DEEP NEURAL NETWORKS COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern Deep Neural Networks (dnn) models used in computer vision applications are compelling. They are widely used to solve a variety of problems and the increase in data size implies that the model could be very large and complex, and therefore increased in computational requirements. The number of parameters in recent state-of-the-art networks makes them difficult to deploy in edges devices like mobile phones, watches or drones where memory and energy are limited. We are working on the implementation of techniques that significantly reduce the size of a very large and powerful vision model while preserving as much of its performance as possible. We built classification models on the mnist dataset and used it with pre-trained models on ImageNet on the Cats & Dogs dataset. We performed a closer examination of the effectiveness (mathematical and implementation aspects) of Knowledge distillation (KD), Pruning and Quantization techniques. Firstly, we implemented transfer learning which consists on modifying the parameters of an already-trained network to adapt to a new task on a new dataset, then secondly, we trained this network by using a gradual pruning approach that requires minimal tuning and can be seamlessly incorporated within the training process. Thirdly, the Quantization has helped us reduce the number of bits required to represent each parameter from 32-bit floats to 8 bits. We significantly reduced bandwidth and storage. On MNIST, we reduced the model from 12.52 MB to 0.57 MB with no loss of accuracy. After the transfer learning and pruning step, we reduced the MobileNet from 12.48 MB with an accuracy of 0.9556 to 2.91 MB with an accuracy of 0.9516. We also empirically show our method's adaptability for classification based architecture vgg16 and VGG19 on datasets Cat & Dogs observing that the entire pruning pipeline plus post-quantification at 8 bits works well up to 70% level of sparsity, suffer only very small losses in accuracy and the size of the model obtained by transfer learning are divided by 10.

Keywords: Deep Learning, Compression model, Knowledge distillation, Pruning, Quantization, Computer vision

1 INTRODUCTION

1.1 PROBLEM STATEMENT

Over the last few years, Deep Learning (dl) has been used to solve several problems, ranging from computer vision, speech recognition, social media, to natural language processing, health, finance, cybersecurity, and others. DL is one way of implementing machine learning via what are called Deep Neural Networks (DNN) that are algorithms that effectively mimic the human brain's structure and function. DNN have recently received lots of attention, been applied to different applications and achieved dramatic accuracy improvements in many tasks.

However, one of the main problems of (dnn) modern models is its huge data storage requirement; the more data increases the deep model also become complex and larger to learn pattern from the data. The large network structure requires **high computational complexity** and **memory cost**. The amount of memory space and lower power consumption they need can make their deployment very prohibitive especially for some real-time applications such as online learning, virtual reality, augmented reality, and smart wearable devices like mobile phones or watches, where memory and energy are limited.

Because the dnn are large (in terms of the number of parameters), they require more space for storage and also more energy to compute the results. For instance, some original architecture like the well-known VGG architecture requires considerably larger memory (528 Mb) to store the weight parameters needed for classifying a single image[1][2]. In a cloud-based environment with abundant computational capabilities enabled by multiple graphical processing units (gpus), such massive memory requirements might not be considered as a restriction, but the connection cost and users number might. However, in the case of mobile or edge-based embedded devices with limited computational capabilities, such resource intensive dnn cannot be readily applied. Recently, the proliferation of deep learning applications on mobile iot (Internet of things) devices, including smartphones, has unveiled this as a major hurdle for a widespread use.

Compressing of dl models allows saving storage space and response time and therefore can have significant impacts on distributed systems, embedded and edge devices by extending dnn applications in these areas. Thus, the design of dnn that require less storage and computation power has established itself as a new research direction. Particularly, the modification of large models that reduces the memory requirements while retaining as much of its performance as possible is referred to as compression of neural networks. Another direction is the design of more memory efficient network architectures from scratch.

It is from those problems and challenges that researchers have developed methods for compressing neural network models. In this thesis we focused on three main methods: **Knowledge distillation (kd), Pruning and Quantization**.

The first method is Knowledge Distillation (kd), the idea behind kd is to distill a knowledge of large and complex network which is called the teacher network and transfer it to a small and simple network which is called the student network. The second method is Pruning, which is a technique to reduce the parameters of a large and complex network by pruning weights/neurons during the training of this large network; Pruning the network means removing the redundant connections and keeping only the most informative connections. The third method is Quantization: It is the process of approximating a neural network (nn) that uses floating-point numbers by a neural network of low bit-width numbers, reducing the number of bits that represent a number (weights, bias).

