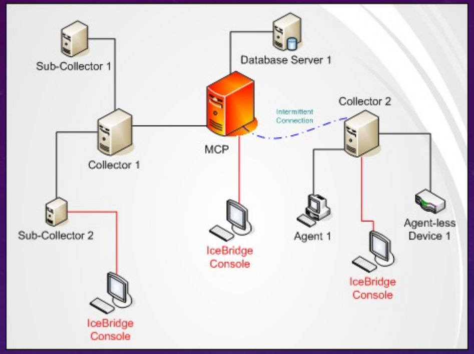
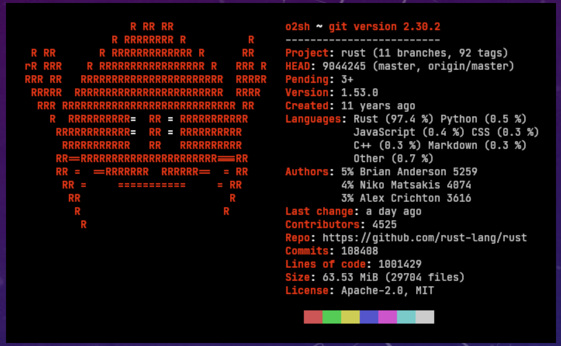
X-RLFLOW: GRAPH REINFORCEMENT LEARNING FOR NEURAL NETWORK SUBGRAPHS TRANSFORMATION

GUOLIANG HE, SEAN PARKER, EIKO YONEKI

UNIVERSITY OF CAMBRIDGE

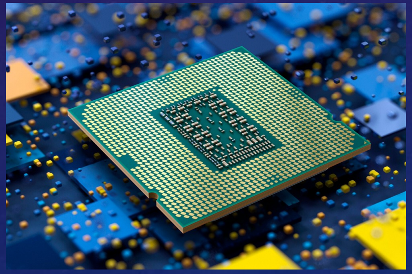


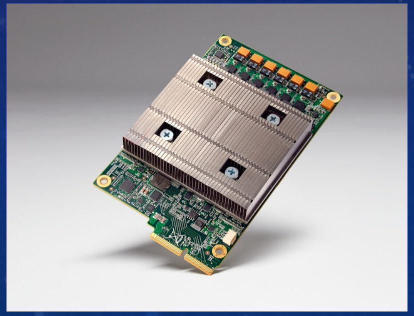


frontend

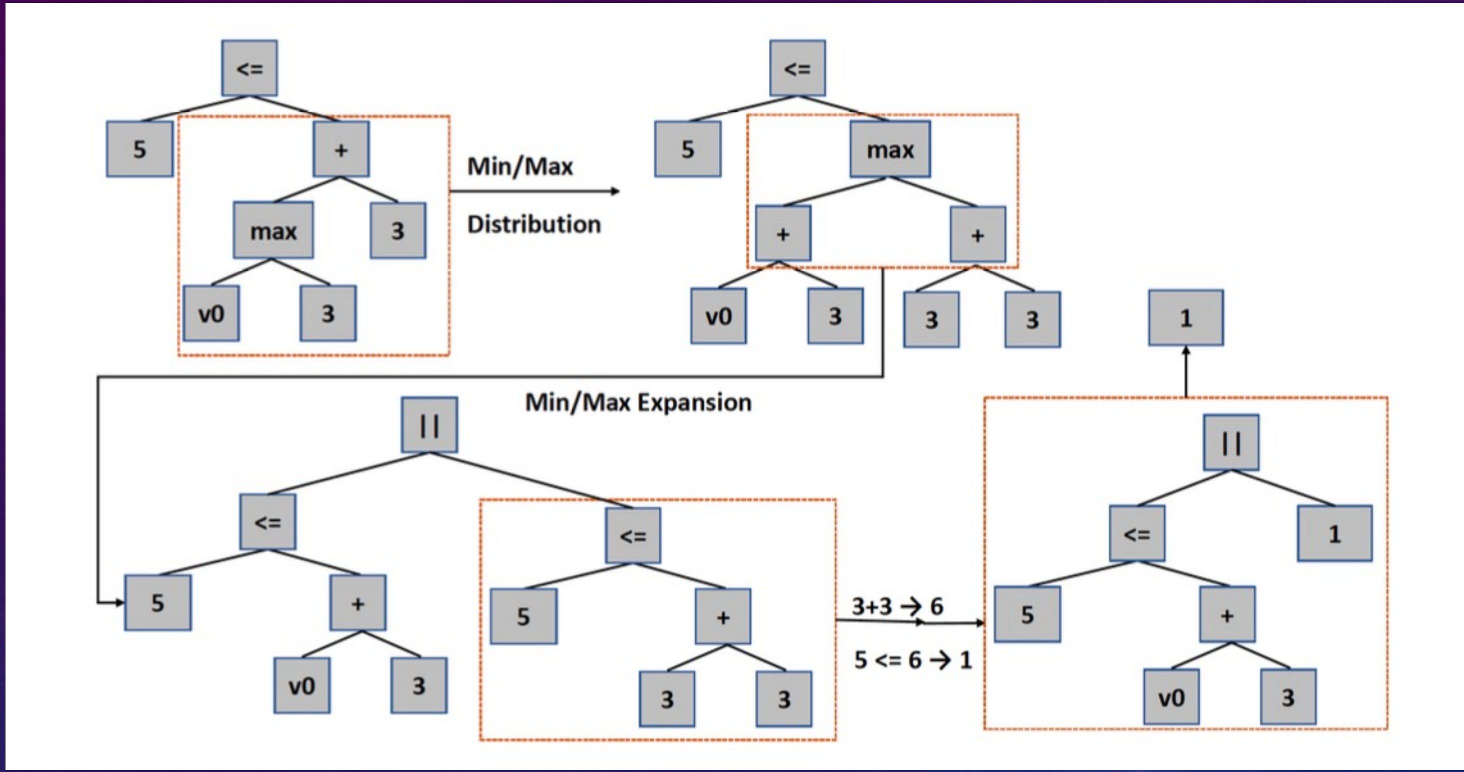
RL-driven compiler optimisation

backend





RELATED WORK: NEUREWRITER



NeuRewriter

Halide

a language for fast, portable computation on images and tensors

Chen, X. et al. 2019. Learning to perform local rewriting for combinatorial optimization

