NEURAL NETWORK COMPRESSION WITH FEEDBACK MAGNITUDE PRUNING FOR AUTOMATIC MODULATION CLASSIFICATION

Jakob Krzyston[1,3], 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 for Automatic Modulation Classification (AMC). Despite the increase in accuracy, these algorithms are notoriously infeasible for integrating into edge computing applications. In this work, we propose an enhanced version of a simple neural network pruning technique, Iterative Magnitude Pruning (IMP), called Feedback Magnitude Pruning (FMP) and demonstrate its effectiveness for the "Lightning-Fast Modulation Classification with Hardware-Efficient Neural Network" 2021 AI for Good: Machine Learning in 5G Challenge hosted by the International Telecommunications Union (ITU) and Xilinx. IMP achieved a compression ratio of 9.313, while our proposed FMP achieved a compression ratio of 831 and normalized cost of 0.0419. Our FMP result was awarded second place, demonstrating the compression and classification accuracy benefits of pruning with feedback.*

Keywords – Artificial intelligence, automatic modulation classification, feedback magnitude pruning, neural network compression

1. INTRODUCTION

Modern wireless communication systems operate at speeds of gigabits per second or higher; signal processing in these systems must occur at the same scale of speed. Such systems are designed to be fast by minimizing computational complexity, memory footprint, and energy expenditure while maximizing accuracy on a specific task. Edge communication systems are deployed in a variety of operating environments (cellular, seaborne, airborne, etc.) where all of these are constrained to legacy technologies, implementation costs, and the physics underlying power dissipation. Neural networks have been successful solving problems in communications, e.g., Convolutional Neural Networks (CNNs) for Automatic Modulation Classification (AMC) [1, 2, 3, 4], but require significant reduction of computation, memory footprint, and storage space for widespread deployment.

Neural network compression is the set of methods and techniques that have been developed to reduce the number of computations and memory required while maintaining sufficient accuracy. Pruning methods remove edges in the neural network [5], and quantization methods to reduce the precision of the values stored in the data, weights, and activations [6]. Other techniques, such as depth-wise separable convolutions and inverted bottleneck residuals [7, 8], have been developed to make computations in these neural networks more efficient.

Deep neural networks occupy large memory footprints, are typically developed in Python libraries like PyTorch, and rely primarily on hardware such as Graphical Processing Units (GPUs) for computation execution in a parallel manner. This works well for cloud-based machine learning applications or other uses where very low latency is not critical. Communication systems often deploy algorithms on flexible 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 [9] that allows for quantized models to be trained and leverages FINN [10] which then translates PyTorch models for execution on FPGAs. With the rising popularity for deep learning in communications, Xilinx teamed up with International Telecommunications Union (ITU) to host a competition, called, "Lightning-Fast Modulation Classification with Hardware-Efficient Neural Networks" that was part of their 2021 AI for Good: Machine Learning in 5G Challenge [11].

2. MACHINE LEARNING IN 5G CHALLENGE

The International Telecommunications Union hosted a competition [11] to create a neural network that classifies modulation formats of communication signals. The network must minimize the inference cost while maintaining at least 56% overall accuracy across all modulation classes and SNRs in the RadioML 2018.01A dataset [2]. The inference cost was measured relative to a baseline network architecture provided by the challenge organizers,

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

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. Baseline code for a network and training loop was provided to challenge participants containing a simple VGG-style neural network [12].

In this paper 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. Our competition submission consists of modifying the baseline approach to incorporate 1) quantization, 2) a simple learning rate "reduce-on-plateau" scheduler, and 3) a new method of pruning called Feedback Magnitude Pruning (FMP). FMP is an advanced form of IMP that leverages feedback from the training to adjust the rate of pruning.

3. BACKGROUND

This core of the challenge is to engineer the trade-off between accuracy and network compression. Pruning, quantization, and network architecture techniques aim to reduce latency and computational demand while various training paradigms aim to increase the accuracy of the network. The authors recognized that utilizing more efficient architectures, such as MobileNets [7] and SqueezeNets [8], is another valid tactic to succeeding in this competition, however it is explicitly avoided in this work to constrain the variable space.

