PatDNN: Achieving Real-Time DNN Execution on Mobile Devices with Pattern-based Weight Pruning

Wei Niu, Xiaolong Ma, Sheng Lin, et al

The International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS), 2020

Presenter: Dawoon Kim

Jun 09, 2020

Neural Network Acceleration Study Season #2

Contents of presentation

- **Background and Motivation**

- **Overview of PatDNN**
  - Pattern-based pruning [kernel pattern pruning, connectivity pruning]
  - Execution code generation stage

- **Evaluation**

- **Discussion and Conclusion**

Background and Motivation

Executing Deep Neural Networks (DNNs) inference on Mobile device is still challenging

- High computation
- Storage demands
- **Real-Time Performance with high accuracy** ⇨ **Weight Pruning of DNNs is proposed**

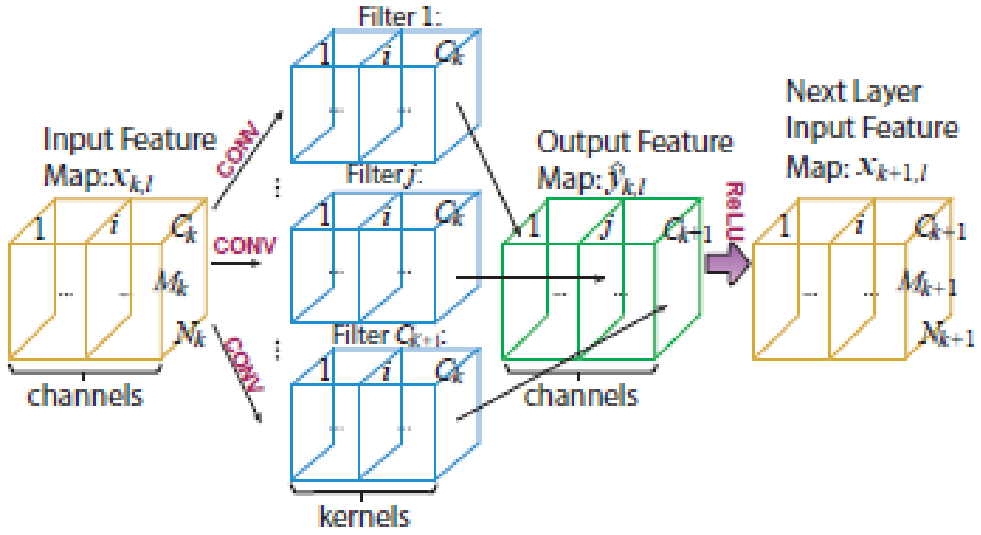

Figure 1. DNN CONV layer computation.

Background and Motivation

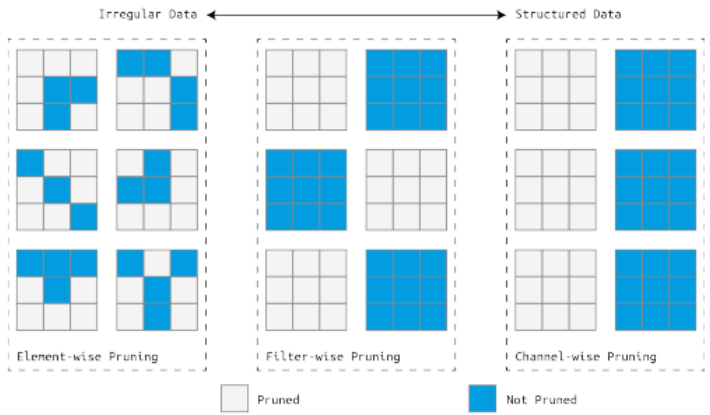

Figure 1. Illustration of pruning regularities, From (l) Irregular data pattern to (r) structured pattern.

