AutoGraph: Optimizing DNN Computation Graph for Parallel GPU Kernel Execution

Yuxuan Zhao, Qi Sun, Zhuolun He, Yang Bai, Bei Yu

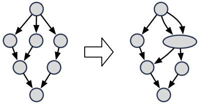
The Chinese University of Hong Kong
`{yxzhao21,byu}@cse.cuhk.edu.hk`

Feb. 07–14, 2023

Introduction

DNN Deployment Stack

DL Frameworks



High-level Computation Graph Optimization

```
for jc.0 in range(None):
  for k.0 in range(None):
    for k.1 in range(None):
for k.2 in range(None):
  for i.0 in range(None):
    for j.1 in range(None):
      C = ...
```

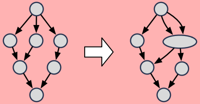
Low-level Tensor Program Optimization

Kernel Dispatching, Kernel Submission

Runtime Optimization

DNN Deployment Stack

DL Frameworks



High-level Computation Graph Optimization

```
for jc.0 in range(None):
    for k.0 in range(None):
        for k.1 in range(None):
for k.2 in range(None):
  for i.0 in range(None):
        for j.1 in range(None):
          C = …
```

Low-level Tensor Program Optimization

Kernel Dispatching, Kernel Submission

Runtime Optimization

Prior Arts

- Equivalent Graph Substitution:
  - TASO[1] takes operator definitions and specifications, then automatically generates and verifies graph substitutions.
- Parallel GPU Kernel Launch:
  - IOS[2] divides the computation into different stages and uses DP to find the optimized launch schedule.
  - Nimble[3] supports parallel kernel launch for the whole model and leverages the AOT scheduler to minimize scheduling overhead.

[1] Zhihao Jia et al. (2019). "TASO: optimizing deep learning computation with automatic generation of graph substitutions". In: *Proc. SOSP*.

[2] Yaoyao Ding et al. (2021). "IOS: Inter-Operator Scheduler for CNN Acceleration". In: *Proc. MLSys*.

[3] Woosuk Kwon et al. (2020). "Nimble: Lightweight and Parallel GPU Task Scheduling for Deep Learning". In: *Proc. NeurIPS*.

Prior Arts

- Equivalent Graph Substitution:
  - TASO[1] takes operator definitions and specifications, then automatically generates and verifies graph substitutions.
- Parallel GPU Kernel Launch:
  - IOS[2] divides the computation into different stages and uses DP to find the optimized launch schedule.
  - Nimble[3] supports parallel kernel launch for the whole model and leverages the AOT scheduler to minimize scheduling overhead.

Can we bridge the gap between them?

[1] Zhihao Jia et al. (2019). "TASO: optimizing deep learning computation with automatic generation of graph substitutions". In: *Proc. SOSP*.

[2] Yaoyao Ding et al. (2021). "IOS: Inter-Operator Scheduler for CNN Acceleration". In: *Proc. MLSys*.

[3] Woosuk Kwon et al. (2020). "Nimble: Lightweight and Parallel GPU Task Scheduling for Deep Learning". In: *Proc. NeurIPS*.

Challenges

Huge graph optimization search space

- Modern DNN models can be complex and large.
- The number of available graph substitutions are huge.

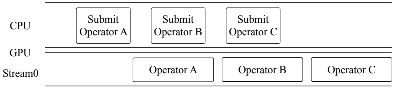
Challenges

Huge graph optimization search space

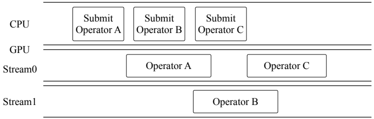
- Modern DNN models can be complex and large.
- The number of available graph substitutions are huge.

Inter-operator parallelism is ignored

- Previous graph optimization methods focus on sequential kernel launch.
- Lack runtime information.



