POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

Attention-based channel-wise pruning with QAT

Final report

Matilda Novshadian

Muhammed Mamdouh Salah Abdelefatah

Professor:

Dr. François Leduc-Primeau

Teacher's Assistant:

Reda Bensaid

Date:

April 23, 2024

1 Introduction

Deep Neural Networks (DNNs) have achieved remarkable success in computer vision tasks, but still face challenges on resource-constrained edge devices, due to their large parameter count and computational demands. To address this issue, quantization and pruning techniques are used to optimize the speed, energy consumption, and storage in neural networks. Quantization reduces the precision of parameters, typically by representing them with fewer bits, and pruning, is the process of selectively removing or setting certain weights, channels, or kernels in the network to zero. These measures reduce model size while keeping the accuracy and speeding up inference.

1.a Pruning

When pruning was first introduced for compression, the procedure followed as first removing redundant connections of a network trained to learn which connections are important, and then fine-tuning the weights of the remaining connections. In one of the recent papers, this loop was performed employing attention-based unstructured pruning [1]. In this paper, an attention scalar value is defined across each layer as in Eq.1 where $f_l(x_{l-1}; w_l)$ is the output of a normal neural network's layer, $\hat{f}_l(x_{l-1}; w_l; a_l)$ is the output of an attention neural network's layer, and a_l is the attention value of layer l. The attention value both highlights the importance of the layer for the network, and determines the pruning ratio using Eq.2, where α is a positive hyper-parameter called the pruning factor that controls the computation of the pruning ratio p_l of layer l [1].

$$\hat{f}_l(x_{l-1}; w_l; a_l) = a_l . f_l(x_{l-1}; w_l) \tag{1}$$

$$p_l = (1 - a_l)^{\alpha} \tag{2}$$

In the loss function of this network (Eq. 3), $\mathcal{L}(.)$ denotes the cross-entropy loss, $\phi(.)$ is the sparsity regularizer, and γ and λ are the coefficients for the sparsity regularizer and L2 regularizer of the weights, respectively [1].

$$\min(\hat{w}, A) \sum_{i=1}^{N} \mathcal{L} f(x_i; \hat{w}, A) + \gamma \phi(A) + \lambda \sum_{j} (\hat{w}_j)^2 \tag{3}$$

1.b Quantization

Quantization of neural networks can be described using the formula below, involving scaling factors s and bias b. These factors relate to the adjustments applied during quantization to map floating-point values to a lower precision, i.e. the scaling factor adjusts the range of values, and the bias shifts the values to ensure they fall within a desired bracket. They can vary based on the method of quantization.

$$y = \sum x_i \cdot w_i + b = \sum (s_x \cdot x_{\text{int}}) \cdot (s_w \cdot w_{\text{int}}) + b_{\text{int}}(s_x s_w) = s_x s_w \sum (w_{\text{int}} \cdot x_{\text{int}}) + b_{\text{int}} \tag{4}$$

In this project, we will implement a modified version of these techniques, to better accomplish our objectives which are stated in the following section.

2 Objectives

If we take the quantized output of a layer of a neural network (Eq. 4), applying the attention-based pruning is as straightforward as a simple multiplication such as in the equation below.

$$\hat{y} = a_l \cdot y = a_l \cdot s_x \cdot s_w \sum (w_{\text{int}} \cdot x_{\text{int}}) + a_l \cdot b_{\text{int}} \tag{5}$$

It can be seen that attention-based pruning scheme is quite compatible with quantization, which is why we propose utilizing both pruning and quantization, to reduce the network size as much as possible while attempting to maintain a reasonable accuracy.
Throughout this report:

- Initially, we explore attention-base channel-wise pruning, instead of layer-wise pruning as suggested in [1]. Layer-wise unstructured pruning often leads to sparse models requiring specialized hardware capable of efficient sparse computations. However, channel-wise pruning results in structured sparsity which is a more hardware-friendly solution.
- Consequently, we explore an alternative loss function that expands the range of attention values within each layer while maximizing the proportion of zeroes thus reaching maximum pruning,
- Finally, we implement both pruning and quantization on the pretrained ResNet32 network using the CIFAR-10 dataset while re-training it, to have a comprehensive report on the

best achievable accuracy for the maximum compression possible using these methods, and analyze the contribution of each part of the training to compression and accuracy.

