LoopTree: Enabling Exploration of Fused-layer Dataflow Accelerators

Michael Gilbert
MIT
Cambridge, US
gilbertm@mit.edu

Yannan Nellie Wu
MIT
Cambridge, US
nelliewu@mit.edu

Angshuman Parashar
NVIDIA
Westford, US
aparashar@nvidia.com

Vivienne Sze
MIT
Cambridge, US
sze@mit.edu

Joel S. Emer
MIT, NVIDIA
Cambridge, US
jsemer@mit.edu

***Abstract*—Many accelerators today process deep neural networks layer by layer. As a consequence of this processing style, every intermediate feature map incurs expensive off-chip transfers. Layer fusion eliminates off-chip transfers of intermediate results, leading to better latency and energy efficiency.**

Prior works have explored only subsets of the fused-layer design space, looking only at a particular choice of tiling, scheduling, and buffering strategy. Their architectural models are also tailored for their proposed dataflow. The lack of a unified, systematic representation of designs and a versatile evaluation method has prevented thorough exploration of the design space.

To enable systematic exploration of this design space, we present LoopTree, a framework for describing and evaluating any design in our expanded fused-layer dataflow design space. With a case study, we explore new designs to show that exploring our larger design space uncovers more efficient designs, especially for recent workloads with diverse layer types. Our design achieves $2.5\times$ speedup and $2\times$ lower energy compared to an optimized layer-by-layer design. Compared to a state-of-the-art fused-layer design, we match latency and energy while using 25% less on-chip buffer space.

***Index Terms*—analytical modeling, layer fusion, accelerators**

I. Introduction

Deep neural networks are composed of multiple layers. Many accelerators process DNN workloads *layer by layer* [1]–[4], where all operations for a layer are scheduled before the operations for the next layer (see Fig. 1(a)). The intermediate feature map, produced by the first layer and consumed by the next, is often too large to fit on-chip, so it is streamed off-chip during the processing of the first layer and back on-chip during the processing of the next.

Layer fusion eliminates off-chip transfers of intermediate results between the fused layers, saving energy and reducing latency. To reduce on-chip buffer size, fused-layer dataflows *tile* the intermediate data and interleave the *scheduling* [5]–[7] of producer and consumer operations (compare Fig.1(a), (b), and (c)). To exploit reuse, a dataflow buffers values on-chip. We call the choice of data to buffer and how long to keep them the *buffering strategy*. One key aspect of fused-layer dataflows is the choice to drop a computed activation (a value in the feature map) from the buffer, saving buffer space, and recompute when needed [7] (compare Fig.1(c) and (d)). The tiling, scheduling, and buffering strategy of a particular design create a *fused-layer dataflow*.

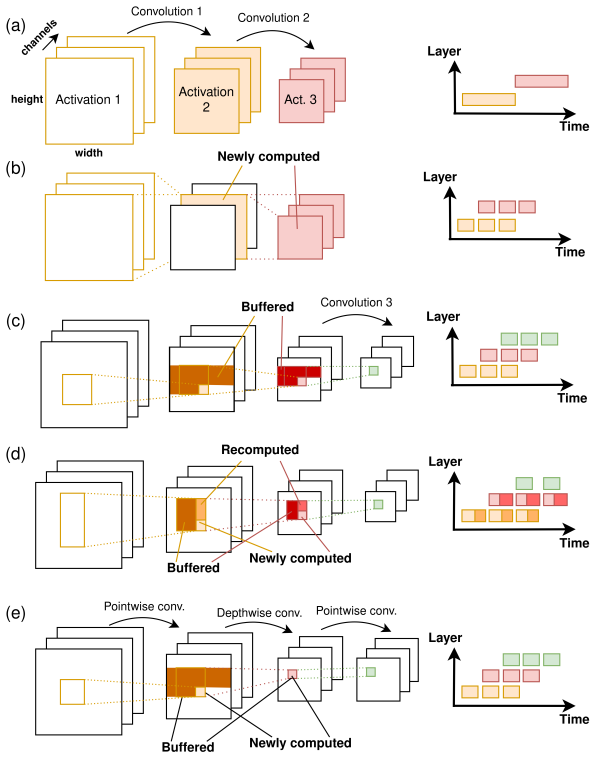

Fig. 1: Layer-by-layer dataflow (a) and fused-layer dataflows (b), (c), (d) with various tiling, scheduling, and buffering strategies. Orange, red, green: first, second, third layers. Dark to light shades: buffered, recomputed, newly computed.