3.1 Neural network pruning

The proliferation in applications for neural networks, accompanied by the desire for greater accuracy improvements, has led to a trend of ever-larger network architectures [12, 13, 14, 15, 16]. As architectures grow, the number of parameters to optimize also increases, with well-known large models containing trillions of parameters. This has been a concern in the field of AI for many years [17, 18]. Research focused on reducing the number of parameters in a network has been an ongoing pursuit by teams of researchers for decades [5, 19, 20, 21, 22]. Network pruning is focused on devising strategies to best remove edges in the network such that the resultant subnetwork performs comparable to the original network. For clarity, we use the term edge synonymously with the terms connection and weight.

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 [23, 24, 25, 26]. In practice, accuracy is of the utmost concern and these sophisticated methods do not outperform a simple method called Iterative Magnitude Pruning (IMP) [20].

IMP works by three simple steps:

1. Train the neural network.
2. Prune the network by removing the smaller weights.
3. Continue training the remaining subnetwork.

Steps 2 and 3 are to be repeated for a specified number of epochs or up to a certain compression ratio, η [27]. Given a pruning percentage p, and a number of pruning epochs e_p, the compression ratio is defined as

$$\eta = \frac{1}{(1-p)^{e_p}}. \quad (2)$$

The compression ratio is inversely proportional to the sparsity of the network, e.g. if 1% of the weights remain after pruning, $\eta = 100$.
The two primary approaches to neural network pruning include unstructured pruning and structured pruning. Unstructured pruning [28] is the removal of specific edges from the computational graph, whereas structured pruning [29] focuses on removing entire nodes of the graph altering the structure. In practice, unstructured pruning yields more compressed networks but leveraging the extreme sparsity for computational benefit remains an outstanding challenge.

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. Training for a network to operate at 16, 8, 4, 2, or even 1 bit integer representation brings about 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 (i.e. the precision of the values), which changes whether using the `float` or `int` data type [6]. Examples of methods for quantizing a neural network include: using adaptive ranges and clipping [30, 31], mixed precision training [32, 33, 34, 35], coding schemes [36], and differentiable quantization [37, 38, 39, 40, 41].

3.3 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. Some baseline deep learning optimizers such as Stochastic Gradient Descent (SGD) [42] and SGD with momentum [43], use a constant learning rate and other commonly used optimizers such as Adam [44] have parameters that adjust the learning rate. Whether a constant or adaptive learning rate, it is difficult to efficiently traverse the loss landscape as well as avoiding getting stuck in a local minimum or even 'stepping' out of a minimum. An easy way to alter the final accuracy of a network is by modifying the LR value through the use of an LR scheduler. Learning rate schedulers adjust the LR as a function of at least one variable, such as number of epochs and/or derivative of validation loss. LR schedulers can be put into categories: fixed, decaying, and cyclical [45]. Fixed LR schedulers imply the LR never changes, which can prevent the network from converging into a local minimum. 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 minimum, then smaller steps to ensure it locates the minimum and does not escape. Cyclical LR schedulers periodically vary the LR within a specified range. Cyclical LRs leverage the properties of decaying LR schedulers and intermittently take larger steps to determine the robustness of the minimum reached [46].

In the PyTorch framework there is a unique, adaptive learning rate scheduler called Reduce LR on Plateau. This method operates by tracking the validation loss in the network and by recording for how long the validation loss does not improve, governed by a parameter called *patience*. No improvement implies no discovery of a local minimum. Reduce LR on Plateau will reduce the learning rate by a set factor to stop jumping over local minima and seek improved accuracy.

4. METHODS

In this competition we began from the neural network provided by the competition and (1) quantized the weights of the network prior to training, (2) implemented a learning rate scheduler, and (3) performed neural network pruning.

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 the dynamic range and trouble matching the distributions between channel-wise and layer-wise distributions in depth-wise separable CNNs [47]. 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-wise separable CNNs, like MobileNet.

Due to these findings, we decided to utilize the provided VGG-style architecture shown in Fig. 1 because of the opportunity to further quantize the network to improve the normalized inference cost while mitigating accuracy losses.

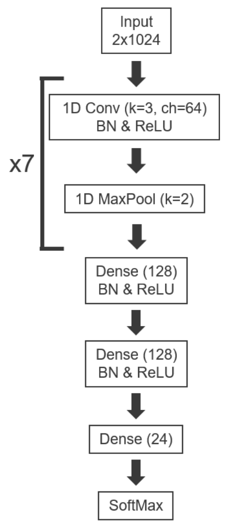

Fig. 1 – VGG-style architecture provided by the competition. **k** denotes the kernel dimension, **ch** denotes the number of channels, and the number in the parenthesis for the dense layers indicates the number of output nodes.

