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
hlabai@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 hyperparameters 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-
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.

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>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChannelPrunedLearner</td>
<td>channel pruning with LASSO-based channel selection [7]</td>
</tr>
<tr>
<td>DisChnPrunedLearner</td>
<td>discrimination-aware channel pruning [18]</td>
</tr>
<tr>
<td>WeightSparseLearner</td>
<td>weight sparsification with dynamic pruning ratio schedule [17]</td>
</tr>
<tr>
<td>UniformQuantLearner</td>
<td>weight quantization with uniform reconstruction levels [9]</td>
</tr>
<tr>
<td>NonUniformQuantLearner</td>
<td>weight quantization with non-uniform reconstruction levels [3]</td>
</tr>
</tbody>
</table>

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 WeightSparse-Learner 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.

![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.](image)

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 of original and compressed models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Computational Complexity (MFLOPs)</th>
<th>Top-5 Classification Accuracy</th>
<th>Model Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChannelPrunedLearner</td>
<td>50.40 MFLOps</td>
<td>92.80%</td>
<td>1.85 MB, 63.69%</td>
</tr>
<tr>
<td>ResNet-44</td>
<td>97.00 MFLOps</td>
<td>92.76%</td>
<td>8.20 MB, 70.05%</td>
</tr>
<tr>
<td>ResNet-56</td>
<td>126.01 MFLOps</td>
<td>93.23%</td>
<td>16.17 MB, 70.82%</td>
</tr>
<tr>
<td>PocketFlow</td>
<td>37.80 MFLOps</td>
<td>92.50%</td>
<td>25.35 MB, 69.80%</td>
</tr>
<tr>
<td>MobileNet</td>
<td>44.60 MFLOps</td>
<td>92.80%</td>
<td>44.60 MB, 70.28%</td>
</tr>
<tr>
<td>Inception-V1</td>
<td>44.60 MFLOps</td>
<td>93.00%</td>
<td>25.35 MB, 69.80%</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>16.17 MB, 70.82%</td>
<td>92.80%</td>
<td>8.20 MB, 70.05%</td>
</tr>
</tbody>
</table>

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

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.

---

1The 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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Learner</th>
<th>Top-1 Acc.</th>
<th>Top-5 Acc.</th>
<th>Size (MB)</th>
<th>Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18</td>
<td>-</td>
<td>70.3%</td>
<td>89.4%</td>
<td>44.63</td>
<td>334.83</td>
</tr>
<tr>
<td></td>
<td>ChannelPrunedLearner</td>
<td>67.1%</td>
<td>87.4%</td>
<td>25.68</td>
<td>219.83</td>
</tr>
<tr>
<td>MobileNet</td>
<td>-</td>
<td>70.9%</td>
<td>89.6%</td>
<td>16.13</td>
<td>157.67</td>
</tr>
<tr>
<td></td>
<td>UniformQuantLearner</td>
<td>70.0%</td>
<td>89.0%</td>
<td>4.08</td>
<td>64.17</td>
</tr>
</tbody>
</table>

References