The fused-layer dataflow design space is rich but has eluded thorough exploration because of the lack of a systematic representation of designs and a versatile method to evaluate designs. Prior works have explored only subsets of the fused-layer dataflow design space, constraining their tiling, scheduling, and buffering strategy to their proposed dataflow [5]–[9]. Similarly, their models can only evaluate a design following their dataflow.

To enable systematic exploration of this design space, we present LoopTree, a framework for systematically describing and evaluating latency, energy consumption, buffer capacity, and bandwidth usage of any design in this expanded design space. The expanded design space contains designs with lower latency and energy consumption. The extra degrees of freedom

especially benefit modern workloads with diverse layer types. In a case study, we explore new regions of this design space to find better designs for the MobileNet bottleneck block [10] which has pointwise and depthwise convolution layers. Our design achieves 2.5× speedup and 2× lower energy compared to an optimized layer-by-layer design. Comparing to a state-of-the-art fused-layer design, we match latency and energy while using 25% less on-chip buffer space.

II. Challenges in Fused-layer Dataflow Design

We discuss the fused-layer dataflow design space, the trade-offs involved, challenges in expanding the design space, and the insights that lead to LoopTree's solution to the challenges.

The first two axes in the design space is tiling and scheduling of operations. The choice of tiling and scheduling affects the number of layers you can fuse. Tiling across the channel dimension, you can only fuse two layers before having to store an entire feature map on-chip (feature map 3 in Fig. 1(b)). Tiling across rows and columns (see Fig. 1(c)) allows more layers to be fused. Tiling also constraints your intra-layer dataflow. Tiling across channels limits parallelism across channels. Lastly, tiling and scheduling may be applicable to multiple kinds of layers. For example, channel-wise tiling applies naturally to fully-connected layer, while tiling rows and colums is specific to convolutional layers.

The buffering strategy of a dataflow refers to which parts of the feature maps get buffered and for how long, which determines the size of on-chip buffer. A tiling and scheduling choice may lead to one obvious buffering strategy (Fig. 1(b) buffers one channel at a time), or multiple choices in buffering strategies which may or may not involve recomputation (Fig. 1(c) buffers rows and Fig. 1(d) buffers tiles with recomputation).

The choice of tiling, scheduling, and buffering strategies interact with each other and with the intra-layer dataflow choice. All these effects need to be considered when designing a fused-layer dataflow. Prior works limit the complexity of their modeling by choosing a particular combination of tiling, scheduling, and buffering strategies. This allows them to create a tailored architectural model for their proposed dataflow. To explore the larger space, we create a more versatile framework capable of capturing the interaction of these design choices for any design in our expanded design space for any workload. We create a specification language which makes it intuitive for the user to specify a set of fused-layer design choices. The framework will analyze the constraints on the intra-layer dataflow imposed by the fused-layer design. Finally, given the intra-layer dataflow, LoopTree can evaluate the latency, energy, buffer capacity, and bandwidth usage of the system.

III. Experimental Results

We implemented LoopTree and validated the accuracy of the model. To show the benefit of expanding the design space, we explore the design space to find better designs for the MobileNetv2 [10] bottleneck block.

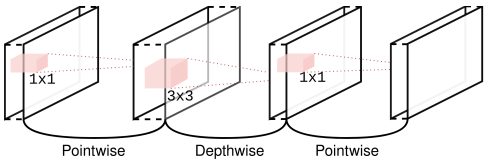

Fig. 2: Dependency pattern in a bottleneck block. The pointwise layers do not have a sliding window effect.

