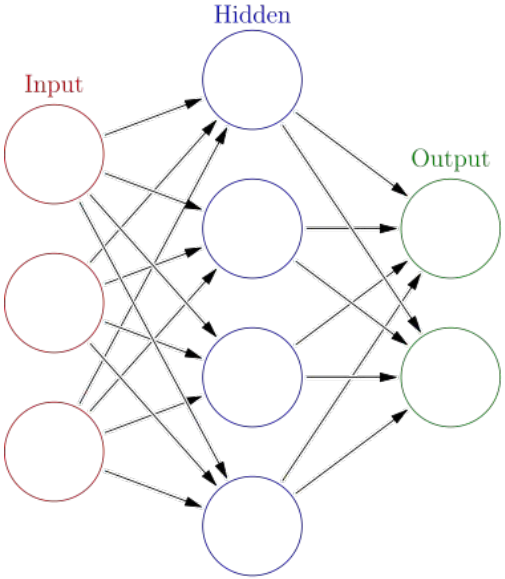
Device Placement Optimisation with Reinforcement Learning

Review

✳: Figure from paper /codebase

Problem Setting

Device Mappings

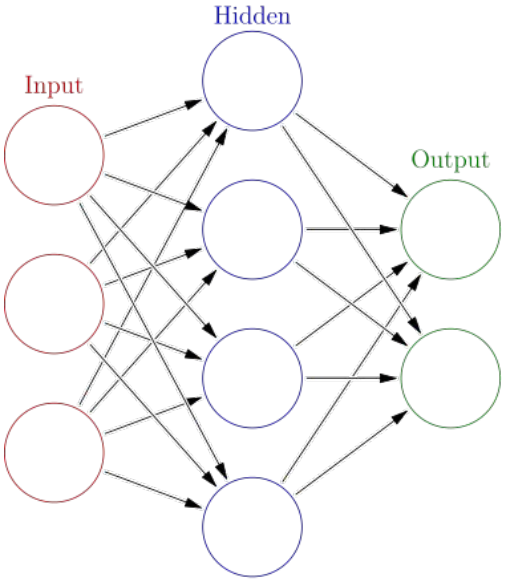


Neural Network

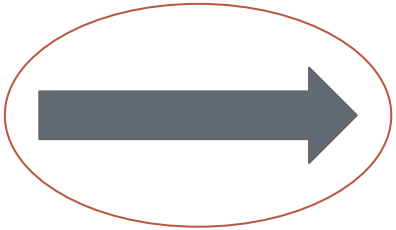
Execution Devices

Images from Wikipedia

Device Mappings



Neural Network



Execution Devices

Images from Wikipedia

Conceptual Model

Source Graph (Computation Dataflow Graph)

- Tensorflow computational graph
- Nodes are operations
- Edges represent data transfer, generally tensors

Target Graph

- Nodes are execution devices (GPU, CPU, TPU etc.).
- Edges are communication links (PCIe, network links etc.)

Goal: find a *placement* – an injective mapping from nodes in the source graph to target graph

Option 1: Expert Designed

torch.nn.to(device=`cuda’)

- Time Consuming
- Relies primarily on intuition
- Possibly suboptimal

Option 2: Graph Partition Heuristics

- We expect more nodes in the source graph relative to the target graph.
- Partition the source graph into different partitions, with each partition mapping onto a node in the target graph.
- Relies on heuristics (a cost model) for partitioning, leading to poor results.

Mirhoseini et al. select software package Scotch (Pellegrini, 2009) as a baseline.

Implementation relies on a collection of methods. Scotch users have the option to manually tune hyper-parameters.

