Frequency Regularization: Efficient Neural Network Compression for Mobile Device Deployment

Wenhao You(wyou1@ualberta.ca), Leo Chang(chewei@ualberta.ca)

Abstract

In general, neural networks require high-specification hardware because of their complexity. However, in daily life, people have the requirements for running neural network applications on their own mobile devices but have limited hardware. Even the researchers have to use the computer in their lab instead of running the pre-trained models by using mobile devices. In this proposal, we will focus on how to implement a simple and creative compression algorithm called Frequency Regularization (FR) [1], which is to compress a basic neural network on Android mobile devices. We also discuss another three existing methods to achieve the goal: upgrade hardware [4], implement NestDNN [2], and implement "One-shot Whole Network Compression" [3]. The reason we did not choose to upgrade mobile hardware is that it is hard to upgrade hardware for only two of us. Moreover, the other two compression algorithms are too hard to implement and not as efficient as FR. That is why we chose FR to do further research.

Introduction

Neural networks have been implemented in many applications and performed great results. However, these neural networks are usually implemented on high-specification hardware such as computers since there are many parameters in different neural networks. The more parameters in the neural networks, the more storage and internet transmission rates are occupied. We try to look for a method that can make the neural networks run on a limited hardware device, such as a mobile device. Although mobile devices have more enhanced hardware nowadays, because of their limited size, the efficacy is still not as powerful as the computers. There are several benefits of running the neural networks on limited hardware devices, specifically, mobile devices. The most important parts are the privacy and the data security. Processing data on-device ensures that personal information does not need to be uploaded or transmitted to the cloud servers, which enhances privacy.

Running neural networks on mobile devices can also bring other advantages. Obviously, it can reduce our dependence on the internet connection to some extent if we can implement it successfully. Moreover, for some applications that need real-time feedback, such as object recognition and image segmentation, neural networks on mobile devices can also make the process faster to improve user experience. The most important part of our idea is that we can keep the data processing and inference in the internal system of mobile devices rather than sending it to a remote server, which can increase the processing speed.

The new method, Frequency regularization, (FR) [1] is based on the frequency domain to compress the neural network. It is divided into two steps: dynamic tail-truncation and inverse discrete cosine transform (IDCT). The main idea of this method is that some of the neural network parameters may be redundant in their representation. The algorithm focuses on using the frequency domain transform for network regularization to restrict information redundancy during the training process and introduce FR as shown in Fig. 1. Therefore, the redundant part can be removed without any side effect on the performance of the network.

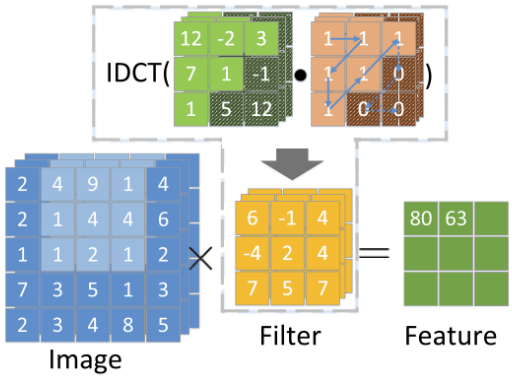


Figure 1: Illustration of the proposed frequency regularization. The tail elements of tensors in the frequency domain are truncated first, then input into the inverse of the discrete cosine transform to reconstruct the spatial tensor for learning features [1].

To implement neural networks on mobile devices, firstly, we need to discuss the limitations and certain characteristics of the mobile devices. There are many popular libraries for deep learning, such as PyTorch and TensorFlow. These two libraries are not complete and perfect to implement on mobile devices directly. Thus, we need to edit some configurations to optimize the mobile devices, in order to deploy on mobile devices. By using this implementation, we can easily convert our standard neural network models into a lighter version that fits the mobile devices' hardware better. Moreover, beyond our idea given above, we can also consider improving the hardware parts such as creating special neural network processors or GPUs for mobile devices.

In conclusion, in this research, we plan to explore a simple, efficient, and creative method to run a large neural network on mobile devices based on the Frequency regularization algorithm. Our purpose is not only to accomplish this implementation, we want to ensure the performance is similar to the result running on high-specification hardware such as computers. We hope to provide a fast and secure neural network application experience on mobile devices.

