NEURAL NETWORK COMPRESSION WITH FEEDBACK MAGNITUDE PRUNING FOR AUTOMATIC MODULATION CLASSIFICATION

Jakob Krzyston[1], Rajib Bhattacharjea[2], Andrew Stark[3]
[1]School of Electrical and Computer Engineering, Georgia Tech, Atlanta, GA USA, [2]DeepSig Inc., Arlington, VA USA, [3]Georgia Tech Research Institute, Atlanta, GA USA

NOTE: Corresponding author: Jakob Krzyston, jakobk@gatech.edu

Abstract *– In the past few years, there have been numerous demonstrations of neural networks outperforming traditional signal processing methods in communications, notably Automatic Modulation Recognition (AMC). Despite the increase in performance, these algorithms are notoriously infeasible for integrating into edge computing applications. In this work, we demonstrate the effectiveness of a simple neural network pruning technique, called Iterative Magnitude Pruning (IMP) and propose an enhanced version of IMP called Feedback Magnitude Pruning (IMP), for the "Lightning-Fast Modulation Classification with Hardware-Efficient Neural Network" AI for Good: Machine Learning in 5G Challenge hosted by ITU and Xilinx. IMP achieved a compression ratio of 9.313 and normalized computational cost of 0.042467 and was awarded second place. Our FMP efforts achieved a compression ratio of 831 and normalized cost of 0.0419 demonstrating the compression and performance benefits of pruning with feedback.*

Keywords – Feedback Magnitude Pruning, Iterative Magnitude Pruning, Automatic Modulation Classification, Neural Network Compression, Artificial Intelligence

1. INTRODUCTION

By the very nature of communications, processing systems be fast and minimize computational complexity, memory footprint, energy expenditure, etc all while maximizing accuracy. These systems are deployed in a variety of operating environments (seaborne, airborne, etc) where all of these constraints may be rigid due to legacy technologies, implementation costs, etc. Although neural networks are praised for their success on problems in communications, such as Convolutional Neural Networks (CNNs) for Automatic Modulation Classification (AMC) [1, 2, 3, 4], they are notorious for their computational demand and infeasibility when trying to deploy these systems into the real world.

In the artificial intelligence research community, numerous methods have been developed to reduce the number of computations and memory required while maintaining performance, called network compression. Within network compression, there are methods for removing weight/edges in the neural network, called pruning, and methods for reducing the precision of the values stored in the weights, called quantization. Further, there have been techniques developed to make computations in these neural networks more efficient.

In addition to being large, with regards to disk space, deep Neural Networks are typically developed in scripting languages like Python, and typically rely on hardware such as Graphical Processing Units (GPUs) to be able to execute the computations in a parallel manner. This may work fine for cloud-based machine learning applications or uses where very low latency is not critical. In communications, algorithms are deployed onto common communications hardware such as a Field Programmable Gate Array (FPGA). This requires specialists who not only understand the hardware, but also the low-level programming necessary to use the specialized hardware.

Recently, Xilinx released a programming package called Brevitas [5] that is able to translate PyTorch scripts to be used on FPGAs. With the rising popularity for deep learning in communications, Xilinx teamed up with ITU to host a competition, called, "Lightning-Fast Modulation Classification with Hardware-Efficient Neural Networks" that was a part of their *AI for Good: Machine Learning in 5G Challenge*.

2. ITU COMPETITION DETAILS

The goal of the competition is to leverage and develop quantization, pruning, architectural design, and training paradigms to create an ANN that minimizes inference cost, while maintaining at least 56% accuracy for modulation classification. The normalized inference cost was defined as

$$cost = \frac{bit\_ops_{final}}{2 \times bit\_ops_{baseline}} + \frac{bit\_mem_{final}}{2 \times bit\_mem_{baseline}} \quad (1)$$

where *bit_ops* is the number of bit operations in the final or the provided baseline model and *bit_mem* is the number of memory bits respectively. The dataset used was the RadioML 2018.01A dataset [2]. There was code provided to contestants with a simple VGG-style

neural network [6], no pruning, no quantization, nor any custom training techniques [7].