Key Challenges

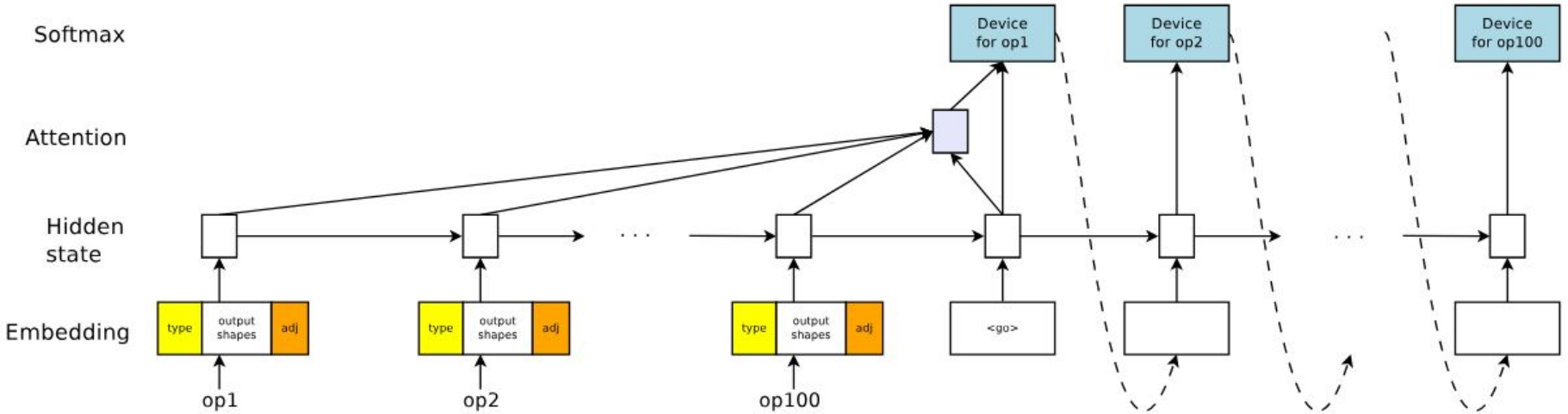
- Inaccurate and expensive cost model
- Exponential search space with number of nodes
- Dynamic and complex environment

Approach

Sequence-to-Sequence Placement Function

- Long Short Term Memory (LSTM) models is a type of recurrent neural network used for sequential tasks in natural language processing.
- Model graph placement as a sequence-to-sequence translation task.
- Additional context-based attentional mechanism.

- The input sequence is the concatenated nodes of the source graph with tunable embeddings, output shape, and one-hot encoded adjacent nodes to capture graph topology. Encoded into a sequence of embeddings by **encoder LSTM**.
- An **attentional decoder LSTM** translates the embedded sequence into a sequence representing the partition mapping.


Softmax
Attention
Hidden state
Embedding
type
output shapes
adj
op1
op2
op100
<go>
Device for op1
Device for op2
Device for op100

Objective

The goal is to find some placement which minimises the execution time. Due to possible interference, the execution time is stochastic. Thus, optimise for the expectation.

$$\mathbb{E}_{\pi}\left[R(\mathcal{P})|\mathcal{G}\right]$$

Where pi is the placement function, R is the execution time, G is the source graph and P is the placement.

(Hack): The authors propose optimising the square-root of running time from empirical observations. *This boosts the training signal when execution time is low.*

Optimisation

R (P|G) is unknown without an analytical form *(Modelling will result in aforementioned inaccuracies with the cost model).*

The gradient can be estimated via policy gradient.

$$\mathbb{E}_{\pi}\left[R(\mathcal{P}) \times \nabla_{\theta} \log p(\mathcal{P}|\mathcal{G};\theta)\right]$$

Device Placement Optimisation with ~~Reinforcement Learning~~ Policy Gradient

Policy Gradient

$$\mathbb{E}_{\pi}\left[R(\mathcal{P}) \times \nabla_{\theta} \log p(\mathcal{P}|\mathcal{G};\theta)\right]$$

The first term can be **sampled**. (Sampling many P from pi, Monte-Carlo sampling)

There is a well known result that a P-dependent term can be subtracted from R without introducing bias. The authors subtract a simple moving average.

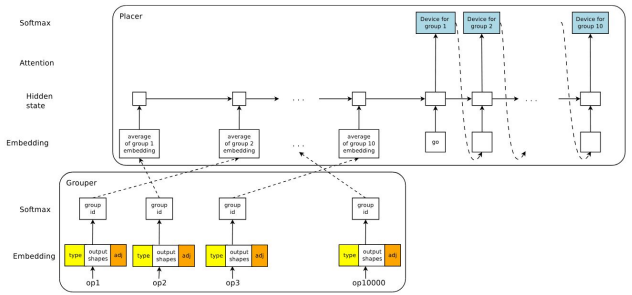
The second term has an analytical form (p is the encoder-decoder)

Gradient descent!

Co-location

- Large numbers of operations cause exploding / vanishing gradients.
- Use heuristics to lower the number of operations (co-location).
- Ex: If output of an operation X is consumed only by Y, then X and Y are co-located (merged).
- Specific rules for CNN and RNN.
- 50-500 fold reduction in experiments.

