Device Placement Optimization with Reinforcement Learning

Azalia Mirhoseini, Hieu Pham, Quoc V. Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, Jeff Dean

What is device placement

- Consider a TensorFlow computational graph G, which consists of M operations $\{o_1, o_2, \ldots, o_M\}$, and a list of D available devices.

- A placement $P = \{p_1, p_2, \ldots, p_M\}$ is an assignment of an operation o_i to a device p_i.

Why device placement

- Trend toward many-device training, bigger models, larger batch sizes

- Growth in size and computational requirements of training and inference

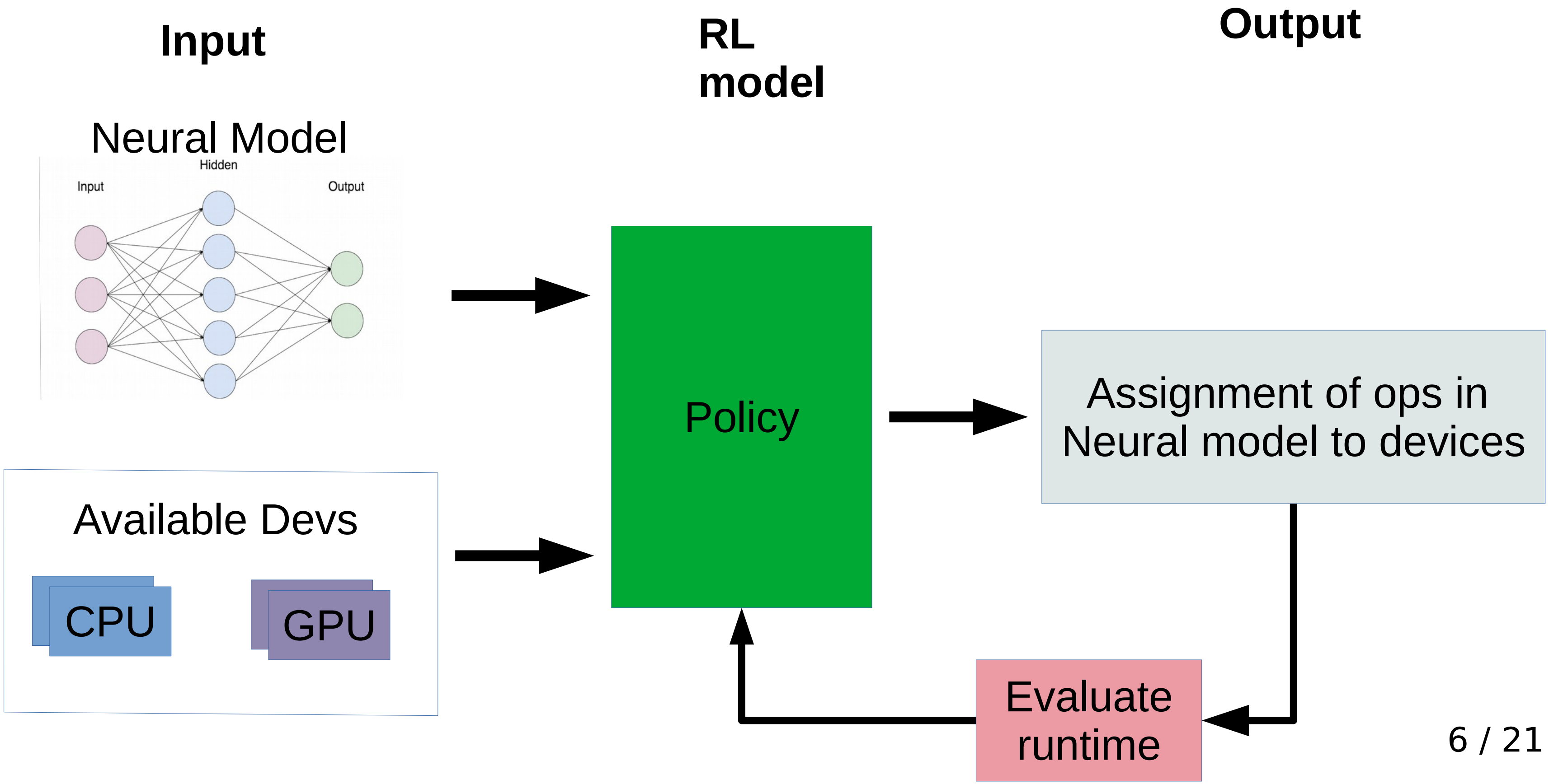
Typical approaches

- Use a heterogeneous distributed environment with a mixture of many CPUs and GPUs
- Often based on greedy heuristics
- Require deep understanding of devices: bandwidth, latency behavior
- Are not flexible enough and does not generalize well

