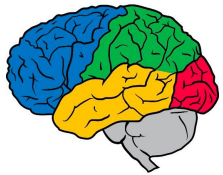
Device Placement Optimization with Reinforcement Learning

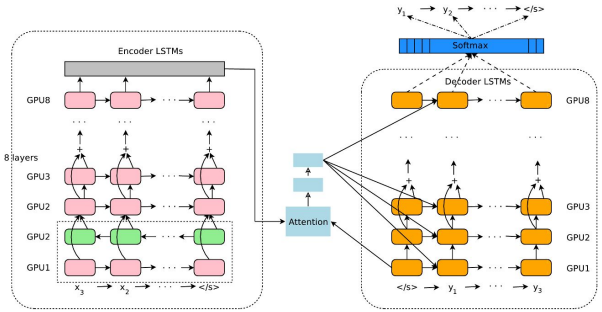
Azalia Mirhoseini, Anna Goldie, Hieu Pham, Benoit Steiner,
Mohammad Norouzi, Naveen Kumar, Rasmus Larsen, Yuefeng Zhou,
Quoc Le, Samy Bengio, and Jeff Dean



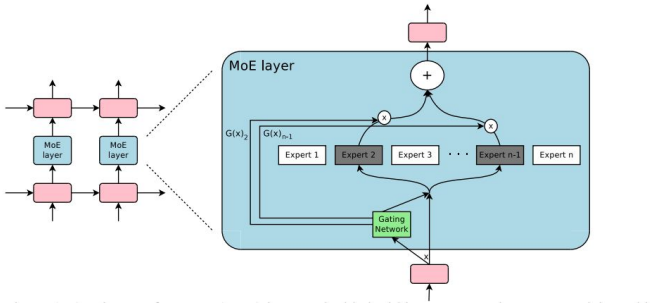
Google Brain

Why device placement?

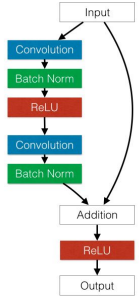
Trend towards many-device training, bigger models, larger batch sizes



Google neural machine translation'16
(trained on 128 GPUs)



Sparsely gated mixture of experts'17
(batch size = 8192, >130 billion parameters,
trained on 128 GPUs)



Training ImageNet in 1-hr'17
(batch size = 8192, trained on
256 GPUs)

Standard practice for device placement

- Often based on greedy heuristics
- Requires deep understanding of devices: nonlinear FLOPs, bandwidth, latency behavior
- Requires modeling parallelism and pipelining
- Does not generalize well

ML for device placement

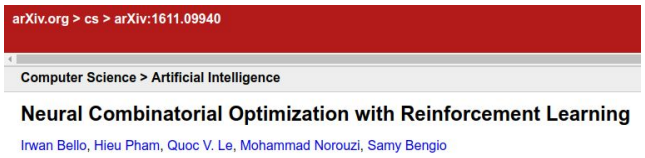
- ML is repeatedly replacing rule based heuristics

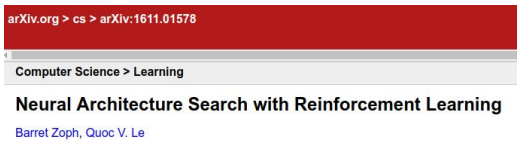
ML for device placement

- ML is repeatedly replacing rule based heuristics
- We show how RL can be applied to device placement
  - Effective search across large state and action spaces
  - Automated learning from environment only based on reward function (e.g. runtime of a program)

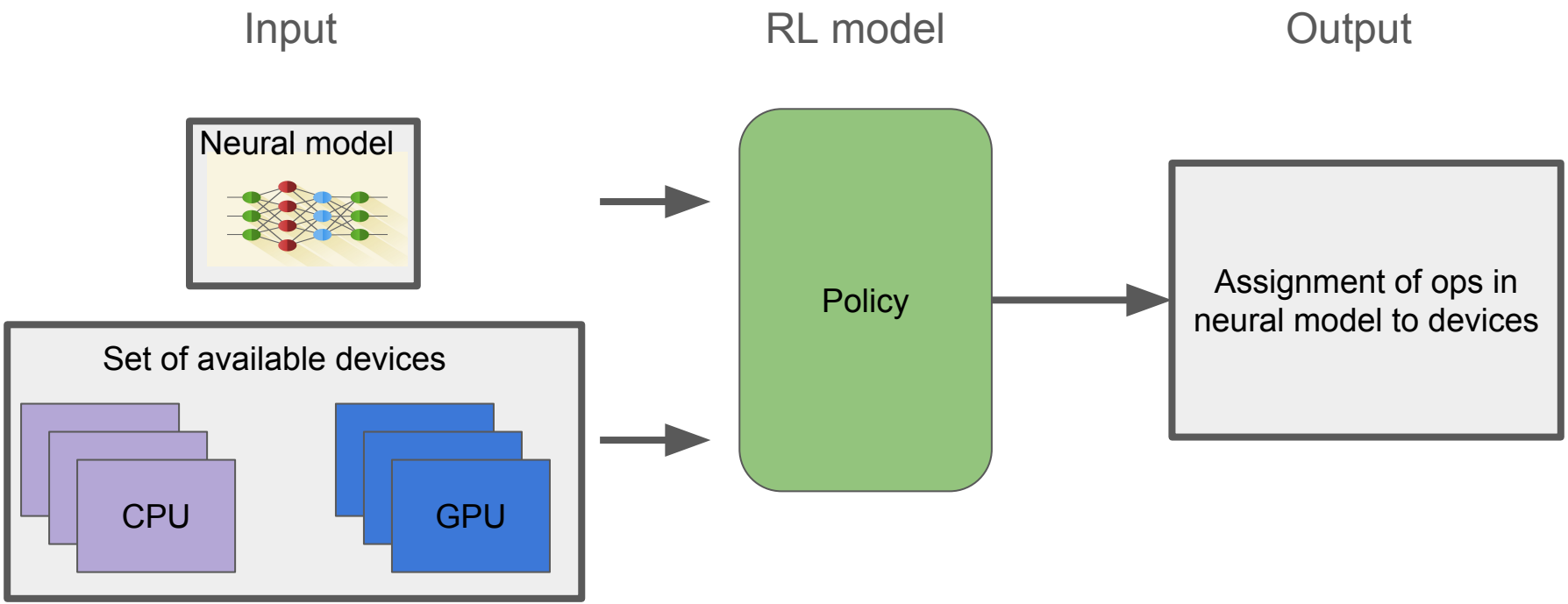
ML for device placement

- ML is repeatedly replacing rule based heuristics
- We show how RL can be applied to device placement
  - Effective search across large state and action spaces
  - Automated learning from environment only based on reward function (e.g. runtime of a program)
- We are inspired by success of ML for learning to search

