is fed, the more relationships that the network can learn and implement into the structure. Typically models with large amounts of training data perform better than models with small amounts due to the number of times the network can adjust its weights throughout training to fit the data.

**UQ Approach/Methodology**

Ensembles are a method of quantifying uncertainty due to the multiple models that predict in parallel. Instead of one model outputting the prediction, multiple models have a “say” in the output. In other words, each model contributes to the prediction and not one model decides the prediction. Ensemble neural networks train multiple neural networks with different random initialization and different shuffling of the training dataset. The uncertainty will be the variance of the models’ predictions. Even though the architecture of the individual models is the same, the weights are different from the start of training. Ensembles provide less variance in the predictions in comparison to the actual values due to the combination of multiple decisions made by multiple networks.

Given 5 different spectra outputs from the 5 networks working in parallel, the arithmetic mean is calculated to determine the final output. Variance occurs across the individual networks and can be statistically shown through standard deviation and confidence intervals.

Figure 16. and Figure 17. Standard Deviation of Ensemble Prediction and Confidence Interval of 95% of Ensemble Prediction

Chart, histogram

Description automatically generated

Figure 16 shows standard deviation of 5 network predictions. The blue line represents the mean of the 5 absorption spectra’s outputted by the individual neural networks. The two red lines represent the upper and lower standard deviations from the mean. The green area represents the area of uncertainty where the models vary. The larger the green area is between the two red lines, the less certain the network is about the prediction.

Figure 17 shows a 95 percent confidence interval over

References

Saha, S., 2022. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. [online] Medium. Available at: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> [Accessed 26 May 2022].

<http://www.math.utah.edu/~palais/pcr/spike/Evaluating%20the%20Goodness%20of%20Fit%20--%20Fitting%20Data%20(Curve%20Fitting%20Toolbox)%20copy.html>

- good reference for goodness of model

[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] Chem. Sci.,2019,10,47

[3] 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].

[4] A. Rakhecha, “Importance of loss function in machine learning,” Medium, 16-Sep-2019. [Online]. Available: https://towardsdatascience.com/importance-of-loss-function-in-machine-learning-eddaaec69519. [Accessed: 09-Jun-2022].

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

[6] A. Mustapha, “Are you confident about that, neural network?,” Medium, 31-Dec-2020. [Online]. Available: https://towardsdatascience.com/are-you-confident-about-that-neural-network-21e87f77b94b. [Accessed: 15-Jun-2022].

[7] P. Sountsov, C. Suter, J. Burnim, and J. V. Dillon, “Regression with probabilistic layers in tensorflow probability,” *The TensorFlow Blog*. [Online]. Available: https://blog.tensorflow.org/2019/03/regression-with-probabilistic-layers-in.html. [Accessed: 16-Jun-2022].