Evaluating Grouped Spatial Pack Convolutions on Edge CPUs

Perry Gibson*[1], José Cano*[1],
Jack Turner†[2], Elliot J. Crowley†[2],
Michael O'Boyle†[3], Amos Storkey†[2]

* *University of Glasgow, United Kingdom*
† *University of Edinburgh, United Kingdom*

ABSTRACT

Grouped convolutions are a drop-in replacement for standard convolutional layers in neural networks. With an adjustable scaling parameter - the number of groups g - they reduce the number of parameters and multiply-accumulate operations (MACs) at the expense of representational power. However, current implementations of grouped convolutions do not perform optimally.In this paper we discuss Grouped Spatial Packed Convolutions (GSPC), a new approach to grouped convolutions, implemented in TVM's tensor compute language. We analyse a set of networks leveraging the full range of the g parameter, and evaluate their performance in terms of inference time. We observe that GSPC achives the best performance in all settings, improving the existing implementations of grouped convolutions in TVM, PyTorch and TensorFlow Lite by 3.4×, 8× and 4× on average respectively. Code is available at `https://github.com/gecLAB/tvm-GSPC/`

KEYWORDS: Deep Learning; Edge Computing; Grouped Convolutions

1 Introduction

Deploying deep neural networks onto mobile and embedded edge devices (e.g. smartphones, wearables, IoT boards, robots, drones, etc) faces barriers due to the resource constraints under which these devices operate. Large, memory intensive neural networks typically perform poorly on edge devices. Thus, there are a wealth of techniques to compress these networks, while attempting to maintain their learning task performance. However, novel approaches from the machine learning community, who have a preference for maintaining accuracy, may have non-trivial implications for hardware efficiency. That is, many neural architecture compression techniques may not work as expected at the system level where one of the main metrics considered is the inference time. Turner et al. [TCR+18] demonstrated that compression at the neural architecture level may have negative effects further down the

[1]Email: {Perry.Gibson, Jose.CanoReyes}@glasgow.ac.uk
[2]Email: {jack.turner, elliot.j.crowley, a.storkey}@ed.ac.uk
[3]Email: mob@inf.ed.ac.uk

Deep Learning Inference Stack, depending on the choices of algorithmic transformation, code generation and optimisation techniques, and the target hardware device.

In this paper we evaluate the performance of grouped convolutions, a network cheapening technique which reduces the number of parameters and multiply-accumulate operations in the convolutional layers. We propose a new implementation of grouped convolutions, that we call *Grouped Spatial Pack Convolutions* (GSPC), which outperforms all previous implementations of grouped convolutions present in current deep learning frameworks.

2 Grouped Convolutions

The pertinent parameter in grouped convolutions is the number of groups g. For example, at $g = 2$ the number of parameters and operations are halved compared to an equivalent standard convolutional layer. If $g = N$, this means the number of groups is equal to the number of input channels to each layer - the maximum number of groups. This special case is also known as *depthwise convolutions*. It is conventional practice to follow grouped convolutional layers with a low-cost *pointwise convolution*. We denote a network using standard convolution as S, and the same architecture using grouped convolution with g groups $G(g)$.

3 Grouped Spatial Pack Convolutions

We expect a 4D input volume in the standard `NCHW` data layout, 4D kernels, and return a 4D output volume in the same `NCHW` layout. At a high level, GSPC is comprised of four stages:

- Reshape 4D kernels into a new 7D volume.
- Reshape 4D padded input data into a 6D volume.
- Perform the grouped convolution, output to a 6D volume.
- Reshape the 6D output volume to the desired 4D output.

GSPC reshapes the data to improve memory locality, thus reducing the potential cost of loads. Similarly, accumulating the convolution on a 6D intermediate array, and reshaping to 4D output is preferable to accumulating directly onto 4D as improved locality can improve cache behaviour. The input and kernels data are reshaped, maintaining the independence of data between groups, divided across an outer dimension for groups. The reshape maps data onto tiles, with the tile size being an option for which the optimal value may vary for different data shapes and hardware environments. Tiling is performed to enable further optimisations such as vectorization, so that multiple data can be computed on simultaneously. The kernel reshaping stage can be computed ahead of time and stored on disk in lieu of the default `NCHW`-compatible layout, since it does not depend on the input data. A small example of the first two stages is shown in Figure 1, with values representing original indices.

