BINGO!: A Novel Neural Network Pruning Mechanism to Allow For Physical Computing in AI Education

Aditya Panangat
Flower Mound High School
adipanangat@gmail.com

Abstract

Over the past decade, ML and AI skills have become increasingly necessary for the workplace. Unfortunately, students and self-learners currently lack engaging ways to acquire these skills. Historically, hands-on physical computing has served as an effective and engaging method for teaching programming. However, physical computing hardware, such as microcontrollers, often lacks the memory and computing power to run large ML and AI models. As a result, learning AI becomes less engaging, leading many to feel apathetic toward the idea. Current methods for shrinking models to fit on microcontrollers, such as iterative magnitude pruning, have shown great accuracy but require an iterative training sequence that is incredibly computationally taxing. To solve this problem, BINGO is introduced. BINGO, during the training pass, studies specific subsets of a neural network one at a time to gauge how significant of a role each weight plays in contributing to a network's accuracy. By the time training is done, BINGO generates a significance score for each weight, allowing for insignificant weights to be pruned in one shot. BINGO provides an accuracy-preserving pruning technique that is less computationally intensive than current methods, allowing for a world where students can learn about AI through engaging physical computing activities.

Motivation

As machine learning (ML) and artificial intelligence (AI) become increasingly valuable skills for the future job market, it has become imperative that students and children learn how to utilize AI. Unfortunately, AI is a complex, intricate topic with a large learning curve. As a result, Maximova (2024) finds that students and self-learners often avoid learning AI topics because they find them daunting, irrelevant, and/or boring.

To solve student apathy towards programming, *physical computing* has been utilized. *Physical computing* refers to the act of building hands-on, interactive systems using hardware, such as Arduino, Raspberry Pi, etc. This approach has empirically engaged students and motivated them to learn programming. Love and Asempapa (2022) concluded that 70% of middle school students found the idea of learning to code through hands-on physical computing activities intriguing. Moreover, Udvaros and Forman (2023) found that students achieve better educational results on average when learning programming using visual tools such as microcontroller units (MCUs).

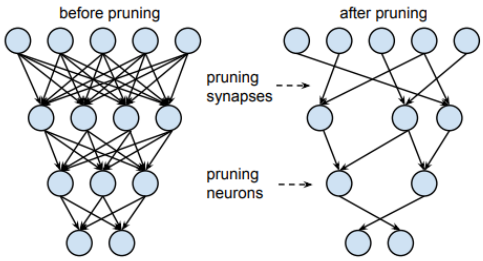
While physical computing platforms like the Arduino, Raspberry Pi, and micro:bit have made programming in general engaging for students, their limited processing power and memory have made it difficult to incorporate AI models, which, as Pernes et. al (2024) find, often require more computational resources than these microcontrollers can support. As a result, Svoboda et. al (2022) find that low-memory microcontrollers are unable to run ML/AI models over 512 KB.

Currently, many ML models are simply too large to employ within a microcontroller. Thus, in order to allow students to learn about AI through engaging and effective physical computing activities, neural networks must be made smaller.

Related Work

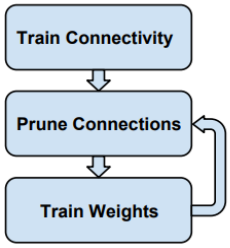
In order to make neural networks smaller, more memory efficient, and increase their ability to be utilized within microcontrollers, *neural network pruning* has been proposed. Neural network pruning attempts to find irrelevant weights within a model and remove them.

Most commonly, this is achieved through *magnitude-based* pruning, where, as Han et. al (2015) explain, weights with magnitudes near zero are pruned out of a model.



Unfortunately, according to Rad and Seuffert (2024), magnitude-based pruning has been shown to drastically reduce model accuracy at higher pruning rates because the mechanism prunes away weights important to the model. Even pruning just 10% of a network this way has been shown to result in overall model accuracy dropping by 30 percentage points.

Magnitude-based pruning can also be performed iteratively in a process called *iterative magnitude-based pruning* (IMP). A model is trained to completion and pruned, and then the remaining network is retrained, all to be pruned again.



Although IMP has been shown to be more efficient in preserving model accuracy, it comes with drawbacks. First, the process is slow, as it requires multiple training rounds. Second, when applied to larger models, it is far too expensive to iteratively train and retrain them. According to Li (2020), it costs OpenAI about 4.6 million dollars to train their GPT-3 model just once. Iteratively training and retraining GPT-3 would hence cost hundreds of millions of dollars. Although most models aren't as large as GPT-3, this shows that IMP isn't a viable option for reducing models to a size where they can be uploaded into an MCU. Students can not utilize engaging physical computing devices, such as Arduinos and Raspberry Pis, for AI education until AI models can be shrunk down in a way that is cheap and accuracy-preserving.

