GTuner: Tuning DNN Computations on GPU via Graph Attention Network

Qi Sun$^1$, Xinyun Zhang$^1$, Hao Geng$^2$, Yuxuan Zhao$^1$, Yang Bai$^1$, Haisheng Zheng$^3$, Bei Yu$^1$

$^1$The Chinese University of Hong Kong $^2$ShanghaiTech University $^3$SmartMore

{qsun,byu}@cse.cuhk.edu.hk

July 14, 2022
Background
Learning-based Learning System

Frameworks

High-level data flow graph and optimizations

Hardware aware Search Space of Optimized Tensor Programs

Machine Learning based Program Optimizer

Hardware

• TVM
Some Concepts

- Computational graph
- Subgraph
Some Concepts

- Computational graph
- Subgraph

- Graph Optimization
  - Operation fusion
  - Constant folding
  - Data layout transformation
  - ...

\[ \text{Conv-3} \rightarrow \text{Conv-3} \rightarrow \text{Max-Pool} \]

\[ \text{Conv-3} \rightarrow \text{ReLU} \rightarrow \text{Conv-3} \]

\[ \text{Conv-1} \rightarrow \text{Fire} \rightarrow \text{Conv-1} \rightarrow \text{Conv-1} \rightarrow \text{Conv-3} \]

\[ \text{Concat} \]
Some Concepts

- Sketch: each subgraph has many sketches (templates)
- Annotation: each sketch has many annotations (groups of parameter values)
• Sketch: each subgraph has many sketches (templates)

• Annotation: each sketch has many annotations (groups of parameter values)

```python
Generated Kernel Code Sketch:
[Placeholder: A, B
  for i.0 in range(None):
    for j.0 in range(None):
      for ic.2 in range(None):
        for jc.2 in range(None):
          for k.0 in range(None):
            for k.1 in range(None):
              for i.3 in range(None):
                for j.3 in range(None):
                  C = ... ]

Annotation 1:
[Placeholder: A, B
  for i.0 in range(32):
    for j.0 in range(64):
      for ic.2 in range(16):
        for jc.2 in range(4):
          for k.1 in range(16):
            for ... i.3 in range(None):
              for j.3 in range(None):
                C = ... ]

Annotation 2:
[Placeholder: A, B
  for i.0 in range(2):
    for j.0 in range(1024):
      for ic.2 in range(32):
        for jc.2 in range(2):
          for k.0 in range(2):
            for k.1 in range(8):
              for k.2 in range(4):
                for i.3 in range(4):
                  for j.3 in range(4):
                    C = ... ]
```
• Sketch: each subgraph has many sketches (templates)

• Annotation: each sketch has many annotations (groups of parameter values)

<table>
<thead>
<tr>
<th>Generated Kernel Code Sketch:</th>
<th>Annotation 1:</th>
<th>Annotation 2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>for i.0 in range(32):</td>
<td>for i.0 in range(32):</td>
<td>for i.0 in range(2):</td>
</tr>
<tr>
<td>for j.0 in range(64):</td>
<td>for j.0 in range(64):</td>
<td>for j.0 in range(1024):</td>
</tr>
<tr>
<td>for ic.2 in range(16):</td>
<td>for ic.2 in range(16):</td>
<td>for ic.2 in range(32):</td>
</tr>
<tr>
<td>for jc.2 in range(4):</td>
<td>for jc.2 in range(4):</td>
<td>for jc.2 in range(2):</td>
</tr>
<tr>
<td>for k.0 in range(2):</td>
<td>for k.0 in range(2):</td>
<td>for k.0 in range(2):</td>
</tr>
<tr>
<td>for k.1 in range(16):</td>
<td>for k.1 in range(16):</td>
<td>for k.1 in range(2):</td>
</tr>
<tr>
<td>for k.2 in range(2):</td>
<td>for k.2 in range(2):</td>
<td>for k.2 in range(4):</td>
</tr>
<tr>
<td>for i.3 in range(2):</td>
<td>for i.3 in range(2):</td>
<td>for i.3 in range(4):</td>
</tr>
<tr>
<td>for j.3 in range(2):</td>
<td>for j.3 in range(2):</td>
<td>for j.3 in range(4):</td>
</tr>
<tr>
<td>C = ... ]</td>
<td>C = ... ]</td>
<td>C = ... ]</td>
</tr>
</tbody>
</table>