2 Research Questions and Objectives

Our main objective in this thesis is to implement methods that significantly compress a dl model while preserving as much as possible its performance. From the perspective of an on-device neural network inference, given a dnn, the question are how to make it more efficient, does a large model (as VGG19, MobileNet) have any information (or connections) it doesn't really need? Even though this model is already quite small, can we make it even smaller without making it lose its accuracy? How to reduce cpus(Central Processing Units) and hardware accelerator latency, processing, power, and model size with little degradation in model accuracy. For a given deep model, how can we arrive at the smallest model with no loss of accuracy?

The next evolution in machine learning will move models from the cloud to edge devices. To answer the research question, our goals are the following:

- Compress a large model to a smaller model: reduce the size of a very large and powerful vision model while preserving as much of its performance as possible.
- Explore the methods for model compression (KD, Pruning, Quantization) : The mathematical formulation as well as their implementation.
- Resume and present a baseline of each approach used (the process) for each of three methods.
- Experimentation, results and discussion.

In this piece of work, we will compare the quality of models obtained through these different approaches, and also the one obtained by using the methods together. Our main insight is that pruning and quantization can compress the network without loss of accuracy. We used transfer Learning to adapt large pretrained networks on Imagenet to specialized tasks on smaller dataset like Cats and Dogs data. Our work makes the following contributions: 1. We combine pruning and quantization to reduce the network complexity, 2. The experimental results are demonstrated on two available datasets, and show that significant improvement in classification accuracy and model size compared to the other method used alone. 3. We provide a comparison of those three methods.

3 Literature Review

Model compression is a critical technique to efficiently deploy nn models on mobile devices which are limited in terms of computational resources and power budgets. The task is the modification of large models that reduces the memory requirements while retaining as much of its performance as possible is referred to as compression of nn. Another direction is the design of more memory efficient network architectures from scratch. In order to learn efficient dnn, a number of methods have been proposed to eliminate redundancy. Denil et al **?** demonstrate that neural networks are often over-parameterized and removing redundancy is possible. In the same order, various methods have been proposed to accelerate the fully connected layer **????**. To compress the whole network, Zang et al **??** presented an algorithm using asymmetric decomposition and additional fine-tuning. Kim et al **?** presented a method called one-shot whole network compression also to compress the entire cnn. The three different approaches that we will investigate in our work lie on the following lines: Knowledge distillation, Pruning, and quantization. In the following, we will discuss the literature review of the different approaches.

3.1 Knowledge Distillation

- **KD from Output** The concept of KD was first proposed by Caruana et al **?** in the context of neural model compression; exploiting Knowledge Transfer(kt) to compress the model. Ba and Caruana **?** adopted the idea of **?** to compress deep networks into shallower but wider ones, where the compressed model mimics the logits or pre-softmax of teacher model during his training. After that, Huang et al **?** proposed a KD by Neuron Selectivity Transfer (nst). They matched the distributions of neuron selectivity patterns between teacher and student networks by devising a new kt loss function by minimizing the Maximum Mean Discrepancy (mmd) metric between these distributions.
- **KD from single/Multiple Layer(s)** In the case that the target model is very deep, it meets the difficulty of convergence problems. Romero et al **?** proposed an improved method by introducing intermediate-level hints, using not only the logits layer but earlier ones too. The core idea of the method is to enable the student model to learn the intermediate representation from the teacher model. The middle layer of the teacher model is called the hint layer and that of the target model is called the guided layer. Unfortunately, there is no principled way to do this. To solve this, Yim et al **?** proposed another concept of knowledge distillation which employs the relationship between layers considered to be more representative of the knowledge than the model output. Therefore, the extracted feature maps from two layers are used to generate the Flow of Solution Procedure (FSP) matrix. This represents the relationship between layers. By minimizing the distance between both FSP matrices of the teacher network and student network, this approach enforces the knowledge of the teacher model to be transferred to the student model.
- **Layer-Level KD** Inspired by **?**, **?** proposed a novel Layer Selectivity Learning, using lsp (Layer Selectivity Procedure) to transfer knowledge from one or more layers based on the analysis of the diversity and discrimination of feature maps in a layer-wise manner. They firstly use an asymmetric dual-model learning framework, called Auxiliary Structure Learning (asl), to train a small model with the help of a larger and well-trained model. Then, the intermediate layer selection scheme, called the Layer Selectivity Procedure (LSP), is exploited to determine the corresponding intermediate layers of source and target models.
- **To improve student performance**, Sau and Balasubramanian **?** added random perturbations(noise) into soft labels(teacher) to simulate learning from multiple teachers to make the student model more robust. Multiple teachers are a way to increase robustness. Zhang et al. **?** presented a method to train a thin deep network by incorporating multiple teacher networks not only in output layer by averaging the softened outputs (dark knowledge) from different networks, but also in the intermediate layers by imposing a constraint about the

