Neural Architecture Search as Program Transformation Exploration Extended Abstract

Jack Turner
University of Edinburgh
jack.turner@ed.ac.uk

Elliot J. Crowley
University of Edinburgh
elliot.j.crowley@ed.ac.uk

Michael F.P. O'Boyle
University of Edinburgh
mob@inf.ed.ac.uk

1. Motivation

Improving the performance of deep neural networks (DNNs) is important to both the compiler and neural architecture search (NAS) communities. Compilers optimize neural networks by applying program transformations, with the aim of exploiting hardware parallelism and memory hierarchy. However, legality concerns mean they fail to exploit the natural robustness of neural networks. In contrast, NAS techniques mutate networks by swapping out operations for different alternatives, such as the grouping or bottlecking of convolutions. In this way, NAS techniques exploit the natural resilience of DNNs to noise.

In this work, we reframe the process of a NAS agent swapping one type of convolution for another as a form of program transformation whose legality depends on a notion of representational capacity. This allows the transformations to be combined with existing ones into a unified optimization framework. This unification allows us to express existing NAS operations as combinations of simpler transformations, to discover new ones, and to quickly generate fast implementations for them. We prototyped the combined framework in TVM and were able to find optimizations that reduce inference time by over 3× across a wide variety of workloads.

2. Limitations of the State of the Art

We have two distinct communities with the same goal but are siloed: the NAS community, and the compiler community. NAS researchers assume the compiler is a black box bundled with the hardware, while compiler writers assume that the network architecture is set in stone. NAS researchers can discover good networks but are limited to a set of pre-implemented operations; compiler writers can efficiently exploit hardware structure but miss larger scale optimization opportunities.

NAS designs are not guaranteed to be correct and have to be separately evaluated through either a full retraining process or on a smaller proxy task. This training process severely limits the search space and can render large scale searches intractable. For example, the seminal work of [7] relied on a restricted design-space, but needed 48,000 GPU-hours to converge on its optimal network. The current state-of-the art approach still requires 216 GPU-hours [5].

Compiler approaches avoid this excessive search time but fail to exploit domain specific opportunities. They are inherently conservative and ignore the inherent ability of neural networks to weather deformation, reshaping and transformation while incurring minimal damage to their ability to learn.

Though the toolchains are exceptionally feature-rich, there is some evidence of highly-engineered implementations for specific workloads, at the cost of general support for newer optimizations [1]. All current approaches, however, are fundamentally limited by their inability to exploit NAS transformations.

What we want is the best of both worlds. We wish to combine neural architecture and compiler optimization in a unified framework. In this paper, we recast neural architecture search as program transformation exploration. By including transformations such as grouping and bottlenecking into the compiler optimization search space, we not only leverage the extensive history of program transformation research, but also discover new forms of neural architecture reduction that would not have been available to us otherwise.

3. Key Insights

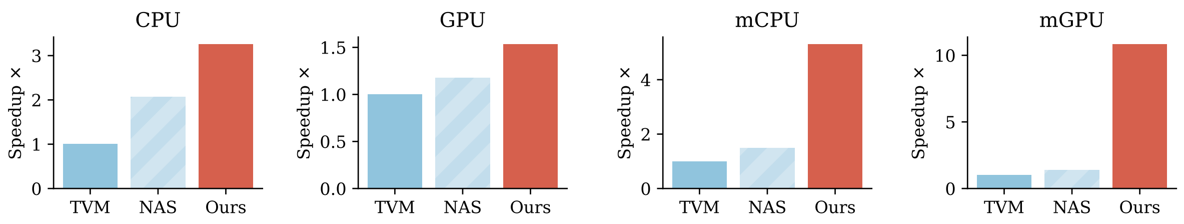
Program transformations are necessarily restricted as they must be safe. Our solution is to unlock the space of *neural transformations* by introducing a new safety metric based on Fisher Potential. It is a compile-time, cheap-to-compute metric that is able to reject transformations that would incur accuracy losses, *eliminating the need to train while searching*.

We, therefore, judge a transformation to be legal, not by data dependence preservation, but by the preservation of *representational capacity*. This unification allows the exploration of a space that leads to new operators with efficient implementations. It also shows that neural architecture options that previously required the engineering efforts of experts to develop, such as spatial bottlenecking, can be expressed as compositions of more fundamental transformations and discovered automatically.