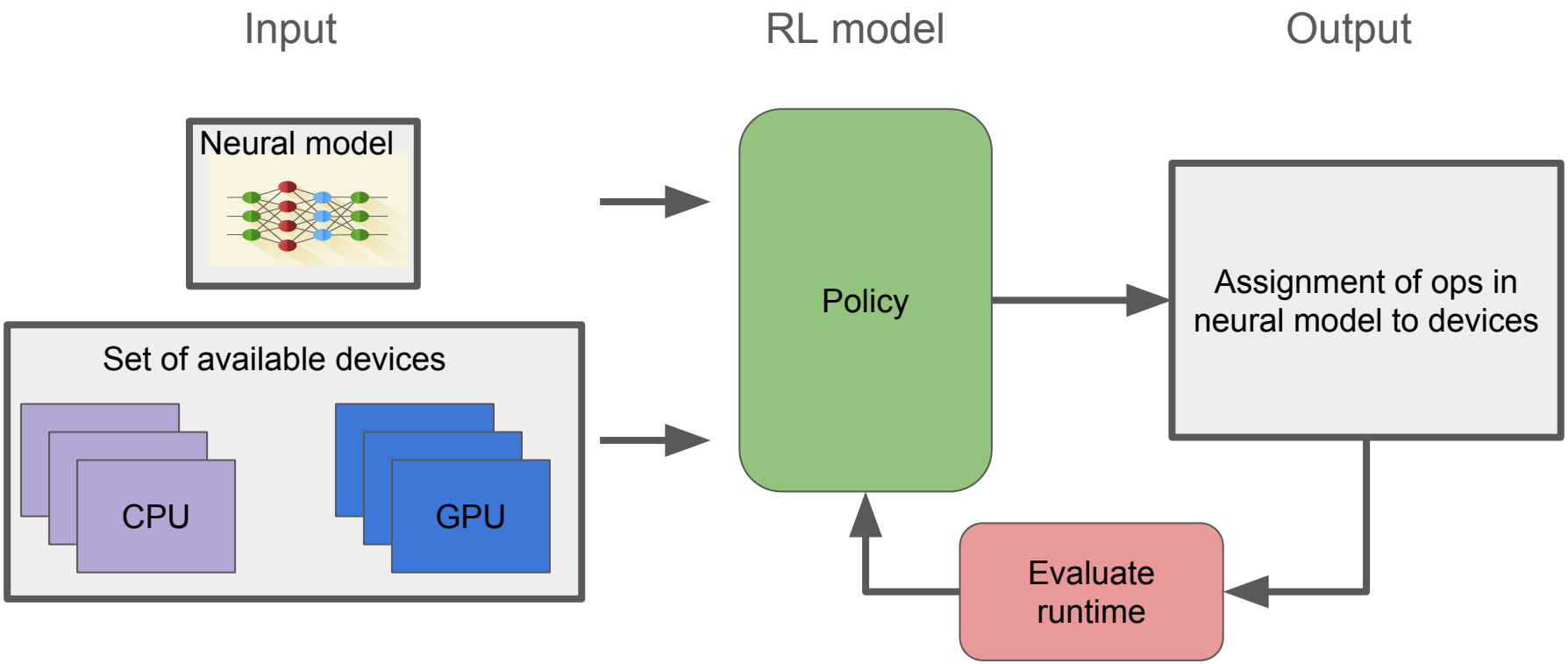




Posing device placement as an RL problem



Posing device placement as an RL problem



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

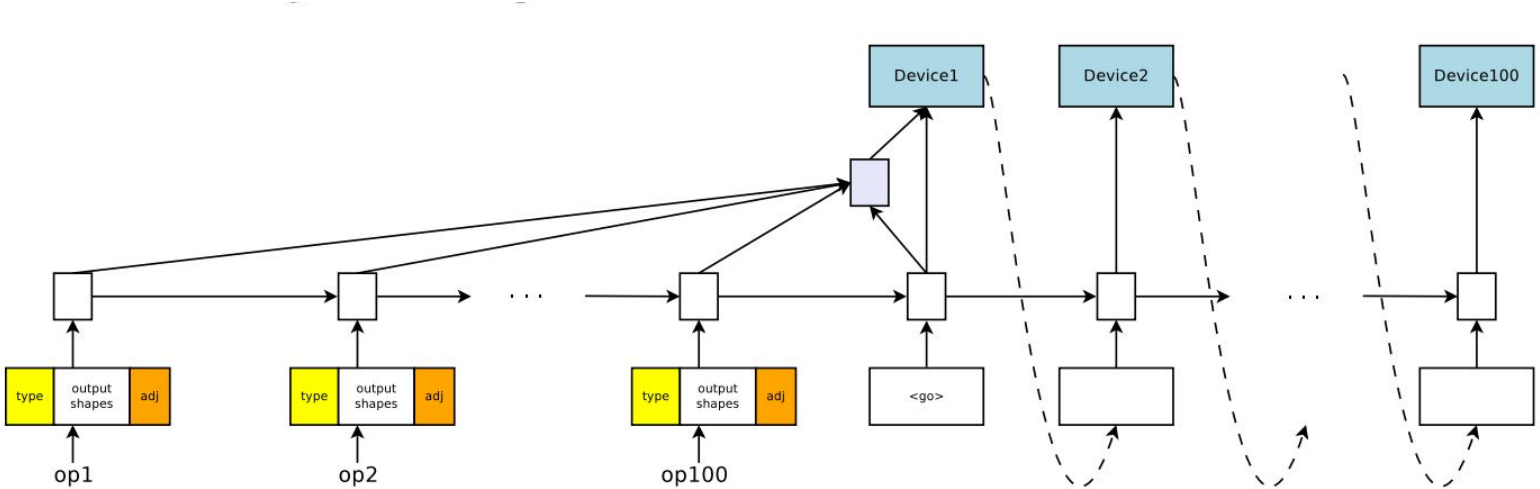
R: runtime

$\pi(\mathcal{P}|g;\theta)$: policy

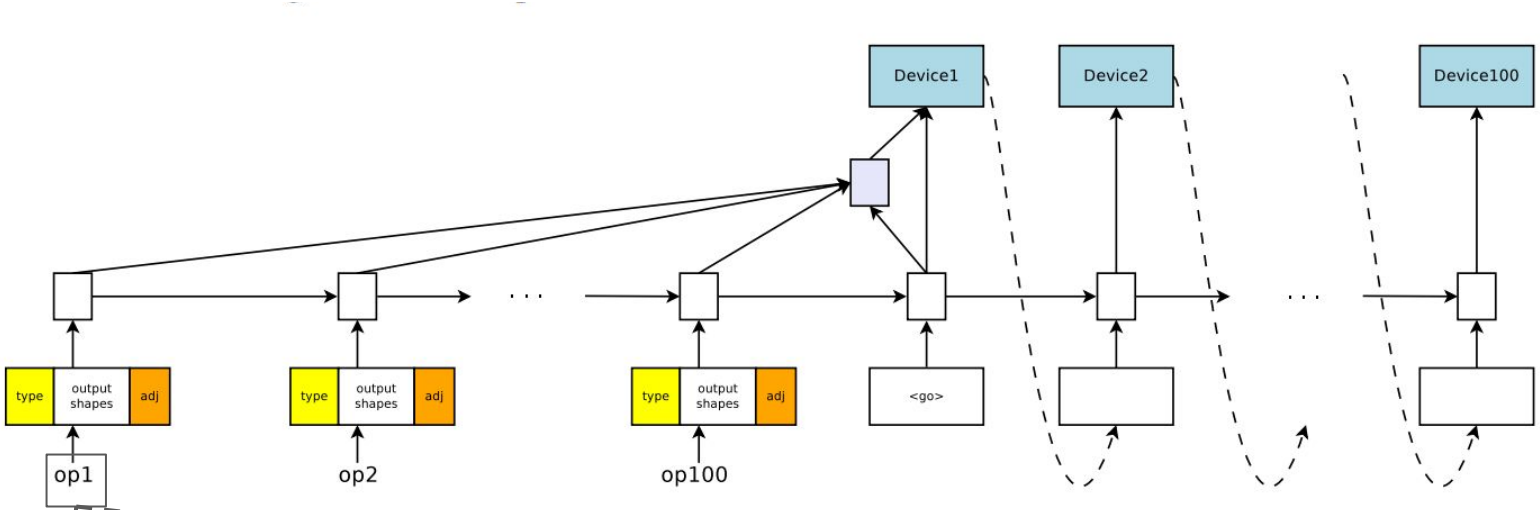
g : input neural graph

$\mathcal{P}$: output placements $\in$ {1,2,..,num_ops}$^{\text{num\_devices}}$

Policy architecture



Policy architecture