DNN Model Compression Method : Pruning[1]

Non-Structured Pruning

- Fine-grained, high accuracy
- Not hardware friendly (Parallelism)

Structured Pruning

- Coarse-grained, hardware friendly
- Higher accuracy loss

[1] DSC: Dense-Sparse Convolution for Vectorized Inference of Convolutional Neural Networks

Background and Motivation

We rethink the design space and observe that **non-structured** and **structured** represent **two extremes** in the design space.

we naturally introduce a new dimension, **fine-grained pruning patterns inside the coarse -grained structures**, revealing a previously unknown point in design space.

→ Pattern-based Pruning

Overview of PatDNN

- Pattern-based pruning [kernel pattern pruning, connectivity pruning]

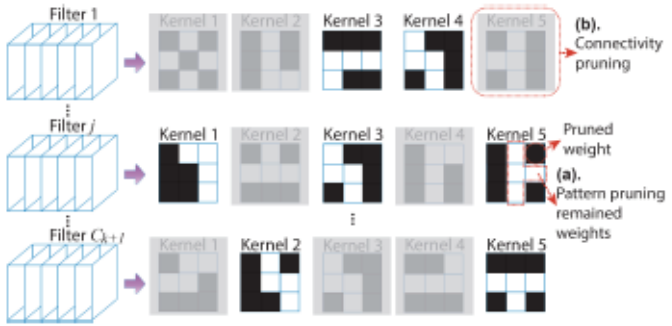

Figure 3. Illustration of (a) kernel pattern pruning on CONV kernels, and (b) connectivity pruning by removing kernels.

Connectivity Pruning (b)
: more flexible than the prior filter/channel pruning schemes that remove whole filters/channels, thereby achieving higher accuracy

Kernel Pattern Pruning (a)
: For each kernel, a fixed number of weights are pruned, and the remaining weights (white cells) form specific "kernel patterns"

+ At Compiler level, the pre-defined pattern allows compiler to re-order and generate codes at filter and kernel level so that kernels with same pattern can be grouped for consecutive executions to maximize instruction-level parallelism.

Overview of PatDNN

- Pattern-based pruning [kernel pattern pruning, connectivity pruning]

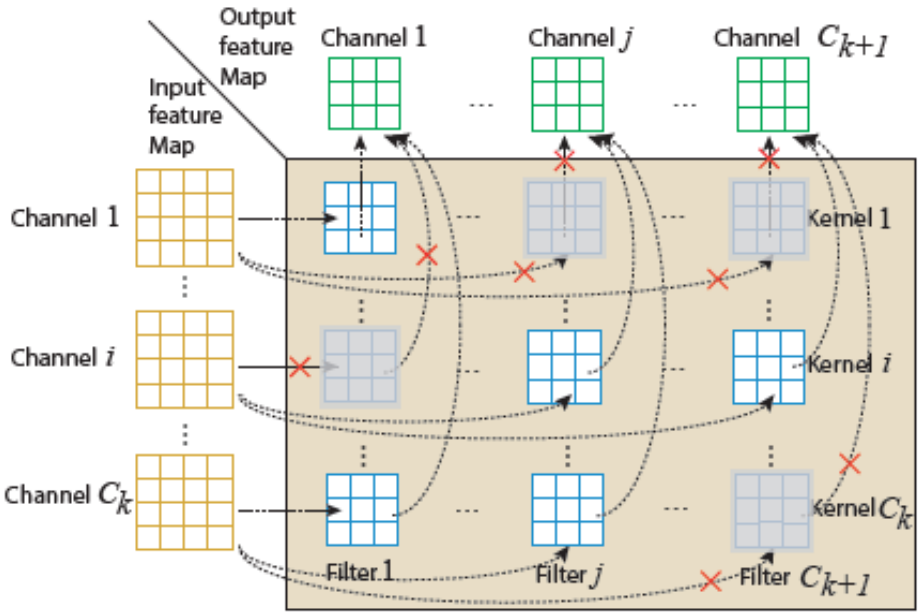

Figure 4. Illustration of connectivity pruning.

Connectivity Pruning

The key insight is **to cut the connections between certain input and output channels**, which is equivalent to removal of corresponding kernels.

This method is proposed for overcoming the limited weight pruning rate by kernel pattern pruning.