3 Methodology

In this section, we go through the details on the modifications of previously mentioned methods and the implementation of our proposed method, in order to support our objectives.

3.a Channel-wise pruning

For channel-wise pruning, we replace the scalar attention value of each layer as in [1] by a vector $A_l \in \mathbb{R}_{[0,1]}^{C\times 1\times 1}$, where C is the number of channels in the l^{th} layer. This attention vector A_l will be multiplied element-wise by the output of the l_{th} layer ($output_l \in \mathbb{R}^{C\times H\times W}$) as in Algorithm.1. Given these modifications, we define a loss function for attention similar to elastic net [2] which combines both $L1$ and $L2$ norms, with weights α and β respectively. The added hyperparameters helps us gain more control over the contribution of attention loss and the speed of attention updates. Attention loss is added to the cross-entropy loss and $L1$-norm of the weights, making up the total loss of the network. For further illustration please see lines 5-8 of Algorithm.2.

Algorithm 1 Forward Pass of Attention Neural Network

```
1: function ForwardPass(Input)
2:     output_0 ← Input
3:     for l in network.layers do
4:         output_l ← WeightedSum(output_{l-1}, weights_l)
5:         A_{l(normalized)} ← A_l / A_{l(max)}        ▷ Normalize attention using maximum
6:         output_l ← output_l * A_{l(normalized)}     ▷ Apply normalized attention
7:     end for
8: end function
```

In the algorithm above, there is a normalization term for attention before loss calculation. That is because, without normalization, the gradient of cross-entropy loss becomes significantly small relative to the attention loss and the network focuses mostly on minimizing attention values. Once all the attention values of a layer become zero, the layer becomes 100% pruned, and disrupts the network by preventing any further forward or backward passes, which is irrecoverable as shown in Fig.1. To ensure that the network does not drive all attention values to zero to minimize the loss, attention values are normalized. They can be normalized to a statistical portion of the data, for example, the 75^{th} percentile. This ensures that in every iteration, any

Algorithm 2 Training Network with Channel-Wise Attention

function TRAINNETWORK(network, attention_epoch, sub_epochs)
  for *attention_epoch* ← 1 **to** *attention_epochs* **do**
    for *recovery_epoch* ← 0 **to** *recovery_epochs* **do**
      output ← **FORWARDPASS(Input)**
      loss ← CROSSENTROPYLOSS($output_l$, *true_labels*)
      attention_loss ← λ_{att}* ELASTICNET($A_{normalized}$, α, β) ▷ Elastic Net loss
      reg_loss ← λ_{reg}* L_1(network.weights) ▷ L1 regularization
      total_loss ← *loss* + *attention_loss* + *reg_loss*
      BACKWARDPASS(*total_loss***)**
      UPDATEWEIGHTS(*network.weights*)
      if *recovery_epoch* = 0 **then** ▷ Only update attention in the first recovery epoch
        UPDATEATTENTION(*network.A*)
      end if
      if *test_acc* $\geq$ *moving_average* **then**
        BREAK ▷ If test accuracy exceeds average accuracy start new attention epoch
      end if
    end for
  end for
end function

attention value as big as or larger than the 75% of data points is mapped to 1 or above, thus effectively keeping 25% of the attention values away from zero. This also sets a layer's pruning upper limit at 75%. Additionally, the closer the pruning ratio of a layer approaches this limit, the slower the pruning becomes. This is a sparsity factor that can be varied as desired. Initially, we considered using the 75^{th} percentile, which in comparison to normalizing to the maximum or higher values, lets the network prune itself much slower and avoids big loss in accuracy. But with enough care in choosing the loss function hyperparameters, we were able to normalize attention values to their maximum in every iteration, without much damage to accuracy. This way we reach maximum sparsity, exploring a space where pruning could be potentially aggressive enough to only leave one channel unpruned.

