# xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx

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

## Abstract

xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx

## Introduction

Neural networks have been implemented on many applications and performed a great results. However, these neural networks are mostly implemented on a high-specification hardware such as computers since there are many parameters in different neural networks. The more parameters in the neural networks, the more storages and internet transmition rates are occupied. In this research, we try to look for a method that can make the neural networks run on a limited hardware devices, such as mobile devices. Although mobile devices have more enhanced hardware nowadays, but because of its 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 part is the privacy and the data security. Processing data on-device ensures that personal information does not need to upload or transmit to the cloud servers, which enhanced privacy.

Running neural networks on mobile devices can bring lots of advantages. Mobile devices, such as smartphones and tablets, play a significant role in our daily lives. If we are able to implement the neural network models directly on the mobile devices, it can reduce our dependence on the internet connection to some extent. Moreover, for some applications which need real-time feedback, such as object recognition and image segmentation, neural networks on mobile devices can also make the process faster and improve the user experience. The most important part of modern life is that we can keep the data processing and inference on our own interal system of the mobile devices rather than sending it to a remote server, which can enhance privacy and security to a great extent.

The new method called Frequency regularization (FR) [1] is based on the frequency domain to compress the neural network. In this way, it can be 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 netwroks parameters may be redundant in their representation. The algorithm focus 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.

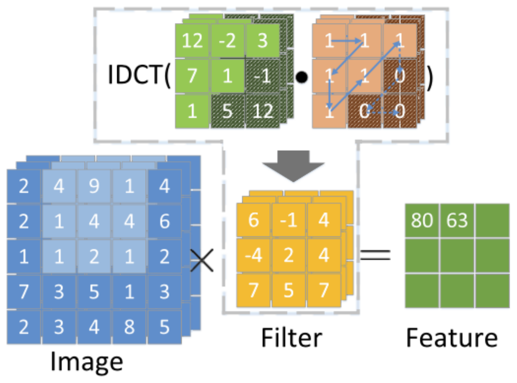

Figure 1: Illustration of the proposed frequency regularization. The tail elements of tensors in the frequency domain are truncated first. then input into the inverse of the discrete cosine transform to reconstruct the spatial tensor for learning features [1].

In order to implement the neural networks on mobile devices, we have to think about the limitations of hardware and its characteristics. Many popular deep learning libraries, such as PyTorch and TensorFlow, are not complete and perfect to implement directly on the mobile devices. Thus, we need to edit some implementation in order to optimize specifically for the mobile devices such as TensorFlow Lite. By using the method, we can convert our standard neural network models into a lite format that can fit the mobile devices hardware better. Moreover, beyond this idea, we can also consider improving the hardware parts such as creating special neural network processors for the mobile. In this paper, we only explore the method in software instead of hardware.

Our paper makes discussion explores a simple, efficient, and creative method to transfer the large neural network into mobile devices, which is based on Frequency regularization. Our purpose is not only to accomplish this implementation but also to make sure the performance which should be the same as running on high-specification hardware such as computers. To sum up, we hope to provide a fast and secure neural network application experience for anyone who uses mobile devices.

## Brief Summary of Existing Work

In this section, we cover the methods about implementing neural network on the mobile over three sections. In Section 0.1, upgrading hardware on mobile devices and which part are inrtoducted. 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.

### 0.1 Upgrade Hardware

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

### 0.2 Algorithm NestDNN

NestDNN is a framework that takes the dynamics of runtime resources into account to enable resourceaware 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 detailedly, 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]. In Table 0.2, it illustrates all the terminologies involved in NestDNN.

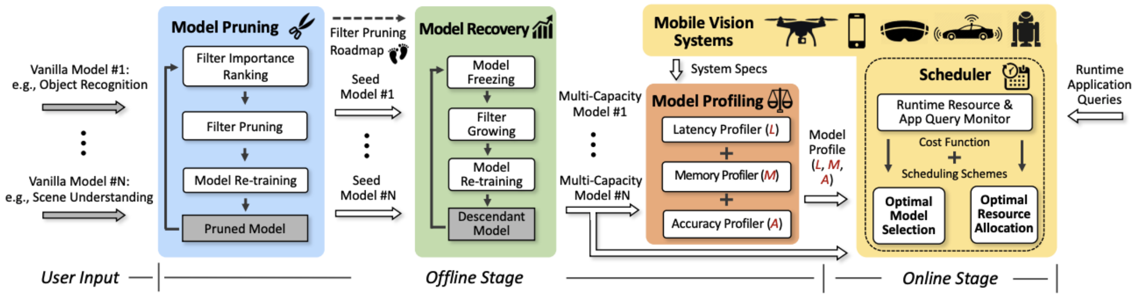

Figure 2: NestDNN architecture [2].

| Terminology | Explanation |
|---|---|
| **Vanilla Model** | Off-the-shelf deep learning model (e.g., ResNet) trained on a given dataset (e.g., ImageNet). |
| **Pruned Model** | Intermediate result obtained in model pruning stage. |
| **Seed Model** | 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. |

Table 1: Terminologies involved in NestDNN [2].

### 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].

## What you Plan To Do

There are four sub-topics for implementing Frequency Regularization on the Android devices. The first one ”Implement Frequency Regularization” have done in Basu’s lab in University of Alberta (2023). The remaining topics will be finished before the lecture ends.

- Implement Frequency Regularization: Implement the Frequency Regularization in order to compress the redundant information on nerual network. Make sure that the accuracy and velocity is acceptable for the nerual 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 libaray to the emulator. Collect and analyze the error information if we get.
- Adjust Dependencies: Potential errors may probably caused by Android PyTorch Library. Since the library on android lacks of improvement not as much as laptop.
- Adjust Frequency Regularization: We can also try to change something on the algorithm to let the new one fitting the Android system.

## How You Plan to Implement Your Ideas

In this research, we are only going to put the compressed neural network to the mobile devices and make it run on the emulator.

## Timeline for Deep Distribution

xxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxx

## Short Description of 5 LABS

Deploy the Frequency Regularization algorithm with the UNet (neural network) on the computer and adjust some parameters for the emulator.

Attempt to implement PyTorch on the emulator and summarize the error messages.

Analyze the error messages and try to do the debug process.

XXX

Test all functions and neural networks to check the effectiveness and accuracy on the emulator after importing the compressed neural network. We compare the result 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.