Rlgraph: Flexible Computation Graphs for Deep Reinforcement learning

MICHAEL SCHAARSCHMIDT, SVEN MIKA, KAI FRICKE, EIKO YONEKI
RLgraph

- Framework that designs and implements RL computation
- Metagraph outlining high-level data flow, followed by execution

| API, Component configuration | Prebuilt models, inference |
|-----------------------------|---------------------------|
| RLgraph component graph     | Model design, dataflow composition |
| TensorFlow                  | Local backends variables/operations |
| PyTorch                     | Distributed execution engine |
| ...                         | Execution, orchestration |
| Distributed TF              | Hardware: CPU, GPU, TPU, FPGAs... |

*Figure 1. RLgraph stack for using and designing RL algorithms.*
Reinforcement Learning

• ML technique that interacts with environment to make decisions
• Expanded use in gaming, robots, 3D scene simulators
RL Execution Difficulties

• Frequent problem environment interaction
• Highly varied states, resources, models
Novelty

- Novel meta-graph that generalizes dataflow to high level
- Claims to be “the first common interface to tensorflow and pytorch”
  - Rlib
  - Distributed Tensorforce
- Works on static and define-by-run graphs
- Updated systems since
  - Rlib Flow
  - MSRL
**Rlgraph Creation**

- Component composition phase
- Assembly phase
- Graph compilation/building phase

*Figure 2. Example memory component with three API methods.*
**RLgraph Execution**

- Graph executors
- Backend support and generalization

![Figure 4. RLgraph execution stack.](image)
Results

- Tested on tensorflow and pytorch
- Low build overhead
- Multiple GPU success

(a) Single worker throughput.  (b) Training times for *Pong*.

*Figure 6*. Distributed sample throughput on *Pong*.

*Figure 7*. Single task throughput and learning comparison.
RLgraph Solution

- Logical component composition separation supports any distributed execution paradigm
- No restrictions on execution supports static and define-by-run backend
- High level abstractions Fast development cycles
- Individually built and tested components Incremental building and testing
**Figure 11.** TensorBoard visualization of DeepMind’s IMPALA learner (left).
### RLgraph Impact & Future Work

- Pluggable
- Open source
- RL use cases only increasing

---

**Fig. A1:** Interface for teachers to write hints and prompts.
• Schaarschmidt, M., Mika, S., Fricke, K., & Yoneki, E. (2018). RLgraph: Flexible Computation Graphs for Deep Reinforcement Learning. *CoRR*, *abs/1810.09028*. Retrieved from [http://arxiv.org/abs/1810.09028](http://arxiv.org/abs/1810.09028)

• Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Gonzalez, J., … Stoica, I. (2017). Ray RLLib: A Composable and Scalable Reinforcement Learning Library. *CoRR*, *abs/1712.09381*. Retrieved from [http://arxiv.org/abs/1712.09381](http://arxiv.org/abs/1712.09381)

• Liang, E., Wu, Z., Luo, M., Mika, S., & Stoica, I. (2020). Distributed Reinforcement Learning is a Dataflow Problem. *CoRR*, *abs/2011.12719*. Retrieved from [https://arxiv.org/abs/2011.12719](https://arxiv.org/abs/2011.12719)

• Zhu, H., Zhao, B., Chen, G., Chen, W., Chen, Y., Shi, L., … Chen, L. (2022). MSRL: Distributed Reinforcement Learning with Dataflow Fragments. *arXiv [Cs.LG]*. Retrieved from [http://arxiv.org/abs/2210.00882](http://arxiv.org/abs/2210.00882)