Even with these precautions, if the network continuously prunes itself, there will be a sharp decline in the network's accuracy beyond recovery. Therefore, we propose the addition of recovery and define two more hyper-parameters: attention_epochs and recovery_epochs. For every iteration in the attention_epochs loop, attention values are updated only for the 0^{th} recovery_epoch, and for the rest of the recovery_epochs, the network pauses attention update and trains its weights to recover from pruning.

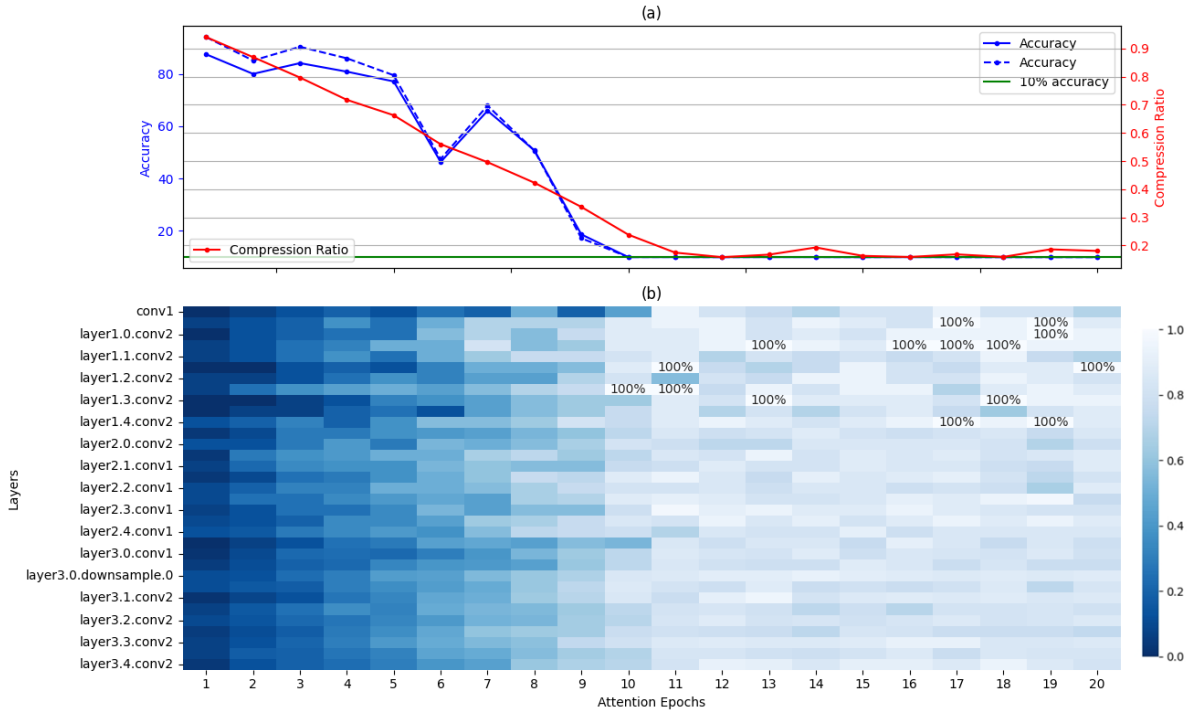

Figure 1: (a) Unnormalized attention training and compression ratio per epoch (b) Color-coded sparsity ratio progression of each layer in unnormalized training

