Dynamic DNNs meet Runtime Resource Management on Mobile and Embedded Platforms

Lei Xun, Jonathon Hare, Geoff V. Merrett
University of Southampton, UK
{lx2u16, jsh2, gvm}@ecs.soton.ac.uk

Bashir M. Al-Hashimi
King's College London, UK
bashir.al-hashimi@kcl.ac.uk

Deep neural network (DNN) inference is increasingly being executed on mobile and embedded platforms due to low latency and better privacy. However, efficient deployment on these platforms is challenging due to the sheer volume of computation and memory access.

Static model pruning is an effective method to reduce the model parameters (i.e. computation demand) at the cost of accuracy loss. Platform-aware model pruning [1] gradually reduces the number of parameters while measuring the latency, and stops pruning once the target latency is achieved in order to keep the accuracy as high as possible. However, the latency measurement is based on a fixed hardware configuration (e.g. CPU/GPU at maximum clock frequency), which is often unavailable at runtime due to thermal throttling or other concurrently executing applications sharing the hardware. The target latency could also change during different execution phases of the same application [2]. Although a conservative hardware configuration can guarantee hardware availability at runtime, more parameters need to be pruned to achieve the same latency target, leading to a model with lower accuracy not fully utilising available hardware.

Unlike static model pruning, which can only fit into a fixed latency target and hardware configuration, dynamic DNNs are trained to have sub-networks with different latency-accuracy trade-offs through either channel scaling [3], [4] or layer scaling [5]. For example, smaller sub-networks are faster but less accurate than larger ones. At runtime, dynamic DNNs can change the architecture among their sub-networks to comply with new latency targets and dynamically available hardware resources. However, existing works treat dynamic DNNs as an algorithm-only approach, without considering hardware trade-off opportunities. This limits the trade-off range and granularity in latency, power and energy. Furthermore, standalone channel/layer scaling cannot fully utilise heterogeneous computing resources on modern mobile/embedded platforms, and their application is limited to convolution neural networks [6].

We propose a holistic system design for DNN performance and energy optimisation, combining the trade-off opportunities in both algorithms and hardware. As shown in Fig 1, a system can be viewed as three abstract layers: the device layer contains heterogeneous computing resources; the application layer has multiple concurrent workloads; and the runtime resource management layer monitors the algorithms' dynamically changing performance targets as well as hardware resources and constraints, and tries to meet them by tuning the algorithm and hardware at the same time. Moreover, We illustrate the runtime approach through a dynamic version of 'once-for-all network [7]' (namely Dynamic-OFA), which can scale the entire DNN architecture to fit heterogeneous computing resources [6] efficiently and has good generalisation for different model architectures such as Transformer [8]. Compared to the state-of-the-art Dynamic DNNs, our experimental results using ImageNet on a Jetson Xavier NX show that the Dynamic-OFA is up to 3.5x (CPU), 2.4x (GPU) faster for similar ImageNet Top-1 accuracy, or 3.8% (CPU), 5.1% (GPU) higher accuracy at similar latency. Furthermore, compared with Linux governor (e.g. performance, schedutil), our runtime approach reduces the energy consumption by 16.5% at similar latency.

These works were supported in part by the Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/S030069/1. No data except those explicitly stated in the paper were generated during the study.

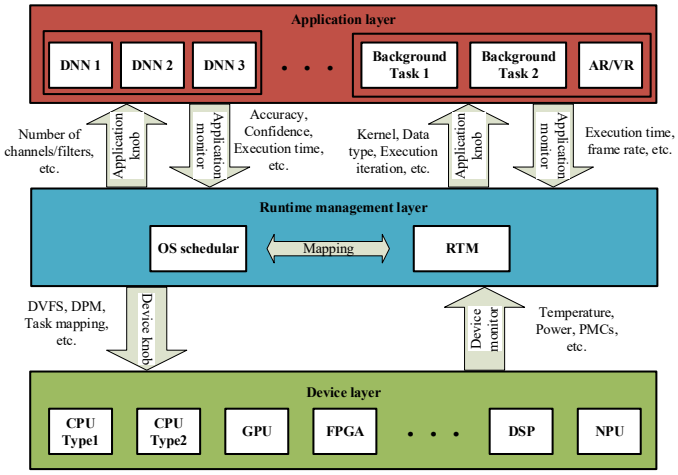

Fig. 1. A holistic system design for DNN performance and energy trade-offs. Algorithm knobs (e.g. dynamic DNNs) and hardware knobs (e.g. task mapping, dynamic voltage and frequency scaling (DVFS)) are combined to meet algorithms' dynamically changing performance targets (e.g. accuracy, latency) as well as hardware resources (e.g. available cores, clock frequency level) and constraints (e.g. power, temperature).

REFERENCES

[1] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, and H. Adam, "NetAdapt: Platform-aware neural network adaptation for mobile applications," in *ECCV*, 2018.
[2] L. Xun, L. Tran-Thanh, B. M. Al-Hashimi, and G. V. Merrett, "Optimising resource management for embedded machine learning," in *DATE*, 2020.
[3] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, "Slimmable neural networks," in *ICLR*, 2019.
[4] L. Xun, L. Tran-Thanh, B. M. Al-Hashimi, and G. V. Merrett, "Incremental training and group convolution pruning for runtime dnn performance scaling on heterogeneous embedded platforms," in *MLCAD*, 2019.
[5] S. Laskaridis, S. I. Venieris, H. Kim, and N. D. Lane, "HAPI: Hardware-aware progressive inference," in *ICCAD*, 2020.
[6] W. Lou, L. Xun, A. Sabet, J. Bi, J. Hare, and G. V. Merrett, "Dynamic-ofa: Runtime dnn architecture switching for performance scaling on heterogeneous embedded platforms," in *CVPR Workshop*, 2021.
[7] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *ICLR*, 2020.
[8] H. Parry, L. Xun, A. Sabet, J. Bi, J. Hare, and G. V. Merrett, "Dynamic transformer for efficient machine translation on embedded devices," in *MLCAD*, 2021.