dissimilarity among examples. KL divergences between pairs of students are added into the loss function to enforce the knowledge transfer among peers. You and al **?** proposed to train a thin deep network by incorporating multiple teacher networks not only in output layer by averaging the softened outputs (dark knowledge) from different networks, but also in the intermediate layers by imposing a constraint about the dissimilarity among examples, a strategy to unify multiple relative dissimilarity information provided by multiple teacher networks. Anil et al. **?** introduced an efficient distributed online distillation framework called co-distillation and argued that distillation can even work when the teacher and student are made by the same network architecture. The idea is to train multiple models in parallel and use distillation loss when they are not converged, in which case the model training is faster and the model quality is also improved. **?** showed that the student network performance degrades when the gap between student and teacher is large. They introduced multi-step knowledge distillation which employs an intermediate-sized network called teacher assistant to bridge the gap between the student and the teacher, a framework Teacher Assistant KD (takd).

3.2 PRUNING

The approach of pruning was an idea from **?**. Based on the idea of learning only the important connections in neural networks, Han et al **?** proposed a compression by reducing the number of weights and **?** proposed pruning, quantization and Huffman encoding methods to reduce computational consumption of large networks. They first pruned the network by learning only the important connections, then weights were clustered to generate the codebook to enforce weight sharing, finally and apply Huffman coding.

This idea was first proposed by Yann Le Cunn et al. back in 1990 in their famous paper called Optimal Brain Damage (obd) and was later applied to modern deep networks **?**. Research on pruning is mostly concerned with the question of how the contribution of the weights should be measured. In OBD, the contribution is measured by the effect on the training error in the case of setting this particular parameter to zero. Obviously, this method becomes computationally infeasible for deep networks. In Deep Compression, Han et al simply prune the weights with lowest absolute value, reducing the number of weights to 10% of its original size for fully connected, and around 60% for convolutional layers at no loss of prediction accuracy **?**. Tu et al. **?** proposed a method to accurately measure the Fisher information associated with weight and used it as a measure for its contribution. Dong et al. **?** proposed Pruning via Layer-wise Optimal Brain Surgeon (obs), where the parameters of each individual layer are pruned independently based on second-order derivatives of a layer-wise error function with respect to the corresponding parameters. In the same idea, Aghasi et al. **?** proposed an algorithm called Net-Trim to prune (sparsity) a trained network layer-wise by removing connections at each layer by solving a convex optimization program.

Leroux et al **?** aimed at reducing computation by evaluation of certain convolutional filters and pruned low-impact filters at runtime. Instead of removing individual weights one at a time as done in previous works, Srinivas et al **?** explored the redundancy among neurons and proposed to remove one neuron at a time.

- **Magnitude based-method: Iterative Pruning + Retraining**:

  For Learning both Weights and Connections for Efficient Neural Networks, **?** described a method to reduce the storage and computation required by neural networks by an order of magnitude without affecting their accuracy by learning only the important connections. Their method prunes redundant connections using a three-step method. First, they train the network to learn which connections are important. Next, they prune unimportant connections. Finally, they retrain the network to fine-tune the weights of the remaining connections.

  In the work Pruning cnn for Resource Efficient Inference, Molchanov et al. **?** interleaved greedy criteria-based pruning with fine-tuning by back-propagation. Focused on transfer learning where large pre-trained networks are adapted to specialized tasks they propose a new criterion based on Taylor expansion that approximates the change in the cost function induced by pruning network parameters.