Overview of PatDNN

- Pattern-based pruning [kernel pattern pruning, connectivity pruning]

Table 2. Qualitative comparison of different pruning schemes on accuracy and speedup under the same pruning rate.

Pruning Scheme	Accuracy				Hardware Speedup			
	Highest	Minor Loss	Moderate Loss	Highest Loss	Highest	High	Moderate	Minor
Non-structured	X							X
Filter/Channel				X	X			
Pattern	**X**				**X**			
Connectivity		**X**				**X**		

PatDNN can achieve the benefits of both non-structured and structured pruning..

The key enabler to achieving this goal is to leverage compiler to maintain the efficiency of structured pruning based on kernel pattern and connectivity pruning.

Overview of PatDNN

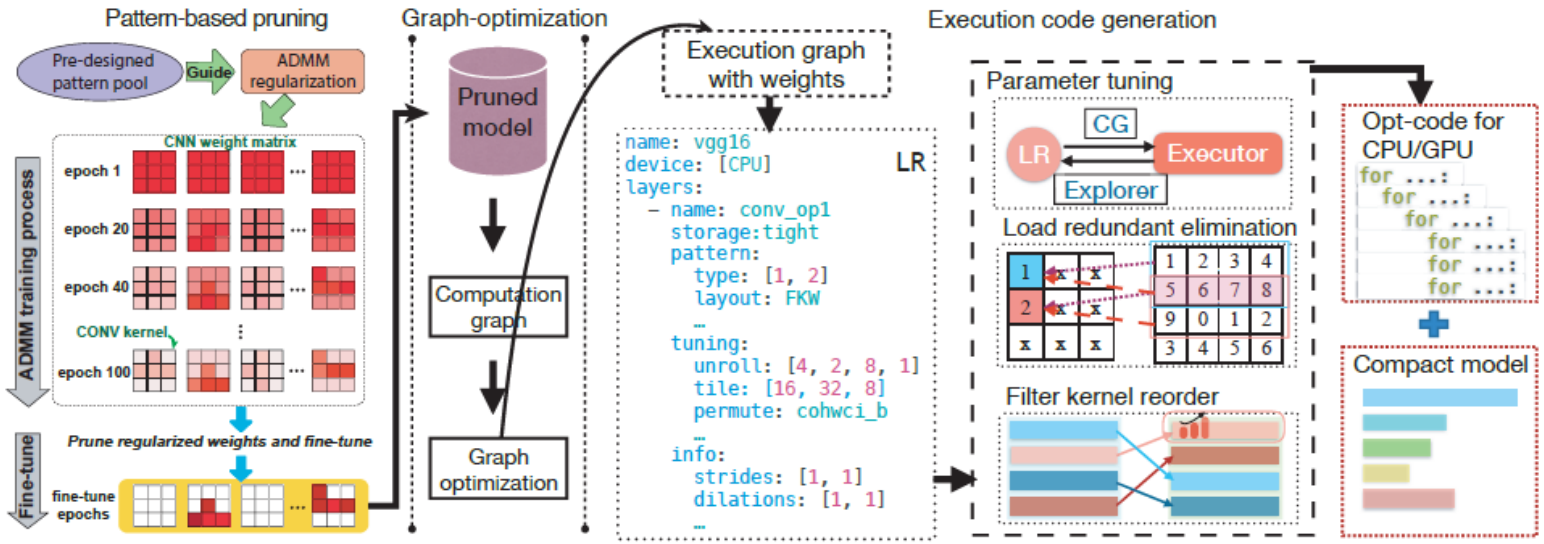

Figure 5. Overview of PatDNN acceleration framework.

pattern-based training stage

which performs kernel pattern and connectivity pru ning with an extended ADMM solution framework.

execution code generation stage

which performs multiple effective optimizations based on the patterns.

