

NDnano Summer Undergraduate Research 2024 Project Summary

1. Student name & home university:

Joseph Park, University of Notre Dame

2. ND faculty name & department:

Professor Ningyuan Cao, Electrical Engineering

3. Summer project title:

Circuit and system design for collaborative machine learning

4. Briefly describe new skills you acquired during your summer research:

This summer, I solidified my understanding of neural networks by learning more in-depth about cost functions, gradient descent, and backpropagation. I was also introduced to Bayesian neural networks which incorporate uncertainty in classification/prediction by using weights with a normal probabilistic distribution. Next, I learned how to create a neural network from scratch that can predict a sine wave and learned what parameters lead to the best results. I also learned how to retrain a Bayesian neural network that takes MNIST data by coming up with my own retraining method. Additionally, I learned how to perform a threshold sweep and to analyze it by doing a cost benefit analysis. Finally, I learned how to present my findings in a professional manner.

5. Briefly share a practical application/end use of your research:

The research that I conducted can be used to quantify and minimize uncertainty in a multitude of industries. One potential industry is the military. For example, a military surveillance drone may be sent out to a location to determine whether there are people in the area. However, due to obstacles such as smoke, the drone may be unsure. Through my research, the drone can then quantify this uncertainty and if it is higher than a specified threshold, it can then move to a different location to minimize this uncertainty. Finally, it will then be able to detect whether there are people in the area.

6. 50- to 75-word abstract of your project:

Basic neural networks have a problem of being overconfident when trying to classify or predict with unfamiliar data. Bayesian neural networks solve this issue by retraining the model when the uncertainty is higher than a specified threshold. Using uncertainty threshold sweeps on Python, we were able to increase the accuracy of a model that classifies both familiar and unfamiliar data from MNIST from ~78% to 99+% after ~5 retraining iterations.

7. References for papers, posters, or presentations of your research:

[1] M. Magris & A. Iosifidis, "Bayesian learning for neural networks: an algorithmic survey"

[2] S. Thakur, H. Hoof, J. Higuera, D. Precup & D. Meger, “Uncertainty Aware Learning from Demonstrations in Multiple Contexts using Bayesian Neural Networks”

[3] C. Blundell, J. Cornebise, K. Kavukcuoglu & D. Wierstra, “Weight Uncertainty in Neural Networks”

[4] J. Gawlikowski, C. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler & X. Zhu “A Survey of Uncertainty in Deep Neural Networks”

One-page project summary that describes problem, project goal and your activities / results:

A major challenge with machine learning models is their “black box” nature. This obscurity may cause difficulty in establishing trust between machine learning models and human users. One key aspect of this challenge is the inability of these machine learning models to effectively communicate the confidence or uncertainty associated with their classifications or predictions. To combat this limitation, statistical machine learning approaches, like Bayesian inference and Bayesian neural networks (BNN), offer a unique advantage. Rather than focusing solely on finding an optimal model with fixed weights, they aim to learn the distribution of model parameters. This intrinsic capability allows Bayesian approaches to quantify and convey the uncertainty related to their learning processes, thereby fostering enhanced trust and collaboration between machine learning models and human users. While powerful, such statistical machine learning approaches are notably resource intensive. The requirement for massive random number generation, coupled with extensive computation to assess inference confidence, can be prohibitive. This computational burden makes it challenging to deploy on resource-constrained edge devices, which are often the primary platforms directly interfacing with human users.

Therefore, the goal of our research is to use Bayesian neural networks (BNN) to quantify and convey uncertainty while maximizing the accuracy and minimizing the energy used. More specifically, my goal is separated into two parts. First, to create and implement a retraining method that can be used to retrain the machine learning model when it is uncertain. And second, to identify the ideal uncertainty threshold which determines when the retraining should be triggered for maximum accuracy and minimum energy-use.

We used the MNIST dataset which contains samples of handwritten digits between 0-9 and trained the model on the digits 0-7 and analyzed how the model deals with unfamiliar digits 8 and 9. To do this, we fed the training data (0-7) into our initially untrained BNN model. Next, we split the testing data (0-9) into validation data (70%) and final testing data (30%). We then fed the validation data into the now trained BNN model and identified the uncertain samples and retrained the model based on these samples along with the original training data. We repeated this process for multiple retraining iterations. Finally, we then fed in the final training data to the updated BNN model to find the accuracy of the model.

Our next step was then to perform uncertainty threshold sweeps to find the ideal uncertainty threshold that we could implement into our retraining method. We based the ideal uncertainty threshold on two factors: accuracy and number of samples needing retraining. To do this, we first did an initial uncertainty threshold sweep to see how each threshold performs in terms of accuracy and size of retraining dataset. Then, we used an equation (Figure 1) to pick the best threshold and actually retrain the model (with both the original training data and high uncertain samples) based on this threshold. Finally, we performed multiple threshold sweeps (Figure 2) as well as multiple re-trainings based on the best threshold for each of these sweeps. Ultimately, we were able to obtain an accuracy of 99.00% after five iterations (Figure 3).

Figure 1: Equation used to determine best threshold

```
# calculate scores
weight_accuracy = 0.8
weight_retraining_size = 0.2
scores = weight_accuracy * (np.array(accuracies) / 100) - weight_retraining_size * (np.array(num_uncertain_samples) / 7000)
```

Figure 2: First 3 iterations of uncertainty threshold sweeps

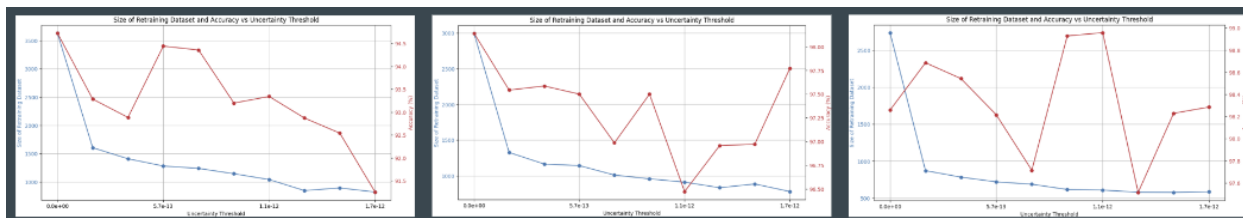


Figure 3: Results of each uncertainty threshold sweep

Iteration	Threshold	Uncertain Dataset Size	Accuracy (Validation Set)	Accuracy (Final Test Set)
0	N/A	0	~74% - 78%	~74% - 78%
1	7.6e-13	1239	94.36%	94.80%
2	1.7e-12	780	97.77%	98.30%
3	1.1e-12	608	98.96%	98.57%
4	1.1e-12	764	98.91%	97.57%
5	1.5e-12	1039	99.13%	99.00%