Methodology

Bingo Overview

Although IMP works extraordinarily well, its major pitfall is the fact that it requires multiple training sessions in order to complete pruning. The ideal technique to reduce the size of models, then, is one that does not require additional training sessions to be done.

So, I introduce BINGO, a novel neural network pruning technique that reduces the size of neural networks not by iteratively finding unnecessary weights through additional training sessions, but by identifying which weights will eventually be unnecessary while the original model itself is training.

Simply put, BINGO solves the issue of needing additional iterative training sessions for model pruning by collecting information during the original training session. With this information, after the model is trained, all unimportant weights can be pruned from the model in one shot. This means that computationally expensive, iterative training of the model is not necessary, drastically reducing the computational cost and time required to shrink models down to MCU size.

Bingo Design

BINGO achieves computationally efficient pruning by contributing 2 novel, creative, and effective mechanisms to the literature.

1. A technique to isolate subsets of a neural network called *lottery ticket searching*.

This technique borrows from neural network dropout, a regularization technique proposed by Srivastava et al. (2014), which attempts to make models more generalizable by deactivating a random subset of neurons during each training pass. In a similar way, lottery ticket searching, just after each training pass, sets a subset of *weights* back to their initial values. It is important to note that this process does not truly affect the weights in the network. Instead, just after each real training pass, this technique simply simulates a "fake training pass," setting some weights back to their initial assignments to simply keep score of what *would have* happened if those were the true neuron weights.

2. A new measurement called *weight significance score*.

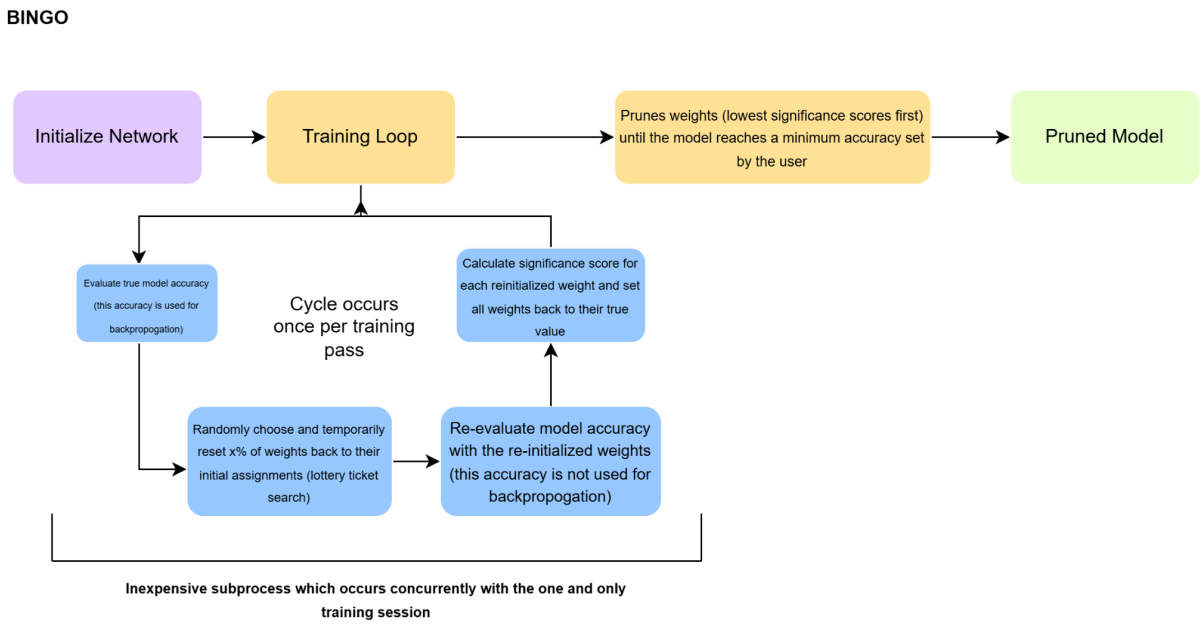
Based on the way that the accuracy of a model is calculated to change during each lottery ticket search, BINGO calculates a weight significance metric for each weight in the neural network. The weight significance for each weight is scaled from 0 to 1, with higher scores representing weights that are believed to contribute more significantly to model accuracy. This metric is calculated using data that is gathered from lottery ticket searching, which keeps track of what happens to accuracy when certain subsets of weights are set back to their initial, untrained values. Data from each lottery ticket search helps calculate a total significance score for every weight that was temporarily reset to its initial value. Initially, weight significance is set to 0.5 for every weight. After a lottery ticket search, the following calculation for weight significance (S_w) is used to update significance scores for the weights that were temporarily reset to their initial values:

$$S_w = \frac{S_p + \left(1 - \frac{\left|A_{before} - A_{after}\right|}{A_{before}}\right)}{2}$$

