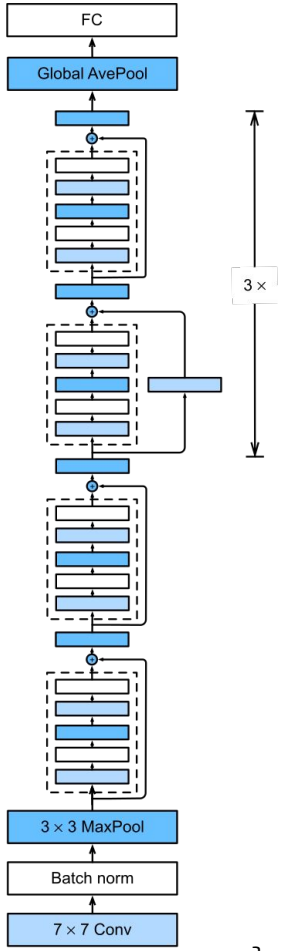
X-RLFLOW: GRAPH REINFORCEMENT LEARNING FOR NEURAL NETWORK SUBGRAPHS TRANSFORMATION

Authors: Guoliang He, Sean Parker, Eiko Yoneki

Presenter: Andrew Krapivin

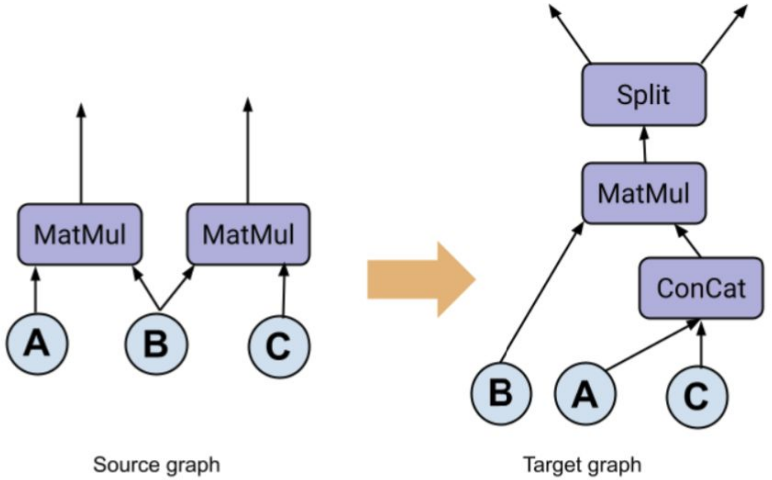
Problem: Optimizing Tensor Programs

- Fancy model, but slooooowww
- Not efficient on target architecture!
- **Goal: have compiler automatically optimize!**



Levels of Optimization

- Instruction level
- Operator level
- **Computation graph level**
  - Nodes: operators
  - Edges: tensors
  - “Graph superoptimization”
  - Must factor in lower levels!



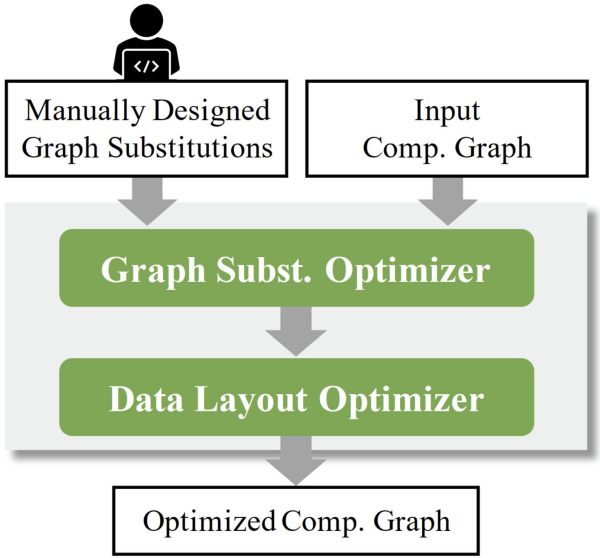
Approaches

Solution #1: Greedy Manual Heuristics

- Design graph transformations
- Apply one after another
- Greedy

Problems:

- Lots of manual effort
- Order matters
- Co-optimize layout and graph



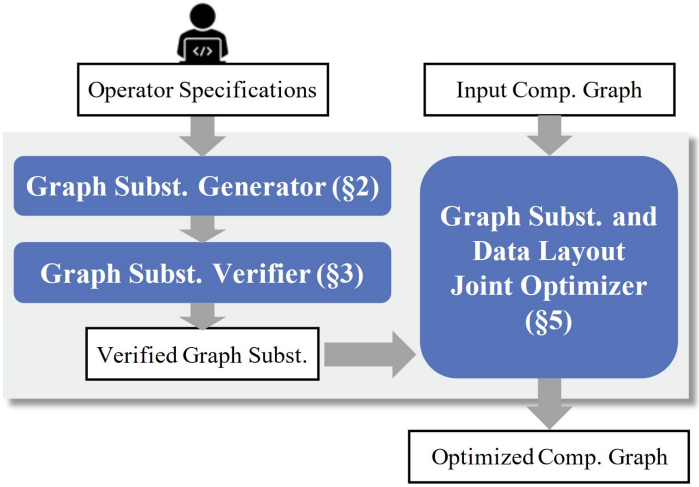
(a) Existing DNN frameworks.