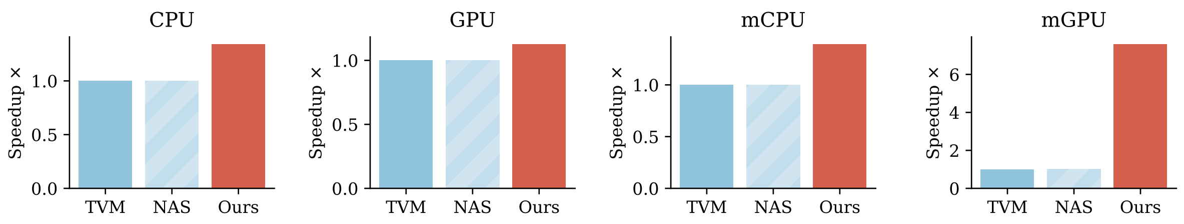
4. Main Artefacts

We have expressed our new transformations as polyhedral transformations and implemented the resulting operators in TVM [2]. We selected TVM as its API allows composition of transformation sequences, and is a well-accepted state-of-the-art optimizing compiler for convolutional neural networks.

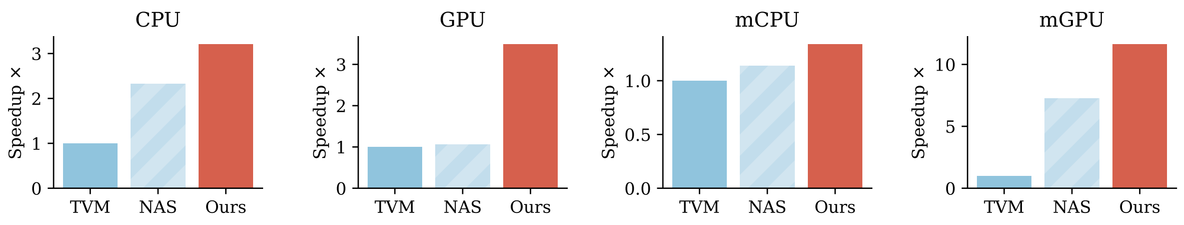
We evaluate our methodology on three popular networks: ResNet-34 [3], ResNeXt-29 [6], and DenseNet-161 [4]. These networks were chosen to represent a range of convolutional



(a) ResNet-34



(b) ResNext-29-2x64d



(c) DenseNet-161

Figure 1: End-to-end performance for several networks on different hardware devices

architectures, from standard 3×3 convolutions in ResNet-34 to grouped convolutions in ResNeXt and a heavy reliance on 1×1 convolutions in DenseNet. We verify the safety of our compiler by training the output networks on two well known datasets: CIFAR-10 and ImageNet.

The networks are implemented with each operation written as a TVM Tensor Expression [2], which is an einsum-style syntax for expressing tensor computations. This is lowered to TVM IR, where our transformations can be employed. This allows for a fair comparison of our approach. Trained models and code are publicly available.

5. Key Results and Contributions

Our contributions are as follows:

1. We reformulate popular Neural Architecture Search techniques as *program transformations*. The NAS community describes different convolution types in ad hoc manner, ignoring structural equivalence while program transformation systems cannot reason about them.
2. We use Fisher Potential to provide transformation safety without the need to train. This means we can consider NAS search as program transformation exploration with standard compiler legality checks; *without any training in the search loop.*
3. We unify the transformation and architecture search spaces, *discovering new types of convolution*. We provide thorough analysis of three examples of new types of convolutional operators available in our framework that give significant performance improvement across networks and hardware.
4. We evaluate 3 networks using these operations, ResNet, ResNext and DenseNet, on 4 platforms and demonstrate, in most cases, more than $3\times$ inference speedup over a TVM baseline. In fact, in certain cases we achieve a $10\times$ improvement (see Figure 1). This is achieved without the extremely long search times of conventional NAS.
5. We are able to explore the accuracy/space co-design space of networks and find new designs including *a new Pareto optimal network.*

6. Why ASPLOS

Neural network efficiency is of great interest to the ASPLOS community. This paper directly connects programming languages and architecture by automatically determining how to transform a program to improve performance on a range of hardware. It also brings together two distinct communities — neural architecture search and compiler optimization — around the common goal of improved hardware performance.

References

[1] Paul Barham and Michael Isard. Machine learning systems are stuck in a rut. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, pages 177–183. ACM, 2019.

[2] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. {TVM}: An automated end-to-end optimizing compiler for deep learning. In *USENIX Symposium on Operating Systems Design and Implementation*, 2018.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[4] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[5] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[6] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[7] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.