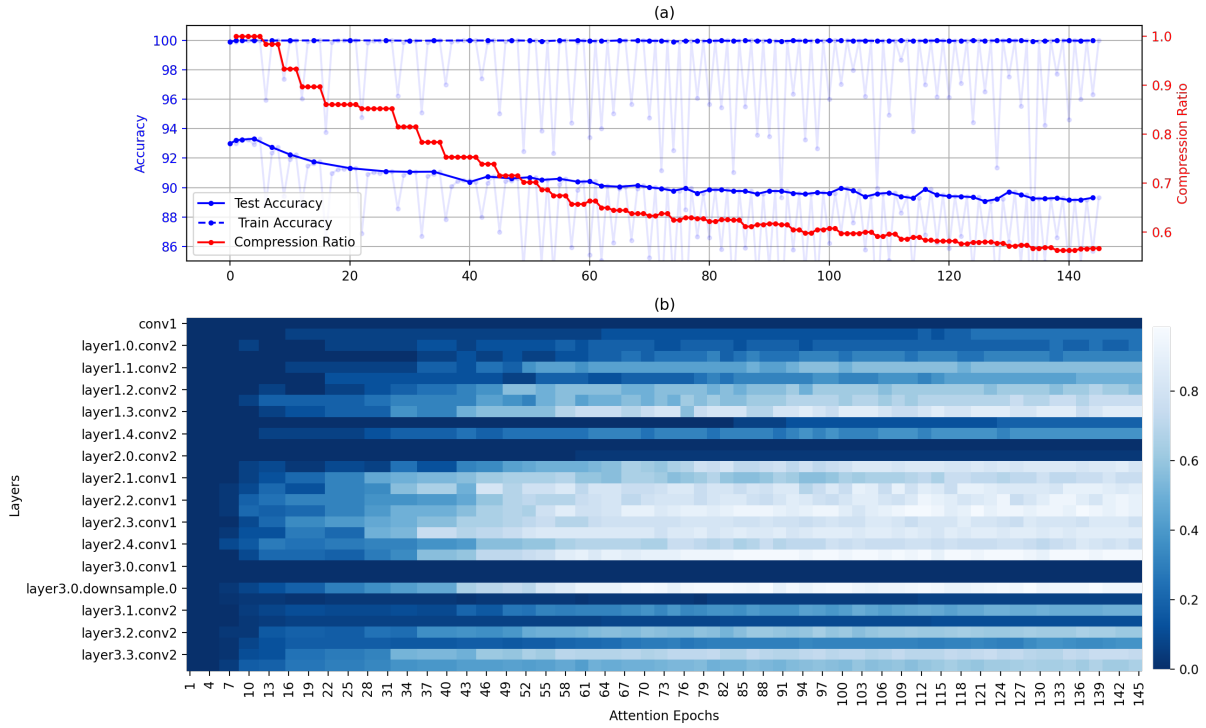

Figure 2: (a) Normalized attention training and compression ratio per epoch (b) Color-coded sparsity ratio progression of each layer in normalized training

This may significantly increase training time due to the infrequent updates of attention values, therefore, we add a test accuracy threshold which lets the network skip recovery if the test accuracy surpasses this threshold. We use the moving average of achieved test accuracies in the history of the network training, which sets a dynamic upper limit for the highest achievable accuracy at each stage. The effect of normalization is shown in Fig.2 where the training and test accuracies are plotted against epochs, highlighting the points at the end of every recovery epoch.

After successfully implementing attention-based channel-wise pruning with the required modifications in the loss function and the training algorithm, we proceed to include quantization in the training scheme as well.

3.b Quantization-aware training

After pruning, we move onto adding quantization-aware training (QAT) by inserting quantization and de-quantization steps into the training process and using the Straight-Through Estimator (STE) [3] to allow gradients to flow through quantization. After applying channel-wise pruning, the weights of different channels within a single layer show significant variation in their ranges as shown in Fig.3. Therefore, it is more logical to implement quantization in a channel-wise manner to account for these large differences in weight ranges across channels.