Op types, e.g., Sum, Conv2d, MatMul
Op output shapes
Op adjacency information

Training with REINFORCE

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

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

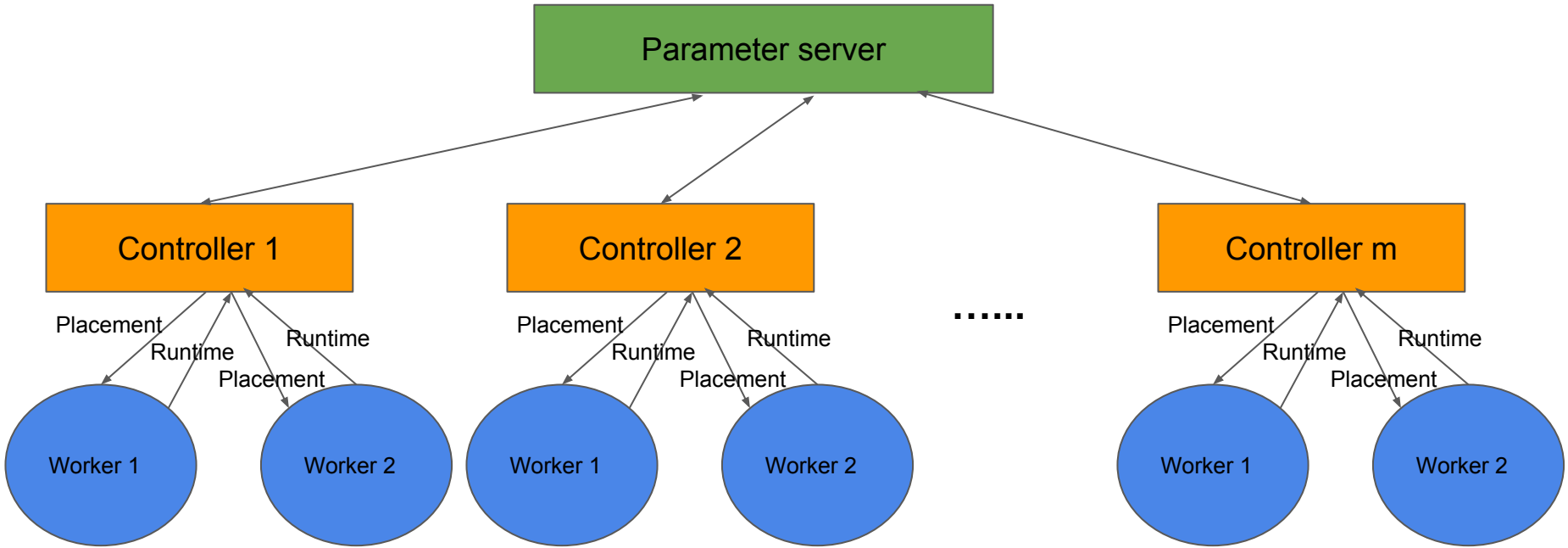
Training with REINFORCE

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

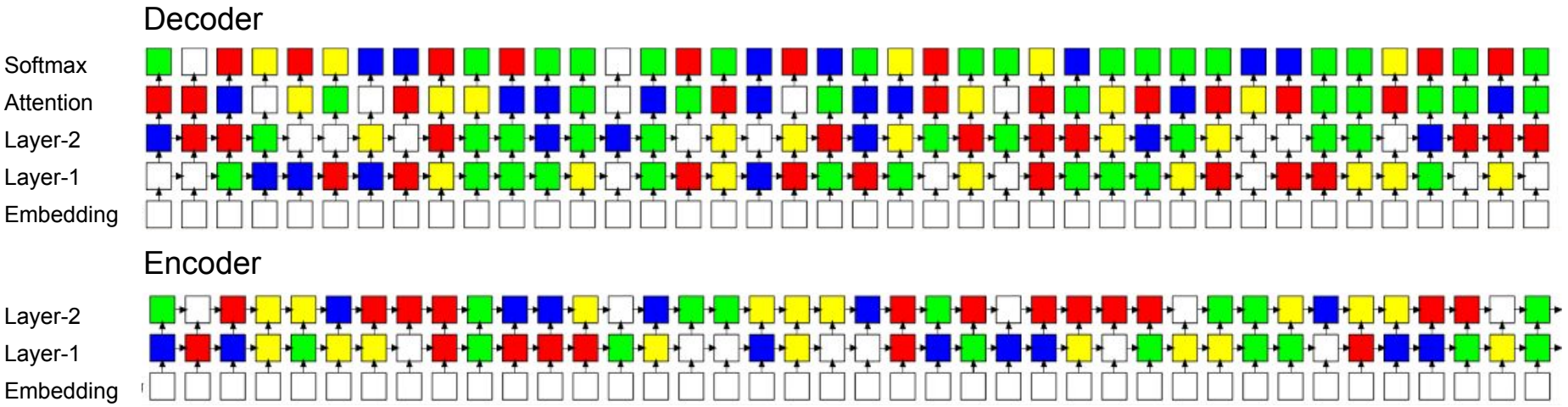
$$\nabla_\theta J(\theta) = \mathbf{E}_{\mathcal{P} \sim \pi(\mathcal{P}|\mathcal{G};\theta)} \left[R\left(\mathcal{P}\right) \cdot \nabla_\theta \log p\left(\mathcal{P}|\mathcal{G};\theta\right) \right]$$

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

Distributed training

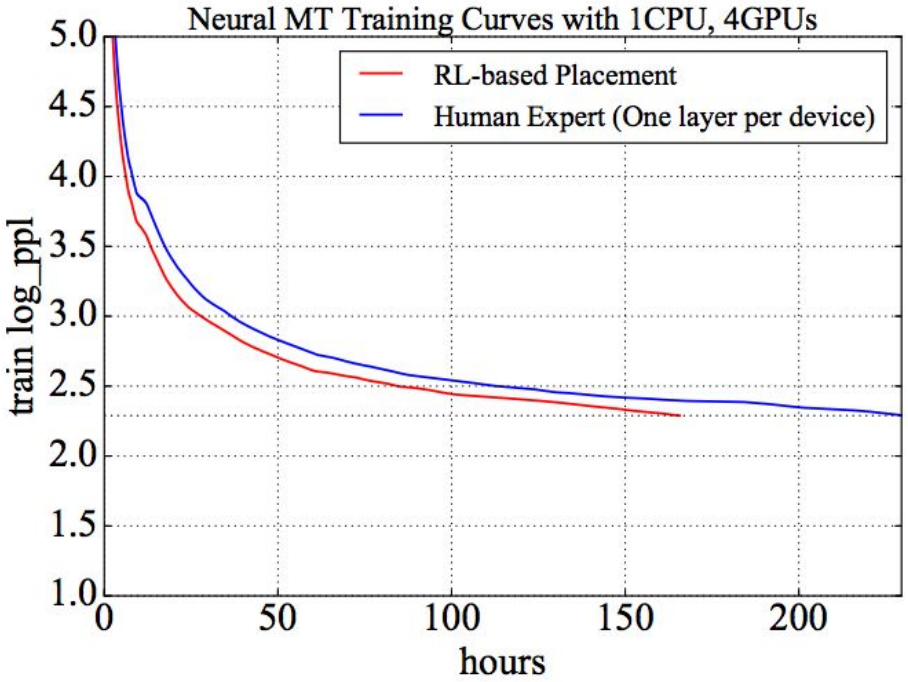


Learned placement on Neural Machine Translation (NMT)



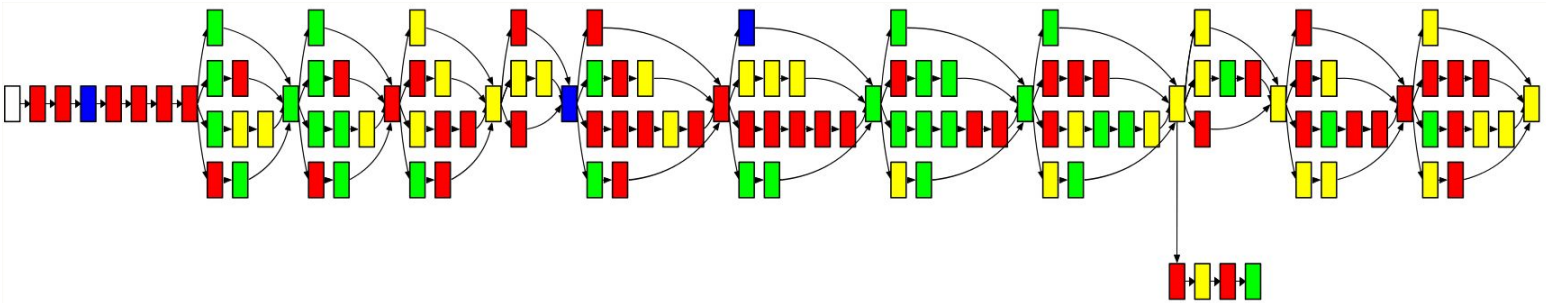
White represents CPU (Ixion Haswell 2300)
Each other color represents a separate GPU (Nvidia Tesla K80)
Searching over a space of 5^280 possible assignments