Solution #2: Automation! (TASO)

- Given: specifications about graph equivalence
- Automatically generates and composes graph substitutions

Problems:

- (Really) inefficient search
- Overly simplistic cost model



(b) TASO.

Problems with Cost Model of TASO

Table 1. Discrepancy between TASO's cost model estimates and TASO's end-to-end inference latency on some unoptimised DNNs. E2E stands for end-to-end inference latency. Time is measured in milliseconds.

DNNs	Cost model	E2E	Diff (%)
DALL-E	1.8269	1.7324	5.2%
InceptionV3	8.3650	9.2098	10.1%
BERT	1.0453	1.1264	7.8%
Squeezenet	1.3082	1.4006	7.1%
ResNext-50	6.1545	7.6498	24%
T-T	2.4828	2.7281	9.9%

The New Contribution: X-RLFLOW (More Automation!)

Goals:

- Speed up search with learned model
- Use hardware for evaluation
  - Without too much computation!

Contributions:

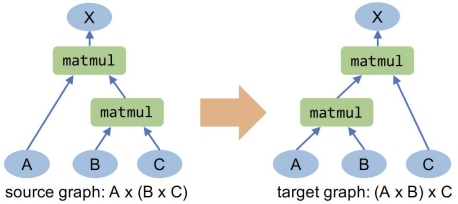
- Reinforcement learning model
  - Graph neural network
- Use hardware for evaluation, *sometimes*

Related Work: Tensat, PET, GO

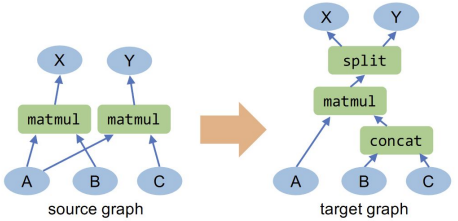
- Tensat: equality saturation
  - (One) issue: running out of memory
- PET: partially equivalent transformations
  - Same cost model as TASO
  - Sensitive to shape of operators
- GO: RL, but different level of optimization

Optimization Process

Graph Rewrite Rules



(a) Associativity of matrix multiplication.



(b) Fusing two matrix multiplications using concatenation and split.

- TASO generates general rewrite rules
- Essentially: patterns

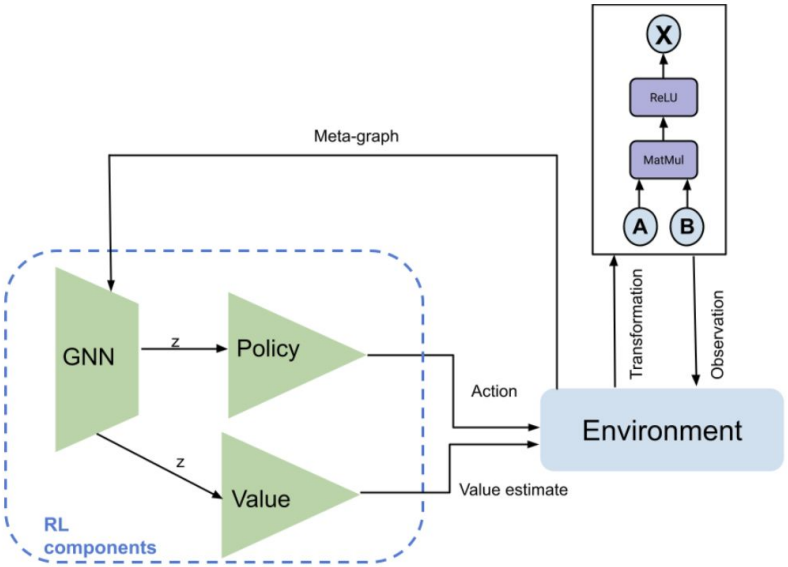
Environment

- OpenAI gym
- Rewrite rules applied to different parts of graph
- RL agent picks one to maximize reward
- (Proxy) Reward is configurable

$$r_t = \frac{RT_{t-1} - RT_t}{RT_0} * 100$$

Architecture

- Encoding:
  - Node: one-hot encoding of operator
  - Edge: tensor shape is used as attribute
- Both current graph and candidates encoded!
  - Batched into “meta-graph”
- After several transformations, aggregate node values to choose best candidate



