Characterizing DNN Models for Edge-Cloud Computing

Chunwei Xia[†§], Jiacheng Zhao[†], Huimin Cui[†§], Xiaobing Feng[†§]
[†]*State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences*
[§]*School of Computer and Control Engineering, University of Chinese Academy of Sciences*
Beijing, China
{xiachunwei, zhaojiacheng, cuihm, fxb}@ict.ac.cn

***Abstract*—Traditionally Deep Neural Networks (DNNs) services are deployed in the cloud, due to the computation-intensive DNN models. In recent years, as emerging edge computing provides new possibilities for DNN applications, we have opportunities to process DNN models in the cloud and on the device collaboratively, i.e., edge-cloud computing. Since cloud and edge devices demonstrate significant diversity on inference latency, network transmission overhead, memory capacity and power consumption, it is a big challenge to determine the DNN model deployment in the cloud and on edge devices. In this paper, we characterize the behaviours of three types of DNN models, i.e., CNN, LSTM and MLP, on four types of platforms, i.e., server-class CPU, server-class GPU, embedded device with GPU, and smart-phones. Our experimental results demonstrate that we can carefully tune a deployment strategy for DNN models in the cloud, and on big and (or) little cores of the edge device, to balance performance and power consumption.**

I. Introduction

In recent years, DNNs are widely adopted in a variety of Artificial Intelligence (AI) applications, such as speech recognition, image classification, and natural language processing. Traditionally these applications are deployed as Request-Response services in the cloud. However, they are moving to the edge, as the emerging AI chips being integrated into smartphones. When a DNN is completely processed in the cloud, the inference latency is minimized, but the data needs to be transferred to the cloud, which introduces extra network transmission overhead. On the contrary, when a DNN is totally processed on device, the network transmission overhead is eliminated, but the inference latency will be increased. An alternative way is to run the DNNs between the edge and the cloud collaboratively to balance the network transmission overhead and on-device inference latency. Thus, it is a big challenge to deploy the DNNs between on-device processing and cloud-only processing coordinately, due to their diversity in inference latency, network transmission overhead, memory capacity and power consumption.

A representative research on edge-cloud processing is Neurosurgeon [1], which automatically partitions DNN computation between mobile devices and data-center at the granularity of neural network layers. Neurosurgeon demonstrates the benefit of collaborative edge-cloud processing. However, there are still several questions to be answered:

- For a given DNN model, how the process time is distributed across its layers on different platforms?
- What are the performance characteristics of different kinds of DNNs on cloud and edge platforms?

To provide an insight into these questions, we present detailed experimental results for frequently-used DNN models on representative cloud and edge platforms. Experimental results show that mobile platforms are suitable for running representative LSTM, MLP and mobile-optimized CNN models, but unsuitable for traditional CNN models while server class CPUs and GPUs are sufficient enough for DNNs' inference. This indicates that it is of the essence to explore collaborative edge-cloud processing of DNN models. Furthermore, for smartphones integrated with powerful/power-hungry cores (big) and slower/battery-saving cores (little), we find that there exists possibility to deploy DNN computation across the big and small cores to balance performance and power consumption.

II. Experimental Setup

In our study, we consider three typical kinds of DNNs: Convolutional Neural Networks (CNNs) [2]–[4], Recurrent Neural Network (RNNs) [5] and Multilayer Perceptrons (MLPs) [6]. Table I summarizes DNNs used in this paper.

TABLE I: DNN specifications

Name	Input	Output	Layers	# Params	FLOPs
Inception V1	[224,224,3]	[1000]	22	6.79M	3.19B
ResNet-50			50	25.6M	3.8B
MobileNet 1.0			15	4.2M	576M
LSTM	[20, 200]	[20,10000]	2	2.65M	14.8M
MLP	[784]	[10]	5	13.9M	13.9M

Table II shows the specifications of cloud and edge platforms evaluated in this paper. For software platforms, we use TensorFlow version r1.4 for both cloud and edge platforms, Intel MKL-DNN as the matrix computation libraries for server class CPUs and CUDA 8 and cuDNN 6 for server class GPUs. For edge platforms, TensorFlow is compiled with Android NDK r14b (targeting *arm64-v8a*). We leverage the

This work is supported in part by the National Key R&D Program of China (2016YFB1000402 and 2017YFB0202002), the National Natural Science Foundation of China (61521092, 61432016, 61432018, 61332009, 61802368).