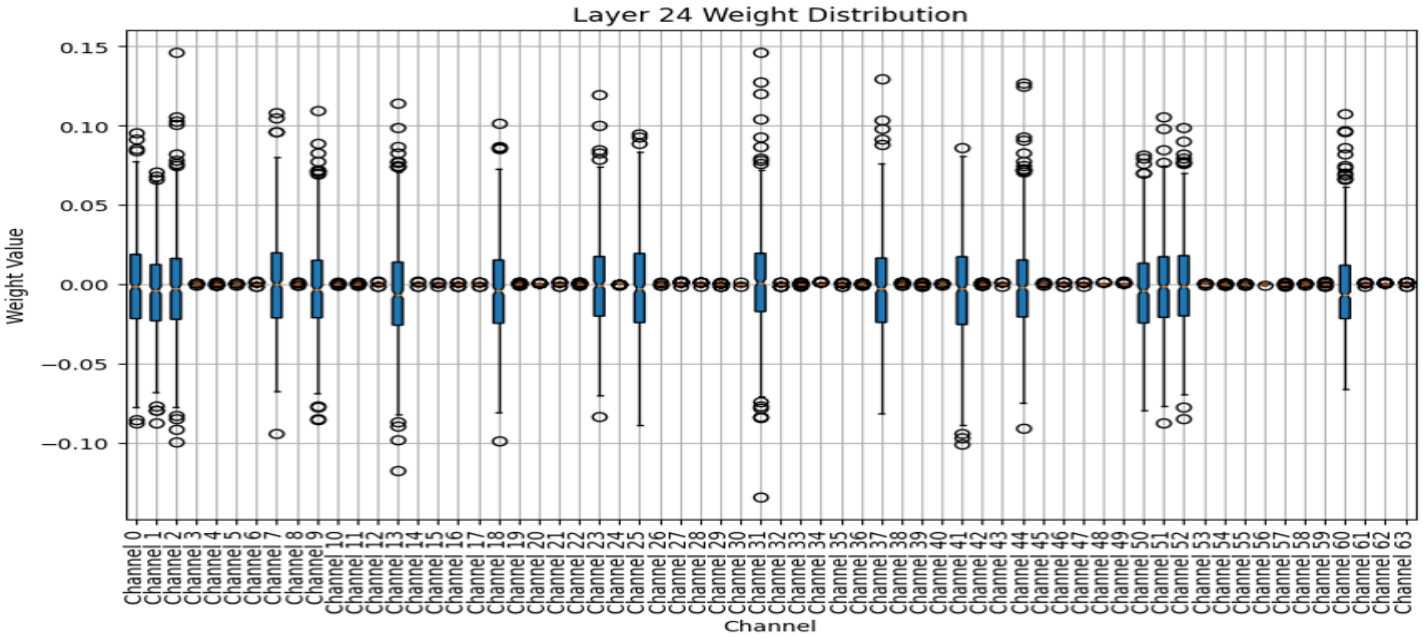

Figure 3: Boxplot of the unquantized channel weights within layer 24 resulted from a pruned network

By adding quantization, we train the network using the quantized forward pass of Algorithm.3 accounting for both attention pruning and quantization. It is good to note that attention can be applied by multiplying either the outputs of the layer or the weights of the layer by the attention values, which are mathematically equivalent in case of using symmetric quantization. We chose to multiply the weights by the attention values then apply channel-wise quantization symmetrically with no bias on the weights.