In this work, we present our submission to the 2021 ITU AI for Good Challenge: "Lightning-Fast Modulation Classification with Hardware-Efficient Neural Networks", which earned second place, and propose a new for of pruning called Feedback Magnitude Pruning (FMP). Our competition submission is comprised of quantizing, learning rate on plateau learning rate scheduler, and the first known demonstration of Iterative magnitude Pruning (IMP) for AMC. FMP is an advanced form of IMP that leverages feedback from the training to adjust the rate of pruning.

3. BACKGROUND

This premise of the challenge is the delicate trade-off between performance and network compression. Pruning, quantization, and network architecture techniques aim to reduce latency and computational demand while various training paradigms aim into increase the accuracy of the network.

3.1 Neural Network Pruning

The proliferation in applications for neural networks, accompanied by the desired for greater accuracy improvements, has led to a trend of ever-larger network architectures [6, 8, 9, 10, 11]. As architectures grow, the number of parameters to optimize also increases leading to trillion parameter networks. This has been a concern in the field of AI for years [12, 13]. Fortunately, research focused on reducing the number of parameters in a network has been an ongoing pursuit by teams of researchers for decades [14, 15, 16, 17, 18]. More formally known as network pruning, this field of research is focused on devising strategies to best rid of edges/weights/connections in the network such that the resultant sub-network performs comparable to the original network.

Typically, edges are removed after training a network. Recent efforts in this area of research have focused on removing edges while using little to no training data at all [19, 20, 21, 22]. In practice, performance is of utmost concern and these sophisticated methods do not outperform a simple method called Iterative Magnitude Pruning (IMP) [16].

3.1.1 Iterative Magnitude Pruning

IMP works by three simple steps: (1) train the neural network, (2) prune the network by removing the smaller weights, and (3) continue training the remaining sub-network. In IMP, steps two and three are be repeated until the performance decreases or until a certain compression ratio, defined as the ratio between the set of all parameters in the original network and the set of parameters in the pruned network, is reached. Although IMP performs very well, the arbitrary nature of what information is being removed does not help researchers understand why features of the network are being removed.

3.2 Network Quantization

Typically when training a neural network on a GPU, the weights, activations, and gradients are stored using 32 floating point bits. By lowering this to 16, eight, four, two, or even one bit opens opportunities for these algorithms to be more readily implemented onto specialized hardware such as FPGAs and Application Specific Integrated Circuits (ASICs). Important considerations when quantizing are the *range* of values represented as well as the *spacing* between the values, which changes whether using `float` or `int` data type. [23]. Examples of methods for quantizing a neural network include: using adaptive ranges and clipping [24, 25], mixed precision training [26, 27, 28, 29], coding schemes [30], and differentiable quantization [31, 32, 33, 34, 35].

3.3 Efficient Network Architectures: MobileNets

With the rise in neural network research after AlexNet [36], the possibility of running one of these algorithms on ever smaller computing platforms became a goal for the field. In 2017, MobileNets [37] were introduced as a convolutional paradigm able to be small enough for embedded applications and have had a strong influence in designing 'deployable' neural networks. In [37], the authors introduce a type of factorized convolutions called depthwise-separable convolutions that aim to reduce computational complexity of the convolutional operation. The computational cost of a traditional convolution is

$$D_K \times D_K \times M \times N \times D_F \times D_F \qquad (2)$$

where $D_K \times D_K$ is the kernel size, M is the number of input channels, N is the number of output channels, and $D_F \times D_F$ is the dimension of the feature map. Leveraging point-wise convolutions, depth-wise separable convolutions separates a traditional convolution into two steps: filtering and then combining. This separation reduces the computational complexity of a convolutional layer to

$$D_K \times D_K \times M \times D_F \times D_F + M \times N \times D_F \times D_F. \quad (3)$$

This reduces the computational complexity by a factor of $\frac{1}{N} + \frac{1}{D_K^2}$. The MobileNet can be tuned using two hyperparameters, *width multiplier* and *resolution multiplier*, which allow for trading off between latency and accuracy.

