Poster: Space and Time Optimal DNN Primitive Selection with Integer Linear Programming

Yuan Wen*, Andrew Anderson*, Valentin Radu†, Michael F.P. O'Boyle†, David Gregg*
*Trinity College Dublin, †University of Edinburgh

***Abstract*—Convolutional neural networks (CNNs) are used in many applications, from industrial robotics to biometric identification on mobile devices. But they can be too resource-hungry for mobile and embedded devices with tightly constrained memory and energy budgets. We propose an ahead-of-time primitive selection for CNNs, based on integer linear programming (ILP). Under a tight memory budget, our ILP solver selects the optimal primitive for each layer such that the entire network is optimized for execution time subject to a memory budget, or vice versa. Our method yields significant speedup and memory reduction compared to existing methods.**

***Keywords*-neural network optimization; computing operators; primitive selection; optimal convolutional layer**

I. Introduction

CNNs have become a dominant technique for computer vision tasks due to their outstanding performance in image classification, image segmentation, objects detection, image style transfer, and others [1]. Emerging computer vision based applications will need to run on small devices to be portable and interactive. On these limited compute resources, large neural network models will need to be specially optimized to meet the constraints of each device [3].

Instead of designing and training custom neural networks for each device, the alternative is to drastically improve the computational and memory efficiency of these large models to exploit target hardware characteristics. While time efficiency optimizations have been explored before [2], similar emphasis should be placed on memory requirements.

In this work we address the problem of optimizing the execution of CNNs under tight memory budgets. Many different implementations exist for computing convolutional layers, some sacrificing memory for performance (*im2col*), while others applying radical transformations to benefit from cheaper operations (*winograd* [4]). Further, the execution time of an algorithm depends partly upon the layout of data in memory. For example, a common data layout for inputs is CHW (channels, height, width), but some algorithms are faster on a HWC data layout. Using a library of 57 convolution primitives with different algorithms and data layouts, we investigate this time-memory tradeoff.

Given a memory budget for the CNN, the best available primitive, operating on the best data layout, is selected for each layer such that these achieve the fastest inference time while having a memory footprint within the memory budget.

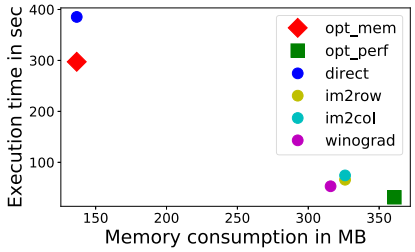

Figure 1: Execution time and memory footprint of GoogleNet on an ARM A15 processor for the same primitive choice across the entire network (circles), the selection of primitives for optimal memory (*opt_mem*) and for optimal time performance (*opt_perf*).

II. Motivation

Current deep neural network models are mostly designed for high performance server-side computations where memory constraints are not critical. Adopting these models on smaller devices is challenging [5]. Figure 1 shows the run-time memory requirement and execution time of GoogleNet (fitting the whole model into memory), implementing the convolutional layer with different primitive choices. One option is to implement the network with a single primitive choice across all convolutional layers (as commonly done by most deep learning frameworks): direct convolution (*direct*), image to row (*im2row*), image to column (*im2col*) and winograd – presented with circles in Figure 1; and the alternative is to select primitives at each convolutional layer to optimize for memory footprint (*opt_mem*) and for inference time performance (*opt_perf*). One choice of primitives across the whole network may result in very poor performance (direct convolution running one inference of GoogleNet in 400sec) or may not be able to run the network at all, if physical memory on device is less than 300MB. In contrast, optimizing primitive selection and data layouts for each layer enables more control, not just better execution time (opt_perf) and minimal run-time memory footprint (opt_mem), but also additional points in between to adjust this trade-off based on specific device or application conditions.

III. ILP Model

This optimization strategy is suitable for devices with enough physical memory to accommodate the entire network

978-1-7281-3613-4/19/$31.00 ©2019 IEEE
DOI 10.1109/PACT.2019.00059

into main memory, optimizing for performance across the whole network. A memory budget can be introduced if sharing the system memory with other tasks, or performance critical conditions to determine the optimal point under these constraints with the ILP optimiser.