RELATED WORK: AUTOPHASE

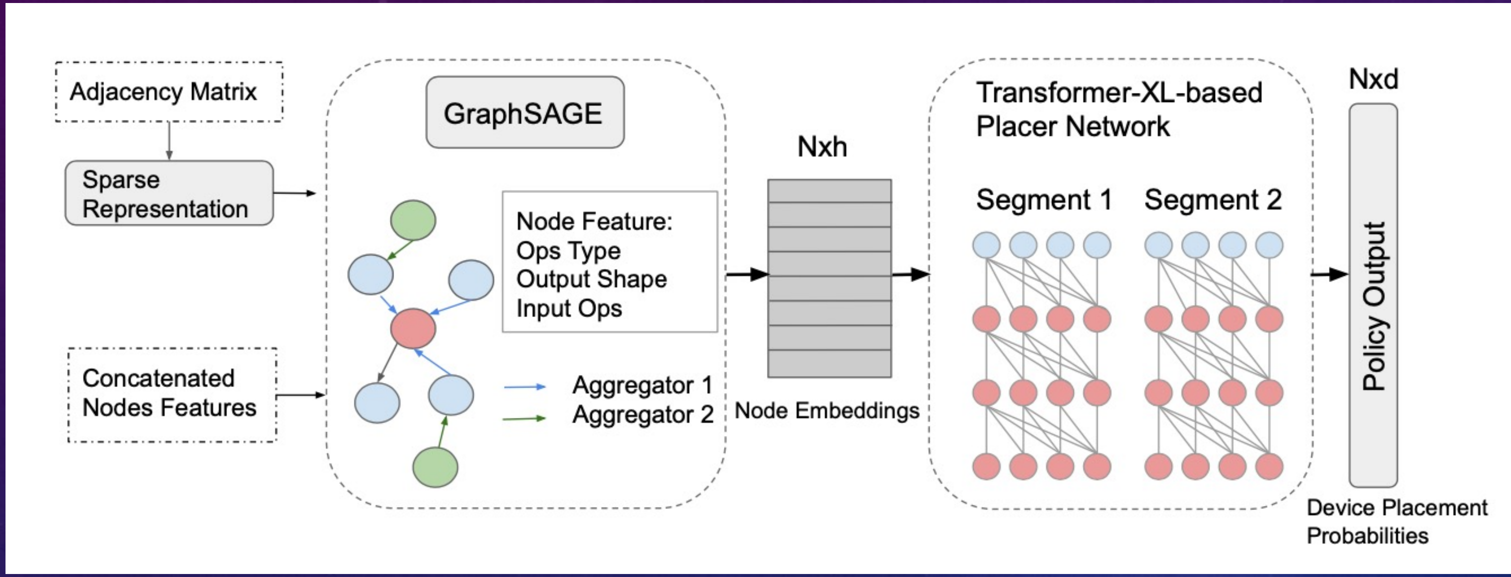
Clang command line argument reference

- Introduction
- Actions
- Compilation flags
  - Preprocessor flags
    - Include path management
    - Dependency file generation
    - Dumping preprocessor state
  - Diagnostic flags
  - Target-independent compilation options
    - OpenCL flags
    - SYCL flags
  - Target-dependent compilation options
    - AARCH64
    - AMDGPU
    - ARM
    - Hexagon
    - SPARC
    - Hexagon
    - M68k
    - MIPS
    - PowerPC
    - WebAssembly
    - WebAssembly Driver
    - X86
    - RISC-V
    - Long double flags
  - Optimization level

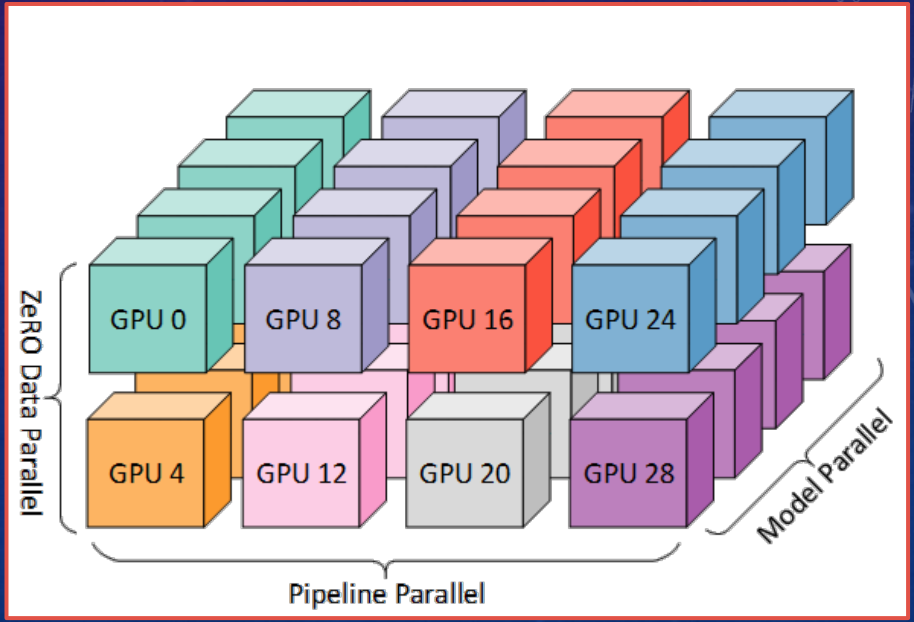
AutoPhase

opt program.c –flag1 –flag2 …

RELATED WORK: DEVICE PLACEMENT



Zhou et al. 2019. Gdp: Generalized device placement for dataflow graphs.