Where:

- S_w is the weight's new weight significance score
- S_p is the weight's previous weight significance score
- A_{before} is the model's accuracy before resetting the weight to its initial value
- A_{after} is the model's accuracy after resetting the weight to its initial value

Finally, after the model is finished training, BINGO will prune weights until a certain minimum model accuracy is reached, pruning weights of lowest significance score first. Thus, BINGO prunes down neural networks to a size where they can be uploaded to MCUs to help educate students.



Results

An ML model was trained on the MNIST dataset produced by LeCun (2009). The following results table compares pruning of this model by IMP and BINGO.

Method	Training + Pruning Time (min)	% of Total Weights Pruned
IMP	122	85%
BINGO	17	71%

Table 1: Pruning results on an MNIST-trained model. IMP and BINGO were both programmed to prune until the model fell to 90% accuracy from an initial 97%

Discussion and Conclusion

When told to prune a model down to the same accuracy, I find that BINGO prunes 86.07% faster than IMP does. At the same time, IMP was able to prune 14 percentage points more weights than BINGO was able to. Compared to IMP, BINGO serves as a mechanism to prune weights that is similar in efficacy and far less computationally taxing.

Because BINGO only takes 17 minutes to prune a model that IMP would take 122 minutes to prune, it serves as a superior method to prune down an AI model before uploading it to an MCU. This ultimately means that students and learners will be better able to interact with and explore ML and AI models on affordable, low-memory microcontrollers like the Raspberry Pi or Arduino. By dramatically reducing the computational barrier to deploying AI models on physical computing platforms, BINGO opens the door for more hands-on, engaging, effective, and accessible AI education.

References

Frankle, J., & Carbin, M. (2018). *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks*. https://arxiv.org/pdf/1803.03635

Han, S., Pool, J., Tran, J., & Dally, W. (2015). *Learning both Weights and Connections for Efficient Neural Networks*. https://arxiv.org/pdf/1506.02626

LeCun, Y. (2009). *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges*. Lecun.com. http://yann.lecun.com/exdb/mnist/

Li, C. (2020, June 3). *OpenAI's GPT-3 Language Model: A Technical Overview*. Lambda.ai; Lambda, Inc. https://lambda.ai/blog/demystifying-gpt-3?srsltid=AfmBOopEBwnjFUkDPIihPXftrT59qpn_ODdaWJp7VNXC83-PRqT1xYMH

Love, T. S., & Asempapa, R. S. (2022). A screen-based or physical computing unit? Examining secondary students' attitudes toward coding. *International Journal of Child-Computer Interaction*, *34*(34), 100543. https://doi.org/10.1016/j.ijcci.2022.100543

Maximova, A. (2024). *Teaching Programming through Multi-Context Physical Computing*. 850–851. https://doi.org/10.1145/3649405.3659482

Pernes, P., Jaroš, M., Podivín, J., & Trenz, O. (2024). MICROCONTROLLERS SUITABLE FOR ARTIFICIAL INTELLIGENCE. *26th Annual International Conference Economic Competitiveness and Sustainability 2024 Proceedings*, 120–126. https://doi.org/10.11118/978-80-7509-990-7-0120

Rad, von, & Seuffert, F. (2024, September 29). *Investigating the Effect of Network Pruning on Performance and Interpretability*. ArXiv.org. https://arxiv.org/abs/2409.19727

Srivastava, N., Hinton, G., Krizhevsky, A., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958. https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

Svoboda, F., Fernandez-Marques, J., Liberis, E., & Lane, N. D. (2022). Deep learning on microcontrollers. *Proceedings of the 2nd European Workshop on Machine Learning and Systems*, 54–63. https://doi.org/10.1145/3517207.3526978

Udvaros, J., & Forman, N. (2023). THE EFFECT OF USING MICROCONTROLLERS IN TEACHING PROGRAMMING. *Conference Proceedings Of ELearning and Software for Education (ELSE)*, *19*(02), 139–151. https://www.ceeol.com/search/article-detail?id=1321992