- ~~Limitation~~ Resolved: follow-on work by same first author, “A hierarchical model for device placement”.
-



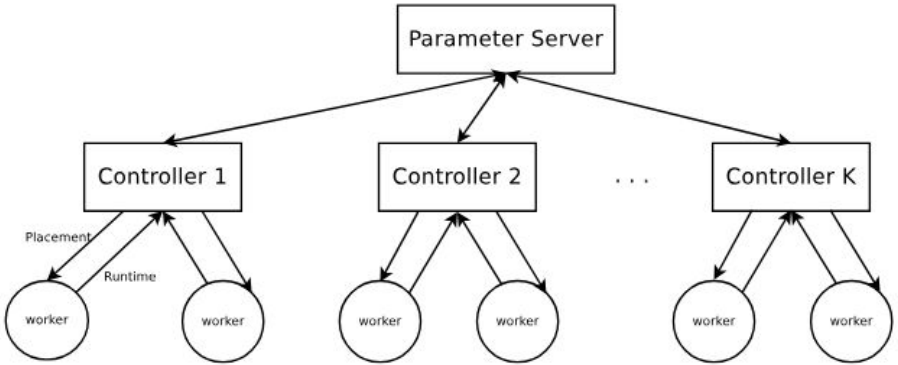
- Classifier replaces heuristics for grouping
- Increased granularity significantly improves performance on specific tasks

Distributed Training

Distributed data collection / environment setup.

Controllers generate placements.
Workers test placements.

20 controllers, 4-8 workers each. 12-27 hours for training.



Miscellaneous

Placement failure (sporadic, resource limits) are manually assigned large times. To avoid sudden large gradient steps, after a certain point parameter updates on placement failure is disabled.

Why REINFORCE?

Why reinforcement learning?

Reinforcement learning has advantages dealing with sequences of actions and possibly delayed effects. There are no (transition) sequences in the current formulation. Thus, the motivation for RL as a solution is uncertain, compared to, for example, Bayesian Optimisation. Alternatively, the environment should be engineered in such a way to benefit from reinforcement learning.

Experiments

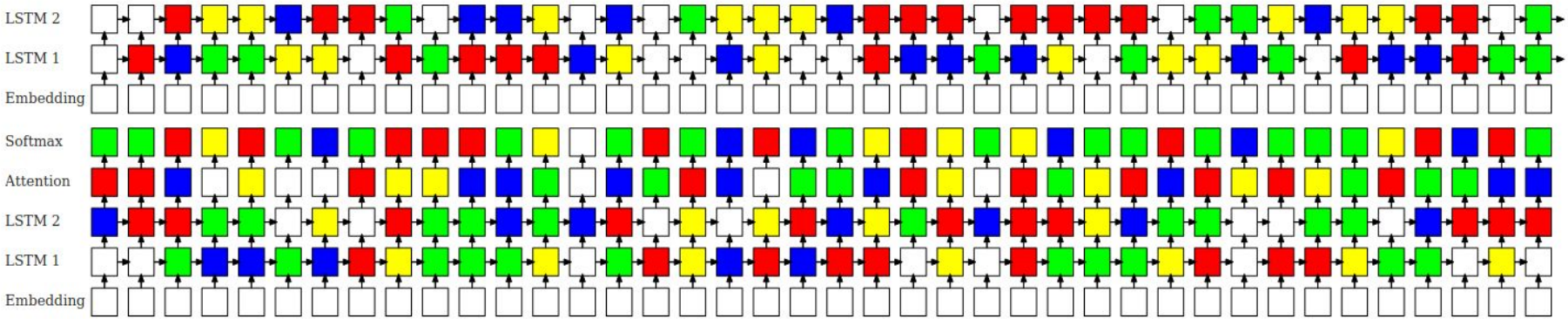
Run-time comparison

Tasks	Single-CPU	Single-GPU	#GPUs	Scotch	MinCut	Expert	RL-based	Speedup
RNNLM (batch 64)	6.89	**1.57**	2	13.43	11.94	3.81	**1.57**	0.0%
			4	11.52	10.44	4.46	**1.57**	0.0%
NMT (batch 64)	10.72	OOM	2	14.19	11.54	4.99	**4.04**	23.5%
			4	11.23	11.78	4.73	**3.92**	20.6%
Inception-V3 (batch 32)	26.21	**4.60**	2	25.24	22.88	11.22	**4.60**	0.0%
			4	23.41	24.52	10.65	**3.85**	19.0%