Algorithm 3 Forward Pass of Quantized Attention Neural Network

function FORWARDPASS(Input)
  $output_0 \leftarrow Input$
  for l **in** $network.layers$ **do**
    $A_{l(normalized)} \leftarrow A_l / A_{l(max)}$ ▷ Normalize attention using maximum
    $attended\_weight_l \leftarrow weight_l * A_{l(normalized)}$ ▷ Apply normalized attention
    QUANTIZE($output_{l-1}$)
    CHANNELQUANTIZE($attended\_weight_l$) ▷ Channel-Wise Quantization
    $output_l \leftarrow$ **WEIGHTEDSUM**($output_{l-1}$,$attended\_weight_l$)
  end for
end function

4 Results

Results of the training progression for the full precision network with normalization to maximum value are represented in Fig.4 where each point in the figure corresponds to a step in the training progress (taken from Fig.2a).

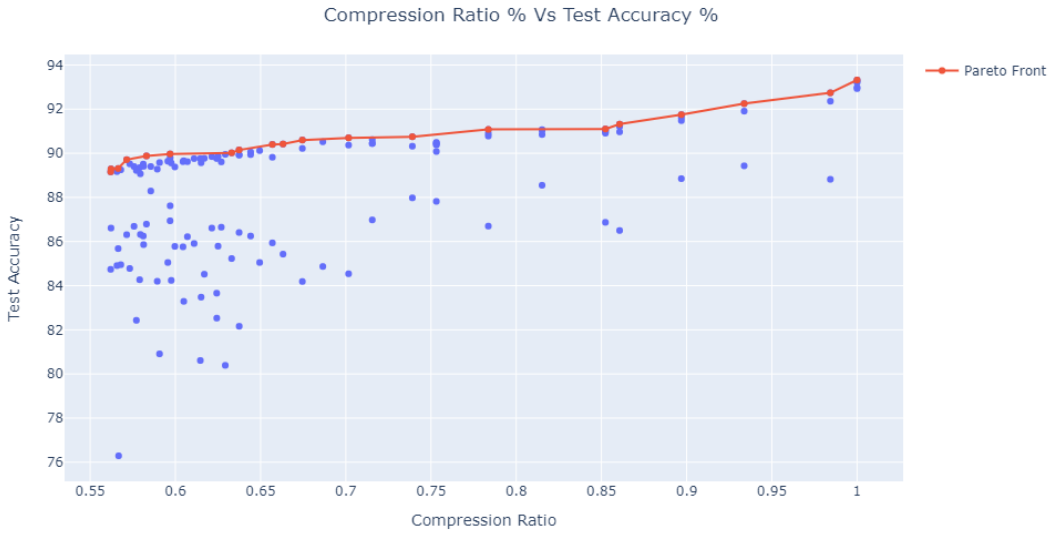

Figure 4: Test accuracy per compression ratio for the attention pruned normalized to maximum experiment

This figure gives us an idea of the highest achievable accuracy for a given compression ratio and sparsity factor in the pruned ResNet32 with our pruning method. We can see that we can maintain more than 89% accuracy while having a 56%reduction in the network size, i.e. 44% pruning. Given that the baseline accuracy is 93%, our method is very effective in maintaining accuracy. Note that we define compression ratio as the size of the compressed network over the original size of the pretrained network.

When we look at the heat map representing the sparsity of each layer in the network (Fig.2b), it can be seen that generally the first convolutional layers of each block, do not become as sparse as others. In ResNet, these layers are crucial for initial feature extraction which helps the network learn more complex and abstract features in deeper layers better, therefore it is expected to see the training go in the direction of avoiding high sparsity in these layers.

To understand how pruning and quantization each affect the results and how much using both of them can help with further compression, Fig.5 shows the Pareto-front for different bit-precision experiments in different colors. The progression of size reduction using pruning is highlighted by the line connecting points of different stages of each case with a specific bit-precision.