MORE RELATED WORKS

- Chen, X. et al. 2019. Learning to perform local rewriting for combinatorial optimization
- Huang, Q. et al. 2020. AutoPhase: Juggling HLS Phase Orderings in Random Forests with Deep Reinforcement Learning
- Haj-Ali, A. et al. 2020. NeuroVectorizer: End-to-End Vectorization with Deep Reinforcement Learning
- Trofin, M. et al. 2021. MLGO: a Machine Learning Guided Compiler Optimizations Framework
- Mirhoseini, A. et al. 2017. Device Placement Optimization with Reinforcement Learning
- And many more…

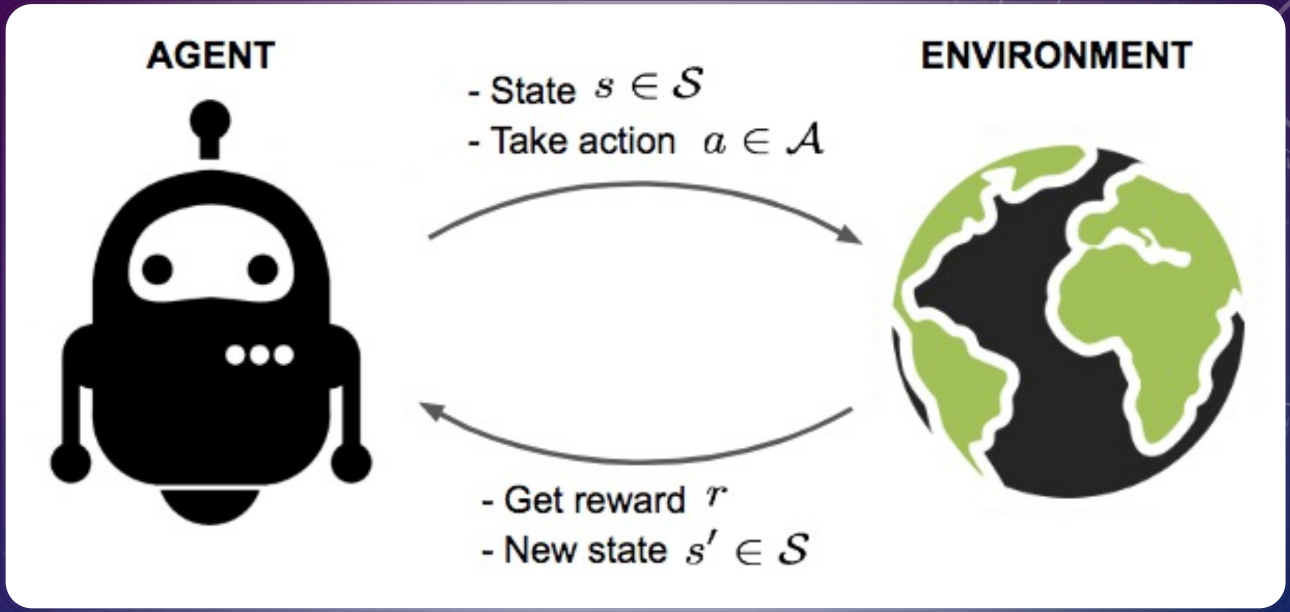
RL BASICS

RL framework:

- Markov decision process (MDP)

Advantages:

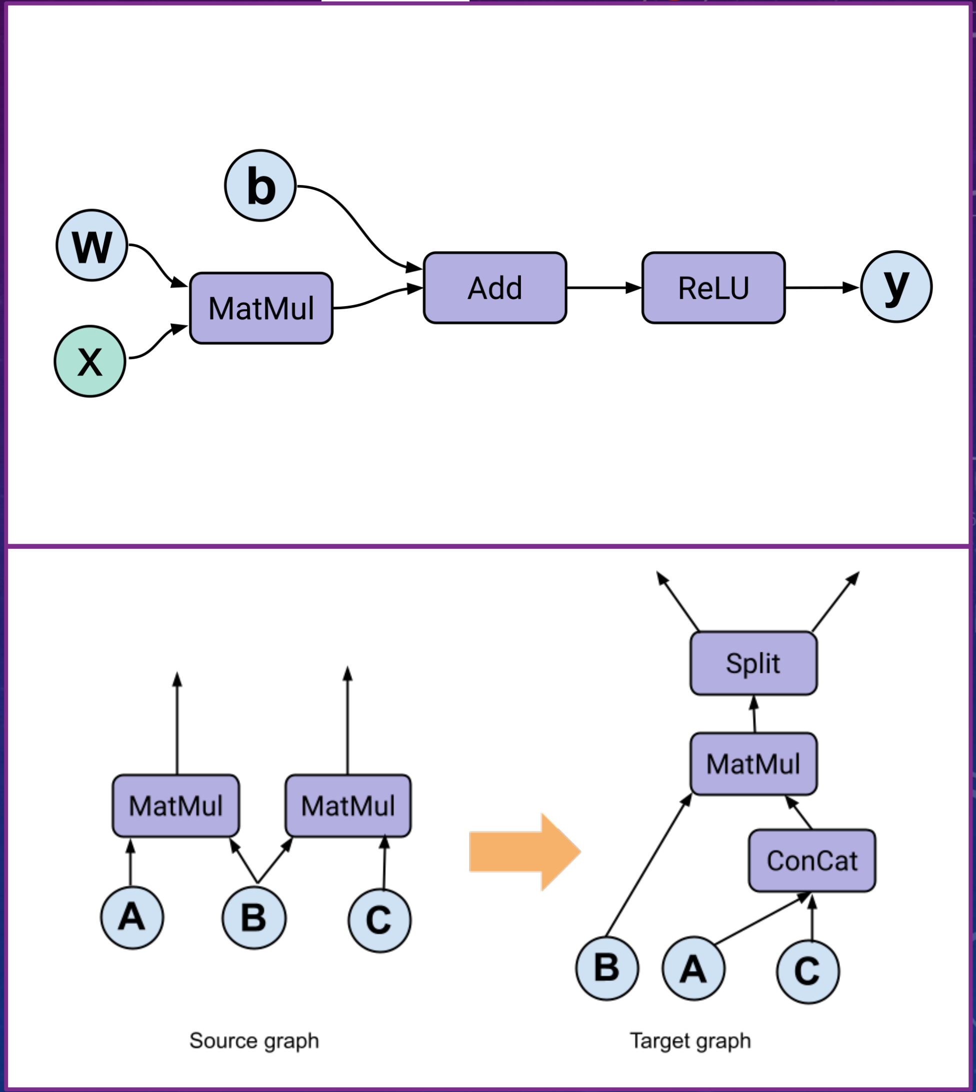
- Sequential decision-making
- Optimality: optimised for ***long-term*** rewards
- Generalisation: can learn to optimise in unseen environments



