Analytical Characterization and Design Space Exploration for Optimization of CNNs

Rui Li, Yufan Xu, Aravind Sukumaran-Rajam, Atanas Rountev, and P. Sadayappan
University of Utah, Washington State University, Ohio State University

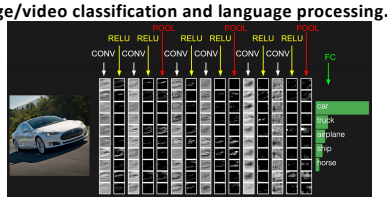
ASPLOS '21

ABSTRACT

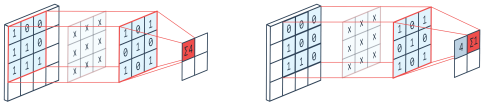
Moving data through the memory hierarchy is a fundamental bottleneck that can limit the performance of convolutional neural networks (CNNs). Loop level optimization, including loop tiling and loop permutation, are fundamental transformations to reduce data movement. However, the search space for finding the best loop-level optimization configuration is explosively large. This paper develops an analytical modeling approach for finding the best loop-level optimization configuration for CNNs on multi-core CPUs. Experimental evaluation shows that this approach achieves comparable or better performance than state-of-the-art libraries and auto-tuning based optimizers for CNNs.

INTRODUCTION

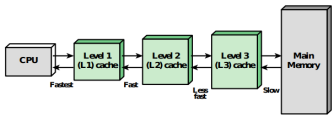
- Convolutional Neural Networks (CNNs) is widely used in image/video classification and language processing.



- Expression: Out[n,k,h,w] += In[n,c,h+r,w+s]*Ker[k,c,r,s]



- Multi-level memory hierarchy is common in modern computer systems



- Places work in the context of the literature
- How fast can modern computers process convolution?
- Wooden Barrel: shortest bar decides performance
- Bars: ALU speed, data movement between caches
- Way to optimize?

METHODS

- Way to improve data movement between caches: Loop Tiling and Loop reordering

```
// Ni/Nj/Nk are perfect multiples of Ti/Tj/Tk
for(it = 0; it < Ni; it+=Ti)
 for(jt = 0; jt < Nj; jt+=Tj)
  for(kt = 0; kt < Nk; kt+=Tk)
   for(i = 0; i < Ti; i++)
    for(j = 0; j < Tj; j++)
     for(k = 0; k < Tk; k++)
       C[i+it][j+jt]+=
       A[i+it][k+kt]*B[k+kt][j+jt];
```

Listing 1: Single-level tiled matrix multiplication

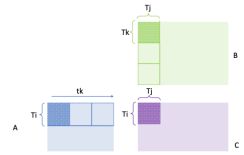


Figure 2: Data reuse in tiled matrix multiplication

- Way to improve ALU speed: Micro-kernel that utilize SIMD instruction and Maximum Instruction level parallelism

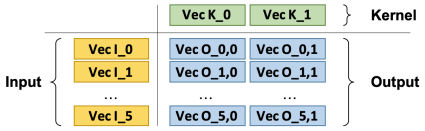


Figure 4: Conceptual view of outer product scheme

- Put it all together: Systematically model the performance and find the best optimization configuration

Figure 1: MOpt Overview

REFERENCES

- Li et. al. Analytical Characterization and Design Space Exploration for Optimization of CNNs. ASPLOS 2021
- Chen et. al. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. OSDI 2018
- Intel oneDNN library, Intel Corporation 2020

RESULTS

- Experiments results on i7-9000k, over benchmarks consist of conv2d in ResNet, MobileNet, and Yolo-9000
- Validation: How close does the model predicted optimal configuration to the actual best configuration, under a given grid search space?

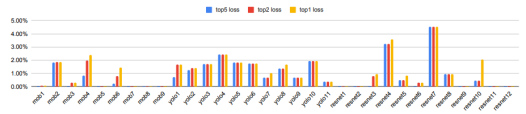
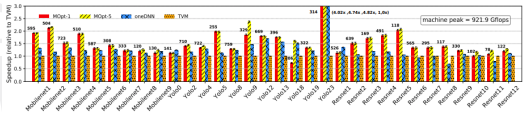


Figure 5: Model prediction performance loss over 100 grid sampling for Mobilenet, Yolo-9000, and Resnet-18 on i7-9700K

- Comparison: How good performance does the model achieve, comparing to the state of art library (Intel oneDNN) and machine learning compiler (AutoTVM)?



- Discussion: Strength/limitations of the state-of-art

	Auto tuning	Micro Kernel	Design Space Exploration
oneDNN	×	Highly optimized	Minimal
TVM	✓	NA	Limited
MOpt	×	Not highly optimized	Comprehensive

Table 2: Strengths/limitations of oneDNN, TVM and MOpt

CONCLUSIONS

- Proposed a new approach to overcome the design-space explosion problem that has thwarted effective compile-time modeling and optimized code generation for CNNs..
- Although the space of possible configurations is extremely large, we devise an effective analytical modeling approach to search in this space.
- Experimental results demonstrate that achieved performance is superior to code generated by TVM and can be comparable to or better than Intel's oneDNN.

ACKNOWLEDGEMENTS

- This work was supported in part by the U.S. National Science Foundation through awards 1946752, 1919122 and 2018016.