ML for device placement

- ML is repeatedly replacing rule based heuristics
- RL can be applied to device placement
  - Effective search across large state and action spaces to find optimal solution
  - Automatic learning from underlying environment only based on reward function

RL based device placement



Problem formulation

$$J(\theta) = \mathbf{E}_{\mathcal{P} \sim \pi(\mathcal{P}|\mathcal{G};\theta)} \left[R\left(\mathcal{P}\right) | \mathcal{G} \right]$$

$J(\theta)$: expected runtime

θ : trainable parameters of policy

R : runtime

$\pi(P|G;\theta)$: policy

P : output placements

Training with REINFORCE

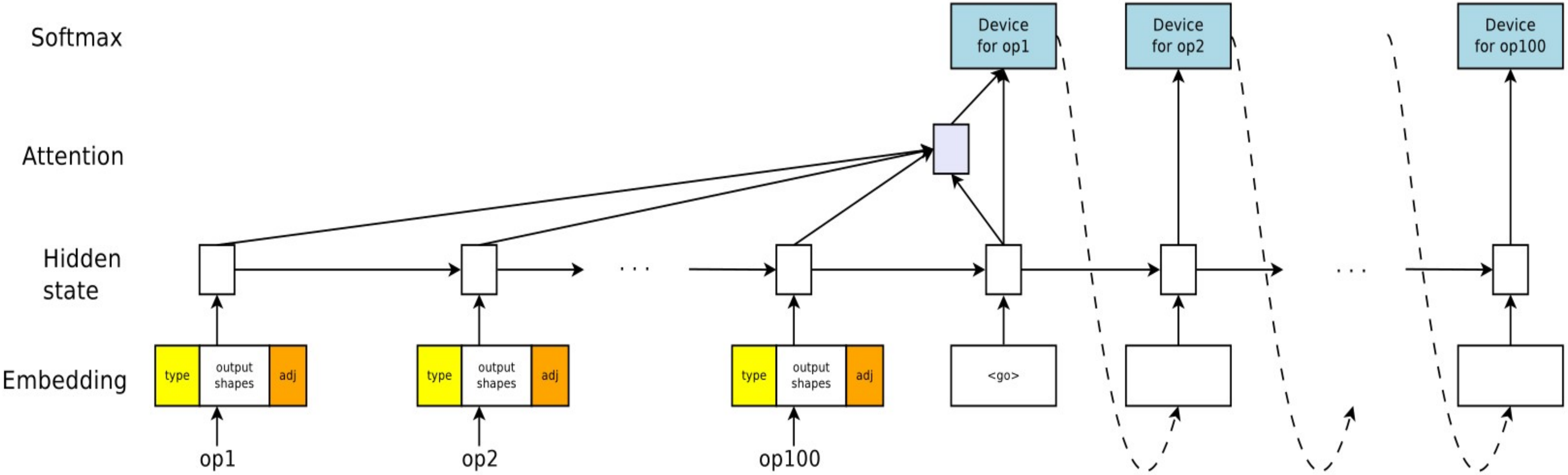
- Learn the network parameters using Adam optimizer based on policy gradients computed via the REINFORCE equation:

$$\nabla_\theta J(\theta) = \mathbf{E}_{\mathcal{P} \sim \pi(\mathcal{P}|\mathcal{G};\theta)} \left[R\left(\mathcal{P}\right) \cdot \nabla_\theta \log p\left(\mathcal{P}|\mathcal{G};\theta\right) \right]$$

- Use K placement samples to estimate policy gradients & use a baseline term B to reduce variance:

$$\nabla_\theta J(\theta) \approx \frac{1}{K} \sum_{i=1}^{K} \left(R\left(\mathcal{P}_i\right) - B \right) \cdot \nabla_\theta \log p\left(\mathcal{P}_i|\mathcal{G};\theta\right)$$

