DESIGN AUTOMATION CONFERENCE

59

JULY 10 - 14, 2022

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

DESIGN AUTOMATION CONFERENCE 59

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

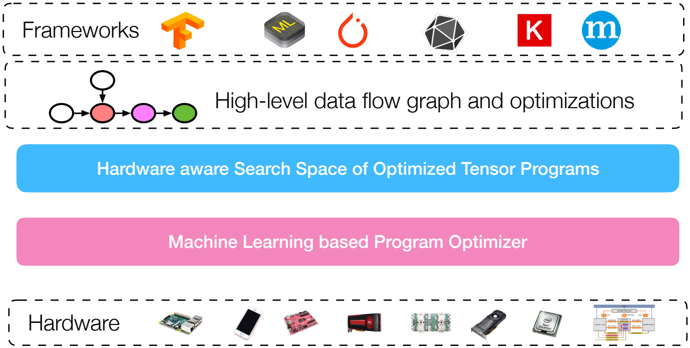
{qsun,byu}@cse.cuhk.edu.hk

July 14, 2022

Background

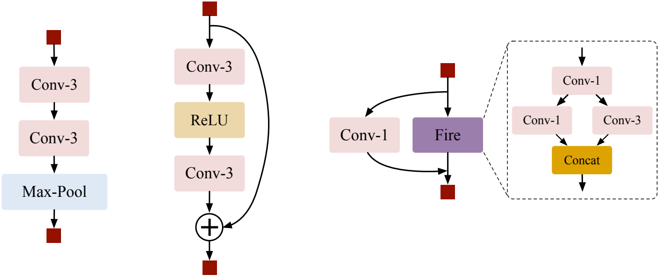
- TVM

Learning-based Learning System



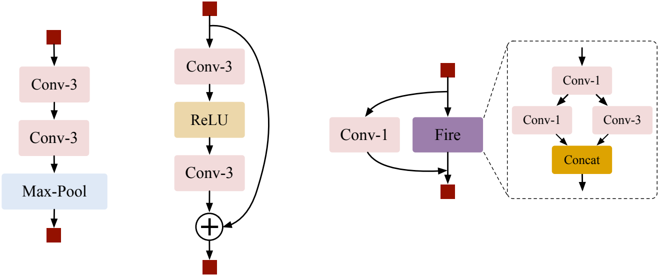
Some Concepts

- Computational graph
- Subgraph



Some Concepts

- Computational graph
- Subgraph



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

Some Concepts

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

Some Concepts

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

```
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 k.2 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.0 in range(2):
         for k.1 in range(16):
             for k.2 in range(2):
             for i.3 in range(2):
             for j.3 in range(2):
                 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 = … ]
```

Some Concepts

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

```
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 k.2 in range(None):
            for i.3 in range(None):
            for j.3 in range(None):
                C = … ]
```

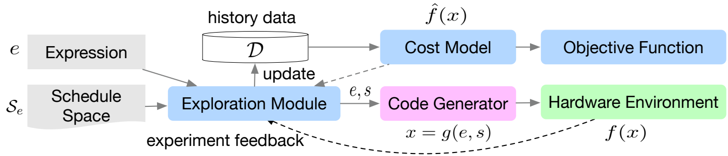
```
Annotation 1:
[Placeholder: A, B
    for i.0 in range(32):
        for j.0 in range(64):
    for ic.2 in range(16):
        for jc.2 in range(4):
            for k.0 in range(2):
        for k.1 in range(16):
            for k.2 in range(2):
            for i.3 in range(2):
            for j.3 in range(2):
                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 = … ]
```

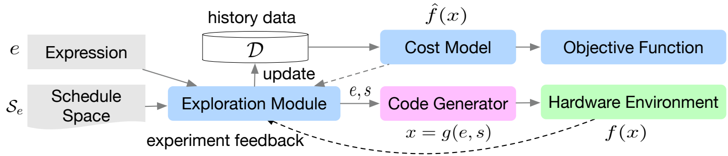
- Target: the optimization target is to find the optimal annotations for each subgraph in the deep learning model

Deep Learning Deployment Methods for GPU

Previous Arts



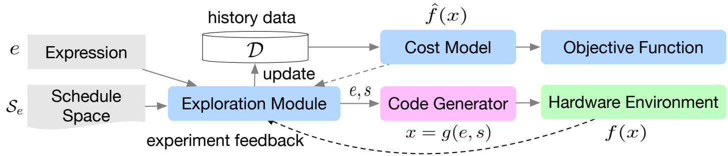
Previous Arts



- AutoTVM (Chen et al. 2018)