Details

- N: hardware evaluation every N iterations
- M: edge normalization constant
- Architecture support fixed # candidates
  - Mask out those don’t need
  - Necessary: #candidates shrinks as graph is more optimized

Learning Algorithm

- PPO: few roll-outs, then update network (round)
- After round: Don’t want too much change in weights

$$\mathcal{L}_{clip} = -\mathbb{E}_G\{\min(\frac{\pi_\theta}{\pi_{\theta_k}} \cdot A^{\pi_{\theta_k}}, clip(\frac{\pi_\theta}{\pi_{\theta_k}}, 1-\epsilon, 1+\epsilon) A^{\pi_{\theta_k}})\}$$

- Value network simply mean-squared error

$$\mathcal{L}_{vf} = \mathbb{E}_G\{(V_\theta(s_t) - V_{target})^2\}$$

$$J = \mathcal{L}_{clip} + c_1\mathcal{L}_{vf} + c_2\mathcal{L}_{entropy}$$

GNN Architecture

- Follows GAT pretty closely
- First layer incorporates edges into node values:
- Several graph attention layers:

$$\vec{h'}_i = \sigma\{W(\sum_{j \in \mathcal{E}_i} \vec{e}_j \| \vec{h}_i)\}$$

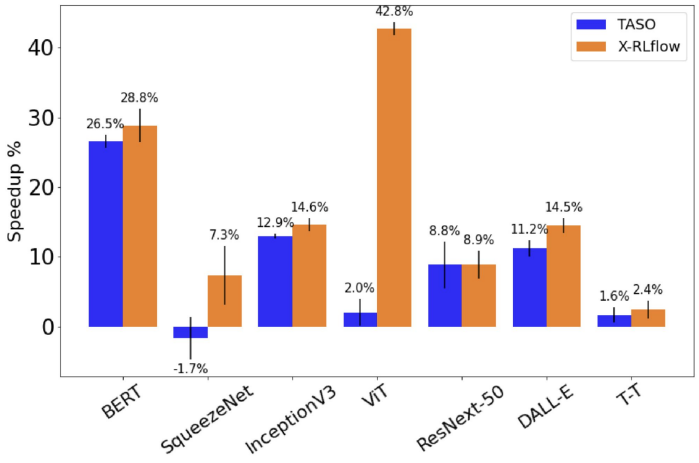
$$\vec{h'}_i = \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{i,j} W \vec{h}_j)$$

- Finally: global pooling

$$\vec{g'} = \sigma(\sum_{\mathcal{N}} \vec{h} \| \vec{g})$$

Evaluation and My Thoughts

X-RLFLOW Evaluation



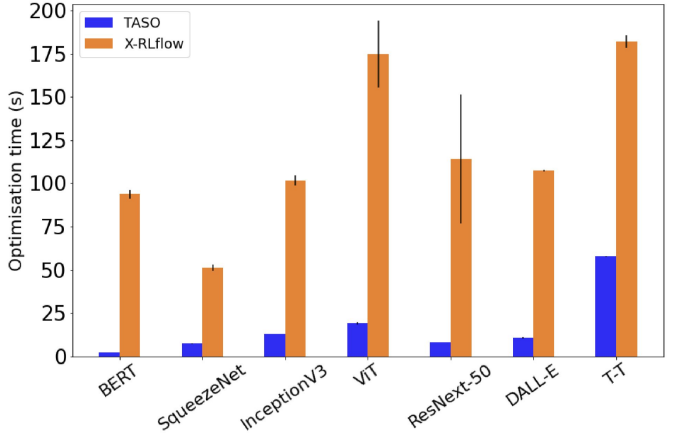
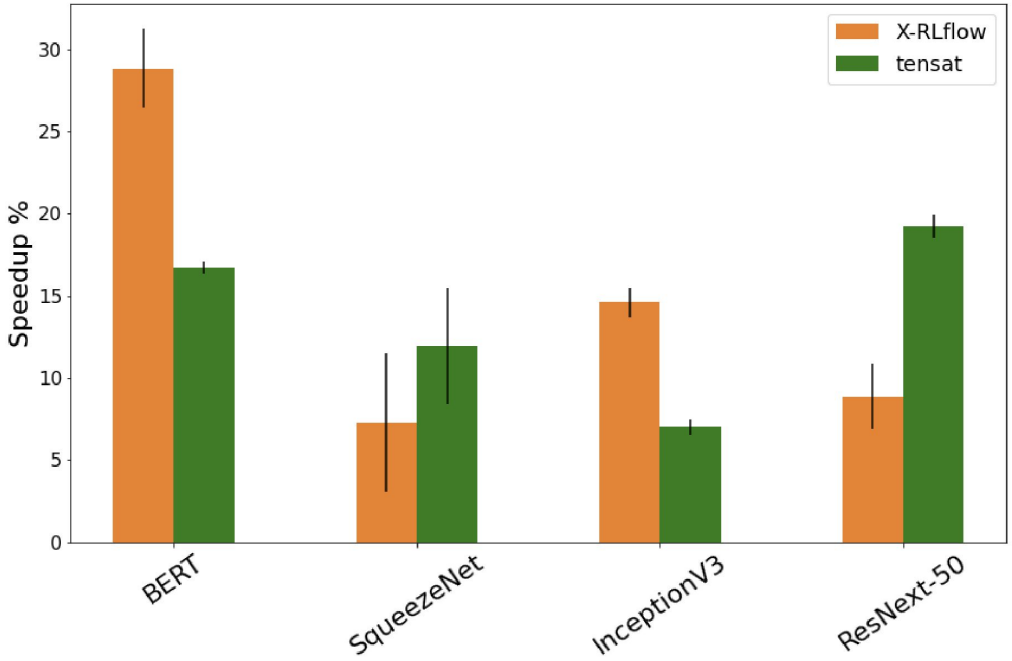