The next iteration, called MobileNetV2 [38], incorporated inverted residual connections. In a traditional

residual connection [8], given a sequence of layers $l_i, i \in [0, 3]$ where layers l_0 and l_3 have the at least as many channels as the other layers, the input layer l_0 and the output layer l_3 have the same number of channels and thus can be added to one another. Also known as a skip connection, this was created to address the vanishing gradient problem seen in deep neural networks. In an inverted residual connection, the interior layers, with fewer channels, are arranged outside of the higher channel layers and the skip connection is formed between the first and last channels in this new arrangement. Additionally, there is no non-linearity included in these first and last channels. This allows for further reduction in computational complexity compared to MobileNets and is analogous to expanding the latent space dimensionality to better disentangle feature representations, akin to the famed *kernel trick* [39].

In 2019, MobileNetV3 [40] leveraged results from a neural architecture search approach, MNasNet [41], and implemented what are called *squeeze and excite* layers. Squeeze and excite layers are extensions of the inverted residual connection from MobileNetV2 with an addition of a fifth layer. The first three layers are the same as the inverted residual connection, but then the third layer is 'squeezed' with respect to $D_K \times D_K$ by a pooling function. This is then passed into a small fully connected layer with ReLU activation to compress depthwise, then another small fully connected layer with a hard activation to reconstruct the original depth. A deconvolutional layer expands the tensor with respect to $D_K \times D_K$, which is then followed by a 1×1 convolutional layer to the same number of channels as l_0. This is added to $l_0$to complete the residual connection.

3.4 Learning Rate Strategies

Neural networks traverse their loss landscape according to an optimization algorithm. A key parameter in these optimization algorithms is the step size, or learning rate (LR) from deep learning parlance. Baseline deep learning methods use a set learning rate for the chosen optimizer, which is difficult to set as to efficiently traverse the loss landscape as well as avoid getting stuck in a local minimum or possibly even 'stepping' out of a minimum. An easy way to alter the ultimate performance of a network is by altering the LR value by having a LR scheduler. Learning rate schedulers alter the LR as a function of at least one variable, i.e. number of epochs, derivative of validation loss, etc. LR schedulers can be put into categories: fixed, decaying, and cyclic. Fixed LR schedulers implies the LR never changes, which can be very risky for the aforementioned reasons. Decaying LR schedulers leverage various decay functions for LR annealing. Decaying LRs anticipate the optimization function will need to take larger steps in the beginning to locate a sufficient minima, then smaller steps to ensure it locates the minima and does not escape. Cyclic LR schedulers cyclically vary the LR within a specified range. Cyclic LRs leverage the properties of decaying LR schedulers, but gradually take larger steps to determine the robustness of the minima reached [42].

4. COMPETITION SUBMISSION

In this competition, we aimed to display the effectiveness of simple methods for the problem of compressing neural networks for AMC. Here we describe the details of our second-place submission to the ""Lightning-Fast Modulation Classification with Hardware-Efficient Neural Networks" challenge.

4.1 Neural Network Architecture

MobileNets are extremely popular in edge applications of deep learning. Recent work analyzing the effects of quantization on MobileNets saw drastic fluctuations in dynamic range and trouble matching the distributions between channel-wise and layer-wise distributions in depth-wise separable CNNs [43]. These issues lead to greater degradation as MobileNets were increasingly quantized as well as greater distributional shift as information propagated through the network. The authors experimentally showed there is less error due to quantization in traditional CNNs, like VGG-style networks, compared to depth-with separable CNNs, like MobileNet. Due to these findings, we decided to utilize the provided VGG-style architecture provided, seen in Figure 1, because of the known efficiency of convolutional layers versus dense layers, as well as the opportunity to further quantize the network to improve the normalized inference cost while mitigating accuracy losses.