NMT end to end training



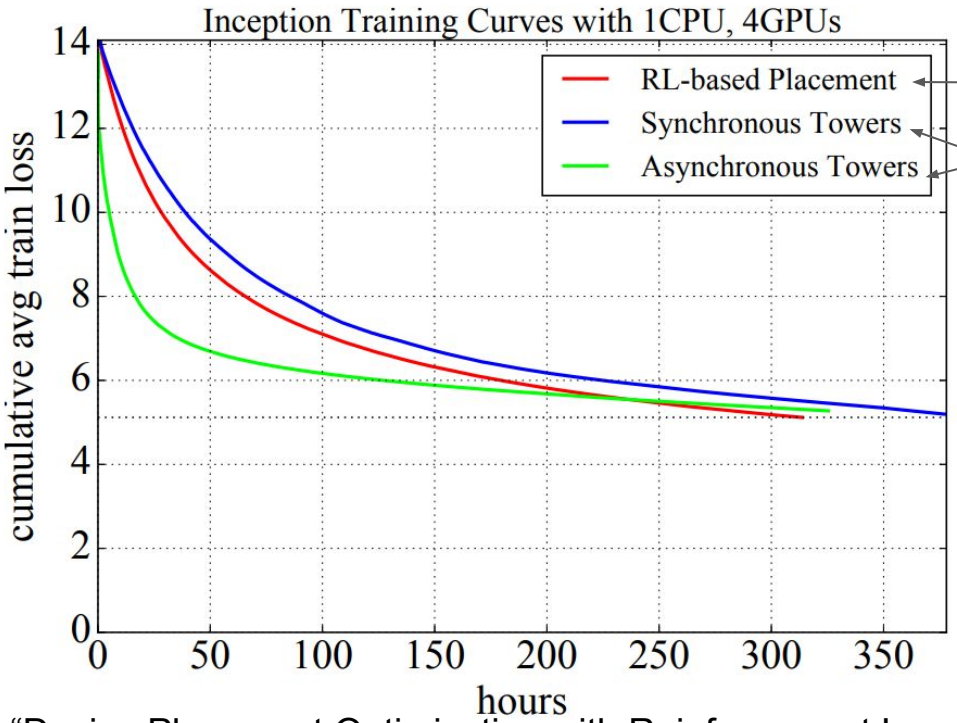
Mirhoseini, Pham et al., “Device Placement Optimization with Reinforcement Learning”, ICML’17

Learned placement on Inception-V3



White represents CPU (Ixion Haswell 2300)
Each other color represents a separate GPU (Nvidia Tesla K80)
Searching over a space of 5^83 possible assignments