4.2 Network quantization

The precision of the input, weights, and activations are determined by three different parameters: `input_bits`, `w_bits` and `a_bits`, respectively. By default these were all set to eight and were equal for all layers of the network. We changed the precision of these parameters to four (`int4`) for all layers of the network.

4.3 Training paradigm

In this work, the Adam optimizer was used with an initial learning rate of 10^{-3}. Reduce LR on Plateau was used in order to help our model find a better local minima as we pruned and quantized the model. The model was trained on an NVIDIA Quadro RTX 4000 GPU, with a batch size of 1024 and an upper bound of 50 training epochs. Additionally, there was an upper bound of 20 pruning epochs.

4.4 Network pruning

In this competition, we test a well-established neural network pruning called Iterative Magnitude Pruning (IMP) as well a novel pruning method we call Feedback Magnitude Pruning (FMP). All pruning was L1 unstructured which was advantageous to the inference cost shown in Equation *(1)*.

4.4.1 Iterative magnitude pruning

When using Iterative Magnitude Pruning (IMP) to prune our quantized architecture, following the advice of the author of [48], we set the pruning percentage, p, to be 20%. 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. In both versions of IMP, p does not change and the models are not reinitialized after each round of pruning. The pseudocode for our implementation of IMP, with the accuracy criterion, is in Algorithm 1.

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

4.4.2 *Feedback magnitude pruning*

Pruning techniques typically specify desired sparsity/compression ratios beforehand or keep the pruning rate constant for a specific number of pruning epochs. Some work regarding changing the level of sparsity as a function of a variable [49, 50, 51] has focused on removing then reallocating the weights later in the training process. In this section we propose a pruning rate scheduler called Feedback Magnitude Pruning (FMP) which leverages feedback to adjust the pruning rate according to a function of a specified criterion without needing to train an unpruned model simultaneously. Similar to existing learning rate schedulers like Reduce LR on Plateau, FMP does not guarantee the next pruning rate nor the schedule by which the pruning rate is changed. Rather, FMP uses information regarding the status of a model with respect to a criterion how much to prune.

In this work, the pruning rate in FMP follows a decaying trajectory parameterized by a normalizing factor n which regulates the rate at which the initial pruning percentage p decreases. In this specific implementation of FMP, p is allowed to decrease until it is less than 5%. This threshold was chosen to save computation time and due to dimin- ishing improvement in normalized inference cost. Pseu- docode of our implementation of FMP for this competi- tion is shown in Algorithm 2.

The pruning percentage p is user defined as is n, the divisor which will dictate how rapidly p will reduce. The authors initialized $p = 0.2$ and $n = 2$.

```
while $p \geq 0.05$ do
    $i$ = 0;
    while i < number of training epochs do
        Train model;
        Evaluate model;
        if model accuracy $\geq 56\%$ 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
```

Algorithm 2: Feedback magnitude pruning

Extending Equation*(2)*, the total compression ratio is

$$\eta = \prod_{r=0}^{R} \frac{1}{(1-p_r)^{e_r}} \tag{3}$$

where p_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, and e_r is the number of pruning epochs at a given pruning rate. Equation *(3)* is an approximation because a percentage does not always result in an exact number of nodes/connections to prune, i.e. 20% of a network with 11 eligible elements.

5. RESULTS

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

Table 1 – Comparison of IMP strategy

Strategy	Prune Epochs	η
Normal IMP	9	7.45
IMP w/ Accuracy	10	9.31

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 IMP networks

Network	Bit_Ops (%)	Bit_Mem (%)	Cost
Baseline	100	100	1
IMP	3.025	5.468	0.0424

Using IMP, we were able to achieve a normalized inference cost of 0.0424 and an overall accuracy of 56.25%.

