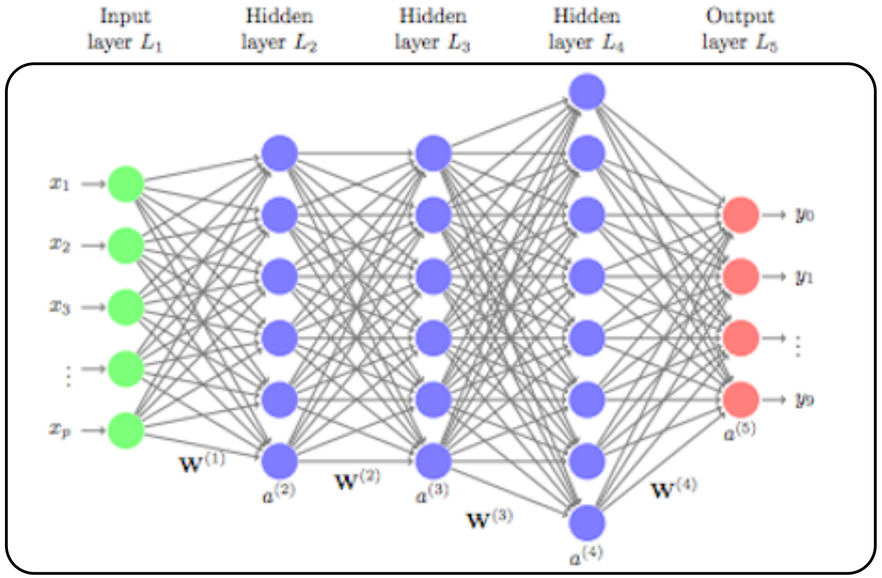
Accelerated Device Placement Optimization with Contrastive Learning

Hao Lan, Li Chen, Baochun Li
University of Toronto
hao.lan@mail.utoronto.ca

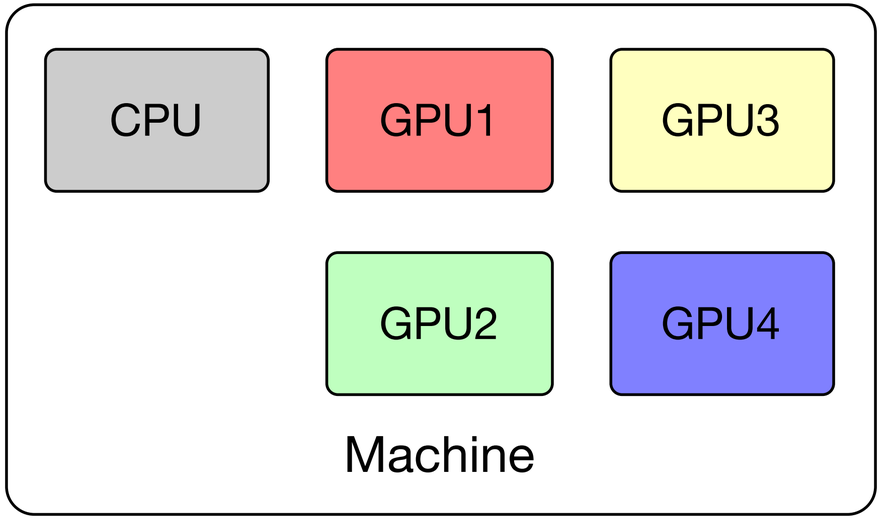
What is device placement?

Background

- Problem: Large DNNs cannot fit into single GPU (mem limitation).
- Solution: Model parallelism, Partition a DNN across multiple GPUs