Inception-V3 end to end training

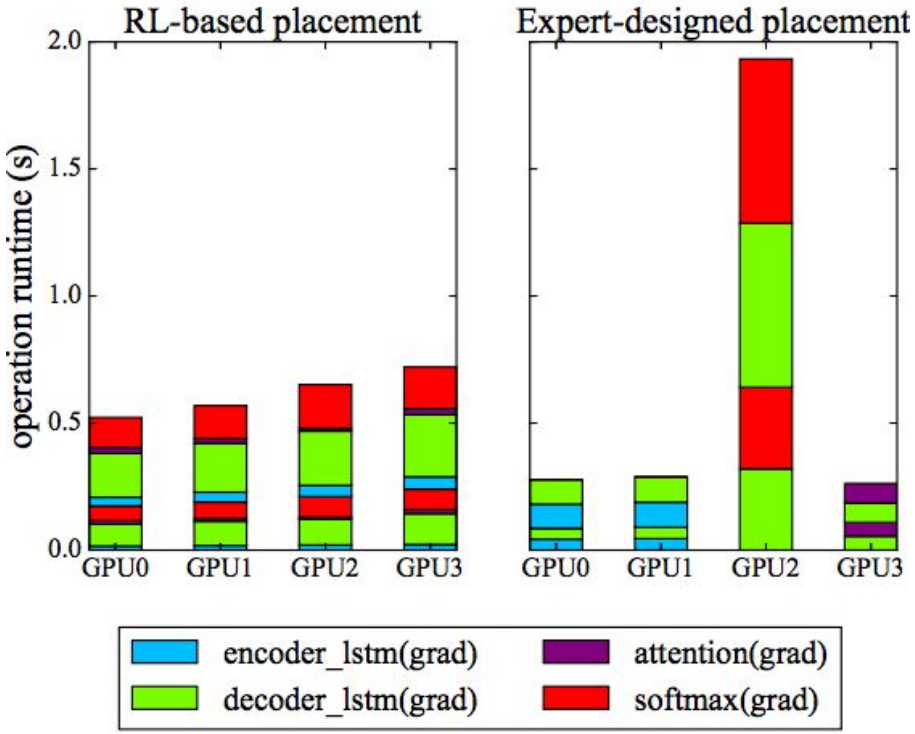


Model-parallelism with batch 128

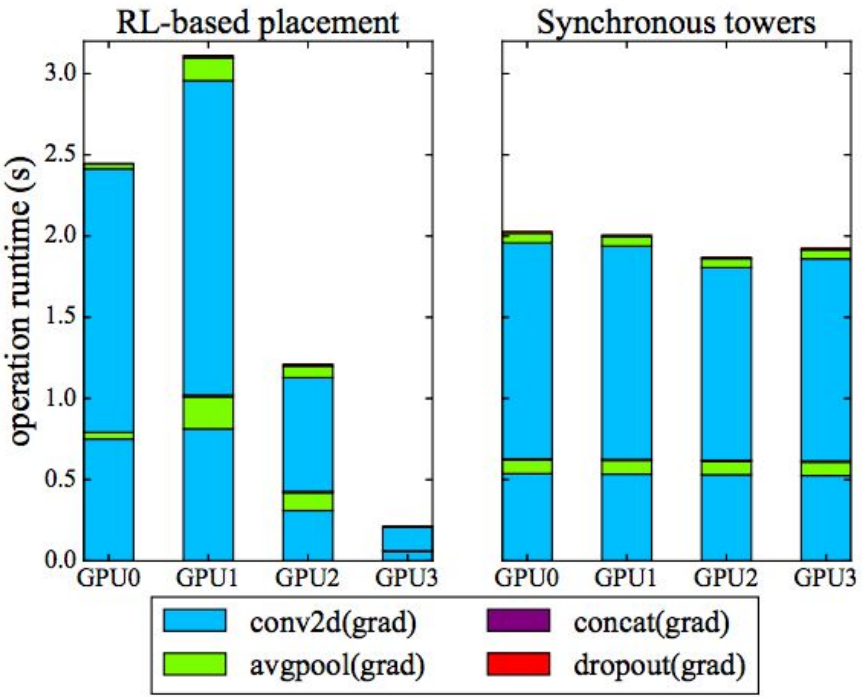
Data-parallelism, 4 towers each with batch 32

Mirhoseini, Pham, et al., “Device Placement Optimization with Reinforcement Learning”, ICML’17

