Concurrent Deep Neural Network Tasks on Constrained Devices

Valentin Radu
University of Edinburgh

Yuan Wen
Trinity College Dublin

Deep neural networks have proven transformative across many domains due to their superior accuracy and their characteristic to scale well on growing available training data. Once trained, these models can be deployed to many devices, each with specific computational and memory constraints, to produce edge intelligence closer to data source. Concurrent applications can utilise generated data to facilitate inference for different purposes, such as recognition and detection on images and on audio streams, exploiting their rich source of information to observe different characteristics. But how can multiple neural networks produce their inference under tight resource constraints and latency budget? We explore optimisations across the inference stack to produce a framework for efficient execution of concurrent deep neural networks. Presented optimisations include primitive selection for efficient management of memory footprint and inference time at the systems level, and features sharing to reduce computations in the machine learning space.

The growing number of recognition tasks explored in the research community, which show a superior performance for deep neural networks compared to all other machine learning solutions, encourages their adoption at scale on mobile devices, on robotics and on devices for entertainment and others. For example, an image taken with the phone camera can be used to understand the location through background landscape recognition, a social network app may recognise familiar faces from this image on the device (without cloud assistance), a mental health app may also process the same image to recognise features that can assess the mood of people and the quality of experiences users are involved in. Similar to how location is shared to many applications by subscription to the location service, the same should be possible to perform for diverse recognition tasks on incoming sensor data. Similarly, in robotics, visual perception can be improved by running frame processing concurrently, starting the execution of incoming frames before previous ones have finished; or to start using incoming frame for planing by another network before perception network has finished.

We profile the execution time of different network structures on embedded devices by evaluating their performance with a range of primitives from 70 candidates. These are then selected at runtime to optimise the memory footprint and inference latency for each concurrent task. Through a well designed optimiser, we show that Pareto-frontier execution points can be selected to identify the best performing primitive for a given memory budget (systems with the ARM Coretx-M processor come with as little memory as 32KB, where most default primitives in other deep learning frameworks would not be able to execute), but also to associate independent memory budgets to each concurrent deep neural network in partitioning the main memory. We are primarily concerned with the main memory judicial allocation in this work, to avoid the big penalty of fetching weights from flash, which is much greater than cache page misses.

At the machine learning level, one approach is to share early feature extracted from input (first Convolutional layers) and to split the final part of the network into Siamese Neural Networks, specialising to different tasks – a common approach for joint face identification and detection tasks. While this works well for similar tasks with transferable features, it is not efficient for diverging recognition tasks especially after network compression (common for network adaptation to run on smaller devices). The alternative is to designate filters to each task, while aiming to share a larger number between tasks (networks) to reuse computations. We evaluate two approaches here, one with repeated shared features at different stages in the longitudinal structure of a network [1] and another method with shared filters transversely in the network architecture [2] to observe their impact and propose the best approach to combine their strengths.

In this work we show that specific optimisations at different levels in the inference stack can enable efficient execution of concurrent deep neural networks to process incoming data on resource constrained devices.

References

[1] Alexandre Boulch. Sharesnet: reducing residual network parameter number by sharing weights. In *arXiv:1702.08782*, 2017.

[2] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *arXiv:1803.10082*, 2018.