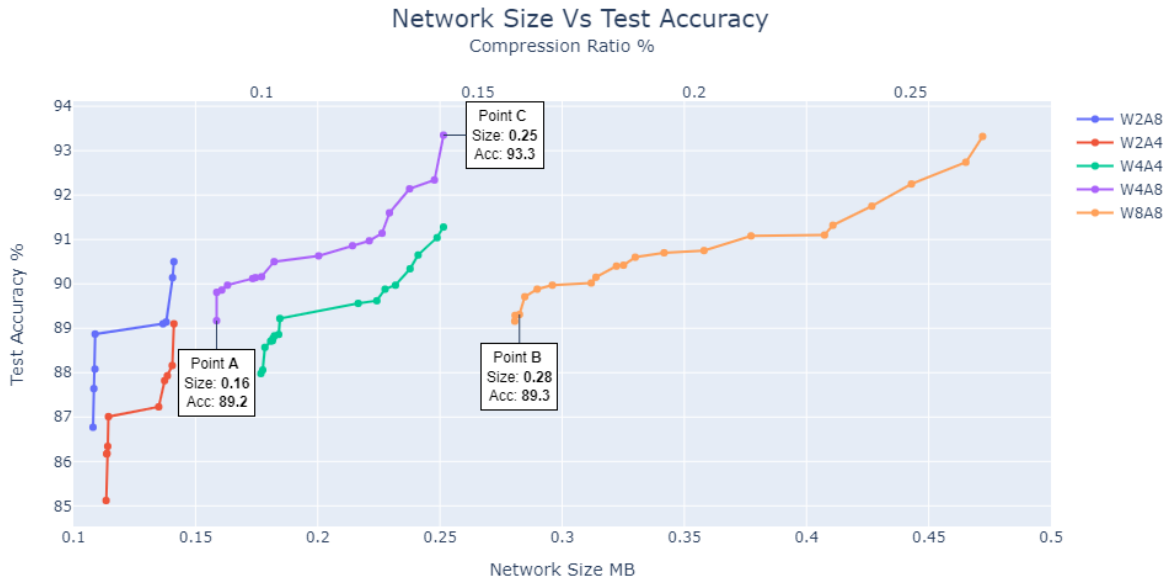

Figure 5: Pareto front of test accuracy per network size (bottom axis) and compression ratio (top axis) for different quantization configurations

Point A, represents the network with 4-bit weights and 8-bit activation (W4A8) with maximum pruning, having about 16% of the size of the original model with 89.2% accuracy. In this case, quantization-aware training did not reduce the model accuracy at all. Comparing point B, the

maximally compressed W8A8 case, to point C, the W4A8 quantized network with no pruning, we can see that point C easily dominates point B both in terms of compression and accuracy, showing that generally quantization is more effective than pruning. However, most hardware is compatible with 8-bit variables, and in cases of not having specialized hardware available, pruning is an effective choice for model compression. It can also be seen that pruning is more effective for networks that have a higher bit precision and for cases such as 2-bit networks, pruning results in big jumps in accuracy degradation.

5 Conclusion

In conclusion, we saw that pruning can almost halve the network size and our proposed adaptive pruning method can maintain the accuracy above 89% for ResNet32 on the CIFAR-10 dataset with 56 % compression. Furthermore, a combination of attention-based channel-wise pruning and channel-wise quantization-aware training helps further reduce the model size without damaging the accuracy. Quantization by itself is more effective for model size reduction than pruning, however, in hardware-constraint situations where the bit precision of variables is fixed, pruning is a good measure for model compression.

6 Appendix

The code for the training process and the collected data, as well as the code for the generation of all the figures included in the report can be found in Github Repository.

References

[1] Qiang He, Wenrui Shi, and Ming Dong. “Learning Pruned Structure and Weights Simultaneously from Scratch: an Attention based Approach”. In: (2021). arXiv: 2111.02399 [cs.LG]. URL: http://arxiv.org/abs/2111.02399.

[2] Hui Zou and Trevor Hastie. “Regularization and Variable Selection via the Elastic Net”. In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. “Estimating or Propagating Gradients through Stochastic Neurons for Conditional Computation”. In: *arXiv preprint arXiv:1308.3432* (2013).