Equation 1 indicates the formulation of the ILP solver used in this work. This optimizes for inference time under constrained imposed by the available *MemoryBudget*. Varying these constrains has immediate impact on the solution determined by the optimizer.

$$
\begin{aligned}
&minimize \\
&Time = \sum_{i=1}^{m} \vec{L_i} \bullet \vec{T_i} + \\
&\qquad \sum_{i=1}^{m-1} \sum_{j=2}^{m} \vec{L_i} \times Trans\_(i,j) \times \vec{L_j}^{\,T} \\
&\text{s.t.} \qquad \sum_{i=1}^{m} \vec{L_i} \bullet \vec{M_i} \leq MemoryBudget \\
&\qquad \sum_{j=1}^{n} \vec{L_{i_j}} = 1, \quad \forall i \in [1,m]
\end{aligned}
\tag{1}
$$

In equation 1, $\vec{L_i}$ represents the one-hot condition variable vector for layer i, $\vec{T_i}$ is the profiled execution time vector for all candidate primitives, $\vec{M_i}$ contains the corresponding memory footprint for each primitive and $Trans\_(i,j)$ is the time required by data layout transformations.

IV. Evaluation

To evaluate our approach, we implemented the ILP model and used it to select primitive routines for convolutional layers on a low power embedded processor. We measure the execution times and memory requirements for each convolutional layer using each of the 57 convolution primitive routines (same as in [2]) on the Arm processor architecture Cortex-15. These measurements are used by our ILP model to select the optimal primitive for each layer. Its selection depends on the execution time, entire neural network memory budget, and on the need to insert data layout transformation routines between different layers if consecutive layer primitives use incompatible data layouts.

As seen in Figure 1, optimal points exist not only for combination of primitives to achieve the fastest execution time (also explored by Anderson et al. [2]) without memory constraints, but also for best performance with minimal memory footprint. More than these, our ILP solver can produce the optimal primitive selection for a given memory budget. The bars in Figure 2 show the execution time of running AlexNet implemented with the optimal selection of primitives for a given memory budget. For reference, statically allocated primitives (same primitive choice across the network) are also marked to indicate the limitation of uniform primitive selection, and also the optimal primitive selection for performance (PBQP), disregarding memory footprint.

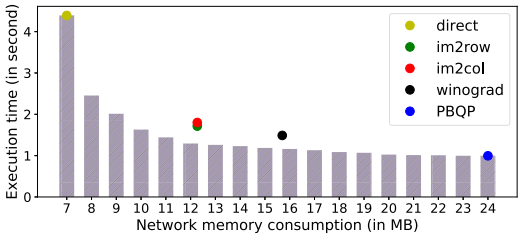

Figure 2: Bars indicate the performance of optimal primitive selection on varying memory budget for the execution of AlexNet on the Cortex-A15 processor. Additional reference points present the performance of a single primitive selection and of PBQP [2] selection.

V. Conclusion

This work addresses a growing challenge, how to make the large deep neural network models available to use on mobile and embedded devices which have limited computational resources. We show that by using an Integer Linear Programming based solver model, hybrid selection of convolutional layer implementations can optimize for both inference latency and memory footprint.

This opens the opportunity for many innovative edge applications to use the higher accuracy of large deep neural network models, which have been unpractical on these small devices until now.

Acknowledgment

This work was supported by ARM and Science Foundation Ireland grant 13/RC/2094 to Lero – the Irish Software Research Centre (www.lero.ie), by the European Unions Horizon 2020 research and innovation programme under grant agreement No. 732204 (Bonseyes), and by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 16.0159.

References

[1] Ian Goodfellow and Yoshua Bengio and Aaron Courville, *Deep Learning*, MIT Press, 2016

[2] Andrew Anderson and David Gregg, *Optimal DNN primitive selection with partitioned boolean quadratic programming*, CGO 2018

[3] Song Han, Huizi Mao, and William J. Dally, *Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding*, ICLR 2016

[4] Andrew Lavin and Scott Gray, *Fast Algorithms for Convolutional Neural Networks*, CVPR 2016

[5] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li and Song Han, *AMC: AutoML for Model Compression and Acceleration on Mobile Devices*, ECCV 2018