Brief Summary of Existing Work

We cover the methods for implementing neural networks on the mobile over four sections. In Section 0.1, upgrading hardware on mobile devices and which parts are introduced. At the algorithm level, there is an algorithm called NestDNN [2] based on the dynamics or runtime resources to enable resource-aware multi-tenant on-device deep learning for mobile vision systems as shown in Section 0.2. Moreover, there is an algorithm called "one-shot whole network compression" [3] to compress convolutional neural networks (CNNs) for mobile deployment as shown in Section 0.3. There is also an algorithm named Frequency Regularization that helps compress the neural network by using the idea of reducing the redundancy of information during the training process. The general steps are as shown and introduced in Section 0.4.

0.1 Upgrade Hardware

Hardware accelerators such as the digital signal processors offer solutions to overcome these challenges. Many manufacturers have already produced better hardware. For example, Apple presented the Metal Performance Shaders with the support of CNNs accelerated by GPU. This is a solution built on top of the Metal API and allows custom operations. Since there is an experiment [4] that confirmed the feasibility of Big Data analysis on modern mobile devices. Moreover, the hardware on Android devices is upgraded by four main companies, such as Qualcomm, HiSilicon, MediaTek, and Samsung [5], which can work with neural networks as well.

0.2 Algorithm NestDNN

NestDNN is a framework that takes the dynamics of runtime resources into account to enable resource-aware multi-tenant on-device deep learning for mobile vision systems. NestDNN enables each deep learning model to offer flexible resource-accuracy trade-offs [2]. Fig. 2 illustrates the architecture of NestDNN in detail, which is split into an offline stage and an online stage. The offline stage consists of three phases: model pruning, model recovery, and model profiling [2]. There are five models involved in NestDNN as shown below:

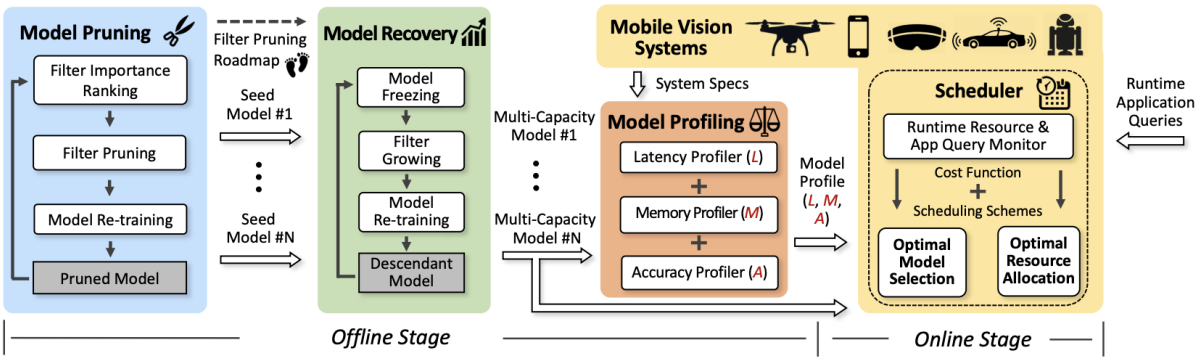


Figure 2: NestDNN architecture [2].

- **Vanilla Model:** Off-the-shelf deep learning model (e.g., ResNet, VGG) trained on a given dataset (e.g., ImageNet, CIFAR-10).
- **Pruned Model:** Intermediate result obtained in model pruning stage. Certain useless weights of the chosen neural network are removed, achieving a more efficient model.
- **Seed Model:** It is the smallest pruned model generated in model pruning which meets the minimum accuracy goal set by the user. It is also the starting point of model recovery stage.
- **Descendant Model:** A model grown upon the seed model in model recovery stage. It has a unique resource-accuracy trade-off.
- **Multi-Capacity Model:** The final descendant model that has the capacities of all the previously generated descendant models.

0.3 Algorithm One-shot Whole Network Compression

”One-shot Whole Network Compression” is a simple way to compress CNNs to make them more suitable for fast and low-power applications on the mobile. This algorithm can be divided into three steps: rank selection, tensor decomposition, and fine-tuning. Each step can be implemented easily by using public tools such as VBMF for rank selection, Tucker decomposition for tensor decomposition, and Caffe for fine-tuning [3].

0.4 Frequency Regularization

The frequency regularization algorithm is divided into two steps, the dynamic tail-truncation and inverse discrete cosine transform (IDCT). The dynamic tail-truncation involves retaining parameters in the frequency domain when training the neural networks. The segments that depict high-frequency zigzag information are truncated. This procedure is executed using a dot product with a zigzag mask matrix to maintain differentiability. [1]. After the procedure, the algorithm utilizes IDCT to rebuild the spatial tensors, which are subsequently employed as standard learning kernels in the networks.