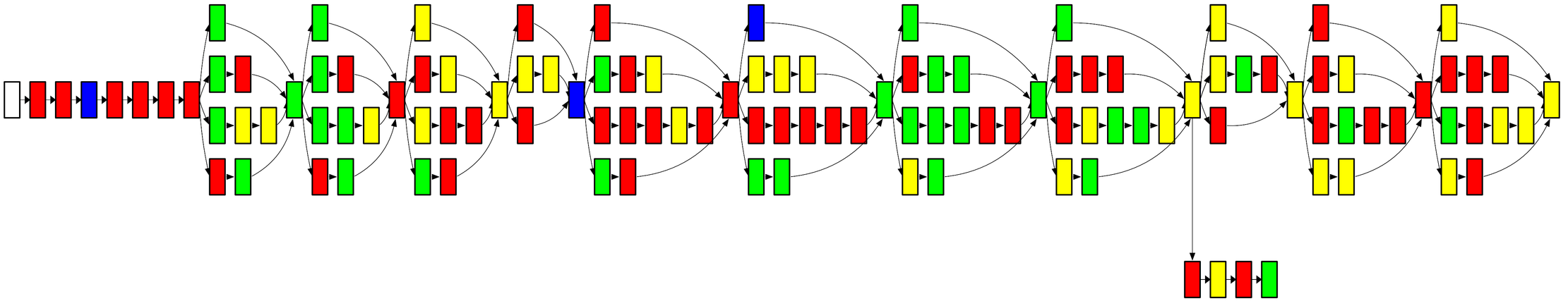
Deep Neural Networks



Computation Resources

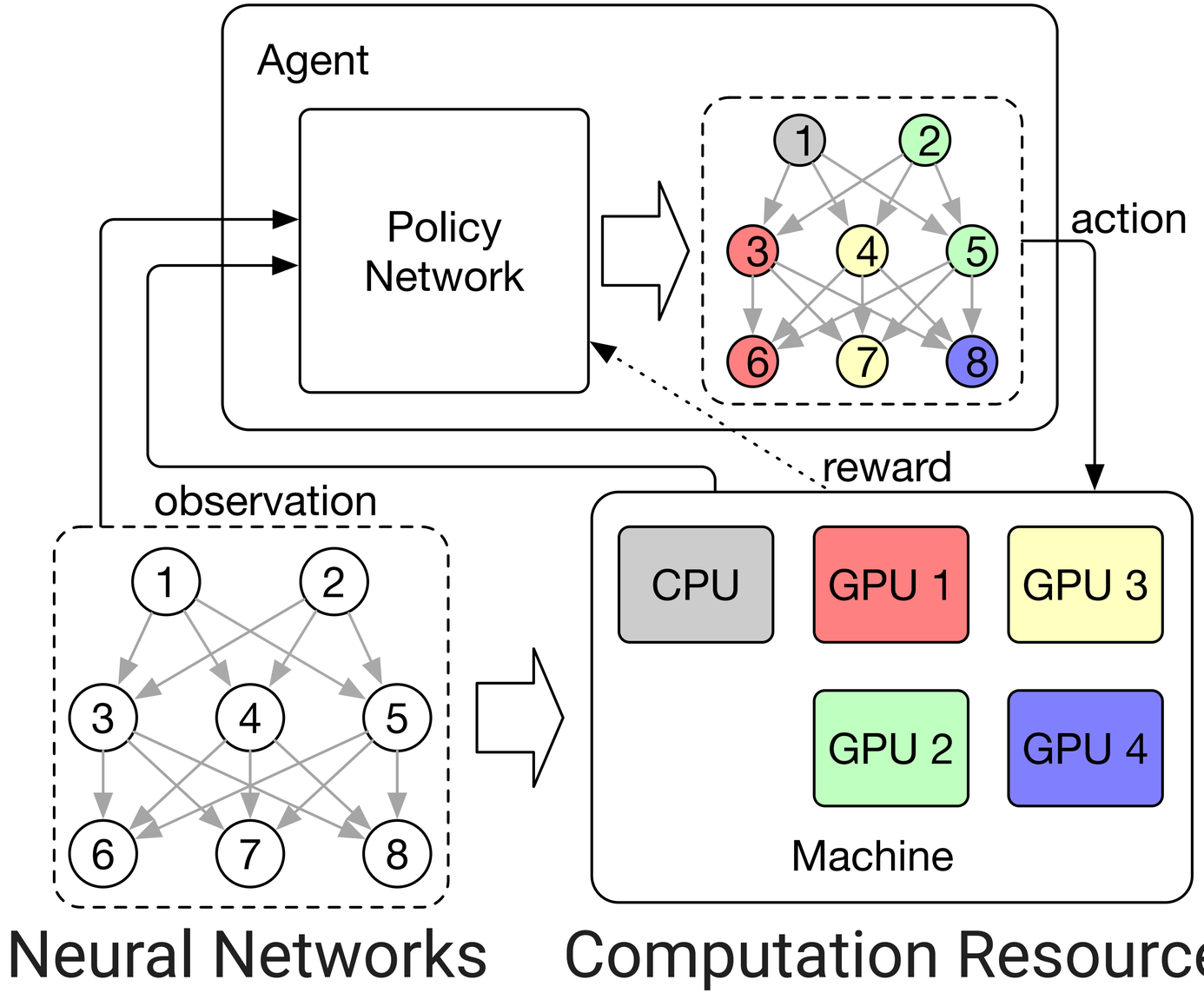
Device placement with deep reinforcement learning