Model architecture

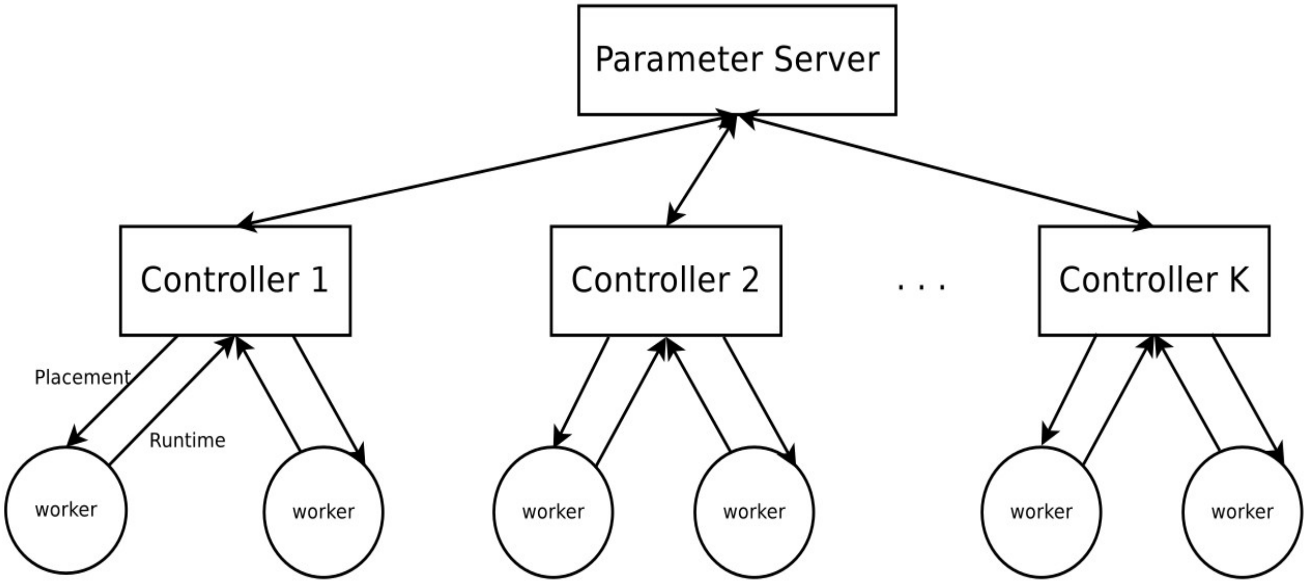


Challenges

- Vanishing
- Exploding gradient issue
- Large memory footprints

Model	#operations	#groups
RNNLM	8943	188
NMT	22097	280
Inception-V3	31180	83