• Target: the optimization target is to find the optimal annotations for each subgraph in the deep learning model
Deep Learning Deployment Methods for GPU

Previous Arts

- **Expression**: $e$
- **Schedule Space**: $S_e$
- **Exploration Module**
  - History data $D$
  - Update $e, s$
  - Experiment feedback
- **Code Generator**: $x = g(e, s)$
- **Hardware Environment**: $f(x)$
- **Cost Model**: $\hat{f}(x)$
- **Objective Function**
Previous Arts

- AutoTVM (Chen et al. 2018)
Previous Arts

- AutoTVM (Chen et al. 2018)
- CHAMELEON: reinforcement learning + adaptive sampling (Ahn et al. 2020)
Previous Arts

- AutoTVM (Chen et al. 2018)
- CHAMELEON: reinforcement learning + adaptive sampling (Ahn et al. 2020)
- GGA: guided genetic algorithm (Mu et al. 2020)
Deep Learning Deployment Methods for GPU

Previous Arts

- AutoTVM (Chen et al. 2018)
- CHAMELEON: reinforcement learning + adaptive sampling (Ahn et al. 2020)
- GGA: guided genetic algorithm (Mu et al. 2020)
- DGP-TL: deep Gaussian process + transfer learning (Q. Sun et al. 2021)
Deep Learning Deployment Methods for GPU

Previous Arts

- AutoTVM (Chen et al. 2018)
- CHAMELEON: reinforcement learning + adaptive sampling (Ahn et al. 2020)
- GGA: guided genetic algorithm (Mu et al. 2020)
- DGP-TL: deep Gaussian process + transfer learning (Q. Sun et al. 2021)
- Ansor: program sampler, sketch, annotation (Zheng et al. 2020)
The *structural* information is not used

- **Structural features**: node types, node connectivities, graph topology
- **Rely only on statistical features**
- **Unable to identify task information and distinguish different tasks**
Challenges

The **structural** information is not used

- Structural features: node types, node connectivities, graph topology
- Rely only on *statistical* features
- Unable to identify task information and distinguish different tasks

The complicated relationships between the features are not considered

- Feature items in the statistical feature vectors are treated equally, despite their physical meanings and relationships
  - XGBoost
  - MLP
  - …
Details of GTuner
• Extract structural and statistical features for the annotations
• Graph attention network (GAT): graph neural network, and multi-head self-attention
• Graph optimization
  • represent subgraphs as Intermediate Representations (IRs)
- Directed Acyclic Graph (DAG) analyzer
  - analyze the IRs to construct the optimized subgraphs
• Generate and sample codes
• Extract structural and statistical features
• Performance learning via Graph Attention Network (GAT)
• Genetic-based iterative optimization

DNN Model

Graph Optimization

Intermediate Representations (IR)

DAG Parser & Analyzer

Kernel Code Sketches

Sample Code Annotations

Optimized Computational Subgraphs

Generated Kernel Code Sketch 1: [Placeholder: A, B]

Generated Kernel Code Sketch 2: [Placeholder: A, B]

GTuner Flow

Genetic Algo. Optimizer

Optimal Code

Extract Code Statistical Features

Extract Graph Structural Features (GNN)

MHSA

MLP

GPU
Graph Attention Network (GAT)

- Define a graph neural network to extract the structural features.
• Define a graph neural network to extract the structural features.
• Use structural features to enhance statistical features.
• The concatenated features are the inputs to the multi-head self-attention.
Graph Network Module

- Graph Neural Network (Morris et al. 2019)

\[
x_i^k = W_1^{k-1} x_i^{k-1} + W_2^{k-1} \sum_{v_t \in N(v_i)} x_t^{k-1},
\]
Multi-head Attention (Vaswani et al. 2017)

Multi-head Attention

Scaled Dot-Product Attention:

$$\text{Attn}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

$$H_i = \text{Attn}(QW_Qi, KW_Ki, VW_Vi), \quad \text{MHA}(Q, K, V) = \text{Concat}(H_1, H_2, \cdots, H_h)W_O$$
Multi-head Attention (Vaswani et al. 2017)

- scaled dot-product attention:
  \[ \text{Attn}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \]
Multi-head Attention (Vaswani et al. 2017)