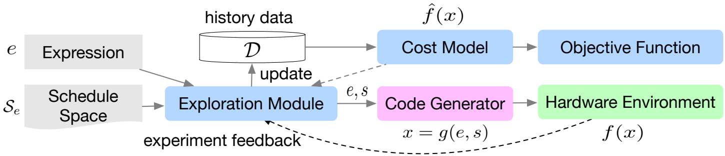
Deep Learning Deployment Methods for GPU

Previous Arts



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

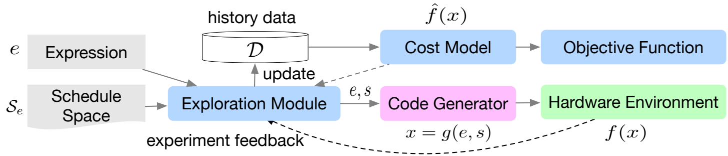
Previous Arts



- 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

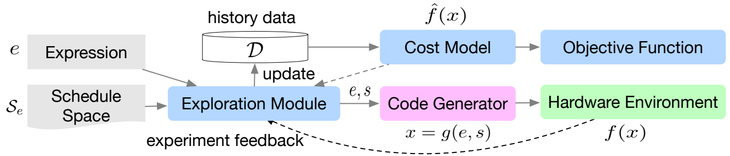
Previous Arts



- 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)

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

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

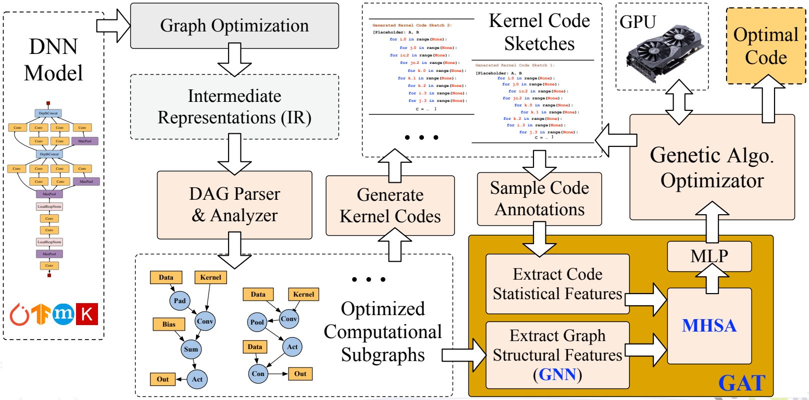
The complicated relationships between the features are not considered

- Feature items in the statistical feature vectors are treated equally, despite their physical meanings and relationships
  - XGBoost
  - MLP
  - ...

Details of GTuner

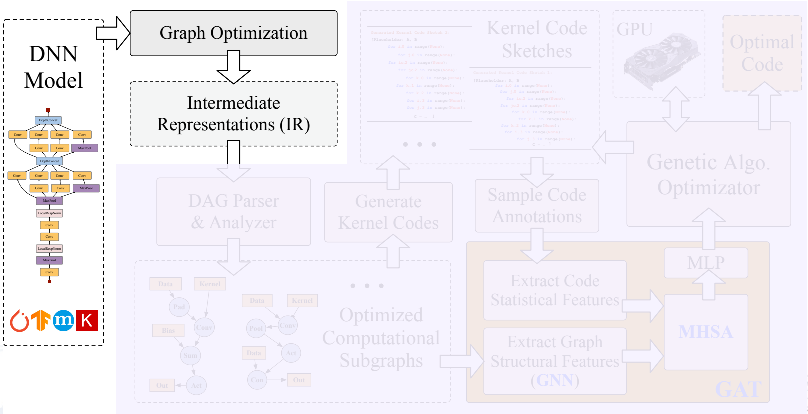
GTuner Flow

- Extract structural and statistical features for the annotations
- Graph attention network (GAT): graph neural network, and multi-head self-attention