The TVM deep learning compiler stack is well suited for these types of layer-specific optimisations. TVM uses ahead-of-time compilation for a given network, allowing a suite of potential workload specific optimisations. Additionally, it features an auto-tuning framework *autoTVM* [CZY+18] for exploring configuration space of these optimisations to improve performance further. Hence we implement GSPC as an extension to TVM, and present auto-tuned results. Figure 2 gives an overview of the relevant parts of the TVM stack.

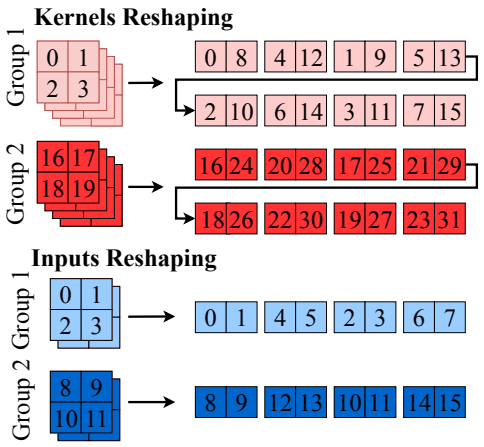


Figure 1: Example of GSPC input reshaping for small input and kernels, and two groups.

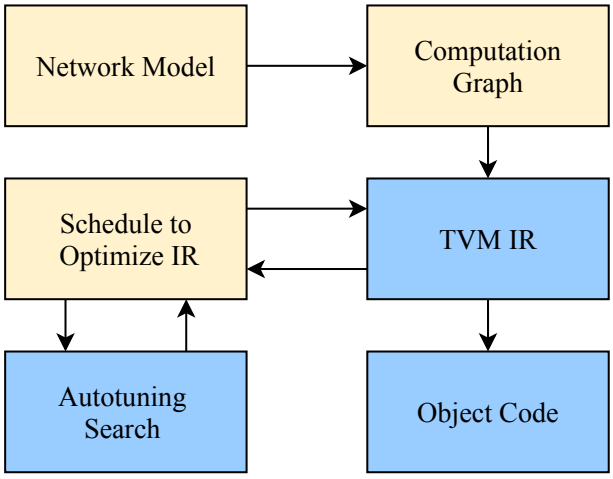


Figure 2: Overview of the relevant parts of TVM's compilation stack.

4 Evaluation

4.1 Experimental Setup

We consider two datasets widely adopted for image classification tasks, CIFAR-10 and ImageNet, and we use the `float32` type to represent data values. We evaluate two deep neural network models: WideResNet-40-2 and ResNet-34.

- WRN-40-2: we use a Wide Residual Network (WRN) with 40 layers, width-multiplier 2, and with 2.2 million parameters. We use the CIFAR-10 definition of the network.
- ResNet-34: we use a Residual Network with 34 layers that requires 21.8 million parameters. We use the network as defined for ImageNet classification.