Objective: to find the **best** way to assign devices to operations to minimize training time

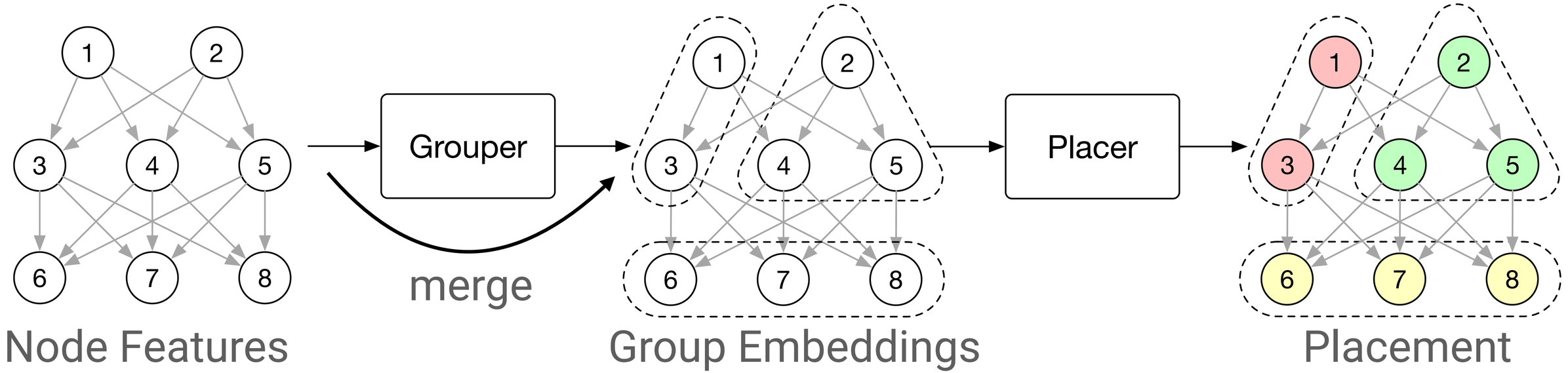


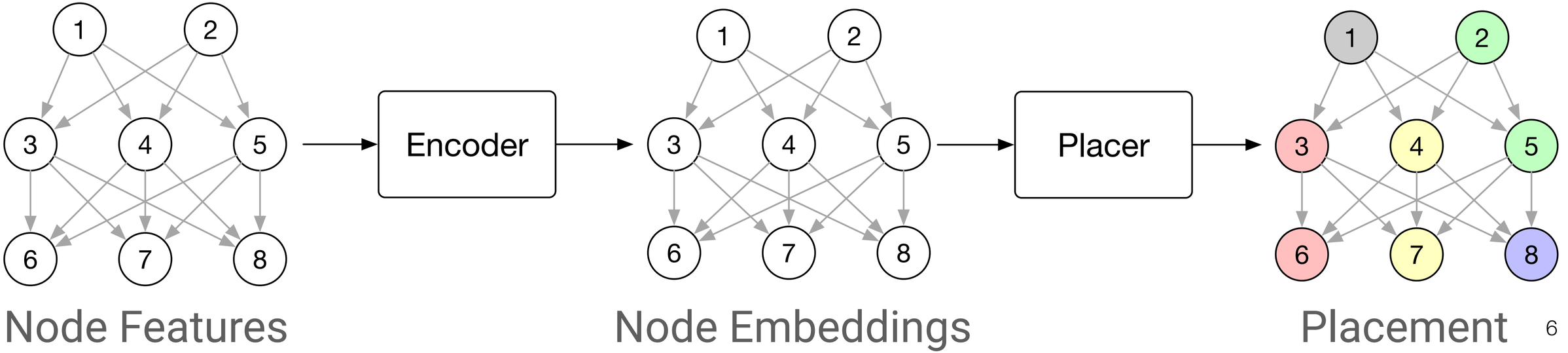
Mirhoseini et al. (Google Inc.), "**Device placement optimization with reinforcement learning,**" in Proc. **ICML 2017**.