- scaled dot-product attention:
\[
\text{Attn}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V
\]

\[
H_i = \text{Attn} \left( QW_{iQ}, KW_{iK}, VW_{iV} \right),
\]
Multi-head Attention (Vaswani et al. 2017)

- scaled dot-product attention:
  \[ \text{Attn}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \]

\[ H_i = \text{Attn} \left( QW_i^Q, KW_i^K, VW_i^V \right), \]

\[ \text{MHA}(Q, K, V) = \text{Concat}(H_1, H_2, \cdots, H_h)W^O \]
Multi-head Self-attention Module

- Input vector $x$ with length $l$
- Reshape: $x^R$ with shape $h \times \frac{l}{h}$
- Number of heads: $h$
- $x^R$ is used as $Q$, $K$, and $V$

Self-attention

$$\text{SelfAttn} \left( x^R W^Q_i, x^R W^K_i, x^R W^V_i \right)$$
Experimental Results
Experimental Settings

- Platform
  - Nvidia GeForce RTX 3090 (Ampere architecture, SM86)
  - CUDA Driver 11.4, PyTorch 1.10, and TVM 0.8-dev
Experimental Settings

- **Platform**
  - Nvidia GeForce RTX 3090 (Ampere architecture, SM86)
  - CUDA Driver 11.4, PyTorch 1.10, and TVM 0.8-dev
- **Training set:** about 170000 annotations (collected from Inception-V3 and VGG-11)
Experimental Settings

• Platform
  • Nvidia GeForce RTX 3090 (Ampere architecture, SM86)
  • CUDA Driver 11.4, PyTorch 1.10, and TVM 0.8-dev

• Training set: about 170000 annotations (collected from Inception-V3 and VGG-11)

• Model structure:
  • two WL-GCN layers
  • a mean pooling layer
  • a concatenation layer
  • a fully-connected layer (512)
  • a four-head multi-head self-attention layer
  • an MLP module (output dimensions: 200-100-20-1)
• Spectral graph convolution (SpecGCN, Kipf and Welling 2017)
  • a first-order approximation of localized spectral filters on the graphs
  • learn filters to represent the nodes in the Fourier domain

Experiments – Ablation Studies on Graph Neural Network

<table>
<thead>
<tr>
<th>Method</th>
<th>Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18</td>
<td>1.073</td>
</tr>
<tr>
<td>Ansor GTuner</td>
<td>0.923</td>
</tr>
<tr>
<td>SpecGCN</td>
<td>1.016</td>
</tr>
<tr>
<td>MaskGAT</td>
<td>1.105</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>1.168</td>
</tr>
</tbody>
</table>
Experiments – Ablation Studies on Graph Neural Network

• Spectral graph convolution (SpecGCN, Kipf and Welling 2017)
  • a first-order approximation of localized spectral filters on the graphs
  • learn filters to represent the nodes in the Fourier domain

• masked attention convolution (MaskGAT, Veličković et al. 2018)
  • introduces the attention-based architecture to compute the hidden
    representations of the nodes by using masks during information aggregation
Experiments – Ablation Studies on Graph Neural Network

• Spectral graph convolution (SpecGCN, Kipf and Welling 2017)
  • a first-order approximation of localized spectral filters on the graphs
  • learn filters to represent the nodes in the Fourier domain

• masked attention convolution (MaskGAT, Veličković et al. 2018)
  • introduces the attention-based architecture to compute the hidden representations of the nodes by using masks during information aggregation

• GraphSAGE (Hamilton, Ying, and Leskovec 2017)
  • generates embeddings by sampling and aggregating features from a node’s local neighborhood to improve the generalization abilities to unseen nodes
• Spectral graph convolution (SpecGCN, Kipf and Welling 2017)
  • a first-order approximation of localized spectral filters on the graphs
  • learn filters to represent the nodes in the Fourier domain

• masked attention convolution (MaskGAT, Veličković et al. 2018)
  • introduces the attention-based architecture to compute the hidden
    representations of the nodes by using masks during information aggregation

• GraphSAGE (Hamilton, Ying, and Leskovec 2017)
  • generates embeddings by sampling and aggregating features from a node’s local
    neighborhood to improve the generalization abilities to unseen nodes