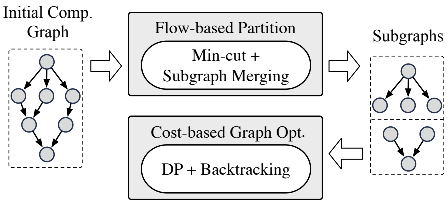
(a) Sequential kernel launch



(b) Parallel kernel launch

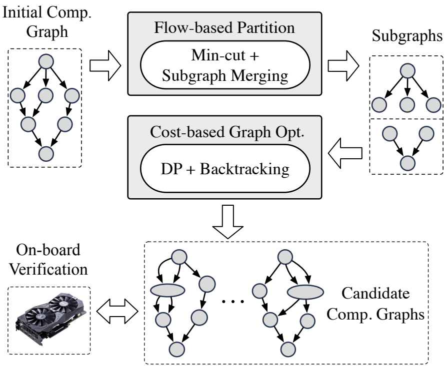
Details of AutoGraph

Overview of AutoGraph



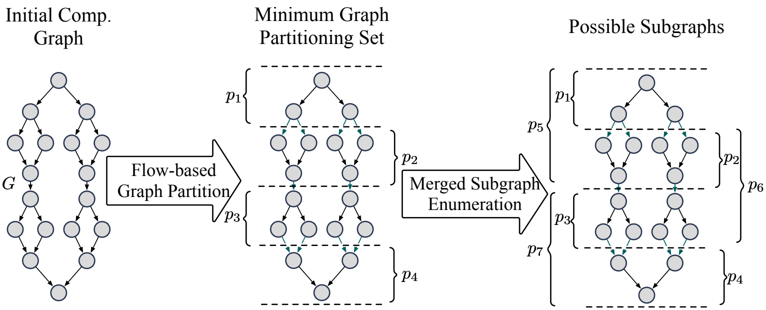
- Tackle huge search space:
  - Flow-based graph partition method.
  - Dynamic programming + backtracking search.
- Tackle inter-operator parallelism:
  - Customized cost function.
  - Runtime information based on GPU Multi-Stream.

Overview of AutoGraph



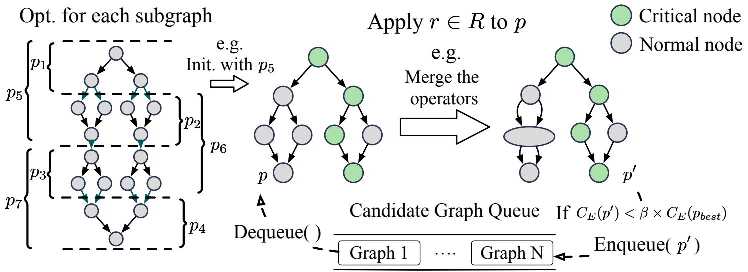
- Tackle huge search space:
  - Flow-based graph partition method.
  - Dynamic programming + backtracking search.
- Tackle inter-operator parallelism:
  - Customized cost function.
  - Runtime information based on GPU Multi-Stream.

Flow-based Graph Partition



- The node capacity is defined as the number of available graph substitutions.
- The entire computation graph is recursively split into independent subgraphs by its minimum cut.
- Adjacent subgraphs are merged to form new subgraphs.

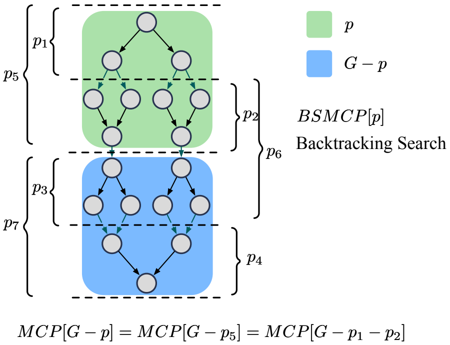
Backtracking Search via Mixed Critical Path Cost



- Backtracking search is used to optimize each subgraph.
- We use the mixed critical path cost in Equation 1 as the selection criteria.

$$\begin{aligned} C_E &= \alpha \sum_{v \in V_c} cost(v) + \sum_{v \in V} cost(v) \\ &= (1+\alpha) \sum_{v \in V_c} cost(v) + \sum_{v \in V - V_c} cost(v). \end{aligned} \tag{1}$$

DP-based Optimization Solution Search

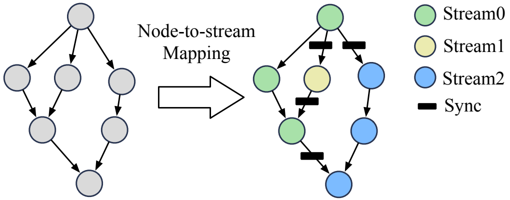


A transition state in our dynamic programming + backtracking search.

- We observe that different graph partitioning sequences share the same sub-sequence.
- The problem can be solved by Equation 2.

$$MCP[G] = \min_{p} \left(MCP[G-p] + BSMCP[p] \right). \tag{2}$$

On-board Verification



GPU stream assignment.