Device placement with deep reinforcement learning



Agent design





Challenges

We notice that Google's ICML 2017 uses about **100** workers and **12 to 27** hours to find the best placement for workloads. It mainly caused by two reasons:

- The **REINFORCE** algorithm is inefficient. It requires large number of samples to train the agent.
- In device placement, the environment is a **real-world** machine with multiple devices(CPU and GPUs). Collecting rewards from a real-world environment is very slow, especially when the workload is a large DNN.

Solution

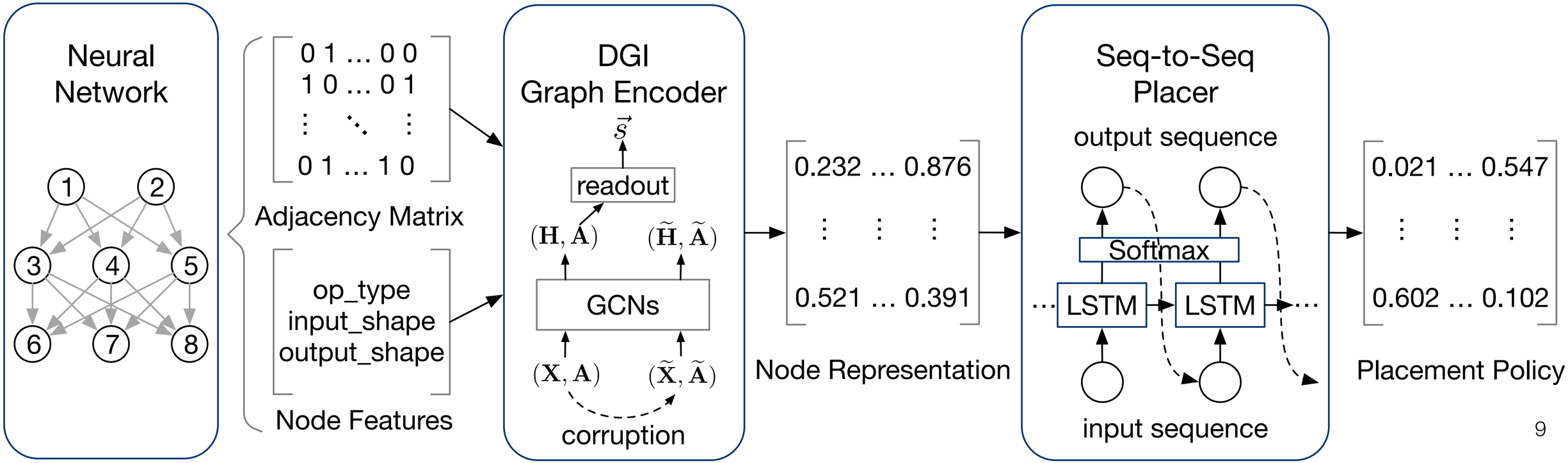
- For the first challenge, we replace the **REINFORCE** algorithm with **PPO** algorithm to improve the sample efficiency, however, it still needs a substantial amount of samples to train the agent.

