Understanding Deep Learning Model Compression on Heterogeneous Mobile Platforms

Qin Qing†, Jie Ren‡, Ling Gao†, Hai Wang†, Zheng Wang*
†Northwest University, China, ‡Shaanxi Normal University, China, *Lancaster University, UK
† qq@stumail.nwu.edu.cn, ‡renjie@snnu.edu.cn, †{gl, hwang}@nwu.edu.cn, *z.wang@lancaster.ac.uk

***Abstract*—**

***Keywords*-workload characterisation**

I. Introduction

In recent years, CNN has become the main technical means of computer vision tasks, and it has been splendid in the direction of image classification, target detection, depth estimation, and semantic segmentation. In particular, since AlexNet won the ILSVRC 2012 ImageNet Image Classification Competition in one fell swoop, the deep neural network boom has swept the entire field of computer vision. Depth model swiftness replaces traditional manual design features and classifiers. Various depth models not only provide an end-to-end processing method, but also significantly refresh the accuracy of each task, and even beyond the accuracy of human knowledge. So far, the accuracy of the image classification model has reached more than 95% (top5).

However, behind the excellent performance of the depth model is a powerful computing demand: while constantly refreshing the limits of the accuracy of the task, its depth and size are also exponentially increasing. In order to preserve user privacy and reduce user-aware query time, these deep neural networks need to be migrated to various new platform applications, including mobile platforms, autonomous systems, and smart devices. Therefore, one of the following problems is that such a huge model can only be used on a limited platform, it cannot be ported to mobile terminals and embedded chips at all, and the high bandwidth consumption is also daunting to many users. On the other hand, large-scale models also pose enormous challenges to equipment power consumption and operating speed. Therefore, the depth model currently faces problems that are difficult to apply in the context of terminal deployment and low-latency requirements.

Therefore, model compression is an indispensable step for the depth model to be truly applied to the mobile terminal. In recent years, this field has achieved great development. The current mainstream compression methods include parameter pruning and sharing, low rank decomposition, and knowledge purification.In summary, the theoretical research on the depth model compression has been established at home and abroad, but compared to other tasks (such as computer vision, speech recognition, etc.), this field is still in an enlightening phase of development, especially in the depth learning model. The real mobile deployment is still very few. In the past two years, related research work has been gradually carried out. To successfully deploy mobile terminals, it is necessary to analyze the performance of different depth models on the one hand. On the other hand, the compression of models is also a necessary research point. Model compression is mainly about reducing the model size and reasoning time for experiments. By quantifying the shared weights, the size of the model can be compressed very well. By pruning the weights, the reasoning time of the model can be further accelerated. However, these methods mainly stay on the experimental conclusion and are not really on the mobile or embedded system. Deploying the model and putting it into practice, performance testing and compression experiments after actually deploying the model on an embedded platform is a necessary step to further verify the feasibility of the model's deployment on the mobile or embedded system, and is also the focus of this article.

In order to better deploy the model to a mobile or embedded system, This paper makes the following contributions:(1) Performance testing and analysis of the most widely used deep learning models in embedded systems, including inference time, model size, and time spent on each operand, and further analysis of these models in embedded systems Performance performance, related research work has not yet appeared, which is the first step in the migration of the deep learning model in the mobile platform. (2) Study the model compression technology based on deep learning, realize the model compression of several classical networks, compare the load characteristics of the model under different compression methods, mainly including the reasoning time change before and after the model compression, the model size change and the accuracy rate The loss, etc., resulting in different compression methods for different models of different applications of the effect of analysis of different models suitable for the compression method, making the use of different models in the mobile terminal is more likely.

II. Background and Motivation

A. 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.[**?**] 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 1 convolution kernel). Depth convolution applies each convolution kernel to each channel, and 1 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.

B. Motivation

The ImageNetFIX:[] competitions has led to great improvement in the accuracy of image classification, which has improved from FIX:[] to FIX:[], by increasing the depth and width of the neural networks such as FIX:model contains contains FIX:number layers and FIX:number million of parameters. However, the excellent performance of these deep networks depends on the powerful computing devices, which also accompany with energy-extensive consumption, to deal with a huge number of operations (up to $O(10^9)$) for one inference, such as memory access and arithmetic. It is expected that the size of the CCNs (i.e., number of weights) will be larger for the more difficult task than the simpler task and thus require more energy.

III. Experimental Setup

IV. Experimental Results

A. Overall Results

B. Compare to Oracle

C. Analysis

V. 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.

VI. Conclusions

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