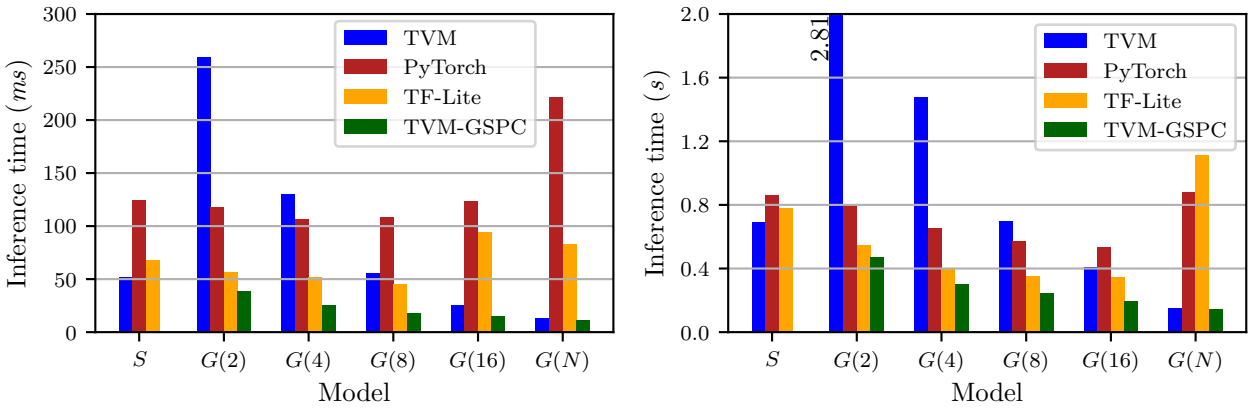
We consider the following grouped convolutions: $G(g)\ \forall\ g \in \{2, 4, 8, 16, N\}$, where N is the number of input channels to each convolutional layer.

We evaluate the two networks on a single Cortex-A73@2.4 GHz core of a Hikey 970.

4.2 Frameworks comparison

Figure 3 shows the inference time of GSPC and other implementations of grouped convolutions in current deep learning frameworks for all the G models of the two networks under study when running on the CPU of the Hikey board. The figure also shows the times for the S models. We consider the tuned version of both GSPC and unmodified TVM. The other frameworks analysed are PyTorch and TensorFlow Lite (TF-Lite).

As we can see, GSPC provides the best results for all the G models of the two networks, clearly outperforming the unmodified TVM and the other two frameworks, up to $8\times$ and $4\times$ better than PyTorch and TensorFlow Lite respectively. To the best of our knowledge, GSPC is the most efficient implementation, in terms of inference time, of grouped convolutions available. We also observe that TensorFlow Lite performs much better than PyTorch for all the G models of WRN-40-2 and for the $G(2)$-$G(16)$ models of ResNet-34, whereas PyTorch is better for $G(N)$ of ResNet-34. However, none of these frameworks scales as expected for the G models according to the number of MAC operations.



(a) WRN-40-2

(b) ResNet-34

Figure 3: Inference time for networks with standard (S) and grouped (G) convolutions. We compare the tuned versions of GSPC and unmodified TVM against PyTorch and TF-Lite.

5 Conclusion

In this paper we have presented Grouped Spatial Pack Convolutions (GSPC) as a new and more efficient implementation of grouped convolutions. We have implemented GSPC in TVM, evaluating two network models implementing grouped convolutions for two datasets on a single *big* core of the Hikey 970. A more in-depth analysis of GSPC and its performance can be found in [GCT+20]. We have compared our implementation against existing solutions in current deep learning frameworks, outperforming them in all settings.

Acknowledgments

This work is partially supported by the European Union's Horizon 2020 programme under grant No. 732204 (Bonseyes), and the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 16.0159. The opinions expressed and arguments employed herein do not necessarily reflect the official views of these funding bodies.

References

[CZY+18] T. Chen, L. Zheng, E. Yan, Z. Jiang, T. Moreau, L. Ceze, C. Guestrin, and A. Krishnamurthy. Learning to Optimize Tensor Programs. In *Advances in Neural Information Processing Systems 31*, pages 3393–3404. December 2018.

[GCT+20] P. Gibson, J. Cano, J. Turner, E. J. Crowley, M. O'Boyle, and A. Storkey. Optimizing grouped convolutions on edge devices. In *2020 IEEE 31th International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, July 2020.

[TCR+18] J. Turner, J. Cano, V. Radu, E. J. Crowley, M. O'Boyle, and A. Storkey. Characterising Across-Stack Optimisations for Deep Convolutional Neural Networks. In *2018 IEEE International Symposium on Workload Characterization (IISWC)*, pages 101–110, September 2018.