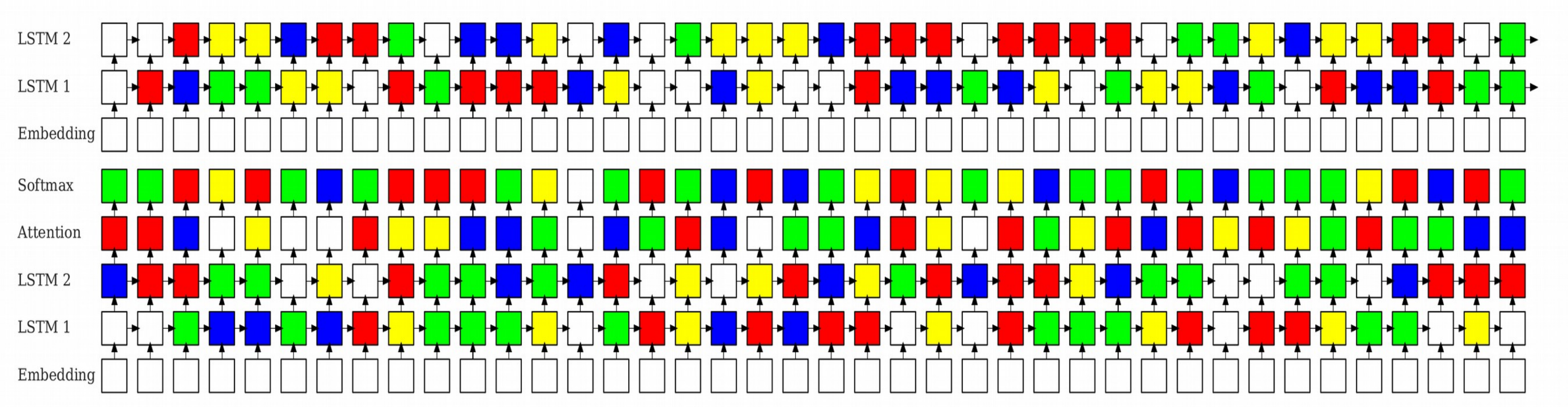
Distributed training



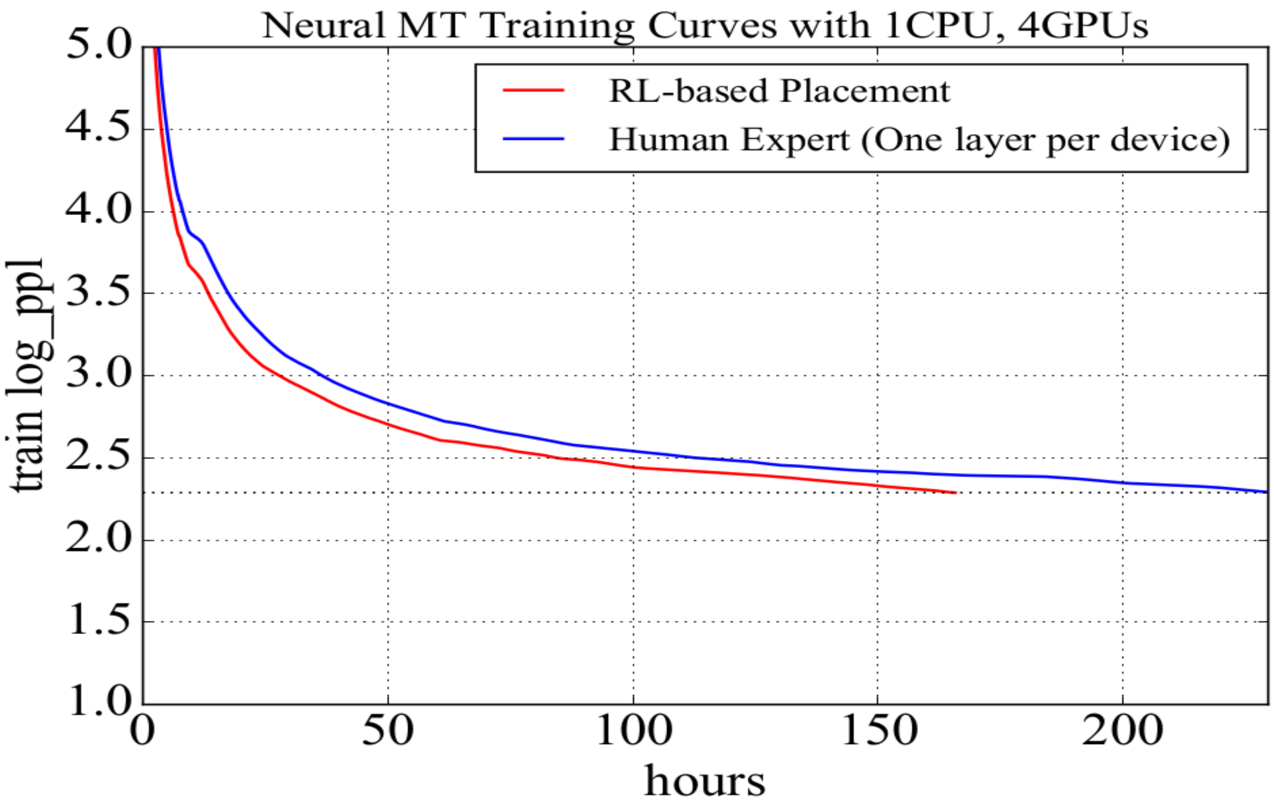
Experiments

- Recurrent Neural Language Model (RNNLM)
- Neural Machine Translation with attention mechanism(NMT)
- Inception-V3