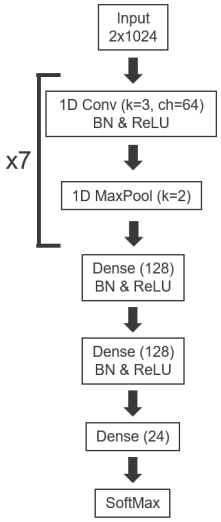

Fig. 1 – VGG-style architecture provided by the competition. **k** denotes the kernel dimension (square kernel), **ch** denotes the number of channels, and the number in the parenthesis for the Dense layers indicates the number of output nodes.

4.2 Network Quantization

In the provided code, there were three parameters determining the precision of the input, weights, and activations, `input_bits`, `w_bits` and `a_bits` respectively. By default, these were all set to eight. In our approach, we changed these parameters to four. This means we used `int4` precision in the input, weights, and activations.

4.3 Training Paradigm

In the PyTorch framework, there is a unique, adaptive learning rate scheduler called Reduce LR on Plateau. This method works by tracking a performance metric of the network being trained, governed by a parameter called *patience*. Reduce LR on Plateau tracks the validation loss in the network and records how long the validation loss has not changed, governed by patience. This implies it is not finding a local minima. Reduce LR on Plateau will reduce the learning rate by a set factor as to stop jumping over the local minima and seek improved performance.

4.4 Network Pruning

Due to the simplicity and proven performance of Iterative Magnitude Pruning, we chose this technique to prune our quantize architecture. We chose the pruning percentage to be 20%, as advised by the author of [44] and chose L1 unstructured pruning as our pruning method. In traditional IMP, the model is trained until convergence, then pruned. To speed up the pruning process, we implemented a criterion where the model would prune once the accuracy was at least 56%, the minimum criteria for the competition. Both the traditional version of IMP (train until convergence) and the IMP with the accuracy criterion were tested in this competition. The pseudocode for our implementation of IMP, with the accuracy criterion, are in Algorithm 1.

```
for number of pruning epochs do
    for number of training epochs do
        Train model;
        Evaluate model;
        if model accuracy> 56% then
            Save model;
            Prune 20% of the weights;
            Break
        end
    end
end
```

Algorithm 1: IMP with Accuracy Criterion

The number of pruning epochs is set to a large number and is not a limiting factor in the pruning process.

5. COMPETITION RESULTS

Two implementations of IMP were compared against one another, one with the accuracy criterion and one that trained for the defined number of training epochs. We compare the number of pruning epochs achieved before yielding a network below 56% accurate as well as the compression ratio η. For a pruning percentage p, and a number of pruning epochs e_p, the compression ratio is defined as

$$\eta = \frac{1}{(1-p)^{e_p}}. \tag{4}$$

As we see in Table 1, the accuracy constraint enabled us to achieve one more pruning epoch, ten compared to nine. This increased compression ratio by nearly 25% from 7.451 to 9.313.

Table 1 – Comparison of IMP Strategy

Strategy	**Prune Epochs**	η
Normal IMP	9	7.451
IMP w/ Accuracy	10	9.313

With our quantized network, we compared the reduced bit operations and weight bits to that of the provided architecture in Table 2. Our pruning and quantization methods resulted in a 96.97% and 94.53% reduction in the number of bit operations and weight bits respectively.

Table 2 – Comparison of Original and Final Networks

Network	**Bit_Ops**	**Bit_Mem**	**Cost**
Baseline	807,699,904	1,244,936	1
Submission	24,436,576	68,072	0.042467

We were able to achieve a normalized inference cost of 0.042467 and an overall accuracy of 56.25% which was good enough to receive second place. In Figure 2 we show the confusion matrix for the submission performed on the modulation patterns over all SNRs. In Figure 3 we show the overall accuracy of the trained model as a function of SNR. In Figure 4, we show the accuracy of the model per modulation format, as a function of SNR.

