Deployment of Frequency Regularization on Android Mobile Devices

1st Wenhao You
Department of Computing Science
University of Alberta
Edmonton, Canada
wyou1@ualberta.ca

2nd Leo Chang
Department of Computing Science
University of Alberta
Edmonton, Canada
basu@ualberta.ca

***Abstract*—xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx**

***Index Terms*—convolutional neural networks, frequency regularization.**

I. Introduction

Currently, people can not live without mobile devices. These compact yet powerful gadgets have become indispensable tools for communication, entertainment, and information. Their portability and versatility make them a constant companion. Meanwhile, Convolutional neural networks play an important role in computer vision applications. However, these neural networks are usually implemented on high-specification hardware. There are several advantages of running convolutional neural networks on mobile devices: privacy, internet, and runtime. For enhancing privacy, personal information does not need to be uploaded or transmitted to the cloud servers anymore. For reading the dependence on the internet connection, the functionality on local devices can replace some internet services. For the runtime, especially some applications that need real-time feedback, without connecting to the cloud server can shorten the processing time. In all, convolutional neural networks can totally replace the usage of many applications on mobile devices, ensuring personal data security.

According to the popularity of mobile devices and the benefits of convolutional neural networks, we want to find a way to deploy some large and complex convolutional neural networks on mobile devices, leading to the question: "How can we deploy large convolutional neural networks on mobile devices?"

We found five methods to achieve our goal: upgrade the hardware of mobile devices; use Extreme Learning Machine (ELM) [1] to allocate the weight of the hidden layers randomly in order to train large models on mobile devices faster; use NestDNN [2] dynamically adjust the size and computational complexity of the network based on available resources on mobile devices; use "One-shot Whole Network Compression" [3] to prune, quantize, and compress the neural networks; use Frequency Regularization (FR) [4] to reduce parameters by removing high-frequency component. We make a more detailed introduction to their drawbacks and limitations in Section II.

After conducting a thorough literature review, considering all the limitations, accuracy, complexity, and future potential, we choose Frequency Regularization (FR) as our target algorithm, deploying it on one of the most popular mobile devices - Android mobile. Our main idea uses FR to compress a convolutional neural network U-Net and then decompresses the uploaded compressed model on an Android mobile device in a short time. After that, use the decompressed model to do image segmentation for the Carvana Image Masking Challenge Dataset [5].

The proposed main idea can be divided into three directions to achieve the final goal respectively:

- Direct deployment and tuning of FR code [6] on Android: Instead of introducing an additional operating system layer, this method focuses on deploying the Frequency Regularization (FR) algorithm within the Android ecosystem directedly. The core idea of this direction is to optimize and adjust the FR parameters specifically for the Android hardware and software architecture.
- Deployment of Linux environment on Android system: It aims to add another layer to the Android system, providing a more controlled and flexible development environment for deploying and testing deep learning models. We choose Termux [7], [8], [9] which is an Android terminal application and Linux environment to deploy.
- Deployment of FR code on Android Studio: This method aims to optimize the FR algorithm for Android's native architecture, ensuring seamless integration and operation within Android. The main idea of this method is similar to the first direction above.

II. Related Work

Extreme learning machine (ELM) has been widely used in artificial intelligence field over the last decades [1], [10], [11], [12], [13]. Although this algorithm has seen significant development, it also has several drawbacks:

- Poor tunability: It has poor controllability of the model since ELM calculates the least squares solution directly, and users cannot analyze the characteristics of the datasets to fine-tune. Adjusting models based on specific performance of mobile devices is important to mobile development.

- Lack of robustness: The performance of the model can be affected significantly while including certain outliers in different datasets, indicating poor robustness. Deployment on mobile devices needs to handle various inputs, including every potential outlier. Although there are many advanced versions of ELM [14], [15], [16], [17] they lacks universality and are not as easy as other algorithms to deploy.
- Overfitting issues: While deploying large convolutional neural networks on mobile devices, model generalization is crucial since overfitting can result in poor performance on unseen data. ELM easily leads to overfitting issues because it is based on empirical risk minimization without considering structural risk. Xue et al. [18]pointed to a regularization strategy to solve this problem by feature-selection.

NestDNN is a framework that takes the dynamics of runtime resources into account [2]. The experiment of Fang et al. [2] achieves as much as 4.2% increase in inference accuracy, 2.0× increase in video frame processing rate and 1.7× reduction in energy consumption. However, NestDNN also comes with some limitations. Its computational cost is significantly higher by using filter pruning method Triplet Response Residual (TRR). The high computational cost could probably exceed the processing capabilities of existing mobile devices and the runtime of model generation may be too long, which is not suitable for our deployment.

”One-shot Whole Network Compression ” [3] includes removing certain parameters or structures, which is irreversible. Moreover, by using this compression method, the accuracy is too low. For example, in the experiment of Kim et al., by using AlexNet, the accuracy of the compressed model can drop by more than 50%. In order to increase its accuracy, we have to make fine-tuning on the compressed model. Increasing accuracy requires at least more than 10 training epochs, which wastes too much time.

