PocketFlow: An Automated Framework for Compressing and Accelerating Deep Neural Networks

Jiaxiang Wu, Yao Zhang, Jinlong Hou, Wei Liu, Wenbing Huang
Tencent AI Lab
{jonathanwu, nandozhang, kinghou, topliu}@tencent.com, hwenbing@126.com

Haoli Bai
Chinese University of Hong Kong
hlbai@cse.cuhk.edu.hk

Huasong Zhong
Tsinghua University
zhonghs16@mails.tsinghua.edu.cn

Junzhou Huang
University of Texas at Arlington
jzhuang@uta.edu

Abstract

Deep neural networks are widely used in various domains, but the prohibitive computational complexity prevents their deployment on mobile devices. Numerous model compression algorithms have been proposed, however, it is often difficult and time-consuming to choose proper hyper-parameters to obtain an efficient compressed model. In this paper, we propose an automated framework for model compression and acceleration, namely PocketFlow. This is an easy-to-use toolkit that integrates a series of model compression algorithms and embeds a hyper-parameter optimization module to automatically search for the optimal combination of hyper-parameters. Furthermore, the compressed model can be converted into the TensorFlow Lite format and easily deployed on mobile devices to speed-up the inference. PocketFlow is now open-source and publicly available at https://github.com/Tencent/PocketFlow.

1 Introduction

Deep learning has been widely used in various areas, such as computer vision, speech recognition, and natural language translation. However, deep learning models are often computational expensive, which limits further applications on mobile devices with limited computational resources.

To address this dilemma between accuracy and computational complexity, numerous algorithms have been proposed to compress and accelerate deep networks with minimal performance degradation. Commonly-used approaches include low-rank decomposition [16, 15], channel pruning (*a.k.a.* structured pruning) [7, 18], weight sparsification (*a.k.a.* non-structured pruning) [17], and weight quantization [2, 3]. However, these algorithms usually involve several hyper-parameters that may have a large impact on the compressed model's performance. It can be quite difficult to efficiently choose proper hyper-parameter combinations for different models and learning tasks. Recently, some researches adopted reinforcement learning methods to automatically determine hyper-parameters for channel pruning [5] and weight sparsification [6] algorithms.

In this paper, we present an automated framework for compressing and accelerating deep neural networks, namely PocketFlow. We aim at providing an easy-to-use toolkit for developers to improve the inference efficiency with little or no performance degradation. PocketFlow has inte-

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

grated a series of model compression algorithms, including structured/non-structured pruning and uniform/non-uniform quantization. A hyper-parameter optimizer is incorporated to automatically determine hyper-parameters for model compression components. After iteratively training candidate compressed models and adjusting hyper-parameters, a final compressed model is obtained to maximally satisfy user's requirements on compression and/or acceleration ratios. The resulting model can be exported as a TensorFlow-Lite file for efficient deployment on mobile devices.

2 Framework Design

The proposed framework mainly consists of two categories of algorithm components, *i.e.* learners and hyper-parameter optimizers, as depicted in Figure 1. Given an uncompressed original model, the learner module generates a candidate compressed model using some randomly chosen hyper-parameter combination. The candidate model's accuracy and computation efficiency is then evaluated and used by hyper-parameter optimizer module as the feedback signal to determine the next hyper-parameter combination to be explored by the learner module. After a few iterations, the best one of all the candidate models is output as the final compressed model.

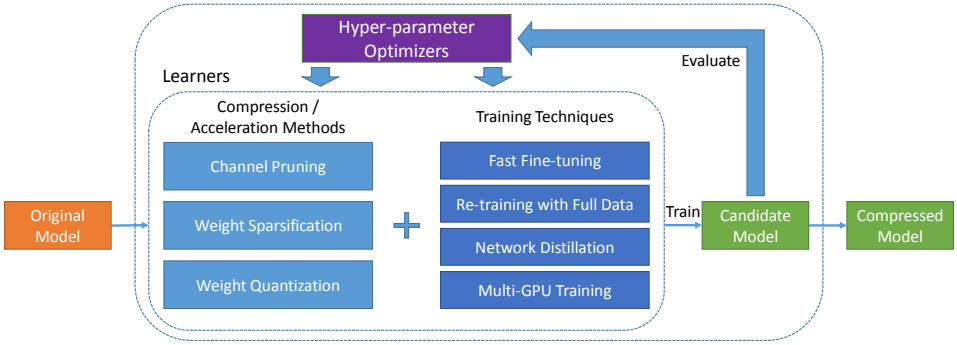


Figure 1: The overall framework of PocketFlow.

2.1 Learners

A learner refers to some model compression algorithm augmented with several training techniques as shown in Figure 1. Below is a list of model compression algorithms supported in PocketFlow:

Table 1: List of learners and corresponding model compression algorithms.

Name	Description
ChannelPrunedLearner	channel pruning with LASSO-based channel selection [7]
DisChnPrunedLearner	discrimination-aware channel pruning [18]
WeightSparseLearner	weight sparsification with dynamic pruning ratio schedule [17]
UniformQuantLearner	weight quantization with uniform reconstruction levels [9]
NonUniformQuantLearner	weight quantization with non-uniform reconstruction levels [3]

All the above model compression algorithms can trained with fast fine-tuning, which is to directly derive a compressed model from the original one by applying either pruning masks or quantization functions. The resulting model can be fine-tuned with a few iterations to recover the accuracy to some extent. Alternatively, the compressed model can be re-trained with the full training data, which leads to higher accuracy but usually takes longer to complete.

To further reduce the compressed model's performance degradation, we adopt network distillation to augment its training process with an extra loss term, using the original uncompressed model's outputs as soft labels. Additionally, multi-GPU distributed training is enabled for all learners to speed-up the time-consuming training process.