- For the second challenge, one of the solution is using a simulated environment instead of a real one (proposed by Placeto). In that sense, we need to build a model for **every single** environment, which is not a **model-free** methods.

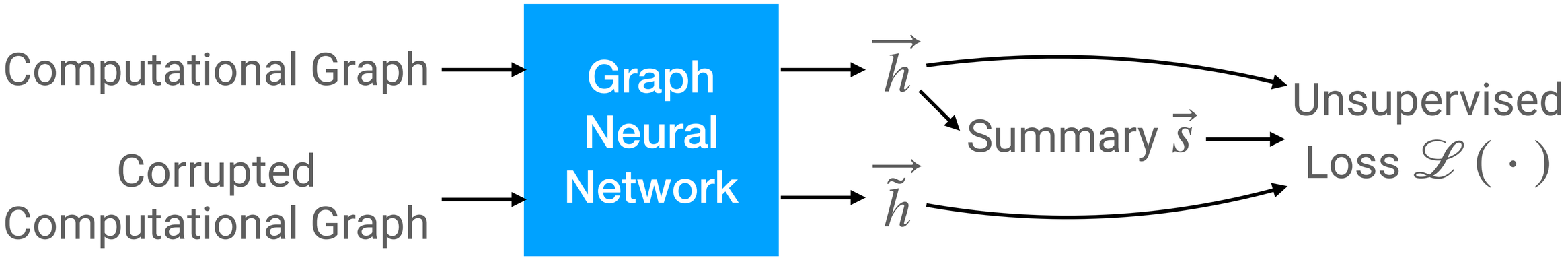
- Is there a way to train the agent with a very few amount of samples or even without any samples?

Our framework

- To address these challenges, we proposed our DRL-based framework, ***Mars***, a graph encoder ***pre-trained*** by ***contrastive learning***, followed by a ***light-weight*** segment-level sequence-to-sequence placer.

