Understanding Deep Learning Model Compression on Heterogeneous Mobile Platforms

QIN QING, Northwest University, China
JIE REN, Shaanxi Normal University, China
LING GAO, Northwest University, China
JIANBIN FANG, National University of Defense Technology, China
YANSONG FENG, Peking University, China
ZHENG WANG, Lancaster University, UK

CCS Concepts: • **Computer systems organization** → **Embedded software**; • **Computing methodologies** → *Parallel computing methodologies*;

Additional Key Words and Phrases: Deep learning, Adaptive computing, Embedded systems

ACM Reference Format:
Qin Qing, Jie Ren, Ling Gao, Jianbin Fang, Yansong Feng, and Zheng Wang. 2018. Understanding Deep Learning Model Compression on Heterogeneous Mobile Platforms. 1, 1 (May 2018), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Deep Learning has led an AI renaissance of sorts in recent years. Image is one of its most prominent success area. Results of the ImageNet Challenge points to a dramatic improvement in object detection, localization and classification. So far, the accuracy of the image classification model has reached more than 95% (top5)FIX:[]. While, the excellent performance of these deep networks depends on the powerful computing devices (i.e cloud platform) to deal with a huge number of operations, up to $O(10^9)$ operationsFIX:cite for one inference It is expected that the size of the deep neural networks (i.e., number of weights) will be larger for the more difficult task than the simpler task and thus require more energy to deal with the growing number of parameters and operations.

The current neural network based mobile applications all rely on the cloud server by putting the load on the server and waiting for the inference results back to the mobile, such as FIX:[]applications name using the cloud-based APIs to recognize the xxx lables from an image, which leverages the power of cloud platform's machine learning technology to give a high level of accuracy. Considering that the network capability is unstable, these mobile services can not guarantee the low latency or even can not be used. Therefore, these deep neural networks need to be migrated to mobile platforms to reduce the inference time and increase the reliability under the unstable network. While the inferring of an advanced deep learning model usually requires a large amount of computing

Authors' addresses: Qin Qing, Northwest University, China, qq@stumail.nwu.edu.cn; Jie Ren, Shaanxi Normal University, China, renjie@snnu.edu.cn; Ling Gao, Northwest University, China, gl@nwu.edu.cn; Jianbin Fang, National University of Defense Technology, China, j.fang@nudt.edu.cn; Yansong Feng, Peking University, China, fengyansong@pku.edu.cn; Zheng Wang, Lancaster University, UK, z.wang@lancaster.ac.uk.

XXXX-XXXX/2018/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

resources, memory, and computing power, which becomes a huge obstacles for mobile devices and IoT devices.

Model compression techniques make it possible to run the inferences on mobile devices by reducing the model size and inference time. In recent years, this field has achieved great development and there are three mainstream compression methods, parameter pruningFIX:[], low rank decompositionFIX:[] and knowledge purificationFIX:[]. However, these methods have not been verfied on any mobile or embedded system. It is a necessary step to further verify the feasibility of the model's deployment on the mobile or embedded system.

In this paper, we compress 10 deep models, which belong to four typical networks, `mobilenet`, `vggnet`, `inception net`, `resnet`, by two typical method, prune and quantization. These models are meant to represent the state of the art in deep learning. Most are taken directly from top-tier research venues and have either set new accuracy records on competitive datasets. We focus on four areas: identifying the types of operations which dominate execution time, compressing the different kinds of models, retraining the compressed model to regain the accuracy, comparing the model size and inference time before and after the compression. We evaluate the compressed models on a representative heterogeneous mobile platform NVIDIA Jetson TX2 and our results show that significant imporvement for compressed model size and inference time can be achieved by making effective use of the heterogeneous mobile architecture.

In summary, our contributions are as follows:

- We test 10 most widely used deep learning models in heterogeneous mobile platform. Testing and analyzing inference time, model size, and breakdown of execution time of operation type for each model. We are the first to do such work, which is the essential step in the migration of the deep learning model in the mobile platform.
- We find that the inference time and compression radio varies for different models after using the quantification and prune.
- The use of quantization can achieve an average reduction of FIX:xxx% for model size, while the inference time rise about FIX:xxx%.

2 BACKGROUND AND MOTIVATION

2.1 Workloads

VGGNet:VGGNet is a deep convolutional neural network developed by the Computer Vision Group at Oxford University and researchers at Google DeepMind. VGGNet explores the relationship between the depth of a convolutional neural network and its performance. By repeatedly stacking 3*3 small convolutional kernels and 2*2 largest pooled layers, VGGNet successfully builds a 16-19 layer deep convolution.[1] Neural Networks. Compared with the previous network structure, VGGNet significantly reduced the error rate, and achieved the second place of the ILSVRC 2014 competition classification project and the first place of the positioning project. At the same time, VGGNet's scalability is very strong, and the generalization performance of migrating to other image data is very good. VGGNet's structure is very simple, the entire network uses the same size of the convolution kernel size and maximum pool size.So far, VGGNet is still often used to extract image features.