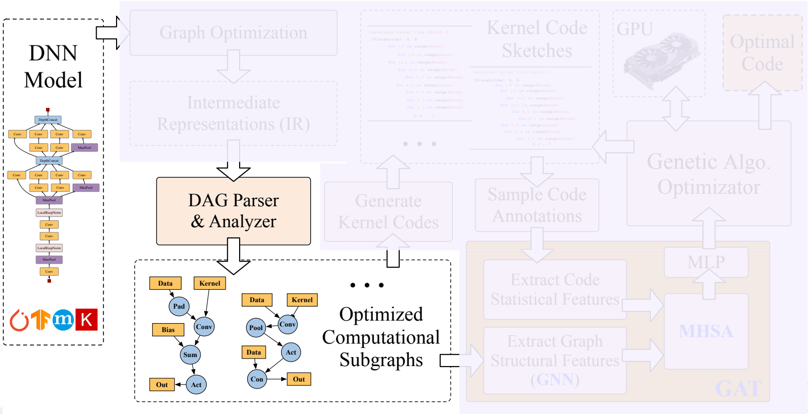
GTuner Flow

- Graph optimization
  - represent subgraphs as Intermediate Representations (IRs)



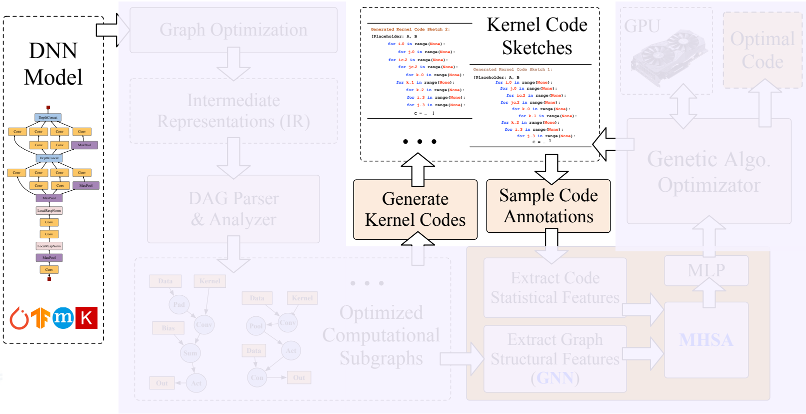
GTuner Flow

- Directed Acyclic Graph (DAG) analyzer
  - analyze the IRs to construct the optimized subgraphs



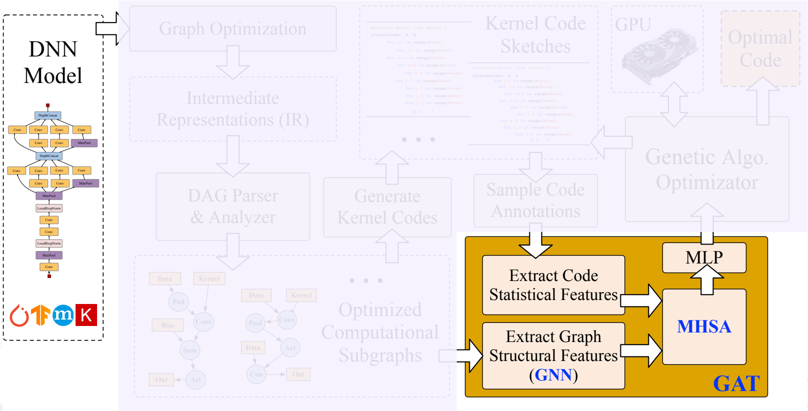
GTuner Flow

- Generate and sample codes



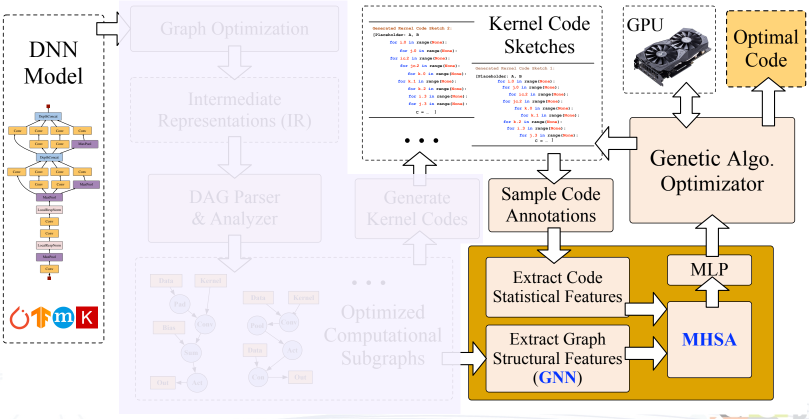
GTuner Flow

- Extract structural and statistical features
- Performance learning via Graph Attention Network (GAT)

