Efficient CNN-LSTM based Image Captioning using Neural Network Compression

Harshit Rampal
Carnegie Mellon University
hrampal@andrew.cmu.edu

Aman Mohanty
Carnegie Mellon University
amanmoha@andrew.cmu.edu

1 Introduction

In recent years, Deep Neural Networks have gained massive popularity for achieving state of the art results on tasks like classification, recognition and prediction. However, such complex networks have a large computational footprint that impedes their portability to low power mobile devices. Modern mobile devices have light and sleek form factors that further constraints their power and thermal capacity. In such constrained environments, it is desirable to increase the efficiency of Deep Learning/AI based applications that require intensive computations. Hence, there is a need to compress Deep Networks to enable their real-time applications on resource-limited devices. Recently, advanced pruning and quantization algorithms have gained momentum in compressing such networks without compromising their performance.

Conventionally, researchers test their compression pipelines on standalone CNNs ranging from AlexNet[5] to MobileNet[3] and even on RNNs and LSTMs. In this project, we intend to follow an unconventional approach of implementing and validating one or more such state-of-the-art compression algorithms [2, 6, 7, 12] on a novel use case i.e. a CNN-LSTM based image captioning model. Such models have huge number of parameters as they use both a CNN (e.g. VGG16 has about 138M) and a LSTM network (having 2.62M). Deploying such huge networks on mobile devices is unfeasible due to the power, space and thermal constraints. Hence, an end-to-end compression of a captioning model is crucial to leverage its real time applications on mobile devices. Interestingly, a compressed version of such an image captioning model can be deployed on wearable electronics for assisting visually impaired people.

We aim to validate a compressed captioning model on MSCOCO[1] [8] and/or flickr8k[2] datasets. The former contains 123,387 images, each with 5 captions and the latter has 8000 images each with 5 captions. The metric for evaluation will be BLEU scores [9], a commonly used method to evaluate a generated sequence with a reference sequence.

For the baseline work, we experiment magnitude-based weight pruning[2] on two simple CNN architectures. Section 4 reports the results of our work and Section 5 delineates our observations on testing these two pruned models on MNIST and CIFAR10 datasets respectively.

2 Related Work

Our literature review focuses on the different use cases of neural network compression and the different compression techniques. While there is an extensive repertoire of work on the different compression techniques, the use cases of such techniques have not yet been widely explored.

Recently, Google launched Live Caption [1], an on device captioning feature on its mobile devices making use of a RNN based model to caption audio sequences. To be able to deploy it on mobile devices, Google used neural network pruning to reduce power consumption to 50% while still being

[1] https://cocodataset.org/#home
[2] https://www.kaggle.com/shadabhussain/flickr8k

able to maintain the efficacy of the network [10]. Kim et al. [4] designed an object recognition system to recognize vehicles on the road using Faster RCNN. They were able to deploy the system on embedded devices by compressing the network using pruning and quantization. Tan et al. [11] proposed pruning techniques on RNN for an image captioning model. While they were able to reach 95% sparsity level without significant loss in performance, the proposed pruning techniques applied only to the RNN layers and not to the convolutional layers. **The above work implemented and validated compression techniques on standalone CNNs, RNNs or LSTM architectures whereas we will be focusing on the effect of compression techniques on an end-to-end CNN-LSTM based image captioning pipeline.**

There has been an immense advancement in research on neural network compression, especially on pruning, in the last decade owing to deploying neural networks on resource-constrained environments. Among this large corpus of work, we review a select few which align best with our project. Han et al. [2] proposed a three step network to prune and fine tune a network to reduce a significant number of parameters in a network without incurring massive performance loss. Along similar lines, Zhu and Gupta [12] validated the efficacy of magnitude based pruning by comparing pruned networks (Large-sparse) and dense networks (small-dense) with similar memory footprints. Li et al. [7] proposed a filter pruning method to remove filters having small effect on output accuracy and thereby reducing the convolution computation costs. The above methods follow an iterative approach to prune the networks after the network has been trained. On the other hand, Lee et al. [6] proposed a pruning algorithm to prune the network once at initialization prior to training.

3 Method

We employed a magnitude based weight pruning technique[2] to compress two relatively simple CNN models. Our motivation to implement this as a baseline stems from its time of origin and simplicity, which in turn makes it a primer to the vast field of Neural Network compression. This method is focused on identifying and retaining sensitive weight connections while pruning the insensitive ones. Here, 'sensitivity' refers to the contribution of the weight value to the network's accuracy. Hence, after training, the connections with weight values below a specific threshold are pruned. Post pruning, retraining is done to learn, and recover, the final weights of the sparse network. At the time of re-training, weight values are not re-initialized as the retained weights post pruning are at an appropriate value to serve as a good initialization point for gradient descent. This sequence of steps makes one iteration. More number of such iterations are required as one iteration is a greedy search to find an optimal number of connections.

4 Preliminary Results

4.1 Baseline Models

To evaluate the magnitude based weight pruning, we trained two rather simple CNNs on the CIFAR-10[3] and MNIST[4] dataset. The network on the CIFAR-10 dataset (4 conv layers, 2 dense layers and relu activation) was trained for 100 epochs with a batch size of 32 and attained 79.29% accuracy on validation. The network on the MNIST dataset (2 conv layers, 1 dense layer and relu activation) was trained for 10 epochs with a batch size of 32 and attained 98.65% accuracy on validation.