Evaluation

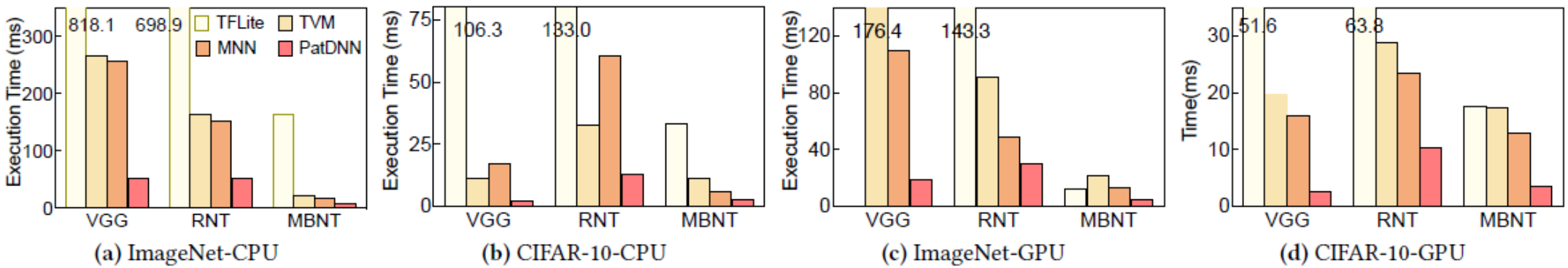

Figure 12. Overall performance: x-axis: different trained DNN models; y-axis: average DNN inference execution time on a single input.

- HW : Samsung Galaxy S10 with Qualcomm Sanpdragon 855
- CPU : Qualcomm Kryo 485 Octa-core
- GPU : Qualcomm Adreno 640

On CPU,
PatDNN achieves 12.3x to 44.5x speed up over TFLite, 2.4x to 5.1x over TVM, and 1.9x to 7.1x over MNN

On GPU,
PatDNN achieves 2.5x to 20x speed up over TFLite, 2.8x to 11.4x over TVM, and 1.6x to 6.2x over MNN

VGG-16 (VGG), ResNet-50 (RNT), Mobile-Net-V2 (MBNT)

Evaluation

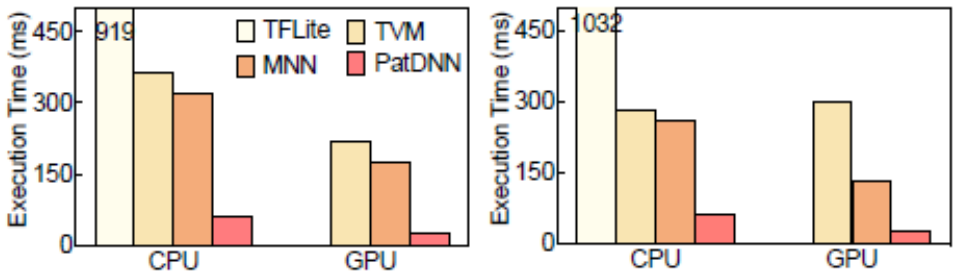

Figure 18. Portability study: performance on two other platforms.

PatDNN is also evaluated on two other platforms to confirm its portability. Figure 18 shows the result. On these platforms, PatDNN also outperforms other frameworks.

Discussion and Conclusion

Generality
: it is a promising research direction to improve PatDNN's portability by incorporating it with TVM th at emphasizes the DNN execution on varied computing devices.

Dense vs. Sparse DNNs
: General end-to-end DNN inference acceleration frameworks like TFLite, TVM, and MNN do not sup port sparse DNN execution.

From this perspective, PatDNN opens a new door to accelerate DNN execution with a compression/c ompileroptimization co-design. With such co-design, sparse (or compressed) DNN execution become s a more promising solution in resource-constraint environments than dense DNN.

VGG-16 (VGG), ResNet-50 (RNT), Mobile-Net-V2 (MBNT)

Discussion and Conclusion

- PatDNN, an end-to-end framework to achieve real-time DNN execution on Mobile devices.
- PatDNN consists of two stages, a pattern-based pruning stage and an optimized execution code gen stage.
- This design allows PatDNN to benefit from both high accuracy and hardware efficiency.

Thank you