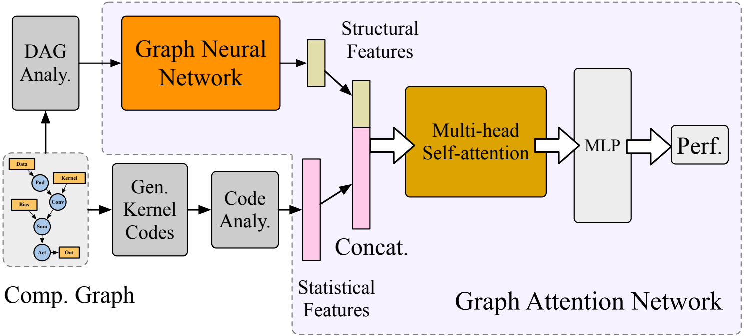


GTuner Flow

- Genetic-based iterative optimization

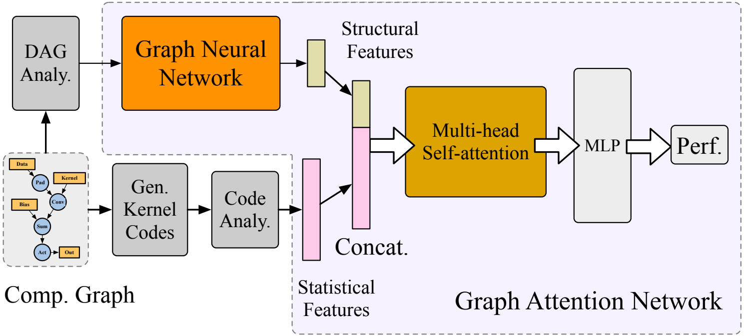


Graph Attention Network (GAT)



- Define a graph neural network to extract the structural features.

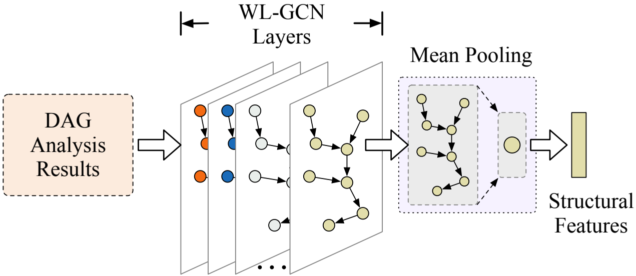
Graph Attention Network (GAT)



- 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.

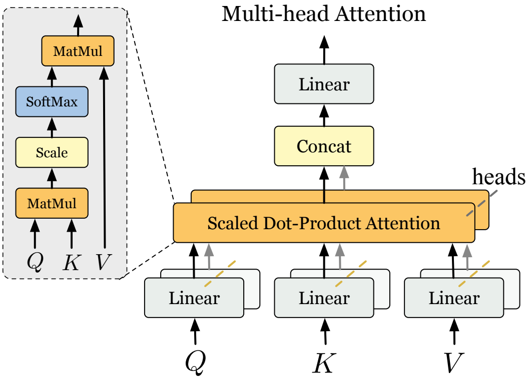
Graph Network Module

- Graph Neural Network (Morris et al. 2019)

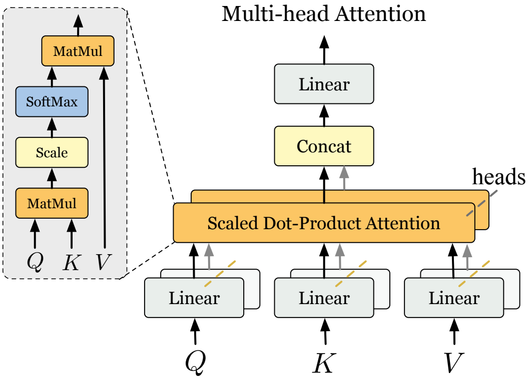


$$x_i^k = W_1^{k-1} x_i^{k-1} + W_2^{k-1} \sum_{v_t \in \mathcal{N}(v_i)} x_t^{k-1},$$

Multi-head Attention (Vaswani et al. 2017)