6. POST-COMPETITION WORK

Pruning techniques typically specify desired sparsities/compression ratios beforehand or keep the pruning rate constant. There have been several works regarding changing the level of sparsity as a function of a variable [45, 46, 47], however those works focus on removing then reallocating/replacing the weights later on in the training process. In this section we propose a pruning rate scheduler which leverages feedback to adjust the pruning rate according to a function of a specified criterion without needing to train an oracle model, called Feedback Magnitude Pruning (FMP). Similar to existing learning rate schedulers, FMP does not guarantee

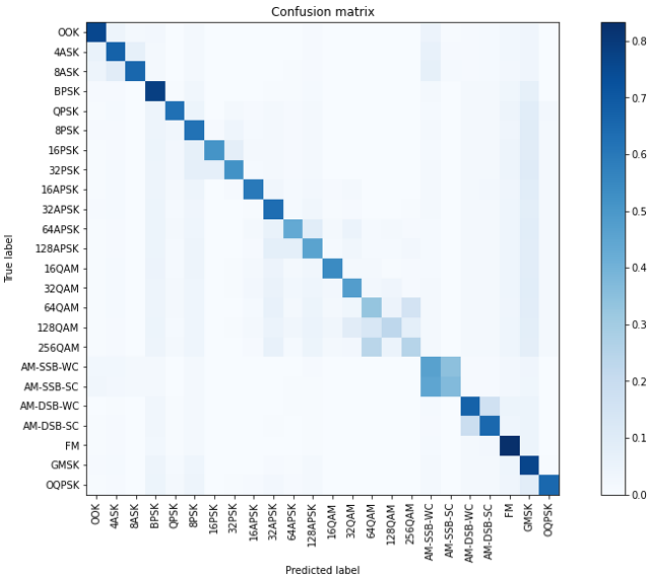

Fig. 2 – Overall confusion matrix from the ITU Competition: Lightning-Fast Modulation Classification with Hardware-Efficient Neural Networks.

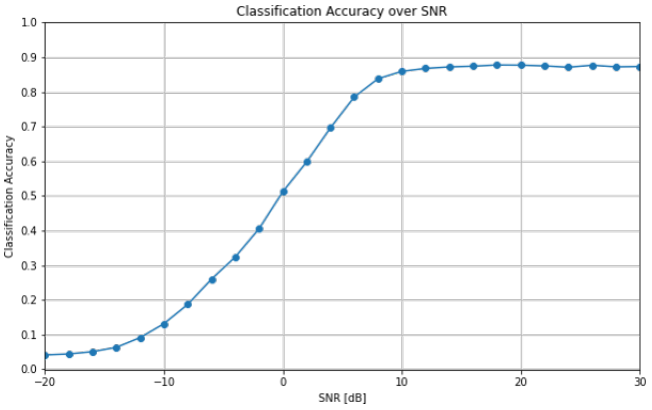

Fig. 3 – Overall accuracy of the submission to the ITU competition, as a function of the input SNR.

the next 'step size' taken to prune a network. Rather, FMP uses information regarding the status of a model with respect to a criterion whether or not to prune, i.e. whether or not the model can meet a specified accuracy threshold.

6.1 Feedback Magnitude Pruning

Inspired by many LR scheduling methods, the pruning rate in FMP will follow a decaying trajectory parameterized by the initial pruning amount, p, and the normalizing factor, n. n regulates the rate at which p decreases. In this specific implementation of FMP, the p continues to decrease until it is less than 5%. This threshold was chosen to save computation time and due to diminishing returns with regards to the normalized inference cost. Pseudocode pertinent to the implementation of FMP to this competition, with the opportunity to combine multiple pruning techniques such as structural, is shown in Algorithm 2.