Metric	Layer-by-layer	Fused-layer	Norm.
Latency (kcycles)	263	115	0.4
DRAM accesses (MB)	2.1	0.6	0.3
DRAM bandw. req. (B/cycle)	8.4	5.5	0.6
Total buffer (KB)	29	11	0.4
Energy (uJ)	195	93	0.5

TABLE I: Fig. 1(e) compared to layer-by-layer design. Total buffer includes intermediate value buffer for the fused-layer design. The rightmost column shows normalized results.

A. Validation

We model a set of architectures to validate against published results. We chose [7], [5], and [8] which represent designs with various tiling, scheduling, and reuse/recompute choices. For direct comparison against the published results, we used AlexNet [11] and VGGNet [12] as workloads. LoopTree achieves 99.4% average accuracy.

B. Case Study

The expanded design space benefits workloads with diverse layer types. In this case study, we explore fused-layer designs for the MobileNetv2 [10] bottleneck block.

The bottleneck block is composed of a pointwise layer, a depthwise layer, and another pointwise layer (see Fig. 2). We want to fuse the entire block, so we will not explore the dataflow in Fig. 1(b). The fused-layer dataflow in Fig. 1(c) (from [7]) is modified for the bottleneck block. The original intra-layer dataflow would not work out of the box because it was designed for vanilla convolutions (channel and window size greater than 1). Using LoopTree, it is easy to transfer the fused-layer design and modify the intra-layer dataflow to work with the bottleneck block.

Because LoopTree allows using different buffering strategies for each layer, we can optimize further by better matching the different data dependency patterns in the layers. Because the pointwise layer does not have a sliding window effect, we can use tile buffering for the input to the last pointwise layer and row buffering for the depthwise layer. Switching to tile buffering saves 25% of on-chip intermediate feature map buffer space. When compared to an optimized layer-by-layer design, we achieve 2× lower latency, 3× lower energy, and 2× lower off-chip bandwidth (see Tab. I).

IV. Conclusion

Using a flexible and intuitive specification language along with a sophisticated model, LoopTree allows the user to express and evaluate energy, performance, buffer capacity, and bandwidth usage of fused-layer dataflow accelerator designs.

REFERENCES

[1] Chen, Yu-Hsin and Krishna, Tushar and Emer, Joel and Sze, Vivienne, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," in *IEEE International Solid-State Circuits Conference, ISSCC 2016, Digest of Technical Papers*, 2016, pp. 262–263.

[2] H. Kwon, A. Samajdar, and T. Krishna, "Maeri: Enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects," *SIGPLAN Not.*, vol. 53, no. 2, p. 461–475, mar 2018. [Online]. Available: https://doi.org/10.1145/3296957.3173176

[3] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *SIGARCH Comput. Archit. News*, vol. 42, no. 1, p. 269–284, feb 2014. [Online]. Available: https://doi.org/10.1145/2654822.2541967

[4] G. Zhou, J. Zhou, and H. Lin, "Research on nvidia deep learning accelerator," in *2018 12th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, 2018, pp. 192–195.

[5] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 14–26.

[6] M. Gao, X. Yang, J. Pu, M. Horowitz, and C. Kozyrakis, "Tangram: Optimized coarse-grained dataflow for scalable nn accelerators," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 807–820. [Online]. Available: https://doi.org/10.1145/3297858.3304014

[7] M. Alwani, H. Chen, M. Ferdman, and P. Milder, "Fused-layer cnn accelerators," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2016, pp. 1–12.

[8] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined reram-based accelerator for deep learning," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, pp. 541–552.

[9] L. Waeijen, S. Sioutas, M. Peemen, M. Lindwer, and H. Corporaal, "Convfusion: A model for layer fusion in convolutional neural networks," *IEEE Access*, vol. 9, pp. 168 245–168 267, 2021.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.