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

---

**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.
Results

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

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

Figure 7. Single task throughput and learning comparison.

Figure 6. Distributed sample throughput on Pong.
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.
Works Cited