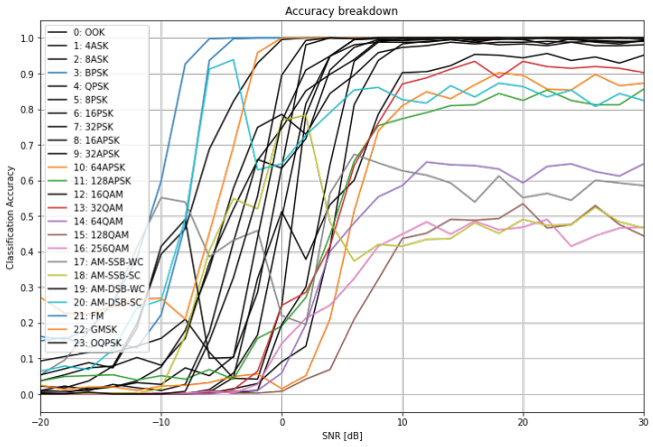

Fig. 4 – Accuracy of the model, per modulation format, of the ITU competition submission, as a function of the input SNR.

The pruning percentage p is user defined as is n, the divisor which will dictate how rapidly p will reduce. The authors used $p = 20\%$ and $n = 2$. When reconstructing the architecture after Structured pruning, create a new architecture with the number of filters and nodes that are non-zero.

Extending Equation *(4)*, the compression ratio for a given pruning method is

$$\eta = \prod_{r=0}^{R} \frac{1}{(1-p_r)^{e_r}} \tag{5}$$

where $p_r, r \in [0, R]$ is the pruning rate as governed by p and n, $R = floor(\log_n(100 * p)) + 1$ is the total number of pruning rates for a method of pruning, and e_r is the number of epochs at a given pruning rate. Equation *(5)* is an approximation because by a percentage, does not always result in an exact number of nodes/connections to prune, i.e. 20% for a network with 11 eligible elements. This formula is less accurate for structural pruning methods as the errors are exacerbated.

The total compression ratio for Algorithm 2 is

$$\eta_{Total} = \prod_{i=0}^{I} \eta_i \tag{6}$$

where $i \in [0, I]$ denotes the pruning method used.

6.2 FMP Results

Feedback Magnitude Pruning was tested on the same neural network architecture seen in Figure 1, however the number of bits in the inputs, activations, and weights was varied. Table 3 shows network architectures with greater numbers of bits were able to be further compressed, however these models did not achieve as low of an inference score as the four bit FMP model, 0.0419, which beat the four bit IMP model. Table 3 shows FMP greatly increasing the ability for a network to be compressed. The four bit VGG network with IMP

```
for method in p_methods do
    while p ≥ 0.05 do
        i = 0;
        while i < number of training epochs do
            Train model;
            Evaluate model;
            if model accuracy≥ 56% then
                Save model;
                Prune weights, per p and method;
                i = 0
            end
            else
                i += 1
            end
        end
        if model accuracy < 56% then
            p = p/n;
            Load most recent 56% accurate model;
        end
    end
    if method ≤ method ∈ p_methods then
        Iterate to the next pruning technique
    end
end
```

Algorithm 2: Feedback Magnitude Pruning

achieved a compression ratio of $\eta = 9.313$, whereas the same network with FMP achieved a compression ratio of $\eta = 813$. We also compare the full eight bit version of the VGG model with FMP, and was able to compress it 5,821x and resulted in an inference cost of 0.0583.

Table 3 – Feedback Magnitude Pruning Results

Bits	η	Bit_Ops	Bit_Mem	Cost
8*	1	807,699,904	1,244,936	1
8†	5,821	49,892,544	68,384	0.0583
7†	5,821	39,056,969	59,836	0.0482
6†	3,072	34,321,896	62,976	0.0465
5†	1,621	28,316,625	64,430	0.0434
4*	9	24,436,576	68,072	0.0424
4†	813	24,601,600	66,616	0.0419

* Seen in Table 2
† Feedback Magnitude Pruning (Unstructured)

7. CONCLUSION

In this work, we detail our submission to the ITU AI for Good Challenge: "Lightning-Fast Modulation Classification with Hardware-Efficient Neural Network" and showcase a proposed pruning technique called Feedback Magnitude Pruning. Our submission demonstrates the potential for simple quantization and learning rate strategies in combination with the first known demonstration of Iterative Magnitude Pruning for Automatic Modulation Classification in the literature. Our submission achieved a normalized inference cost of 0.042467 while maintaining greater then 56% classification accuracy. Our proposed Feedback Magnitude Pruning showed the ability to greatly compress neural networks, from a compression ratio of 9.3 to 813, and improve our normalized inference score down to 0.0419.