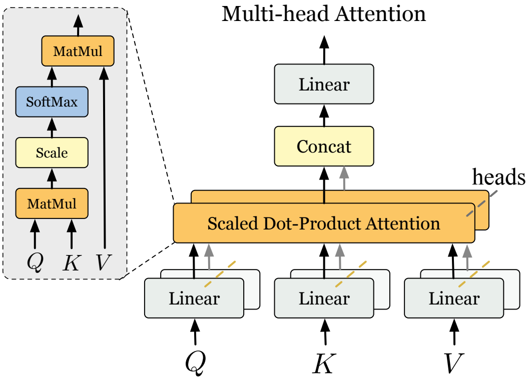
Multi-head Attention (Vaswani et al. 2017)



- scaled dot-product attention:

$$\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{softmax}\left(\boldsymbol{Q}\boldsymbol{K}^T / \sqrt{d_k}\right) \boldsymbol{V}$$

Multi-head Attention (Vaswani et al. 2017)

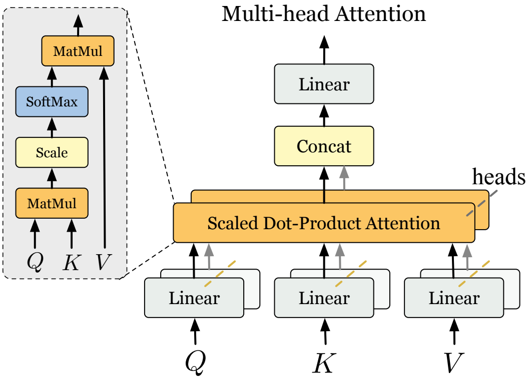


- scaled dot-product attention:

$$\mathrm{Attn}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = \mathrm{softmax}\left(\boldsymbol{Q}\boldsymbol{K}^T/\sqrt{d_k}\right)\boldsymbol{V}$$

$$\boldsymbol{H}_i = \mathrm{Attn}\left(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V\right),$$

Multi-head Attention (Vaswani et al. 2017)



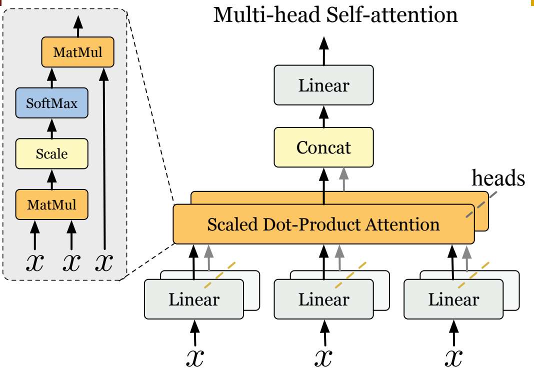
- scaled dot-product attention:

$$\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{softmax}\left(\boldsymbol{Q}\boldsymbol{K}^T / \sqrt{d_k}\right) \boldsymbol{V}$$

$$\boldsymbol{H}_i = \mathrm{Attn}\left(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V\right),$$

$$\mathrm{MHA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = Concat(\boldsymbol{H}_1, \boldsymbol{H}_2, \cdots, \boldsymbol{H}_h)\boldsymbol{W}^O$$

Multi-head Self-attention Module



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

Self-attention

$$\mathrm{SelfAttn}\left(\boldsymbol{x}^R\boldsymbol{W}_i^Q, \boldsymbol{x}^R\boldsymbol{W}_i^K, \boldsymbol{x}^R\boldsymbol{W}_i^V\right)$$

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)

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

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

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

ResNet-18	Ansor	GTuner	SpecGCN	MaskGAT	GraphSAGE
Latency (ms)	1.073	0.923	1.016	1.105	1.168

Experiments – Ablation Studies on Model Structure

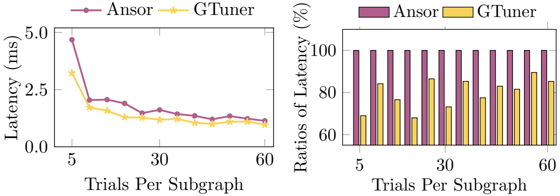
- GNN + MHSA
- MHSA
- GNN + MLP

Table: Performance without GNN or MHSA

ResNet-18	MHSA	GNN + MLP	GTuner (GNN + MHSA)
Latency (ms)	0.963	1.121	0.923

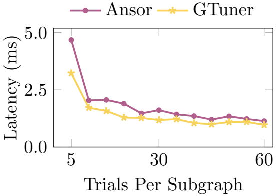
Ablation Studies – ResNet-18

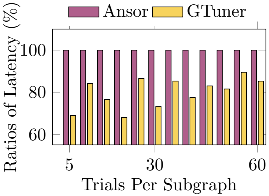
Trials of the genetic-based optimization