- **Pruning Filters for Efficient ConvNets**: **?** proposed to evaluate the importance of the filters and remove the unimportant ones in line with previous works. For **?** if you could rank the neurons in the network according to how much they contribute, you could then remove the low ranking neurons from the network, resulting in a smaller and faster network.
- **Channel Pruning for Accelerating Very Deep Neural Networks**: He et al. **?** propose an iterative two-step algorithm to effectively prune each layer by a lasso regression-based channel selection and least square reconstruction. We further generalize this algorithm to multi-layer and multi-branch cases.
- **dsd: Dense-Sparse-Dense Training for Deep Neural Networks**: **?** proposed DSD, a dense-sparse-dense training flow for regularizing dnn and achieving better optimization performance. In the first D (Dense) step, they trained a dense network to learn connection weights and importance. In the S (Sparse) step, they regularize the network by pruning the unimportant connections with small weights and retraining the network given the sparsity constraint. In the final D (re-Dense) step, they increase the model capacity by removing the sparsity constraint, re-initialize the pruned parameters from zero and retrain the whole dense network.
- **Exploiting Sparsity in Recurrent Neural Network (rnn)**: Narang et al **?** pruned weights during initial training in order to gain sparsity of the given RNN model by using hyper-parameters of pruning to determine the threshold. For approximately the same number of parameters, gradual pruning (proposed) is 7% to 9% better than hard pruning (all parameters below a certain threshold are pruned at a particular epoch).

3.3 QUANTIZATION

Quantization techniques aim is to reduce the number of bits required to represent each parameter from 32-bit floats to 16, 8 bits or fewer. The use of vector quantization methods to compress CNN parameters is mainly inspired by the work of Denil et al **?**, who demonstrated the redundancies in neural network parameters. They show that the weights within one layer can be accurately predicted by a small (5%) subset of the parameters, which indicates that the neural network is over-parameterized. Gong et al **?** applied vector quantization (k-means clustering to the weights) methods to explore the redundancy in parameter space, focusing on compressing dense connected layers, to avoid the storage problem. Wei et al **?** did Quantization Mimic, First they quantized the large network, then mimicked the quantized small network.

3.4 PUT ALL THOSE METHODS TOGETHER

Song Han et al **? ?** had successfully combined the above techniques. In their work called Deep Compression, first, a trained network is pruned by setting connections to zero if the absolute value of the weight is below a certain threshold. Second, quantization and weight sharing are applied. The weights are clustered into 256 groups for convolutional, and 32 groups for fc-layers, respectively, using k-means. Hence, a weight can be represented using 8 bit (in convolutional layers) and 5 bit (in fc-layers) indices representing the centroids of the corresponding cluster. Weights are not shared across layers. The centroids are then fine-tuned in an additional retraining phase, where the loss with respect to the centroid of a cluster is simply given by the additive loss of the weights that belong to it. Finally, Huffman-coding is applied to the indices and centroids to further compress the representation of the parameters. Applying this method to SqueezeNet resulted in 10x compression using 64 clusters (i.e. 6 bit representations for weights) without loss of accuracy on a network architecture which is already optimized for compression.

In the next section, we will introduce the concept of deep learning; explain how it works, and describe the transfer learning technique. We will go into more detail on the approaches chosen for each of the compression methods: Knowledge Distillation, Pruning, and Quantization.

4 CONCLUSION

In this chapter, we conclude the thesis with a summary of its finding and suggest directions for further work.

4.1 Discussion and Recommendations

The relevance of compression of Deep Learning Model is on applications of Artificial Intelligence for real world; It will be easy to deploy dl Model in a small device with a camera or using images (like phones, smart watches, sensors, drones). In the literature review, the concept of each method KD, Pruning and Quantization is well defined but the approaches are changing from one paper to another. Each one has its particularity. After experimentation, we briefly summarize these three types of methods in table **??**.

- KD : The KD can only be applied to classification tasks with softmax loss function. The model assumptions are too strict to make the performance competitive with other types of approaches, the performance are sensitive to applications and network structure and only support train from scratch.
- Pruning : Robust to various settings but can achieve good performance. Pruning method can support both trained from scratch and pre-trained models. the pruning approach implemented is a gradual pruning technique that can be applied across different architectures and can be incorporated within the training process.
- Quantization : It works well for large models, but with small models with less redundant weights, the loss in precision adversely affects accuracy. The simplest approach to quantizing a neural network is to first train it in full precision, and then simply quantize the weights to a fixed-point. As said before , the challenge after pruning is to make the network more immune to precision loss. By default a trained model is 32-bit floating point representation. Quantization of the parameters to a reduced precision number representation, at 8 bit-integer is an effective method for model compression.