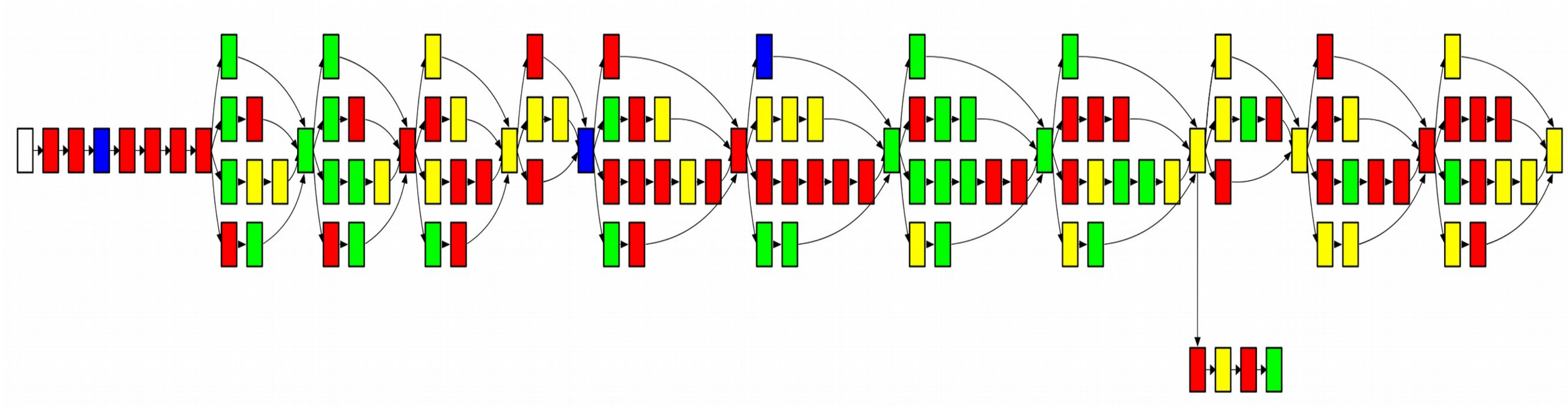
Learned placement on NMT



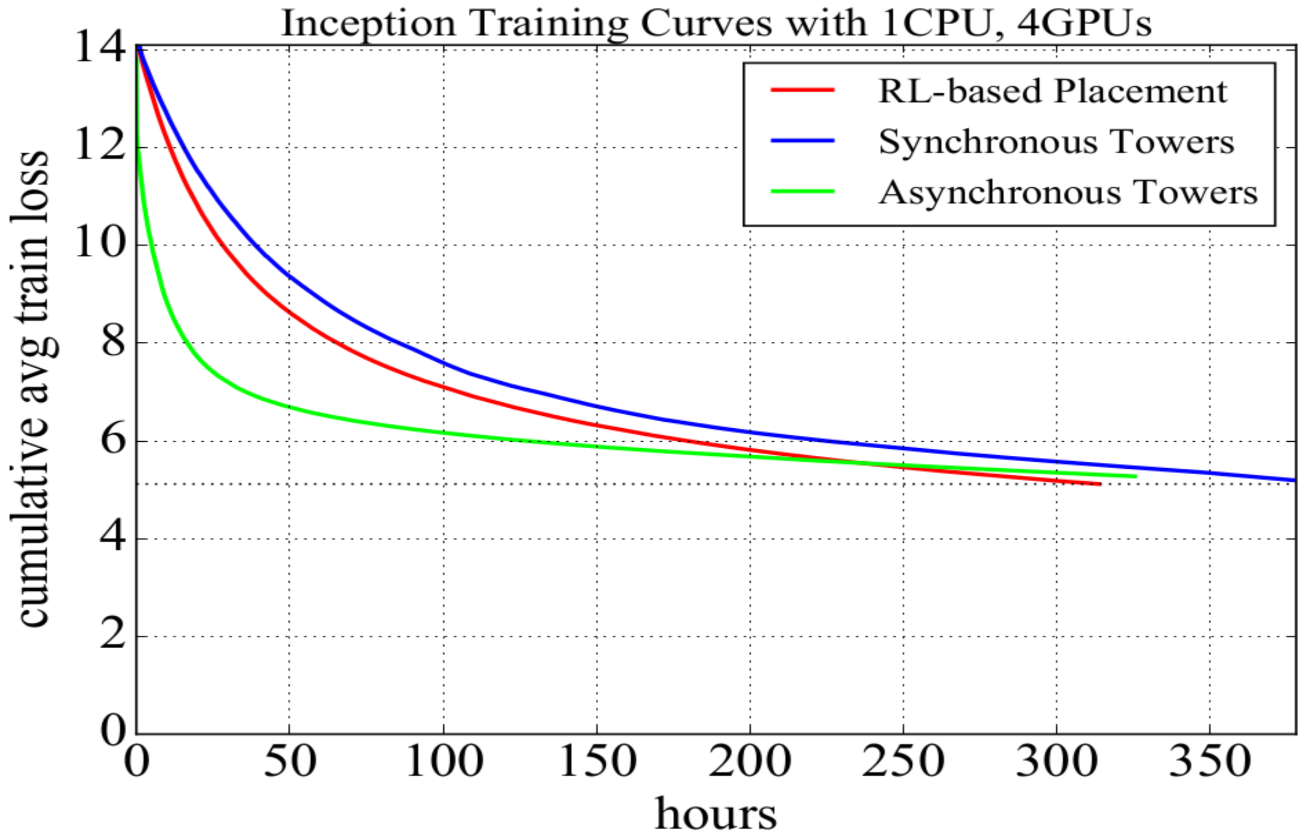
NMT end-to-end runtime



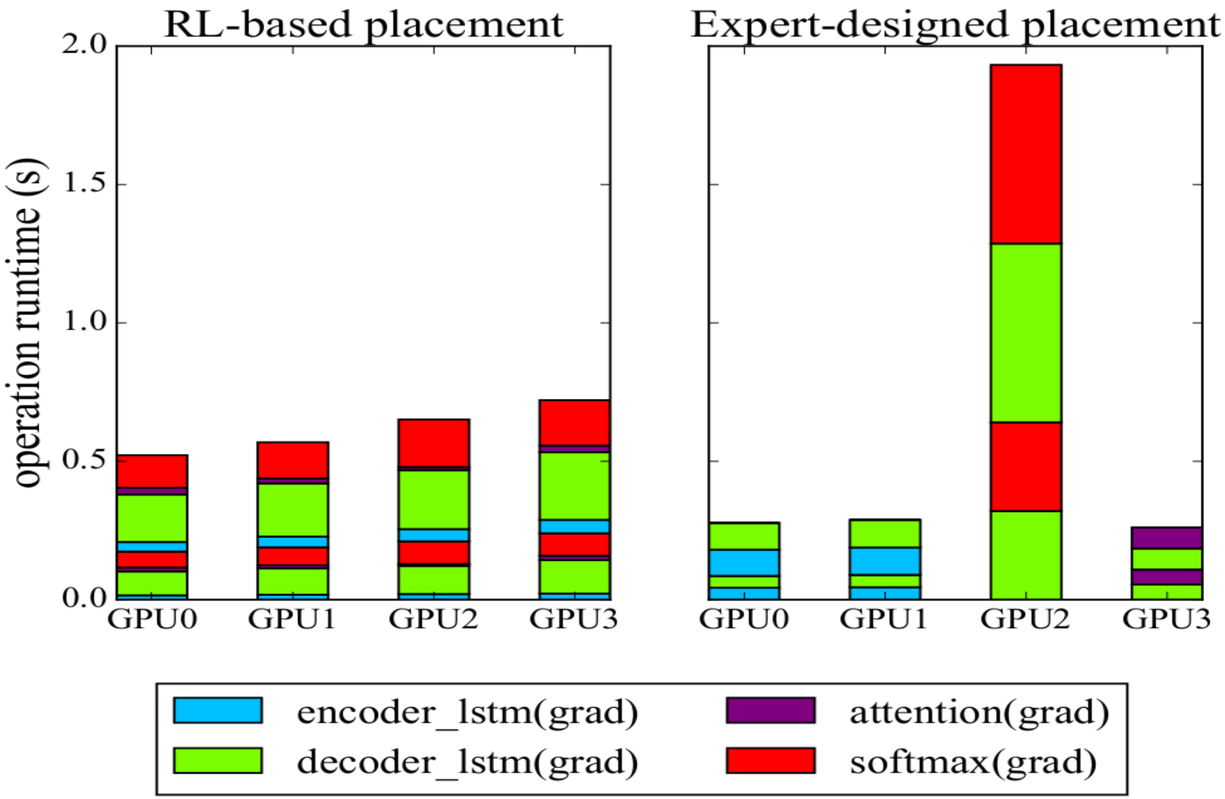
Learned placement on Inception-V3