- The operator nodes on different branches are assigned to different streams with proper synchronization events inserted.
- CUDA Graph is used to launch the computation graph.
- We sample the top-k candidates for on-board verification each time.

Experimental Results

Experimental Settings

- Platform:
  - NVIDIA GeForce RTX 2080Ti GPU.
  - CUDA 11.0, cuDNN 8.0.5, and PyTorch 1.7.

Experimental Settings

- Platform:
  - NVIDIA GeForce RTX 2080Ti GPU.
  - CUDA 11.0, cuDNN 8.0.5, and PyTorch 1.7.
- Seven modern DNNs are benchmarked:

Table: DNN Models Used in Our Experiments.

Type	Name	block#	input shape
CNN	Inception-v3	11	[1, 3, 299, 299]
	ResNet-50	16	[1, 3, 224, 224]
	ResNeXt-50	16	[1, 3, 224, 224]
	NasNet-A	18	[1, 3, 224, 224]
	NasNet-Mobile	12	[1, 3, 224, 224]
RNN	RNNTC-SRU	10	[1×10, 1024]
Transformer	BERT	8	[16×64, 1024]

End-to-end Model Inference Latency

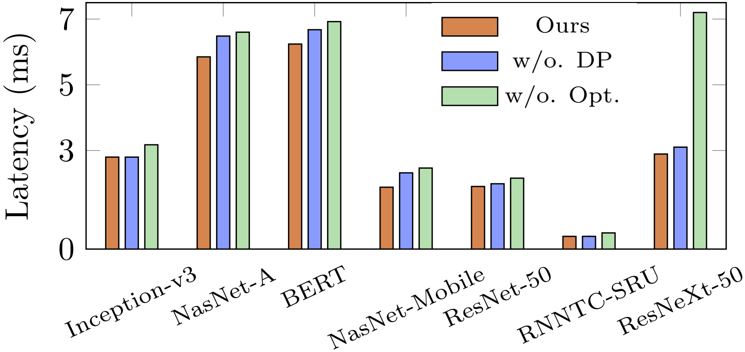
Table: Model inference latency results (ms).

Model	JIT	TASO+JIT	IOS	Nimble	TASO+Nimble	Ours
Inception-v3	8.839	7.819	3.788	3.174	2.928	**2.799**
ResNet-50	4.566	4.554	3.284	2.144	1.988	**1.905**
ResNeXt-50	7.540	7.369	3.056	7.708	5.933	**2.892**
NasNet-A	13.891	10.843	9.583	6.483	13.086	**5.850**
NasNet-Mobile	10.155	8.085	3.821	2.320	6.540	**1.883**
RNNTC-SRU	1.496	1.307	-	0.486	**0.387**	**0.387**
BERT	11.011	9.026	-	6.923	6.473	**6.240**

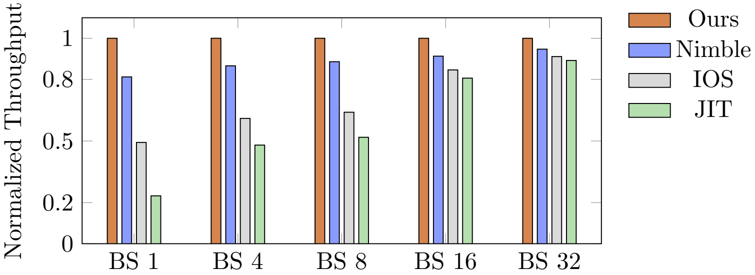
- Compare with TASO, our method achieves speedup ranging from 1.04× to 3.47× on parallel kernel launch framework.
- Compare with IOS and Nimble, our method achieves speedup ranging from 1.06× to 1.26× on the benchmark models.

Ablation Studies on Different Settings

- "w/o. Opt." means directly measuring the initial computation graph.
- "w/o. DP" means directly using the minimum partitioning set without our DP-based method.



Ablation Studies on Different Batch Sizes



The normalized throughput comparisons of different frameworks on various batch sizes for NasNet-Mobile.

- A larger batch size provides more intra-operator parallelism.
- We can still exploit inter-operator parallelism and graph optimization to further improve the inference performance.

Conclusion

Conclusion

- Existing graph optimization methods fails to utilize inter-operator parallelism and thus impair system capability within a parallel kernel launch framework.
- We propose AutoGraph to bridge the gap. Experimental results demonstrate that our method achieves up to 3.47× speedup over previous arts.
- Moreover, AutoGraph outperforms state-of-the-art parallel kernel launch frameworks by up to 1.26×.

THANK YOU!