$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t]$$

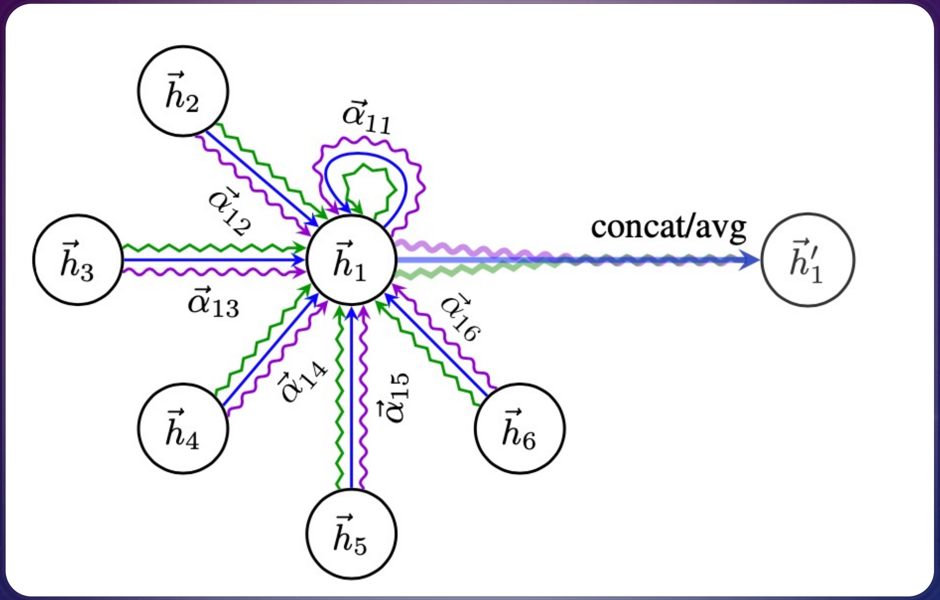
NEW CHALLENGES

- Recent advances in ML compilers present ***graph-level*** transformation
- New challenge to RL-driven compiler optimisation: ***graph domain***
- Existing program features are ***not expressive enough*** to represent graph relationship
- Graph changes ***dynamically***