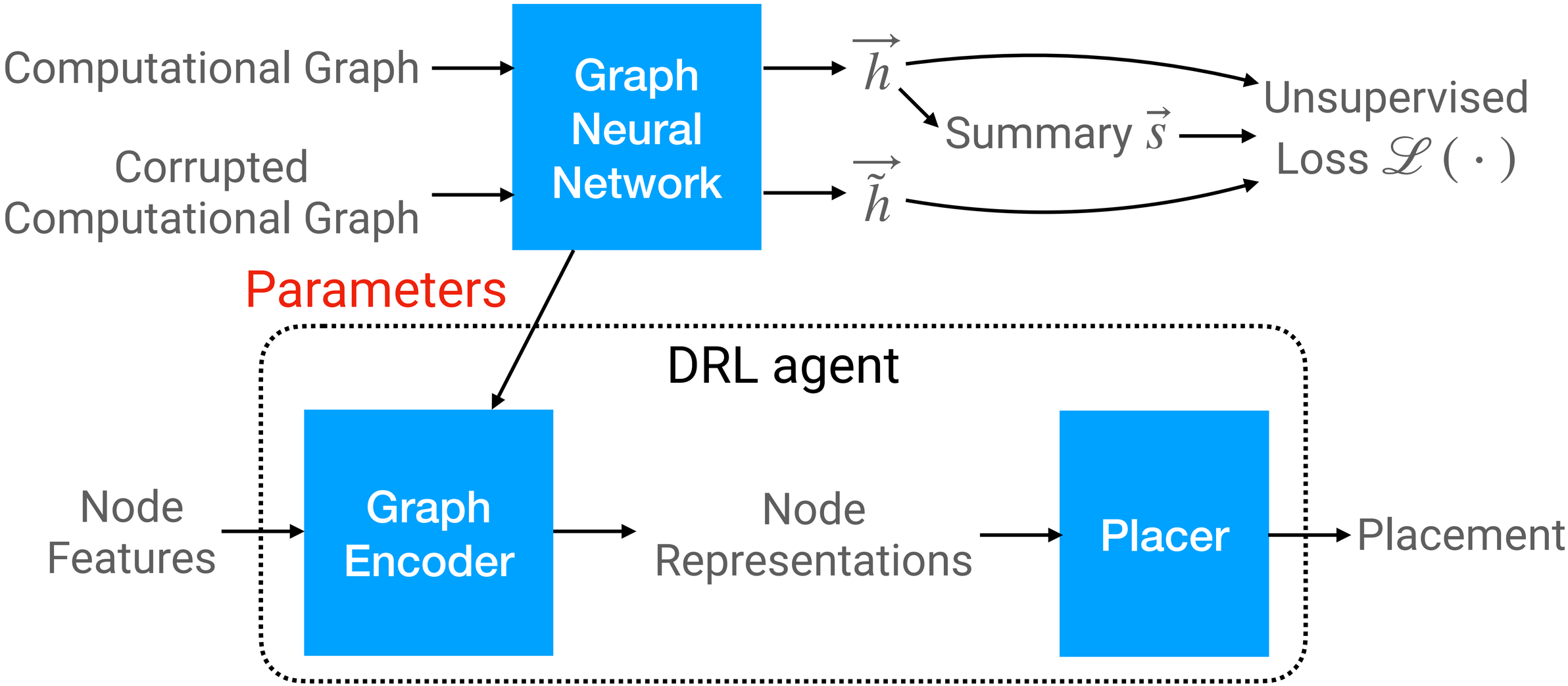


Contrastive Learning



We use contrastive learning to pre-train the graph encoder without any labeled data. After pre-training, the graph encoder can encode the operation in computational graph into a node representation $\overrightarrow{h}$, which represents the operation in a vector space.