TABLE II: Device notations for cloud & edge platforms

Platform	Notation	Mobile	SoC	Notation.
E5-2620 v4	`Server CPU A`	OnePlus 5T	Sd. 835	`Edge CPU A`
E5-1603 v4	`Server CPU B`	OnePlus 3	Sd. 820	`Edge CPU B`
Tesla K40c	`Server GPU A`	RedMi	Sd. 625	`Edge CPU C`
GTX 1070	`Server GPU B`	Jetson TX2	Pascal GPU	`Edge GPU A`

'Sd.' is short for Snapdragon. RedMi typically means RedMi Note 4x in this paper.

`benchmark_model` [7] utility provided by TensorFlow for performance analysis. The batch size is set to 1 as we only focus on inference phase.

III. Evaluation Result

In this section, we first present overall evaluation results of five DNN models across eight different platforms. Then, we take the `Edge CPU A` as an example to investigate how the number of cores/threads as well as big.LITTLE architecture impacts the inference latency.

Fig. 1 depicts the computation latency of five DNN networks on eight platforms, with X-axis representing DNN models and Y-axis representing computation latency (note that the Y-axis is log-scaled). For a given DNN model on a specific platform, we report the optimal results by trying all available configurations. Fig. 1 leads to the following findings:

- Modern edge CPUs are powerful enough to process DNNs like LSTM, MLP and mobile-optimized CNNs while still insufficient for traditional CNNs.
- GPU (Server or Edge) is around an order of magnitude faster than corresponding CPU for CNNs and MLP while no such performance gap is observed for LSTM.

To balance performance and energy consumption, big.LITTLE architecture is commonly adopted in modern smartphone devices. We evaluate how inference latency of DNN models scale with number of threads for big.LITTLE architecture. In Fig. 2, the three CNNs (Inception V1, ResNet-50, MobileNet 1.0) are plotted against left Y-axis while LSTM and MLP are against right Y-axis. Note that 8 threads (Big+Little) represent using all cores. According to Fig. 2, we observe that:

- CNNs exhibit well scalability when enjoying more threads/cores while LSTM and MLP benefit little from extra threads.

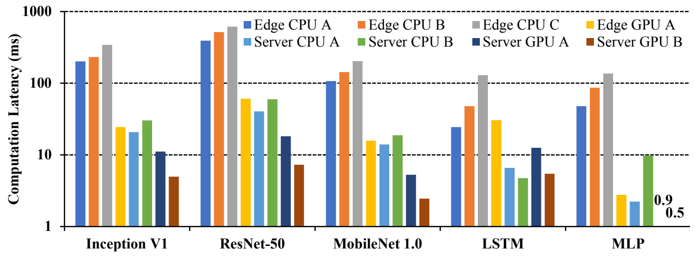

Fig. 1: Computation latency of five DNN models on eight platforms with optimal configuration.

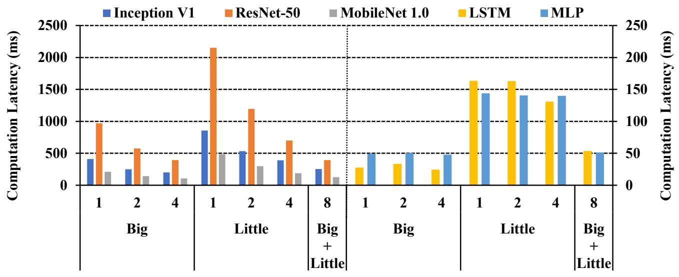

Fig. 2: Computation latency of five DNNs on `Edge CPU A` with 1, 2, 4, 8 threads when using Big and/or Little cores.

- More resources may harm performance of DNN models on big.LITTLE architectures. The computation latency of all five DNNs increases when moving from 4 big cores to 4 big + 4 little cores in Fig. 2. This phenomenon is possibly caused by the operating system scheduler or characteristic of big.LITTLE architecture (e.g. no cache sharing between big and little cores).

IV. Conclusion

In this paper, we provide in-depth evaluation of five DNN models on four types of platforms. Experimental results demonstrate the computation latency of DNN models show much variance due to the diversity of platforms. More specifically, dynamic deploy strategy shall be carefully designed to balance performance and energy consumption on mobile big.LITTLE architecture. In future work, we will evaluate the energy consumption of edge platforms, and explore opportunities for collaborative edge-cloud processing of DNN models.

References

[1] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '17. New York, NY, USA: ACM, 2017, pp. 615–629.

[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[5] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent Neural Network Regularization," *ArXiv e-prints*, Sep. 2014.

[6] S. Shi, Q. Wang, P. Xu, and X. Chu, "Benchmarking state-of-the-art deep learning software tools," in *Cloud Computing and Big Data (CCBD), 2016 7th International Conference on*. IEEE, 2016, pp. 99–104.

[7] (2017) tensorflow/tools/benchmark/. [Online]. Available: https://github.com/tensorflow/tensorflow/tree/master/tensorflow/tools/benchmark