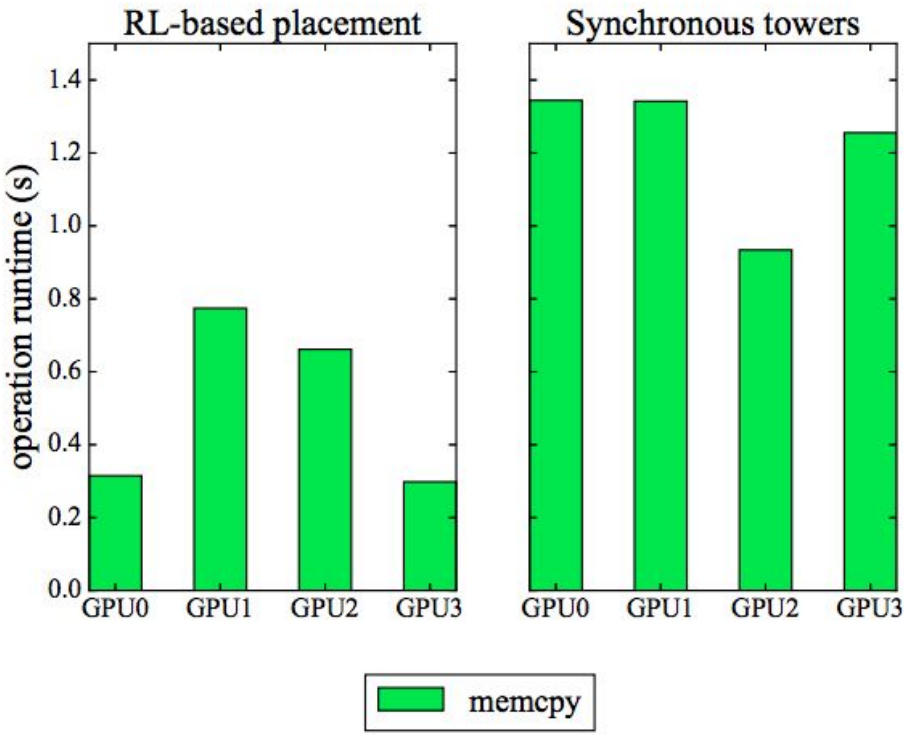
Profiling placement on NMT



Profiling placement on Inception-V3

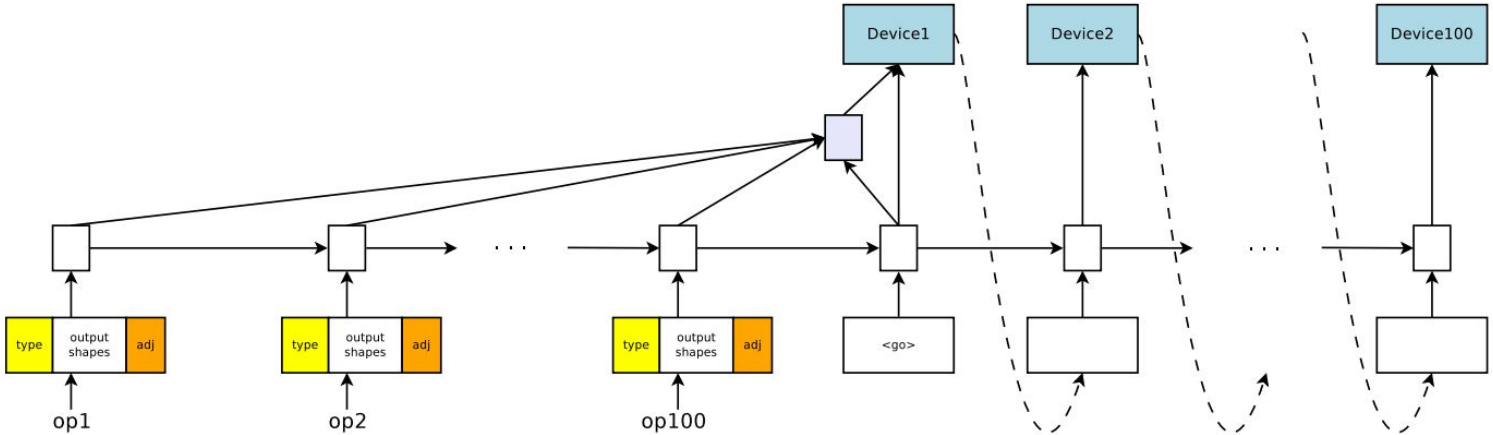


Profiling placement on Inception-V3

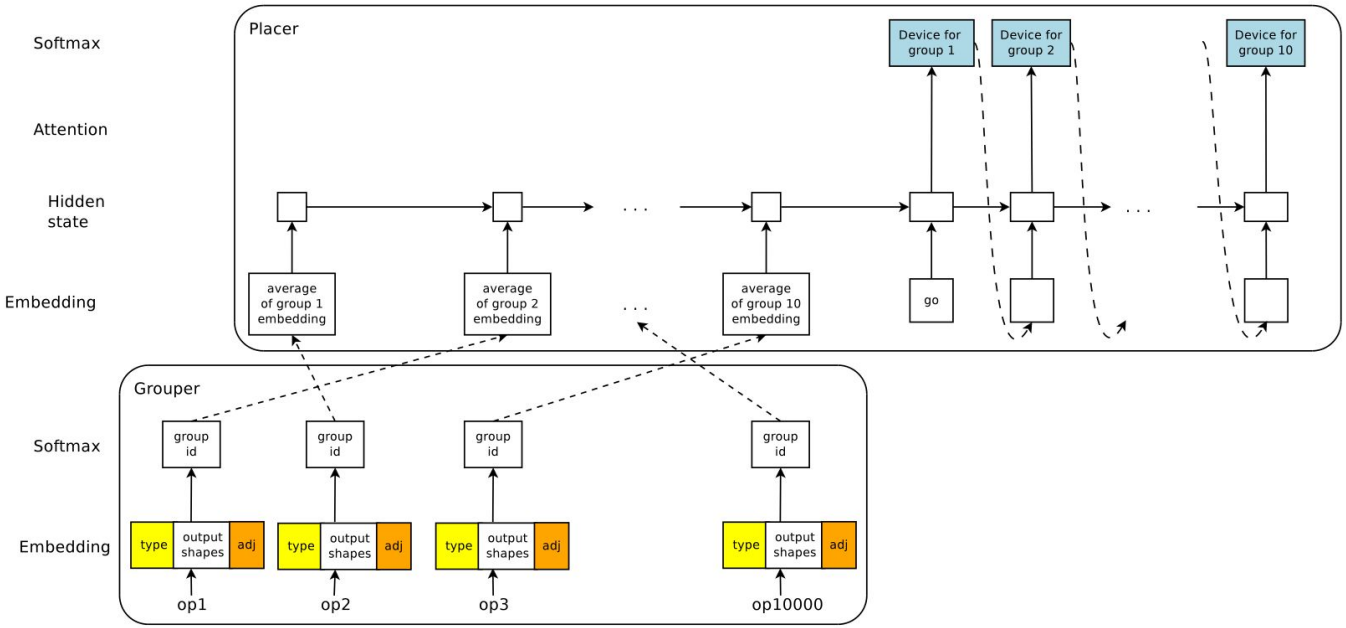


Shortcomings of initially proposed model

- Seq2seq models cannot be unrolled for more than few hundred steps
- Most TensorFlow graphs contain tens of thousands of operations
- Manual grouping of operations hampers scalability



An end-to-end hierarchical placement model



Problem formulation for hierarchical placement

Objective: Minimize expected runtime for predicted placement d

$$J(\theta_g, \theta_d) = \mathbf{E}_{\mathbf{P}(\mathbf{d};\theta_\mathbf{g},\theta_\mathbf{d})}[R_d] = \sum_{g\sim\pi_g} \sum_{d\sim\pi_d} p(g;\theta_g)p(d|g;\theta_d)R_d$$

$J(\theta_g, \theta_d)$: expected runtime

θ_g: trainable parameters of Grouper

θ_d: trainable parameters of Placer

R_d: runtime for placement d

Problem formulation for hierarchical placement

Objective: Minimize expected runtime for predicted placement d

$$J(\theta_g, \theta_d) = \mathbf{E}_{\mathbf{P}(\mathbf{d};\theta_\mathbf{g},\theta_\mathbf{d})}[R_d] = \sum_{g\sim\pi_g} \sum_{d\sim\pi_d} p(g;\theta_g)p(d|g;\theta_d)R_d$$

$J(\theta_g, \theta_d)$: expected runtime

θ_g: trainable parameters of Grouper

θ_d: trainable parameters of Placer

R_d: runtime for placement d

Problem formulation for hierarchical placement

$$J(\theta_g, \theta_d) = \mathbf{E}_{\mathbf{P}(\mathbf{d};\theta_\mathbf{g},\theta_\mathbf{d})}[R_d] = \sum_{g\sim\pi_g} \sum_{d\sim\pi_d} p(g;\theta_g) p(d|g;\theta_d) R_d$$

Probability of predicted group assignment of operations

Problem formulation for hierarchical placement

$$J(\theta_g, \theta_d) = \mathbf{E}_{\mathbf{P}(\mathbf{d};\theta_\mathbf{g},\theta_\mathbf{d})}[R_d] = \sum_{g\sim\pi_g} \sum_{d\sim\pi_d} p(g;\theta_g) p(d|g;\theta_d) R_d$$

Probability of predicted device placement conditioned on grouping results

Gradient update for Grouper

$$J(\theta_g, \theta_d) = \mathbf{E}_{\mathbf{P(d};\theta_\mathbf{g},\theta_\mathbf{d})}[R_d] = \sum_{g \sim \pi_g} \sum_{d \sim \pi_d} p(g;\theta_g) p(d|g;\theta_d) R_d$$

$$\nabla_{\theta_g} J(\theta_g, \theta_d) = \sum_{g \sim \pi_g} \nabla_{\theta_g} p(g;\theta_g) \sum_{d \sim \pi_d} p(d|g;\theta_d) R_d$$

$$\approx \frac{1}{m} \sum_{g_i \sim \pi_g}^{1 \le i \le m} \nabla_{\theta_g} \log p(g_i;\theta_g) . \frac{1}{k} (\sum_{d_j \sim \pi_d}^{1 \le j \le k} R_{d_j})$$