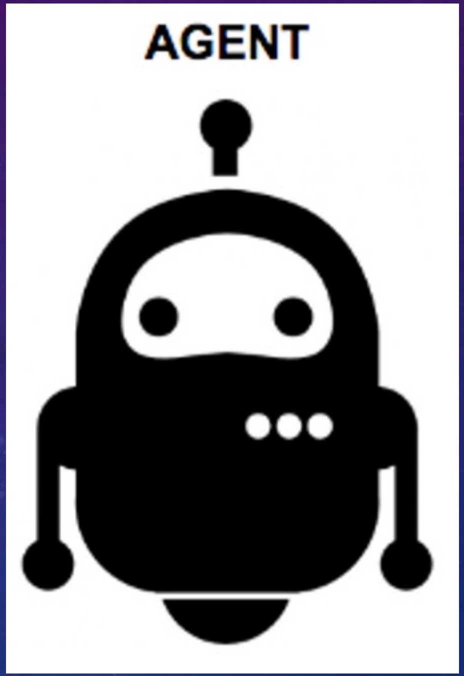


OUR IDEA

GNN + RL = X-RLflow

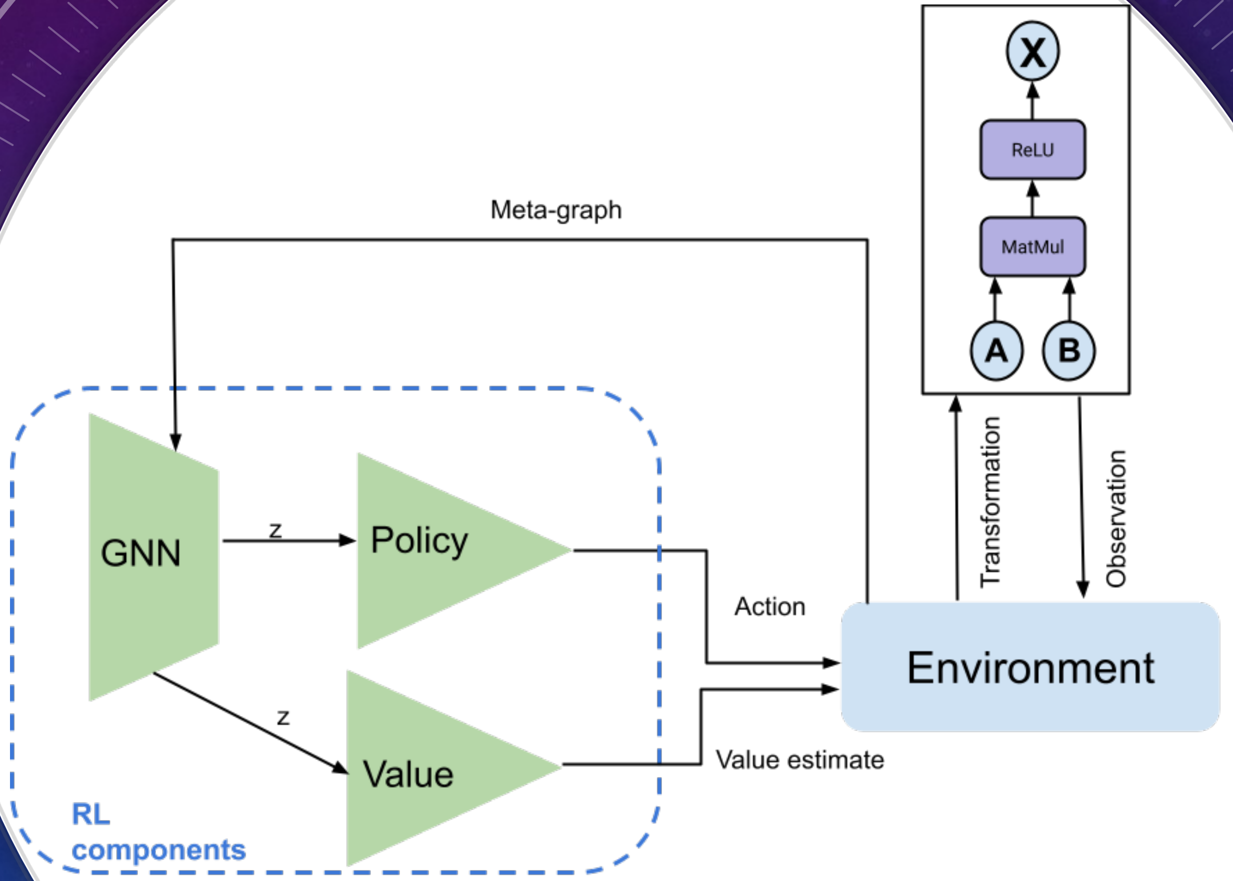


Velickovic et al. 2018. Graph attention networks

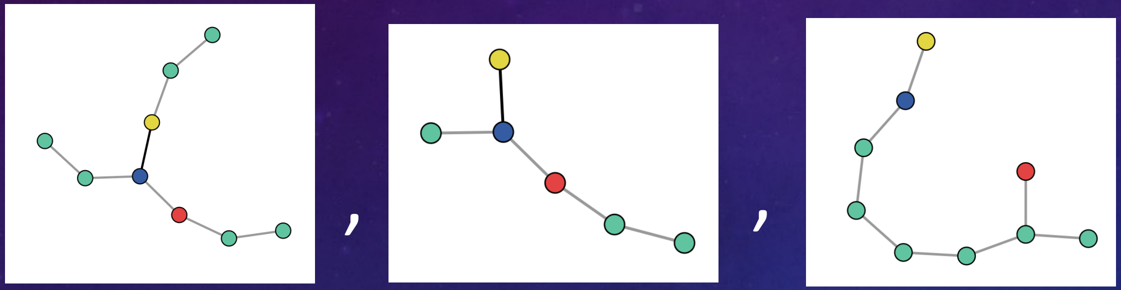


X-RLFLOW

- The environment encapsulates the dataflow graph transformation
- A list of candidates generated by applying rewrite rules are concatenated to a meta-graph
- The meta-graph is fed into a GNN for embedding
- The policy head and value head produce actions and value estimates respectively



STATE SPACE

[]

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & & \\ & \ddots & \\ & & \mathbf{A}_n \end{bmatrix},$$

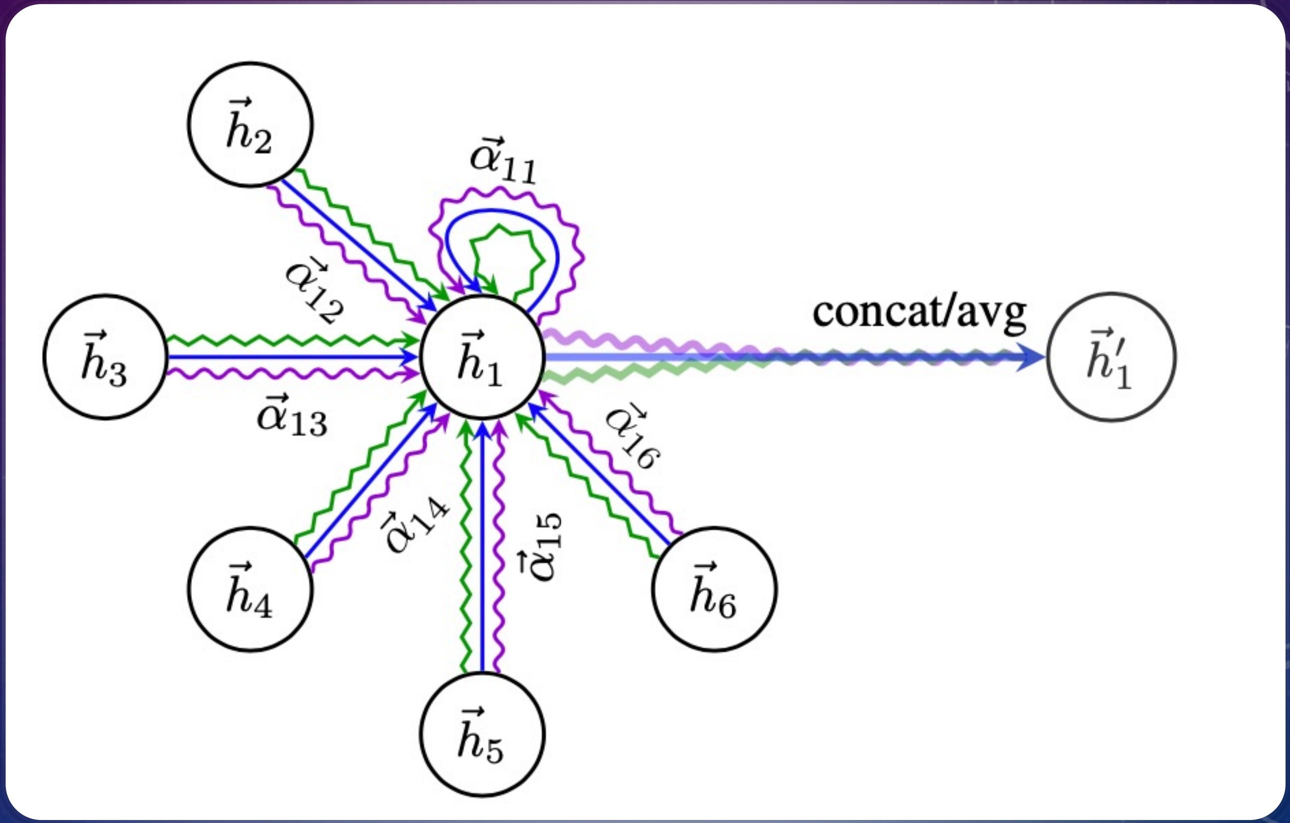
$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_n \end{bmatrix}, \qquad \mathbf{Y} = \begin{bmatrix} \mathbf{Y}_1 \\ \vdots \\ \mathbf{Y}_n \end{bmatrix}.$$