By combining pruning and quantization, there is a small sacrifice of accuracy but we get a small size of dnn.

- If we prune too much at once, the network might be damaged so much it won't be able to recover.
- Pruning + Quantization work well together
- We can prune a larger dense network to achieve better than baseline performance while still reducing the total number of parameters significantly. Pruning a DNN reduces the size of the model, and can also help achieve significant inference time speed-up using sparse matrix multiply.
- In moving from 32-bits to 8-bits, we get (almost) 4x reduction in memory straightaway. Lighter deployment models mean they hog less storage space, are easier to share over smaller bandwidths and easier to update.

We believe this kind of work opens new frontiers for real world AI applications. We strongly recommend to use Pruning + Quantization as a tool for compressing a nn for deployment in resource-constrained environments and to further explore Pruning as compression method.

5 Results and Discussions

5.1 Pruning and Quantization

The pruning and quantification results of the VGG16 network are given in the figure 5.1. The diagram below shows the trade-off between the accuracy and size of the different VGG16 models for Cats & Dogs classification in pruned and pruned + quantized modes. We can observe that the entire pruning pipeline plus quantification works well up to 70% level of sparsity, suffers only very small losses in accuracy and the size is divided by 10.

The summarized results obtained from Cats and Dogs data on VGG19 are given in figure 5.1.

This figure represents the variation of the size and the accuracy as function of different levels of sparsity added to the quantization at 8 bit precision of the model obtained by Transfer Learning of VGG19 on Cats & Dogs dataset. We can observe that the size and the accuracy, represented by the yellow and gray line are decreasing when the levels of sparsity are increasing.

It can be noticed that when the sparsity level is low, the two lines can be confused, which means that the accuracy is not changing, but the size yes, and we have a significant size reduction after pruning + Quantization (represented by the orange bar). As we increase the sparsity of the model more precisely from 70% we observe a slight shift between the two lines, which means that we lose in accuracy even if the size is reduced.

Finally, we have described the experimentation of TL, KD, Pruning and Quantization performed on MobileNet, VGG, etc. We provided and commented some results. The last step will be a summary discussion and a general conclusion.

6 CONCLUSION AND FUTURE WORK

The topic of this paper was ”Deep Neural Network Models Compression ”. The principal objective was to compress a given dl model while preserving as much as possible its performance. The main research question was to find how to compress a model, which method is the most appropriate given a model and also how we can combine different approaches to get better results. To answer these questions, we started by studying how the deep models work and exploring the concept of tl which is a machine learning method where a model developed for a task is reused at the starting point for a model on a second task.

For compression methods, we studied three main methods : Knowledge distillation, Pruning and Quantization. We studied the mathematical part of each method and implemented them using Tensorflow. We designed models from scratch on the MNIST dataset and used pre-trained models like MobileNet, VGG16 and VGG19 on the Cats & Dogs. On MNIST, we reduced the model from 12.52 MB to 0.57 MB with no loss of accuracy. After Transfer learning and the pruning step, we reduced the MobileNet from 12.48 Mb with and accuracy of 95.56% to 2.91 Mb with accuracy 95.16%.

From the original VGG16 and VGG19 pre-trained on Imagenet with an accuracy of 71.3% for each one and the size 524 Mb and 529 Mb respectively, we showed that our method's adaptability for classification based architecture VGG16 et VGG19 on datasets Cat & Dogs which is tl, pruning + quantization we obtained the models $100\times$ smaller with accuracy of 96%. As given in figure 5.1 and 5.1.

The main suggested extensions to this work is focused on pruning. To improve the results we mainly propose to study more approach of pruning because this method requires fine tuned hyper-parameters but can gives the best results.

In future work we will investigate on optimization pruning. Find the advanced method to prune and to combine the compression methods. Therefore, we were focused on CNN, we can extend the same experiments on another architecture like rnn.

REFERENCES

A APPENDIX

[width=0.9]PQ$_V GG161.PNG$

Figure 1: Results of Pruning and Quantization with Cats & Dogs on VGG16

[width=0.9]PQ$_V GG191.PNG$

Figure 2: Results of Pruning and Quantization with Cats & Dogs on VGG19

[width=0.9]PQ$_V GG191.PNG$

Figure 3: Sample figure caption.