Figure 6. Optimisation time taken for TASO and X-RLflow.

- Equal or better than TASO everywhere
- Optimization times much slower

Comparison to Tensat



My Thoughts

Good:

- Better performance than TASO
- Usually better than Tensat
- Reinforcement learning better at exploring search space
- Hardware evaluation leads to better optimization
- Can generalize to different tensor shapes

Bad:

- Scalability?
  - What if big graph has many rewrite candidates?
- Cannot generalize to different computation graphs
- Questionable justification to avoid comparing with PET
- Only better than TASO on two examples
- Some details could be better explained (ex “meta-graph”)
- **Optimization times do not include training**

We can also observe there are two DNNs where X-RLflow achieves much better speedup than TASO. In the case of SqueezeNet, TASO achieves a negative speedup. This is because the cost model is inaccurate, and it misleads the substitution engine. Note that the cost modelling depends on

We further compare PET and TASO on two similar DNNs, but their performances are very different. As shown in Table 2, PET outperforms TASO in ResNet-18 but falls short in ResNext-50. We hypothesise this result is because PET’s partially equivalent transformation is very sensitive to the shape of operators. Choosing the right operator shapes may bring significant improvement to partially equivalent transformation, and PET’s paper also mentions a larger batch size offers more optimisation opportunities. However, understanding when partially equivalent transformation performs well is beyond the scope of this paper, and as a result, we will focus on TASO in this work.

Table 2. Comparison of the optimised graph inference latency between PET and TASO in ResNet-18 and ResNext-50. Time is measured in milliseconds.

	ResNet-18	ResNext-50
PET	1.9619	10.6694
TASO	2.5534	6.6453

Questions?

Generalization to Different Tensor Shapes

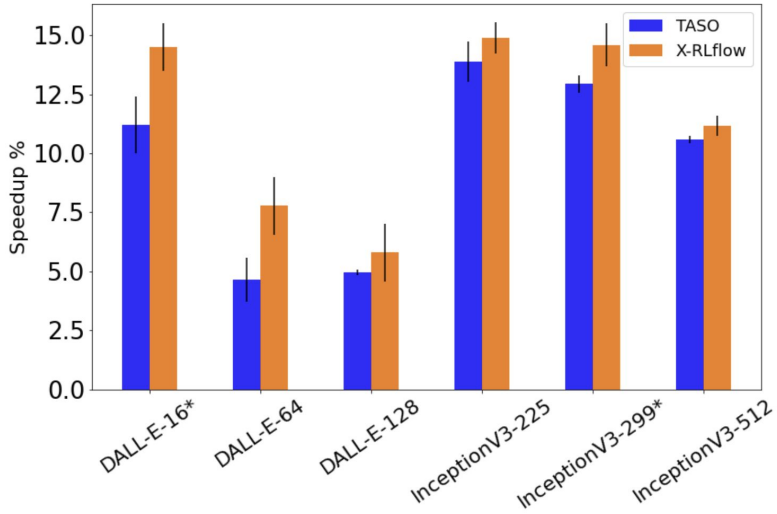


Figure 7. Generalisation to different tensor shapes on DALL-E and InceptionV3. The suffix number following the name of DNNs indicates the input tensor shape. For example, 'InceptionV3-225' indicates the input image has a height and width of 225. '*' indicates the DNNs where X-RLflow is trained to optimise.