Recurrent Neural Network Language Model, Neural Machine Translation, InceptionV3

Trade-offs learnt between parallelism and costs of inter-device communication

Learns Device Properties



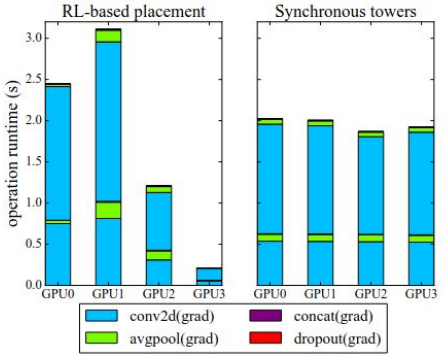
Non-trivial placement learnt for Neural MT Graph - colors represent devices

Transparent denotes CPU - learns specific capabilities of each device, as embeddings are handled by CPU, which are computationally less expensive

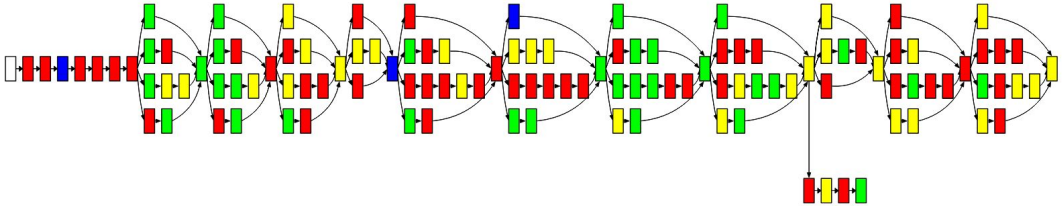
Learns Graph Properties

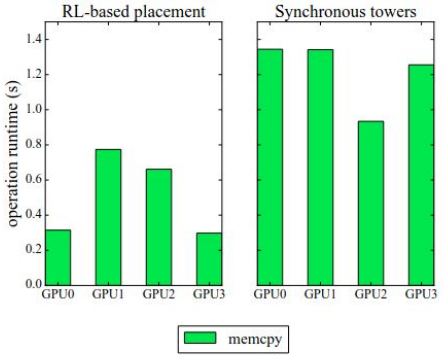
InceptionV3

Non-balanced computational load learnt: the network has more dependencies so model parallelism is not viable



However, this tradeoff allows it to reduce communication costs.



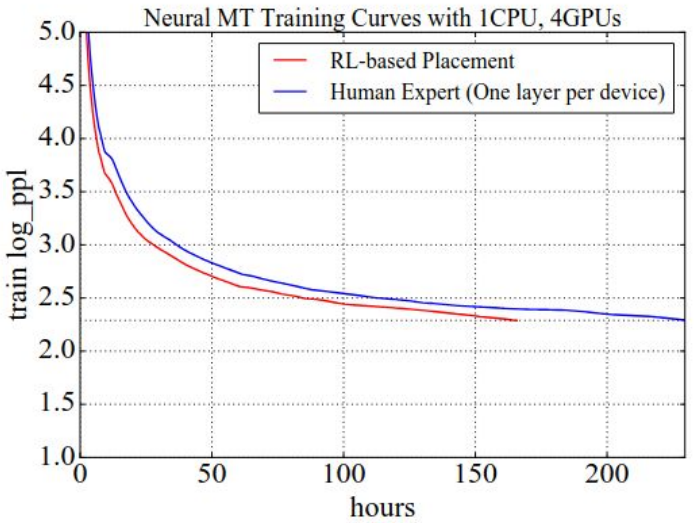


Speeds-up Full Training Process

27.8% speed-up in training time on Neural MT

RL placements balances workload better than human expert. The imbalance of humans is more significant without backpropagation, suggesting factors not considered in design by experts.

Training speedup (19.7%) in InceptionV3



Conclusion

Seminal work applying reinforcement learning to systems optimisation.

Enables end-to-end optimisation and learning complex tradeoffs.

Development of a domain specific sequence-to-sequence architecture for device placement

Surpasses heuristics partitioning solvers and human experts

Weaknesses

Generalisability wrt. Computation graph (Addanki et al. Placeto)

Generalisability wrt. Device and Hardware Available

Architecture and reinforcement learning algorithms are outdated. (Orthogonal). I would expect recent work to use GNNs (A graph placement methodology for fast chip design) and PPO.

Architecture unable to adapt to larger number of nodes, hence grouping heuristic. (A Hierarchical Mode for Device Placement) supports larger computation graphs and placements with higher granularity.