Node features: one-hot encoding tensor operators

Edge features: tensor shapes

THE GNN

- Update nodes via edge features
- Several GAT layers to update node representation
- A finally global layer to update the global representation
- Similar ideas exist for cost modelling
    - Kaufman et al. 2021. A learned performance model for tensor processing units



Velickovic et al. 2018. Graph attention networks

ACTION SPACE

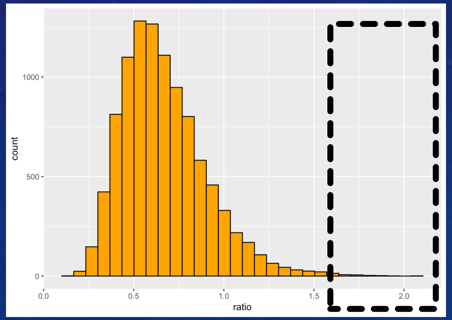
$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & & \\ & \ddots & \\ & & \mathbf{A}_n \end{bmatrix},$$

$Pr(\text{Action } 1)$

$\ddots$

$Pr(\text{Action } n)$

- Append the current dataflow graph to the end as a No-Op action to allow early termination
- Action masking for dynamically changing graph



REWARD FUNCTION

Relative runtime improvement %

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

- r_t: the reward for iteration t
- RT_{t-1}: the current graph runtime
- RT_t: the last graph runtime
- RT_0: the initial graph runtime

ON-POLICY TRAINING

Learning algorithm:

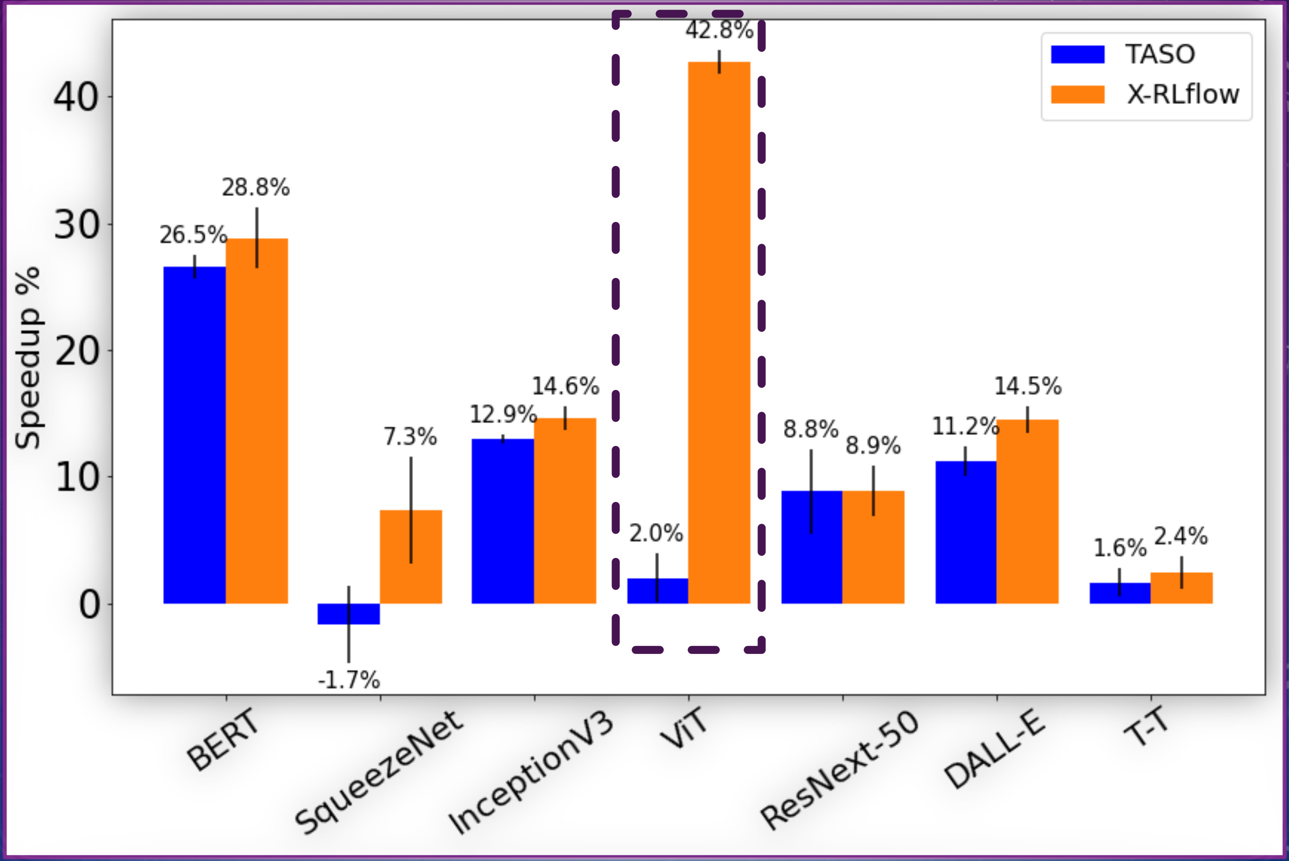
- PPO
- Gradients are backpropagated to all learnable components end-to-end