5.2 FMP results

Feedback magnitude pruning was tested on the same neural network architecture seen in Fig. 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 an inference score as the 4 bit FMP model, 0.0419, which beat the 4 bit IMP model and achieved second place. Table 3 shows FMP greatly increasing the ability for a network to be compressed. The 4 bit VGG network with IMP 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 8 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	100	100	1
$8^\dagger$	5,821	6.177	5.493	0.0583
$7^\dagger$	5,821	4.836	4.806	0.0482
$6^\dagger$	3,072	4.429	5.059	0.0465
$5^\dagger$	1,621	3.506	5.175	0.0434
$4^\ddagger$	9.31	3.025	5.468	0.0424
$4^\dagger$	813	3.046	5.351	0.0419

* Baseline model provided by ITU
† Feedback magnitude pruning
‡ Iterative magnitude pruning

In the bottom two rows of Table 3, trials utilizing 4 bit quantization, the normalized inference cost did not change in proportion to the increase in compression as a result of using FMP. This stems from how the normalized inference cost was defined in Equation *(1)*. The weight bits and bit operations are not proportionally reduced as the pruning is increased.

The accuracies of the 4 bit IMP network and the 4 bit FMP network were very similar, thus we report classification results for the FMP network. In Fig. 2 we show the confusion matrix for the submission performed on the modulation patterns over all SNRs. In Fig. 3 we show the overall accuracy of the trained model as a function of SNR. In Fig. 4, we show the accuracy of the model per modulation format, as a function of SNR.

Inspection of figures 2 through 4 reveals overall classification behavior of the neural network. When averaged over all SNRs, the network has a difficult time distinguishing between 64-QAM, 128-QAM, and 256-QAM, between the AM-SSB formats, and between the AM-DSB formats. This behavior makes intuitive sense because the high-order QAM formats are nearly analog in nature and exhibit relatively high SNR requirements for high accuracy. Other formats were readily identifiable: accuracy is nearly 100% for formats of order below 16-PSK. Some stand-out accuracies include >90% classification accuracy for 4ASK, 8QSK, and BPSK at SNR as low as 0 dB.

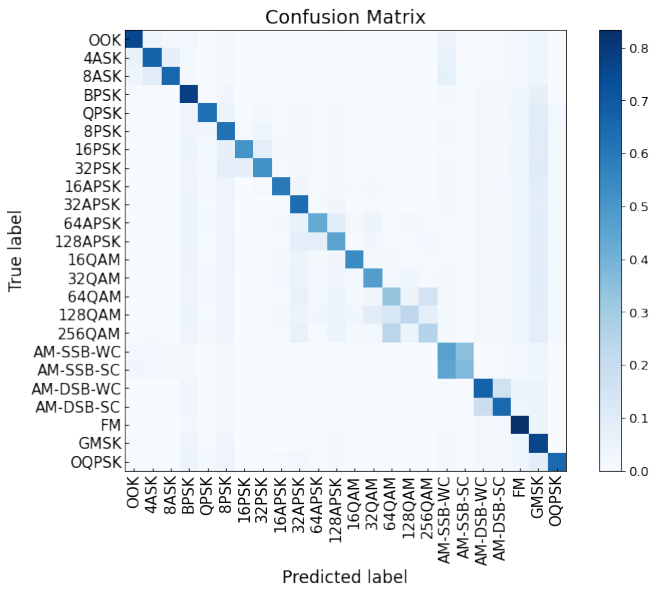

Fig. 2 – Overall confusion matrix from the ITU Competition: Lightning-Fast Modulation Classification with Hardware-Efficient Neural Networks.

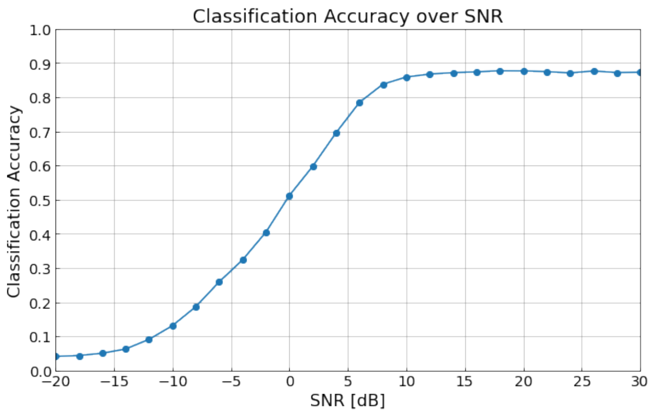

Fig. 3 – Overall accuracy of the submission to the ITU competition, as a function of the input SNR.