One of the outstanding issues in network compression, specifically pruning, is understanding why the resulting computational graph is of significance. Future work from our group will be exploring significance of pruned neural networks as well as developing methods to compress neural networks in a manner that allows researchers to understand why the remaining architecture is of importance.

All code used in this work can be found at https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-007-Feedback-Magnitude-Pruning.

REFERENCES

[1] Timothy J O'Shea, Johnathan Corgan, and T Charles Clancy. "Convolutional radio modulation recognition networks". In: *International conference on engineering applications of neural networks*. Springer. 2016, pp. 213–226.

[2] Timothy James O'Shea, Tamoghna Roy, and T Charles Clancy. "Over-the-air deep learning based radio signal classification". In: *IEEE Journal of Selected Topics in Signal Processing* 12.1 (2018), pp. 168–179.

[3] Jakob Krzyston, Rajib Bhattacharjea, and Andrew Stark. "Complex-Valued Convolutions for Modulation Recognition using Deep Learning". In: *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE. 2020, pp. 1–6.

[4] Jakob Krzyston, Rajib Bhattacharjea, and Andrew Stark. "Modulation Pattern Detection Using Complex Convolutions in Deep Learning". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 2233–2239.

[5] Alessandro Pappalardo. *Xilinx/brevitas*. 2021. DOI: 10.5281/zenodo.3333552. URL: https://doi.org/10.5281/zenodo.3333552.

[6] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[7] *ITU-ML5G-PS-007: Lightning-Fast Modulation Classification with Hardware-Efficient Neural Networks*. URL: https://challenge.aiforgood.itu.int/match/matchitem/34.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[11] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners". In: *arXiv preprint arXiv:2005.14165* (2020).

[12] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. "Exploiting linear structure within convolutional networks for efficient evaluation". In: *Advances in neural information processing systems*. 2014, pp. 1269–1277.

[13] Lei Jimmy Ba and Rich Caruana. "Do deep nets really need to be deep?" In: *arXiv preprint arXiv:1312.6184* (2013).

[14] Yann LeCun, John S Denker, and Sara A Solla. "Optimal brain damage". In: *Advances in neural information processing systems*. 1990, pp. 598–605.

[15] Babak Hassibi and David G Stork. *Second order derivatives for network pruning: Optimal brain surgeon*. Morgan Kaufmann, 1993.

[16] Song Han, Jeff Pool, John Tran, and William J Dally. "Learning both weights and connections for efficient neural networks". In: *arXiv preprint arXiv:1506.02626* (2015).

[17] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. "Pruning filters for efficient convnets". In: *arXiv preprint arXiv:1608.08710* (2016).

[18] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. "Rethinking the Value of Network Pruning". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=rJlnB3C5Ym.

[19] Chaoqi Wang, Guodong Zhang, and Roger Grosse. "Picking winning tickets before training by preserving gradient flow". In: *arXiv preprint arXiv:2002.07376* (2020).

[20] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. "Snip: Single-shot network pruning based on connection sensitivity". In: *arXiv preprint arXiv:1810.02340* (2018).

[21] Hidenori Tanaka, Daniel Kunin, Daniel LK Yamins, and Surya Ganguli. "Pruning neural networks without any data by iteratively conserving synaptic flow". In: *arXiv preprint arXiv:2006.05467* (2020).

[22] Shreyas Malakarjun Patil and Constantine Dovrolis. "PHEW: Constructing Sparse Networks that Learn Fast and Generalize Well without Training Data". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8432–8442.