EXPERIMENT RESULTS

- Workloads: 7 different DNNs
- Platform: NVIDIA GeForce GTX 1080
- Dataflow graph transformation baselines: TASO and Tensat

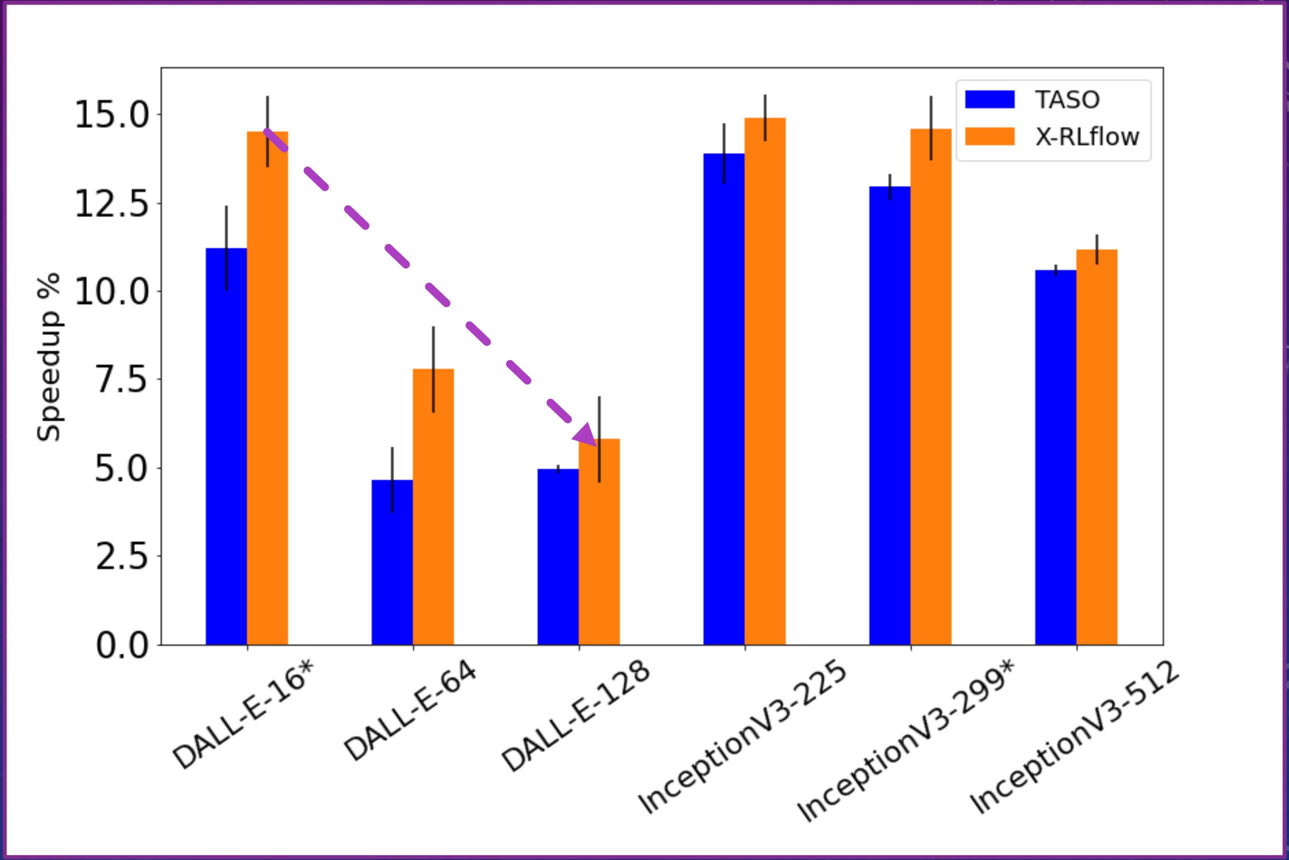
END-TO-END SPEEDUPS

- X-RLflow has better speedups in almost all cases
- The special case ViT shows more opportunities by combining the optimisation pipeline