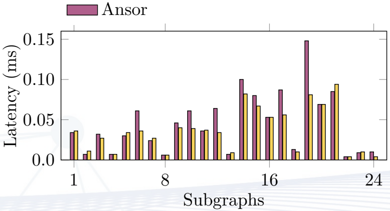
Ablation Studies – ResNet-18

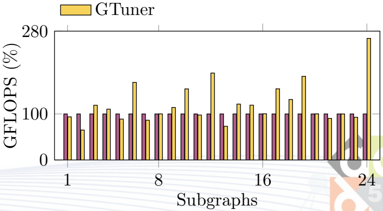
Trials of the genetic-based optimization





Performance of Subgraphs





End-to-end Performance

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

Model	PyTorch	PyTorch-JIT	AutoTVM (Chen et al. 2018)	Ansor (Zheng et al. 2020)	MHSA	**GTuner** +
ResNet-18	27.180	4.119	1.056	1.073	0.963	**0.923** (**13.98%**)
ResNet-34	48.988	5.929	1.180	0.968	0.907	**0.872** (**9.92%**)
SqueezeNet	16.658	3.648	0.311	0.207	0.201	**0.197** (**4.83%**)
MobileNet	30.324	6.972	0.513	0.242	0.252	**0.227** (**6.20%**)

+ Ratios are performance improvements compared with Ansor.

DESIGN AUTOMATION CONFERENCE

End-to-end Performance

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

Model	PyTorch	PyTorch-JIT	AutoTVM (Chen et al. 2018)	Ansor (Zheng et al. 2020)	MHSA	**GTuner** +
ResNet-18	27.180	4.119	1.056	1.073	0.963	**0.923** (**13.98%**)
ResNet-34	48.988	5.929	1.180	0.968	0.907	**0.872** (**9.92%**)
SqueezeNet	16.658	3.648	0.311	0.207	0.201	**0.197** (**4.83%**)
MobileNet	30.324	6.972	0.513	0.242	0.252	**0.227** (**6.20%**)

+ Ratios are performance improvements compared with Ansor.

Table: Time Costs (minutes) of the Optimization Processes

Model	AutoTVM	Ansor	MHSA	GTuner
ResNet-18	65.22	45.57	45.95	46.94
ResNet-34	54.86	46.66	48.89	50.71
SqueezeNet	63.90	43.53	44.40	45.91
MobileNet	61.60	42.88	43.80	44.20

Bibliography I

Byung Hoon Ahn et al. (2020). "CHAMELEON: Adaptive Code Optimization for Expedited Deep Neural Network Compilation". In: *International Conference on Learning Representations (ICLR).*

Tianqi Chen et al. (2018). "Learning to optimize tensor programs". In: *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 3389–3400.

Will Hamilton, Zhitao Ying, and Jure Leskovec (2017). "Inductive representation learning on large graphs". In: *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 1024–1034.

Thomas N Kipf and Max Welling (2017). "Semi-supervised classification with graph convolutional networks". In: *International Conference on Learning Representations (ICLR).*

Christopher Morris et al. (2019). "Weisfeiler and Leman go neural: Higher-order graph neural networks". In: *AAAI Conference on Artificial Intelligence (AAAI).* Vol. 33. 01, pp. 4602–4609.

Jiandong Mu et al. (2020). "A History-Based Auto-Tuning Framework for Fast and High-Performance DNN Design on GPU". In: *ACM/IEEE Design Automation Conference (DAC)*. IEEE, pp. 1–6.

Qi Sun et al. (Oct. 2021). "Fast and Efficient DNN Deployment via Deep Gaussian Transfer Learning". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 5380–5390.

Ashish Vaswani et al. (2017). "Attention is all you need". In: *Conference on Neural Information Processing Systems (NeurIPS)*.

Petar Veličković et al. (2018). "Graph Attention Networks". In: *International Conference on Learning Representations (ICLR)*.

Lianmin Zheng et al. (2020). "Ansor: Generating high-performance tensor programs for deep learning". In: *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 863–879.

THANK YOU!