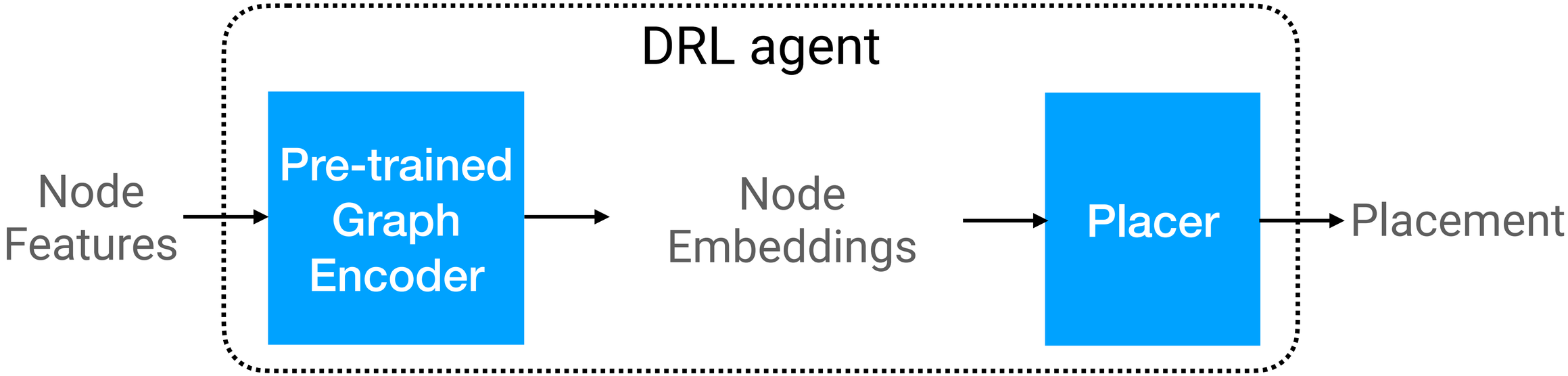
Contrastive Learning



Light-weight Placer

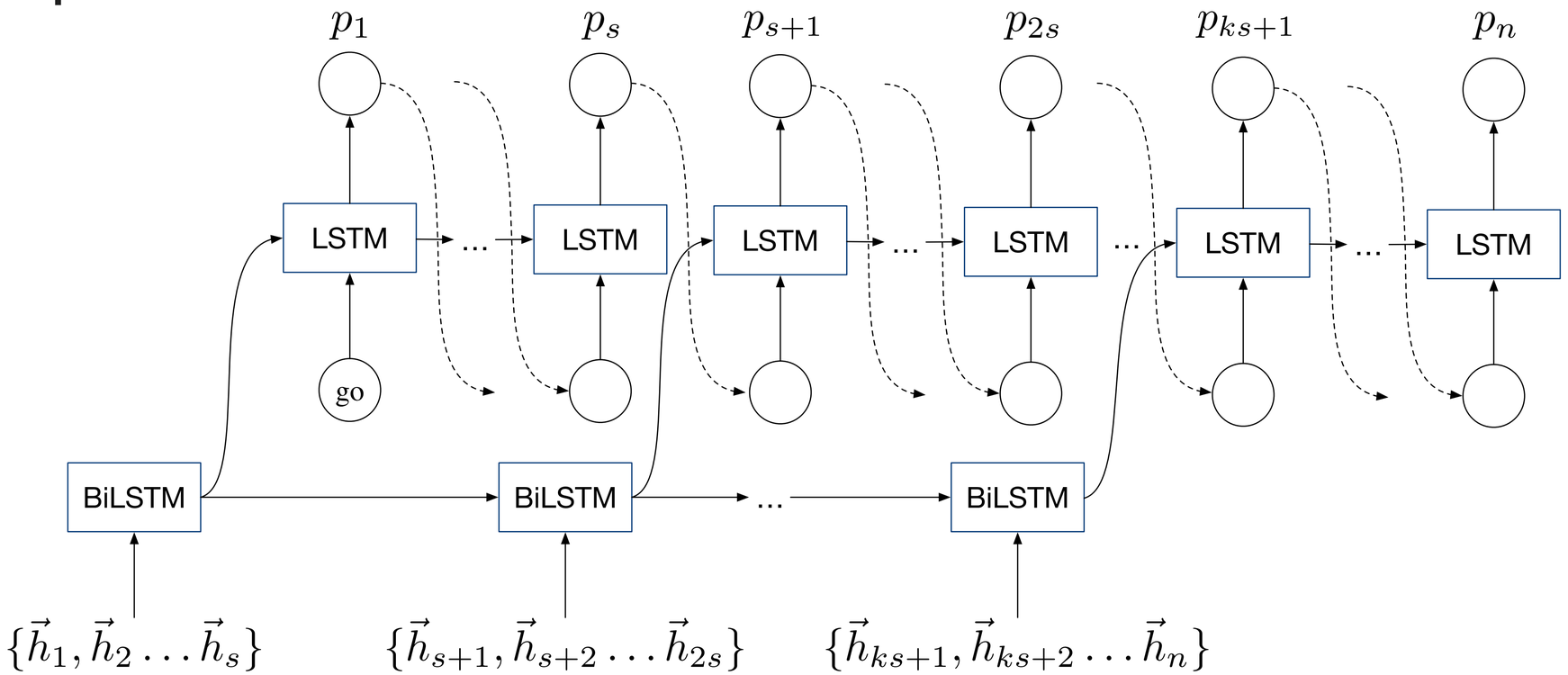
- In node classification task, DGI add a logistic regression layer as classifier after the pre-trained graph encoder, and train it with the labeled data. To reduce the amount of labeled data required for training, the classifier added should be simple and easy to train.

- Following this idea, we use a light-weight placer in our agent design, a segment-level sequence-to-sequence neural network.



Light-weight Placer

- A segment-level sequence-to-sequence placer has been designed to avoid placing extremely long operation sequence at a time.
- The light-weight placer can utilize the pre-trained parameters of the graph encoder better.