Derivative w.r.t. parameters of Grouper

Gradient update for Placer

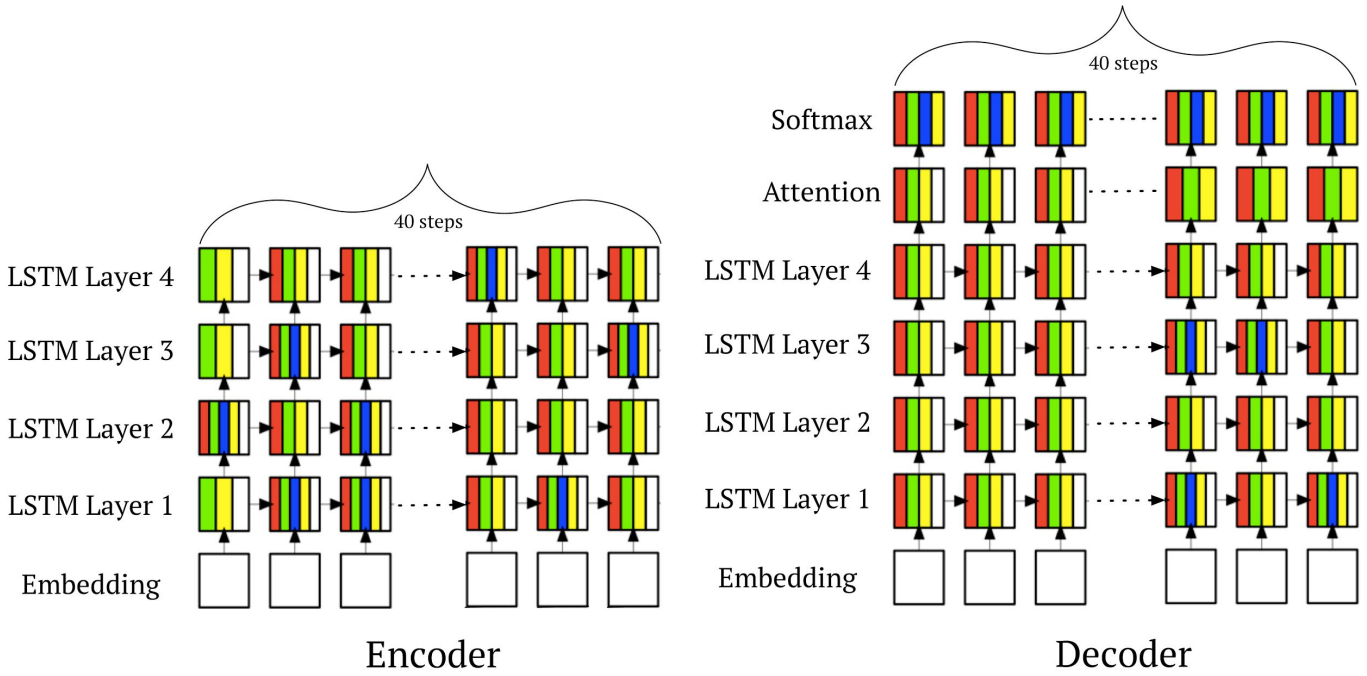
$$J(\theta_g, \theta_d) = \mathbf{E}_{\mathbf{P}(\mathbf{d};\theta_\mathbf{g},\theta_\mathbf{d})}[R_d] = \sum_{g\sim\pi_g} \sum_{d\sim\pi_d} p(g;\theta_g) p(d|g;\theta_d) R_d$$

$$\nabla_{\theta_d} J(\theta_g, \theta_d) = \sum_{d\sim\pi_d} \sum_{g\sim\pi_g} p(g;\theta_g) \nabla_{\theta_d} p(d|g;\theta_d) R_d$$

$$\approx \frac{1}{k} \sum_{d_j\sim\pi_d}^{1\le j\le k} \frac{1}{m} \Big(\sum_{g_i\sim\pi_g}^{1\le i\le m} \nabla_{\theta_d} \log p(d_j|g_i;\theta_d) R_{d_j} \Big)$$

Derivative w.r.t. parameters of Placer

Learned placements on NMT



White represents CPU (Ixion Haswell 2300)
Each other color represents a separate GPU (Nvidia Tesla K80)
Searching over a space of ~5^40000 possible assignments

Results (runtime in seconds)

Tasks	CPU Only	GPU Only	#GPUs	Human Expert	Scotch	MinCut	Hierarchical Planner	Runtime Reduction
Inception-V3	0.61	0.15	2	0.15	0.93	0.82	**0.13**	16.3%
ResNet	-	1.18	2	1.18	6.27	2.92	**1.18**	0%
RNNLM	6.89	1.57	2	1.57	5.62	5.21	**1.57**	0%
NMT (2-layer)	6.46	OOM	2	2.13	3.21	5.34	**0.84**	60.6%
NMT (4-layer)	10.68	OOM	4	3.64	11.18	11.63	**1.69**	53.7%
NMT (8-layer)	11.52	OOM	8	**3.88**	17.85	19.01	4.07	-4.9%

Results (runtime in seconds)

Tasks	CPU Only	GPU Only	#GPUs	Human Expert	Scotch	MinCut	Hierarchical Planner	Runtime Reduction
Inception-V3	0.61	0.15	2	0.15	0.93	0.82	**0.13**	16.3%
ResNet	-	1.18	2	1.18	6.27	2.92	**1.18**	0%
RNNLM	6.89	1.57	2	1.57	5.62	5.21	**1.57**	0%
NMT (2-layer)	6.46	OOM	2	2.13	3.21	5.34	**0.84**	60.6%
NMT (4-layer)	10.68	OOM	4	3.64	11.18	11.63	**1.69**	53.7%
NMT (8-layer)	11.52	OOM	8	**3.88**	17.85	19.01	4.07	-4.9%

Results (runtime in seconds)

Tasks	CPU Only	GPU Only	#GPUs	Human Expert	Scotch	MinCut	Hierarchical Planner	Runtime Reduction
Inception-V3	0.61	0.15	2	0.15	0.93	0.82	**0.13**	16.3%
ResNet	-	1.18	2	1.18	6.27	2.92	**1.18**	0%
RNNLM	6.89	1.57	2	1.57	5.62	5.21	**1.57**	0%
NMT (2-layer)	6.46	OOM	2	2.13	3.21	5.34	**0.84**	60.6%
NMT (4-layer)	10.68	OOM	4	3.64	11.18	11.63	**1.69**	53.7%
NMT (8-layer)	11.52	OOM	8	**3.88**	17.85	19.01	4.07	-4.9%

Summary

- Pose device placement as an RL problem
- Use policy gradient to learn placements
- Policy finds non-trivial assignment of operations to devices that outperform heuristic approaches
- Profiling of results show policy learns implicit trade-offs between computational and communication capabilities of underlying devices