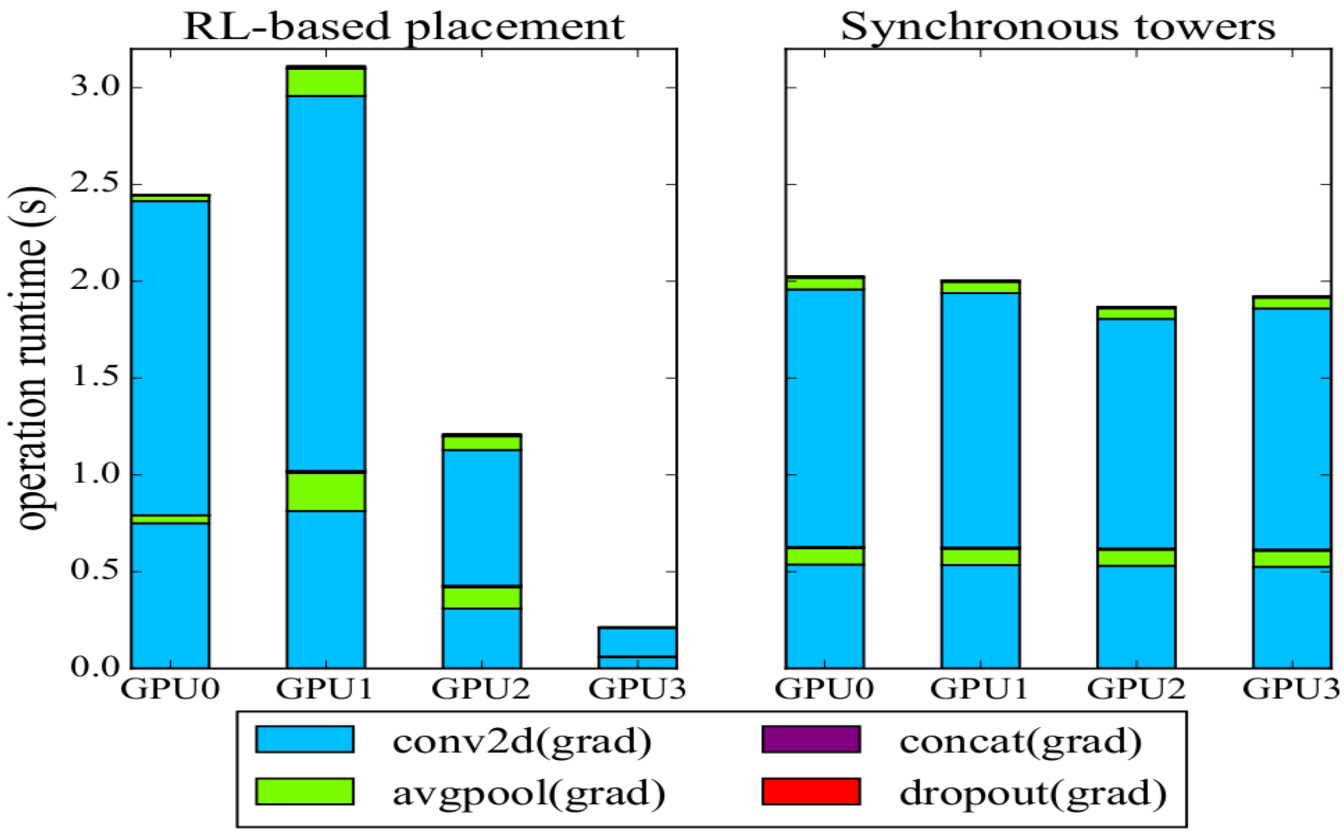
Inception-V3 end-to-end runtime



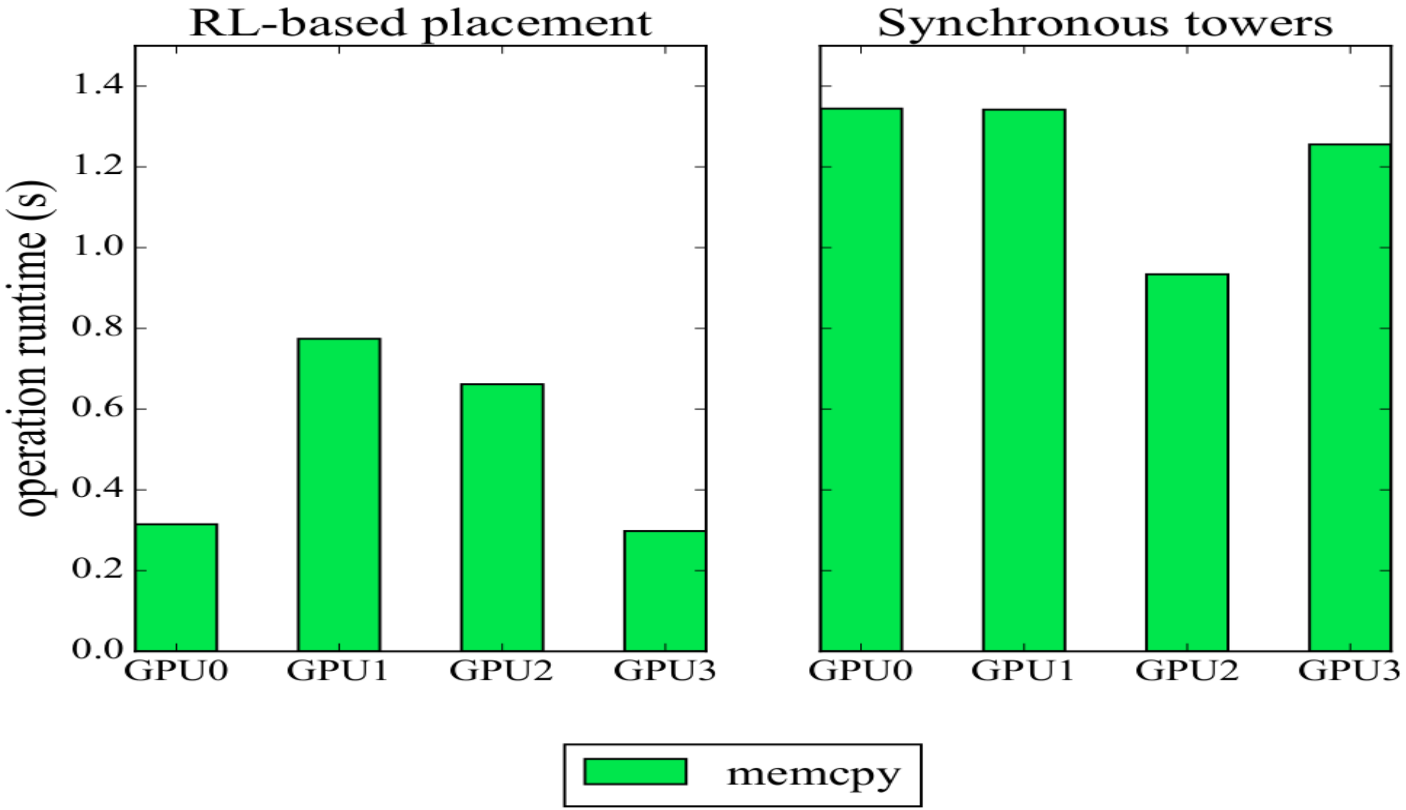
Profling on NMT



Profling on Inception-V3



Profling on Inception-V3



Running times (in seconds)

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%

Summary

- Propose a RL model to optimize device placements for neural networks
- Use policy gradient to learn parameters
- Policy finds non-trival assignment of operations to devices that outperform heuristic approaches
- Profiling of results show policy learns implicit trade-offs between computation and communication in hardware