The proposed frequency regularization (FR) [4] works by restricting the non-zero elements of network parameters in the frequency domain, thus reducing information redundancy. Table I illustrates the evaluation of the proposed frequency regularization on UNet, according to compression rate, number of parameters, and dice score. Dice score is a metric for assessing the quality of image segmentation and ranges from 0 to 1, where 0 indicates no overlap and 1 indicates perfect overlap. The data under the dashed line represents the result under the most extreme condition in which only 759 float16 parameters are kept in UNet-v4. Thus, according to the surprised and satisfying experiment outcomes, we choose the frequency regularization as our compression method, to deploy it on mobile devices (i.e. Android system).

TABLE I

EVALUATION OF THE PROPOSED FREQUENCY REGULARIZATION ON UNET FOR IMAGE SEGMENTATION TASKS USING CARVANA IMAGE MASKING CHALLENGE DATASET [4], [5].

	Dice Score	Compression Rate	# of Parameters
UNet-ref	99.13%	100%(1×)	31,043,586
UNet-v1	99.51%	1%(100×)	310,964
UNet-v2	99.37%	0.1%(1000×)	31,096
UNet-v3	98.86%	0.0094%(10573×)	2,936
UNet-v4	97.19%	0.0012%(81801×)	759(float16)

Termux [7], [8], [9] is an Android terminal application and Linux environment. It works directly with no rooting or setup required. To use Termux, the system needs to meet some requirements: Android 5.0 - 12.0; CPU: AArch64, ARM, i686, x86_64; at least 300 MB of disk space. It is open source and can be accessed at https://github.com/termux/termux-app. The instruction of deploy Termux on Android devices is available at https://github.com/btxcy/NeuralOnMobile#readme.

III. METHODOLOGY

xxxx

IV. EXPERIMENT

xxxx

V. LIMITATION AND FUTURE WORK

xxxx

VI. CONCLUSION

xxxx

REFERENCES

[1] A. Akusok, L. Leal, K.-M. Björk, and A. Lendasse, “High-performance elm for memory constrained edge computing devices with metal performance shaders,” in *Proceedings of ELM2019*, ser. Proceedings in Adaptation, Learning and Optimization, J. Cao, C. Vong, Y. Miche, and A. Lendasse, Eds. International: Springer, 2021, pp. 79–88.

[2] B. Fang, X. Zeng, and M. Zhang, “Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision,” in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom ’18. ACM, Oct. 2018. [Online]. Available: http://dx.doi.org/10.1145/3241539.3241559

[3] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, “Compression of deep convolutional neural networks for fast and low power mobile applications,” 2016.

[4] C. Zhao, G. Dong, S. Zhang, Z. Tan, and A. Basu, “Frequency regularization: Reducing information redundancy in convolutional neural networks,” *IEEE Access*, vol. 11, pp. 106 793–106 802, 2023.

[5] M. M. M. P. W. C. Brian Shaler, DanGill, “Carvana image masking challenge,” 2017. [Online]. Available: https://kaggle.com/competitions/carvana-image-masking-challenge

[6] G. Dong, “Frequency regularization,” https://github.com/guanfangdong/pytorch-frequency-regularization, 2023.

[7] termux, “Termux application,” 2023. [Online]. Available: https://github.com/termux/termux-packages

[8] M. Rana, “Open and accessible education with virtual reality,” 2022. [Online]. Available: https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=75298.

[9] termux, “The termux wiki,” 2023. [Online]. Available: https://wiki.termux.com/wiki/Main_Page

[10] R. N. Shifei Ding, Xinzheng Xu, “Extreme learning machine and its applications,” *Neural Computing and Applications*, vol. 25, no. 3, pp. 549–556, 2014. [Online]. Available: https://doi.org/10.1007/s00521-013-1522-8

[11] S.-H. W. Y.-D. Z. Jian Wang, Siyuan Lu, "A review on extreme learning machine," *Multimedia Tools and Applications*, vol. 81, no. 29, pp. 41 611–41 660, 2022. [Online]. Available: https://doi.org/10.1007/s11042-021-11007-7
[12] C. Deng, G. Huang, J. Xu, and J. Tang, "Extreme learning machines: new trends and applications," *Science China Information Sciences*, vol. 58, no. 2, pp. 1–16, 2015.
[13] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006, neural Networks. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231206000385
[14] J. M. Fossaceca, T. A. Mazzuchi, and S. Sarkani, "Mark-elm: Application of a novel multiple kernel learning framework for improving the robustness of network intrusion detection," *Expert Systems with Applications*, vol. 42, no. 8, pp. 4062–4080, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417414008197
[15] K. Zhang and M. Luo, "Outlier-robust extreme learning machine for regression problems," *Neurocomputing*, vol. 151, pp. 1519–1527, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231214012053
[16] Q.-Y. Zhu, A. Qin, P. Suganthan, and G.-B. Huang, "Evolutionary extreme learning machine," *Pattern Recognition*, vol. 38, no. 10, pp. 1759–1763, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320305001809
[17] K. Sun, J. Zhang, C. Zhang, and J. Hu, "Generalized extreme learning machine autoencoder and a new deep neural network," *Neurocomputing*, vol. 230, pp. 374–381, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092523121631503X
[18] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, vol. 1168, no. 2, p. 022022, feb 2019. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1168/2/022022