4.2 Pruning

Pruning was implemented using polynomial decay function starting at 0% sparsity (the baseline model). We pruned the baseline models to four sparsity levels - 50%, 80%, 90% and 95%. The network on CIFAR-10 dataset was re-trained for 50 epochs and the network on MNIST dataset was re-trained for 10 epochs, which means that gradually over 50 and 10 epochs respectively, the desired sparsity level is reached. Table-1 summarizes the effect of magnitude based pruning on the baseline models trained on CIFAR-10 and MNIST dataset. Figure-1 demonstrates the effect of pruning during re-training of the networks with different sparsity levels for networks trained on CIFAR-10 and MNIST dataset.

[3] https://keras.io/api/datasets/cifar10

[4] https://keras.io/api/datasets/mnist

	Validation Accuracy(%)					*Non-zero Parameter Reduction*				
Dataset	*Baseline*	*50%*	*80%*	*90%*	*95%*	*Baseline*	*50%*	*80%*	*90%*	*95%*
CIFAR-10	79.29	80.12	79.14	74.01	61.11	1×	2×	5×	10×	20×
MNIST	98.65	98.96	98.36	96.74	84.39	1×	2×	5×	9×	17.5×

Table 1: Effect of magnitude based pruning on the validation accuracy and the number of non-zero parameters for the baseline model and pruned models at different sparsity levels trained on CIFAR-10 and MNIST dataset. `Validation accuracy` is reported in terms of correct predictions over total predictions and `non-zero parameter reduction` is reported in terms of by how many times the number of non-zero parameters is reduced compared to the baseline.

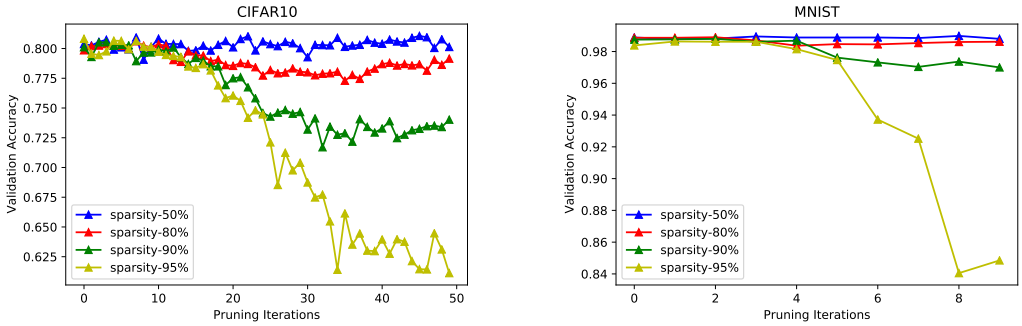

Figure 1: Effect of magnitude based pruning during re-training of the networks with different final sparsity levels for networks trained on CIFAR-10 and MNIST dataset.

Our implementation can be found here: https://github.com/amanmohanty/idl-nncompress.

5 Evaluation

Table-1 clearly shows that **with increasing final sparsity level of the pruned model, the number of non-zero parameters reduce**. With 50% sparsity level (half the number of original parameters), we, even, achieve about 0.8% and 0.3% increase in accuracy for networks trained on CIFAR-10 and MNIST datasets respectively. With 80% sparsity level, we can achieve 5 times reduction in model parameters with just a minimal decrease in performance. As we increase the sparsity level, the performance gradually decreases. Figure-1 demonstrates the effect of sparsity during re-training of the network. For 50% and 80% sparsity levels, the models are able to recover their weights quickly during the re-training process. While for 90% sparsity level, the models are able to sustain sparsity after about 30 epochs for CIFAR-10 trained model and 7 epochs for MNIST trained model, for 95% sparsity, the models are not able to recover their weights resulting in loss of performance.

6 Future Work

This baseline experiment validates the compression of simple CNNs using magnitude based pruning. Our motive is to extend this work to develop an end-to-end compressed CNN-LSTM based image captioning pipeline using one or more state-of-the-art compression techniques [2, 6, 7, 12]. The metric for evaluation will be BLEU scores [9]. Both of us plan to implement up to two of the above stated pipelines on the image captioning model.

7 Work Division

An extensive literature review on Network Compression and its related use cases was done. Aman recommended [2, 12] while Harshit came up with [6, 7]. Aman and Harshit developed and tested their compressed models on CIFAR-10 and MNIST data sets respectively.

References

[1] https://ai.googleblog.com/2019/10/on-device-captioning-with-live-caption.html, 2019.

[2] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[3] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[4] B. Kim, Y. Jeon, H. Park, D. Han, and Y. Baek. Design and implementation of the vehicular camera system using deep neural network compression. In *Proceedings of the 1st International Workshop on Deep Learning for Mobile Systems and Applications*, EMDL '17, page 25–30, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349628. doi: 10.1145/3089801.3089803. URL https://doi.org/10.1145/3089801.3089803.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[6] N. Lee, T. Ajanthan, and P. Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2018.

[7] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[9] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[10] Y. Shangguan, J. Li, L. Qiao, R. Alvarez, and I. McGraw. Optimizing speech recognition for the edge. *arXiv preprint arXiv:1909.12408*, 2019.

[11] J. H. Tan, C. S. Chan, and J. H. Chuah. Image captioning with sparse recurrent neural network. *arXiv preprint arXiv:1908.10797*, 2019.

[12] M. Zhu and S. Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.