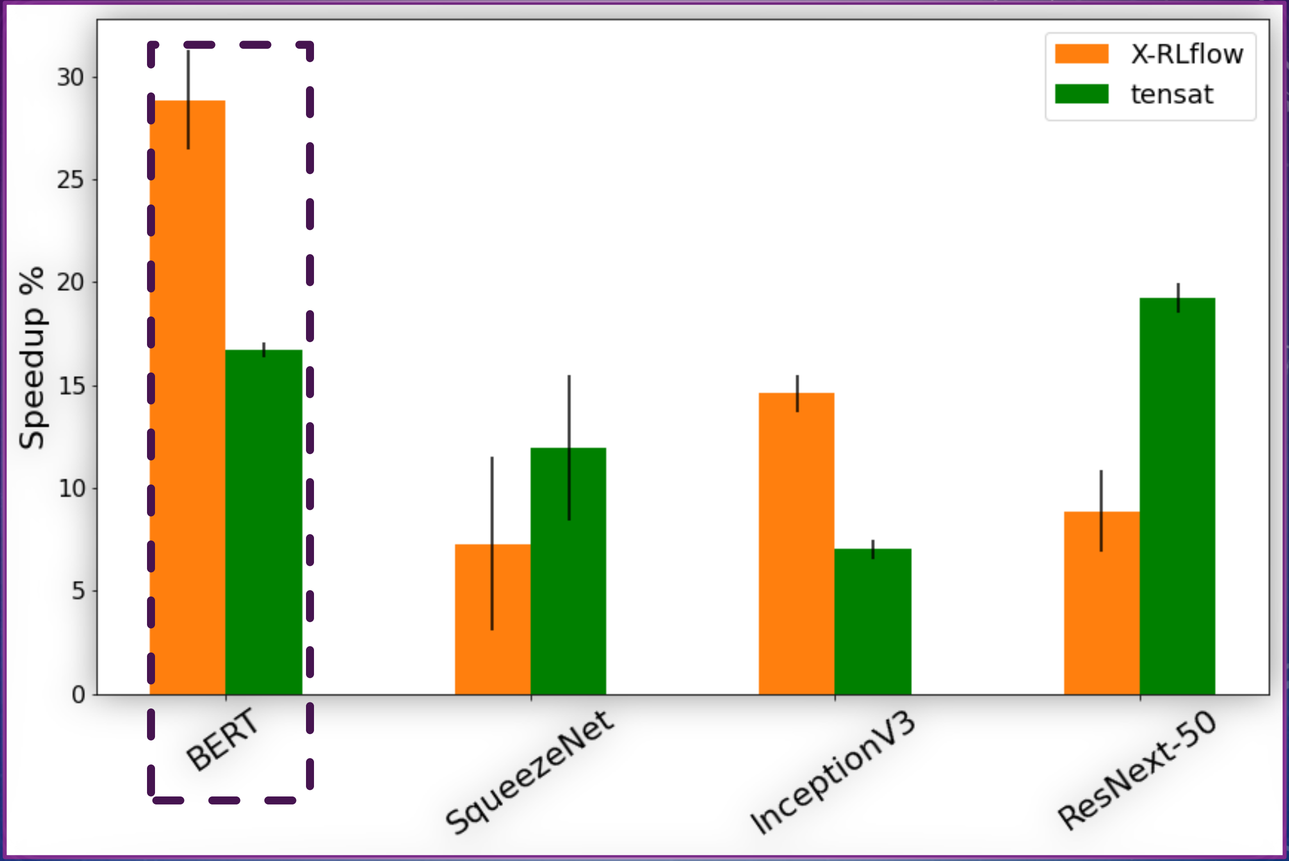
GENERALISATION

- X-RLflow can optimise in unseen environments
- Larger tensors result in less optimisation opportunities

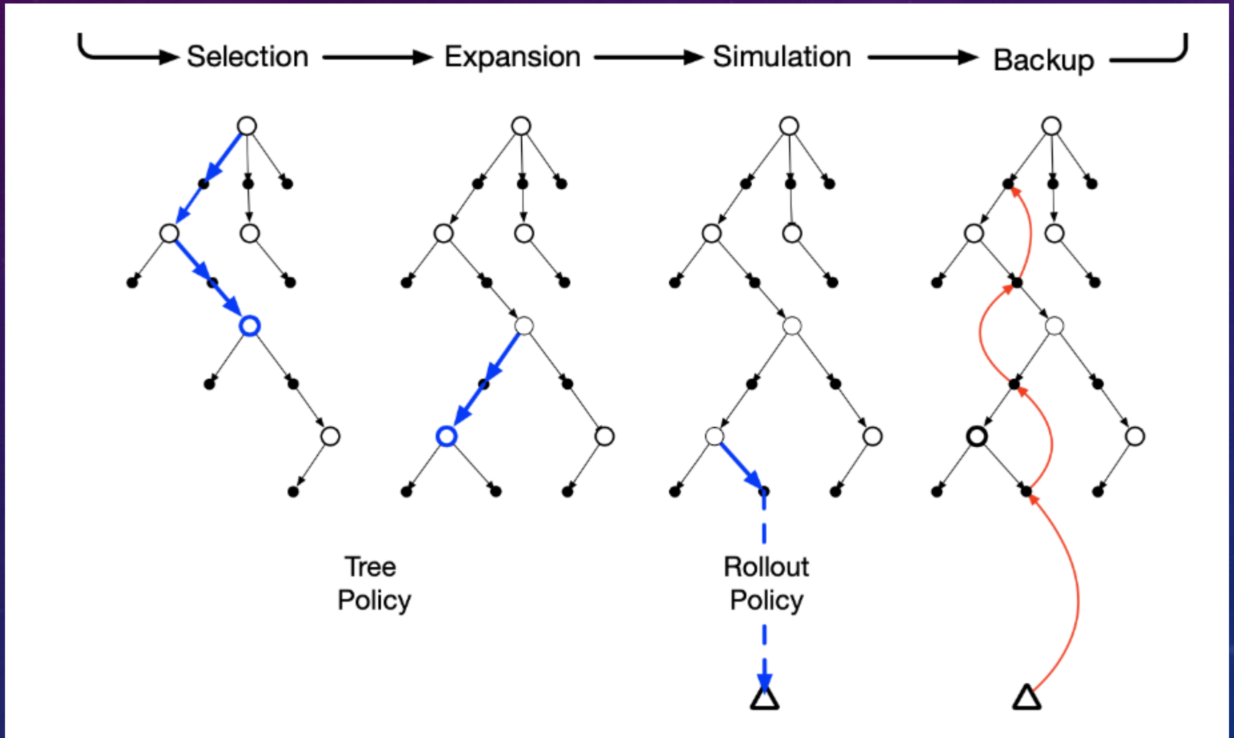


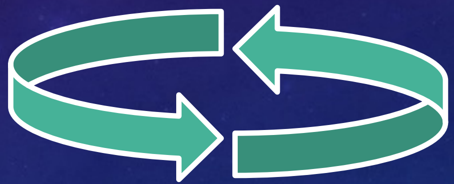
TENSAT

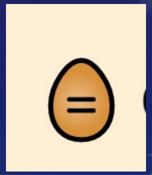
- Tensat: E-graphs to optimise transformation sequences
  - Build E-graph
  - Extract from E-graph
- X-RLflow outperforms in 2 out of 4 test cases
- RL favours complex subgraph patterns



MCTS FOR BUILDING E-GRAPH







He, G. et al. 2023. MCTS-GEB: Monte Carlo Tree Search is a Good E-graph Builder.

Willsey, M. et al. 2021. egg: Fast and Extensible Equality Saturation

SUMMARY

X-RLflow:

- Enable RL-driven compiler optimisation to dataflow graph transformation domain

Future works:

- Combine the optimisation pipeline
- Evaluate on more graph transformation domains

https://github.com/ucamrl/xrlflow

Questions?

Email: gh512@cam.ac.uk

Thank you!