Inception Net: Google Inception Net won first place in the ILSVRC 2014 competition. The Inception Net in the game is usually called Inception V1. Its greatest feature is that it controls the calculations and parameters while achieving very good classification performance. Inception V1 proposes Inception Module to increase the utilization of parameters. Although his depth reaches 22 levels, the parameter amount is much lower than ALexNet and VGGNet. Subsequently, Inception V2, Inception V3 and Inception V4 emerged successively in 2015 and 2016. Inception V2 refers to VGGNet and uses two 3*3 convolutions instead of a 5*5 convolution to reduce the amount of parameters. At the same time, the famous method of Batch Normalization was proposed to speed up the training of the network, and the accuracy of the classification after convergence can also be improved. Inception V3 splits the larger two-dimensional convolutional network into a smaller one-dimensional convolutional network. On the one hand, it saves a lot of parameters, speeds up calculations, and reduces over-fitting. It also adds a layer of non-linear expansion model. Ability to express so that the network can handle more and more abundant spatial features and increase the diversity of features. On the other hand, Inception V3 optimizes the structure of the Inception Module. Subsequent Inception V4 further integrated Resnet on the basis of Inception V3 to further improve accuracy.

Resnet: Microsoft Researcher Kaiming He and others proposed Resnet and won the championship in ILSVRC 2015. Resnet's TOP5 error rate was only 3.75%.Resnet uses a connection method called "shortcut connection" to reduce network parameters, and stack the input and output of the block. This simple addition will not add extra parameters and calculations to the network, but it can greatly increase the training speed of the model, improve the training effect, and when the number of layers of the model deepens, this simple structure can solve the problem of degradation well. There are two different learning units in Resnet's structure: two levels of residual learning units and three levels of residual learning units. The two-level residual learning unit contains two 3*3 convolutions with the same number of output channels. The three-level residual network uses a 1*1 convolution before and after the middle 3*3 convolution. The 50,101,152-layer networks we use are mainly three-tier residual structures.

MobileNet: MobileNet was proposed by Horward et al. in 2017. Its architecture consists of a standard convolution layer acting on the input image, a deep separable convolutional stack, and a final average pool and fully connected layer. Mobilenet uses a deeply separable convolution to build a lightweight, deep neural network that can integrate standard volumes into a deep convolution and a dot convolution (1 ÃŮ 1 convolution kernel). Depth convolution applies each convolution kernel to each channel, and 1 ÃŮ 1 convolution is used to combine the output of channel convolutions. This decomposition can effectively reduce the computation and reduce the model size. MobileNet has a clear advantage in terms of computational volume and model size.

2.2 Motivation

3 EXPERIMENTAL SETUP

3.1 Platform and Models

Hardware. Our experimental platform is the NVIDIA Jetson TX2 embedded deep learning platform. The system has a 64 bit dual-core Denver2 and a 64 bit quad-core ARM Cortex-A57 running at 2.0 Ghz, and a 256-core NVIDIA Pascal GPU running at 1.3 Ghz. The board has 8 GB of LPDDR4 RAM and 96 GB of storage (32 GB eMMC plus 64 GB SD card).

System Software. Our evaluation platform runs Ubuntu 16.04 with Linux kernel v4.4.15. We use Tensorflow v.1.0.1, cuDNN (v6.0) and CUDA (v8.0.64).

Deep Learning Models. We consider FIX:14 pre-trained CNN models for image recognition from the TensorFlow-Slim library [?]. The models are built using TensorFlow and trained on the ImageNet ILSVRC 2012 training set.

3.2 Evaluation Methodology

Evaluation Metrics We consider the following metrics:

- ***Inference time*** *(lower is better)*. Wall clock time between a model taking in an input and producing an output, excluding the model load time.
- ***Energy consumption*** *(lower is better)*. The energy used by a model for inference. We deduct the static power used by the hardware when the system is idle.
- ***Accuracy*** *(higher is better)*. The ratio of correctly labeled images to the total number of testing images.
- ***Precision*** *(higher is better)*. The ratio of a correctly predicted images to the total number of images that are predicted to have a specific object. This metric answers e.g., "*Of all the images that are labeled to have a cat, how many actually have a cat?*".
- ***Recall*** *(higher is better)*. The ratio of correctly predicted images to the total number of test images that belong to an object class. This metric answers e.g., "*Of all the test images that have a cat, how many are actually labeled to have a cat?*".
- ***F1 score*** *(higher is better)*. The weighted average of Precision and Recall, calculated as $2 \times \frac{Recall \times Precision}{Recall + Precision}$. It is useful when the test datasets have an uneven distribution of object classes.

Performance Report. To collect inference time and energy consumption, we run each model on each input repeatedly until the 95% confidence bound per model per input is smaller than 5%. In the experiments, we exclude the loading time of the CNN models as they only need to be loaded once in practice. To measure energy consumption, we developed a lightweight runtime to take readings from the on-board energy sensors at a frequency of 1,000 samples per second. We then matched the energy readings against the time stamps of model execution to calculate the energy consumption.

4 EXPERIMENTAL RESULTS

4.1 Overall Results

4.2 Compare to Oracle

4.3 Analysis

5 RELATED WORK

Our work lies at the intersection of multiple research areas: web browsing optimization, task scheduling, energy optimization and predictive modeling. There is no existing work that is similar to ours, in respect to optimizing web workloads across multiple optimization objectives on heterogeneous mobile platforms.

6 CONCLUSIONS

This paper has presented an automatic approach to optimize the mobile web

REFERENCES

[1] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).