**Authors**
Azalia Mirhoseini * 1 2 Hieu Pham * 1 2 Quoc V. Le 1 Benoit Steiner 1
Rasmus Larsen 1 Yuefeng Zhou 1
Naveen Kumar 3 Mohammad Norouzi 1 Samy Bengio 1 Jeff Dean 1

# DEVICE PLACEMENT OPTIMIZATION WITH REINFORCEMENT LEARNING

# DISCUSSION POINT

Presentation by Alexander Wickman

# Reinforcement Learning for device placement

The paper's punchline is that their RL-based device placement model outperforms expert defined heuristics and algorithm-based graph partitioning

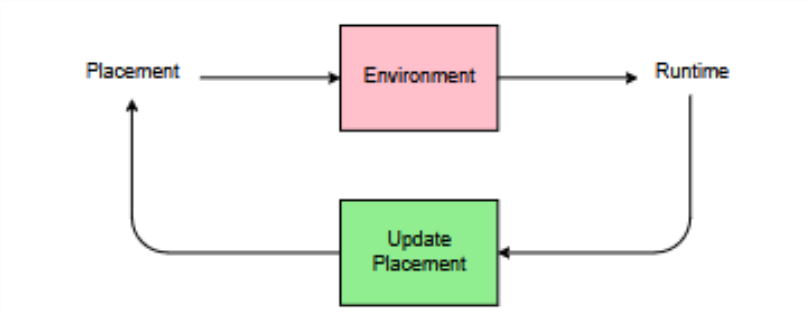



*Figure 1.* An overview of the RL based device placement model.

**Are the choices regarding the RL model as effective as they could be?**

# RL Model – Hand picked design decisions

Reward signal: Square-rooted measured running time: $R(P) = \sqrt{r(P)}$

Failing signal (for crashed placements) as a large constant, **set by hand for every model, depending on the input graph**

After step 5000: model updated only with placements that ran

# Criticising the model design

A key issue: the RL model had issues with memory robustness

- Placements that overload memory of devices caused crashes
- The reward signal or input does not account for memory constraints of devices
- The author's solution is trial and error
  - Results in wasted computations
- Introduced the high constant failing signal to account for crashed placements
  - Something which arguably **distorts reward signals**
  - **What if the constant is set poorly?**

**Are there better ways to account for memory constraints?**

# Memory robustness - alternative

The only hardware consideration in the model is **the number of devices**
Consider the memory capacity of each device as an input?

- Reject placements which fail to load memory
- But memory consumption is **variable**: peak memory depends on the order of graph operations

Potential solution: Cost model to predict memory usage and rule out specific placements?

- Cost models can be wrong
  - Throw away valid/good placements
  - Paper argues optimising directly for running time is more accurate
- But a good memory cost model will save wasted computations

**Is designing a good cost model to predict memory more practical than trial and error?**

# RL for optimal device placement

Is RL the best optimisation method
The paper doesn't compare against other adaptive methods

- No simpler search as a baseline (e.g. random search, simulated annealing) over the same search space and number of places evaluated

How about using a modelled runtime?

- Bayesian optimisation?
  - Why it could be suitable: data is expensive
  - Why it could struggle: growing dimensions, how do we model device placement?

**Is RL the most suitable ML method for this problem? What alternatives could be considered?**

# Discussion

**Are there better ways to account for memory constraints?**
**Is designing a good cost model to predict memory more practical than brute force?**
**Is RL the most suitable ML method for this problem? What alternatives could be considered?**

Other areas

- Co-location groups reduce operations considerably (reducing placement model's memory and combinations). Never isolated in experiments
- Scotch (graph partitioning) as a baseline