What you Plan To Do

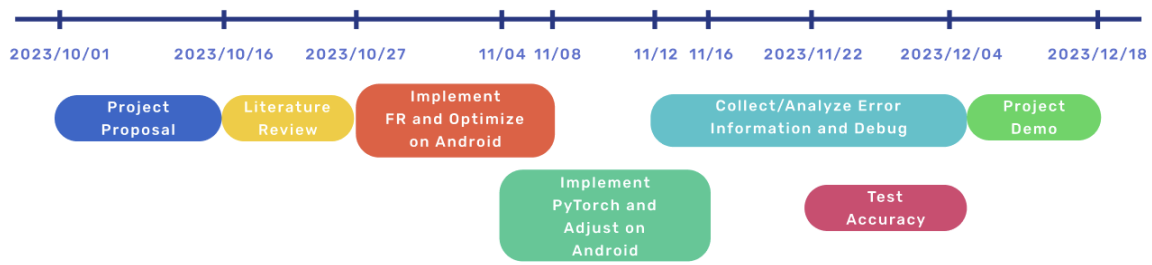
There are five sub-topics for implementing Frequency Regularization on Android devices. The first one ”Implement Frequency Regularization” was done in Basu’s lab at the University of Alberta (2023). The remaining topics are expected to be finished before the course ends.

- **Implement Frequency Regularization:** Implement the Frequency Regularization in order to compress the redundant information on the neural network. Make sure that the accuracy and velocity are acceptable for the neural network.
- **Configure Emulator:** In this paper, we choose Android Studio as our emulator. Install it successfully on our own local machines.
- **Implement PyTorch Library:** Implement PyTorch Android library to the emulator. Collect and analyze the error information if we get it.
- **Adjust Dependencies:** Potential errors may probably caused by Android PyTorch Library. Since the library on Android lacks improvement not as much as laptop.
- **Adjust Frequency Regularization:** We can also try to change something on the algorithm to let the new one fit the Android system.

How You Plan to Implement Your Ideas

For the class 414, we will only focus on the Frequency Regularization (FR). From our own perspectives, upgrading mobile hardware is impossible for two of us student researchers. And the existing algorithms NestDNN and "One-show Whole Network Compression" are not as accurate as FR. In order to do further research on FR, our basic and important idea is to analyze the existing paper in Basu's lab [1]. After that, we also need to explore the basic structure of the PyTorch Mobile [6].

Timeline for Whole Project (Class 414)



Short Description of 5 LABS

In this project, we plan to divide the project into five labs (milestones), which can help us to double-check our progress and adjust it timely. Here are the short descriptions of 5 labs:

- Deploy the Frequency Regularization algorithm with the UNet (one basic Neural Network) on the computer and adjust some parameters for the Android emulator.
- Attempt to implement PyTorch library on the Android emulator and summarize the error messages in order to further analyze.
- Analyze the error messages and explore the methods to change the PyTorch library on Android.
- Attempt to change something on the FR algorithm to make it match the current Android system.

- Test all functions and neural networks to check the effectiveness and accuracy of the Android system after importing the compressed neural network. We will compare the results between the computer and mobile devices.

References

[1] C. Zhao, G. Dong, S. Zhang, Z. Tan, and A. Basu. Frequency regularization: Reducing information redundancy in convolutional neural networks. *IEEE Access*, September 2023. Department of Computing Science, University of Alberta, Edmonton, AB.

[2] Biyi Fang, Xiao Zeng, and Mi Zhang. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. pages 115–127, 10 2018.

[3] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. 11 2015.

[4] Anton Akusok, Leonardo Espinosa Leal, Kaj-Mikael Björk, and Amaury Lendasse. High-performance elm for memory constrained edge computing devices with metal performance shaders. In Jiuwen Cao, Chi Man Vong, Yoan Miche, and Amaury Lendasse, editors, *Proceedings of ELM2019*, pages 79–88, Cham, 2021. Springer International Publishing.

[5] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley, and Luc Van Gool. Ai benchmark: Running deep neural networks on android smartphones. In Laura Leal-Taixé and Stefan Roth, editors, *Computer Vision – ECCV 2018 Workshops*, pages 288–314, Cham, 2019. Springer International Publishing.

[6] Pytorch mobile, 2023.