<table>
<thead>
<tr>
<th></th>
<th>ResNet-18</th>
<th>Ansor</th>
<th>GTuner</th>
<th>SpecGCN</th>
<th>MaskGAT</th>
<th>GraphSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency (ms)</td>
<td>1.073</td>
<td>0.923</td>
<td>1.016</td>
<td>1.105</td>
<td>1.168</td>
<td></td>
</tr>
</tbody>
</table>
Experiments – Ablation Studies on Model Structure

- GNN + MHSA
- MHSA
- GNN + MLP

Table: Performance without GNN or MHSA

<table>
<thead>
<tr>
<th>ResNet-18</th>
<th>MHSA</th>
<th>GNN + MLP</th>
<th>GTuner (GNN + MHSA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency (ms)</td>
<td>0.963</td>
<td>1.121</td>
<td>0.923</td>
</tr>
</tbody>
</table>
Trials of the genetic-based optimization

- Trials Per Subgraph
- Latency (ms)
- Ansor
- GTuner

- Trials Per Subgraph
- Ratios of Latency (%)
- Ansor
- GTuner

- Subgraphs
- Latency (ms)
- Ansor

- Subgraphs
- GFLOPS (%)
- GTuner

Ablation Studies – ResNet-18
Ablation Studies – ResNet-18

Trials of the genetic-based optimization

Performance of Subgraphs
## Table: End-to-end Model Inference Latency (ms)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18</td>
<td>27.180</td>
<td>4.119</td>
<td>1.056</td>
<td>1.073</td>
<td>0.963</td>
<td>0.923 (13.98%)</td>
</tr>
<tr>
<td>ResNet-34</td>
<td>48.988</td>
<td>5.929</td>
<td>1.180</td>
<td>0.968</td>
<td>0.907</td>
<td>0.872 (9.92%)</td>
</tr>
<tr>
<td>SqueezeNet</td>
<td>16.658</td>
<td>3.648</td>
<td>0.311</td>
<td>0.207</td>
<td>0.201</td>
<td>0.197 (4.83%)</td>
</tr>
<tr>
<td>MobileNet</td>
<td>30.324</td>
<td>6.972</td>
<td>0.513</td>
<td>0.242</td>
<td>0.252</td>
<td>0.227 (6.20%)</td>
</tr>
</tbody>
</table>

* Ratios are performance improvements compared with Ansor.
## End-to-end Performance

### Table: End-to-end Model Inference Latency (ms)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18</td>
<td>27.180</td>
<td>4.119</td>
<td>1.056</td>
<td>1.073</td>
<td>0.963</td>
<td>0.923 (13.98%)</td>
</tr>
<tr>
<td>ResNet-34</td>
<td>48.988</td>
<td>5.929</td>
<td>1.180</td>
<td>0.968</td>
<td>0.907</td>
<td>0.872 (9.92%)</td>
</tr>
<tr>
<td>SqueezeNet</td>
<td>16.658</td>
<td>3.648</td>
<td>0.311</td>
<td>0.207</td>
<td>0.201</td>
<td>0.197 (4.83%)</td>
</tr>
<tr>
<td>MobileNet</td>
<td>30.324</td>
<td>6.972</td>
<td>0.513</td>
<td>0.242</td>
<td>0.252</td>
<td>0.227 (6.20%)</td>
</tr>
</tbody>
</table>

+ Ratios are performance improvements compared with Ansor.

### Table: Time Costs (minutes) of the Optimization Processes

<table>
<thead>
<tr>
<th>Model</th>
<th>AutoTVM</th>
<th>Ansor</th>
<th>MHSA</th>
<th>GTuner</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18</td>
<td>65.22</td>
<td>45.57</td>
<td>45.95</td>
<td>46.94</td>
</tr>
<tr>
<td>ResNet-34</td>
<td>54.86</td>
<td>46.66</td>
<td>48.89</td>
<td>50.71</td>
</tr>
<tr>
<td>SqueezeNet</td>
<td>63.90</td>
<td>43.53</td>
<td>44.40</td>
<td>45.91</td>
</tr>
<tr>
<td>MobileNet</td>
<td>61.60</td>
<td>42.88</td>
<td>43.80</td>
<td>44.20</td>
</tr>
</tbody>
</table>


THANK YOU!