[23] James O' Neill. "An overview of neural network compression". In: *arXiv preprint arXiv:2006.03669* (2020).

[24] Eunhyeok Park, Sungjoo Yoo, and Peter Vajda. "Value-aware quantization for training and inference of neural networks". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 580–595.

[25] Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. "Aciq: Analytical clipping for integer quantization of neural networks". In: (2018).

[26] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. "Mixed precision training". In: *arXiv preprint arXiv:1710.03740* (2017).

[27] Fengfu Li, Bo Zhang, and Bin Liu. "Ternary weight networks". In: *arXiv preprint arXiv:1605.04711* (2016).

[28] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. "Trained ternary quantization". In: *arXiv preprint arXiv:1612.01064* (2016).

[29] Dipankar Das, Naveen Mellempudi, Dheevatsa Mudigere, Dhiraj Kalamkar, Sasikanth Avancha, Kunal Banerjee, Srinivas Sridharan, Karthik Vaidyanathan, Bharat Kaul, Evangelos Georganas, et al. "Mixed precision training of convolutional neural networks using integer operations". In: *arXiv preprint arXiv:1802.00930* (2018).

[30] S Han, H Mao, and WJ Dally. "Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv 2015". In: *arXiv preprint arXiv:1510.00149* ().

[31] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. "Differentiable soft quantization: Bridging full-precision and low-bit neural networks". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4852–4861.

[32] Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc Van Gool. "Soft-to-hard vector quantization for end-to-end learning compressible representations". In: *arXiv preprint arXiv:1704.00648* (2017).

[33] Pierre Stock, Armand Joulin, Rémi Gribonval, Benjamin Graham, and Hervé Jégou. "And the bit goes down: Revisiting the quantization of neural networks". In: *arXiv preprint arXiv:1907.05686* (2019).

[34] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. "Quantization and training of neural networks for efficient integer-arithmetic-only inference". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2704–2713.

[35] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. "Hawq: Hessian aware quantization of neural networks with mixed-precision". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 293–302.

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pp. 1097–1105. URL: `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[37] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[38] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[39] Ralf Herbrich. *Learning kernel classifiers: theory and algorithms*. MIT press, 2001.

[40] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. "Searching for mobilenetv3". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1314–1324.

[41] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. "Mnasnet: Platform-aware neural architecture search for mobile". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2820–2828.

[42] Yanzhao Wu, Ling Liu, Juhyun Bae, Ka-Ho Chow, Arun Iyengar, Calton Pu, Wenqi Wei, Lei Yu, and Qi Zhang. "Demystifying learning rate policies for high accuracy training of deep neural networks". In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 1971–1980.

[43] Stone Yun and Alexander Wong. "Do All MobileNets Quantize Poorly? Gaining Insights into the Effect of Quantization on Depthwise Separable Convolutional Networks Through the Eyes of Multi-scale Distributional Dynamics". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2447–2456.

[44] Alex Renda, Jonathan Frankle, and Michael Carbin. "Comparing rewinding and fine-tuning in neural network pruning". In: *arXiv preprint arXiv:2003.02389* (2020).

[45] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science". In: *Nature communications* 9.1 (2018), pp. 1–12.

[46] Hesham Mostafa and Xin Wang. "Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 4646–4655.

[47] Tao Lin, Sebastian U Stich, Luis Barba, Daniil Dmitriev, and Martin Jaggi. "Dynamic model pruning with feedback". In: *arXiv preprint arXiv:2006.07253* (2020).

AUTHORS

Jakob Krzyston Jakob received his BS in Biomedical Engineering ('17) from Rochester Institute of Technology and an MS in Electrical and Computer Engineering ('20) from Georgia Institute of Technology. Currently, he is an Electrical and Computer Engineering PhD Student in the Terabit Optical Networking Lab at Georgia Institute of Technology and full time Research Engineer at GTRI.

Rajib Bhattacharjea .

Andrew Stark .