6. CONCLUSION

In this work, we detail our submission to the 2021 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 our novel Feedback Magnitude Pruning technique. Our submission, compared to Itera-

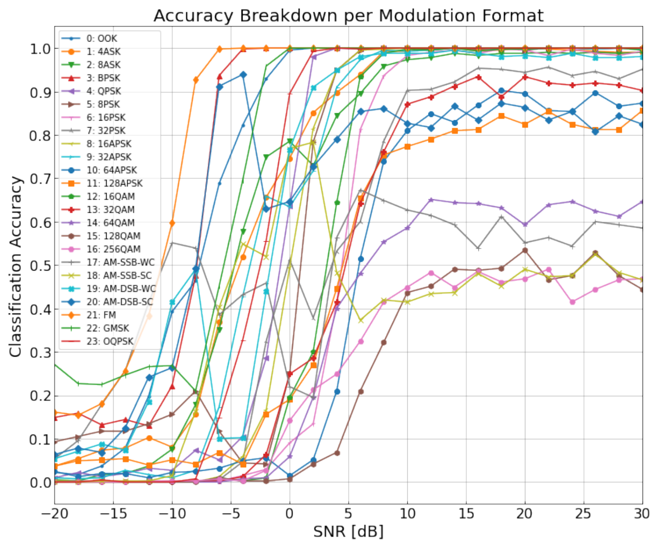

Fig. 4 – Accuracy of the model, per modulation format, of the ITU competition submission, as a function of the input SNR.

tive Magnitude Pruning, achieved a normalized inference cost of 0.0419 versus 0.042467, yielded a compression ratio of 813 versus 9.3, and was awarded second place.

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

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

[7] 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).

[8] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size". In: *arXiv preprint arXiv:1602.07360* (2016).

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

[10] Y Umuroglu, NJ Fraser, G Gambardella, M Blott, P Leong, M Jahre, and K Vissers. "Finn: A framework for fast, scalable binarized neural network inference. Acm". In: *SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. 2016, pp. 65–74.

[11] *ITU-ML5G-PS-007: Lightning-Fast Modulation Classification with Hardware-Efficient Neural Networks*. June 2021. URL: `https://challenge.aiforgood.itu.int/match/matchitem/34`.

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

[13] 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.

[14] 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.

[15] 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).

[16] 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).

[17] 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.

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

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

[20] 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).

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

[22] 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`.

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

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

[25] 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).

[26] 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.

[27] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. "What is the state of neural network pruning?" In: *Proceedings of machine learning and systems* 2 (2020), pp. 129–146.

[28] Stephen Hanson and Lorien Pratt. "Comparing biases for minimal network construction with backpropagation". In: *Advances in neural information processing systems* 1 (1988).

[29] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. "Learning structured sparsity in deep neural networks". In: *Advances in neural information processing systems* 29 (2016).

[30] 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.

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

[32] 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).

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

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

[35] 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).

[36] 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* ().

[37] 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.

[38] 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).

[39] 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).

[40] 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.

[41] 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.

[42] Léon Bottou et al. "Online learning and stochastic approximations". In: *On-line learning in neural networks* 17.9 (1998), p. 142.

[43] Boris T Polyak. "Some methods of speeding up the convergence of iteration methods". In: *Ussr computational mathematics and mathematical physics* 4.5 (1964), pp. 1–17.

[44] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[45] Leslie N Smith. "Cyclical learning rates for training neural networks". In: *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE. 2017, pp. 464–472.

[46] 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.

[47] 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.

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

[49] 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.

[50] 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.

[51] 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 received his BS in biomedical engineering ('17) from Rochester Institute of Technology and an MS in electrical and computer engineering ('20) from the Georgia Institute of Technology. Currently, he is an electrical and computer engineering PhD candidate in the Terabit Optical Networking Consortium at Georgia Institute of Technology and a research engineer at Georgia Tech Research Institute.

Rajib Bhattacharjea received his BS, MSECE, and PhD degrees from The Georgia Institute of Technology. He spent the next five years as a member of the research faculty at the Georgia Tech Research Institute, where he served as technical lead and principal investigator on dozens of programs. He recently joined DeepSig, Inc., a technology startup company working at the intersection of artificial intelligence and radio frequency signal processing, where he is a key technical contributor to DeepSig's federal and defense programs.

Andrew Stark completed his PhD in electrical engineering at the Georgia Institute of Technology in 2012 as a part of the Terabit Optical Networking Consortium, investigating ultra-high speed optical communications and digital signal processing for terabit-per-second signaling. Since then he has worked for a year at Adtran Inc in passive optical networks and at the Georgia Tech Research Institute in the intersection of photonic, radar, and electronic warfare systems.