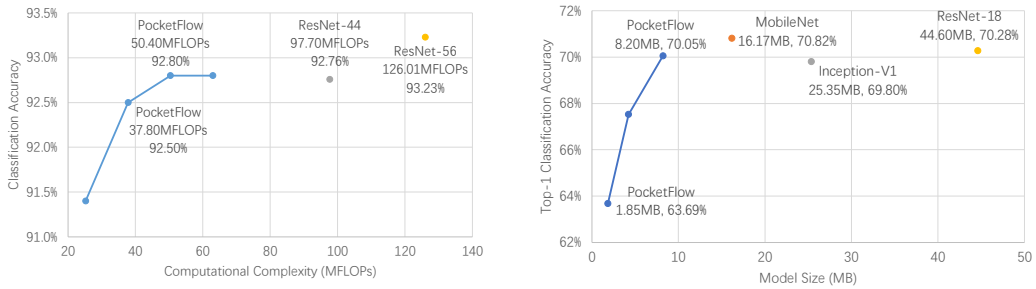
2.2 Hyper-parameter Optimizers

For model compression algorithms, there are several hyper-parameters that may have a large impact on the final compressed model's performance. It can be quite difficult to manually determine proper values for these hyper-parameters, especially for developers that are not very familiar with algorithm details. Therefore, we introduce the hyper-parameter optimizer module to iteratively search for the optimal hyper-parameter setting.

In PocketFlow, we provide several implementations of hyper-parameter optimizer, based on models including Gaussian Processes (GP) [12], Tree-structured Parzen Estimator (TPE) [1], and Deterministic Deep Policy Gradients (DDPG) [11]. The hyper-parameter setting is optimized through an iterative process. In each iteration, the hyper-parameter optimizer chooses a combination of hyper-parameter values, and the learner generates a candidate model with fast fast-tuning. The candidate model is evaluated to calculate the reward of the current hyper-parameter setting. After that, the hyper-parameter optimizer updates its model to improve its estimation on the hyper-parameter space. Finally, when the best candidate model (and corresponding hyper-parameter setting) is selected after some iterations, this model can be re-trained with full data to further reduce the performance loss.

3 Experimental Results

For empirical evaluation, we adopt PocketFlow to compress and accelerate classification models on the CIFAR-10 [10] and ILSVRC-12 [13] data sets. In Figure 2a, we use *ChannelPrunedLearner* to speed-up ResNet-56 [4] to reduce its computational complexity. We observe that the accuracy loss under 2.5× acceleration is 0.4% and under 3.3× acceleration is 0.7%, and compressed models are more efficient and effective that the shallower ResNet-44 model. In Figure 2b, we use *WeightSparseLearner* to compress MobileNet [8] to reduce its model size. We discover that the compressed model achieves similar classification accuracy with much smaller model size than MobileNet, Inception-v1 [14], and ResNet-18 models.



(a) CIFAR-10 (b) ILSVRC-12

Figure 2: Comparison on the computational complexity, model size, and classification accuracy of various models on CIFAR-10 (left) and ILSVRC-12 (right) data sets.

The compressed models generated by PocketFlow can be exported as TensorFlow Lite models and directly deployed on mobile devices using the mobile-optimized interpreter. In Table 2, we compare the classification accuracy, model size, and inference latency[1] of original and compressed models. With *ChannelPrunedLearner*, the compressed model achieves 1.53× speed-up with 2.0% loss in the top-5 classification accuracy. With *UniformQuantLearner*, we achieve 2.46× speed-up after applying 8-bit quantization on the MobileNet model, while the top-5 accuracy loss is merely 0.6%.

4 Conclusion

In this paper, we present the PocketFlow framework to boost the deployment of deep learning models on mobile devices. Various model compression algorithms are integrated and hyper-parameter optimizers are introduced into the training process to automatically generate highly-accurate compressed models with minimal human effort.

[1]The inference latency is measured as the average elapsed time of 6 executions.

Table 2: Comparison on the classification accuracy, model size, and inference latency of compressed ResNet-18 and MobileNet models.

Model	Learner	Top-1 Acc.	Top-5 Acc.	Size (MB)	Latency (ms)
ResNet-18	-	70.3%	89.4%	44.63	334.83
	ChannelPrunedLearner	67.1%	87.4%	**25.68**	**219.83**
MobileNet	-	70.9%	89.6%	16.13	157.67
	UniformQuantLearner	70.0%	89.0%	**4.08**	**64.17**

References

[1] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning (ICML)*, pages 115–123, Jun 2013.

[2] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Low precision arithmetic for deep learning. *CoRR*, abs/1412.7024, 2014.

[3] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision (ECCV)*, pages 630–645, 2016.

[5] Yihui He and Song Han. ADC: Automated deep compression and acceleration with reinforcement learning. *CoRR*, abs/1802.03494, 2018.

[6] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. AMC: Automl for model compression and acceleration on mobile devices. In *European Conference on Computer Vision (ECCV)*, pages 784–800, Sept 2018.

[7] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1398–1406, Oct 2017.

[8] 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. *CoRR*, abs/1704.04861, 2017.

[9] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713, June 2018.

[10] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Department of Computer Science, University of Toronto, 2009.

[11] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

[12] J Močkus. On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404, 1975.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015.

[15] Cheng Tai, Tong Xiao, Xiaogang Wang, and Weinan E. Convolutional neural networks with low-rank regularization. In *International Conference on Learning Representations (ICLR)*, 2016.

[16] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(10):1943–1955, Oct 2016.

[17] Michael Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. *CoRR*, abs/1710.01878, 2017.

[18] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Jiezhang Cao, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 2018.