Experimental Results

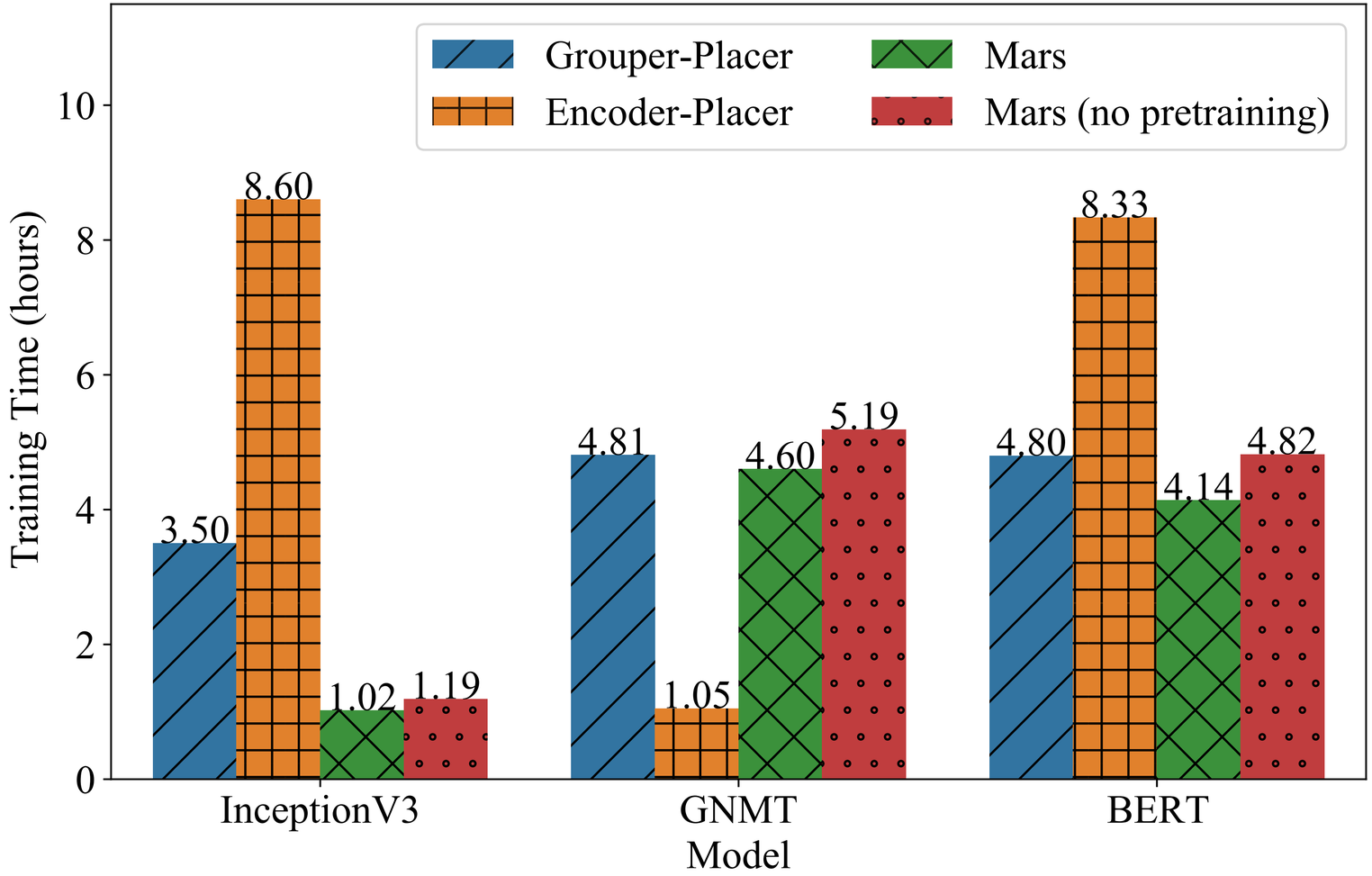
- From experimental results, ours approach outperforms all state-of-the-arts.

Per-step runtime (in seconds) of placements found by different approaches.

Models	Human Experts	GPU Only	Grouper-Placer	Encoder-Placer	Mars	Mars (no pre-training)
Inception-V3	0.071	0.071	**0.067**	**0.067**	**0.067**	**0.067**
GNMT-4	1.661	OOM	1.418	1.437	**1.379**	1.396
BERT	OOM	OOM	12.661	11.737	**9.214**	11.363

Experimental Results

- With unsupervised pre-training, we achieved better performance while reducing the agent training time by 13.2% on average.



Thank you