Building Scalable Framework and Environment of Reinforcement Learning

Yuandong Tian
Facebook AI Research
Reinforcement Learning

\[ S_t, r_t, S_{t+1}, a_t \rightarrow \text{Agent} \]

\[ \text{Environment} \]

Trajectory

\[ S_t, a_t, S_{t+1}, a_{t+1}, \ldots, S_{t+n} \]

[R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction]
Game as a testbed of Reinforcement Learning

- Go
- Shogi
- StarCraft II
- Chess
- Quake 3
- Dota 2
Game as a testbed of Reinforcement Learning

- Infinite supply of fully labeled data
- Controllable and replicable
- Low cost per sample
- Faster than real-time
- Less safety and ethical concerns
- Complicated dynamics with simple rules.
Game as a testbed of Reinforcement Learning

Need good simulator

Require a lot of data/resources.

Sim2real issue

Applications?
Our work

ELF Framework

ELF OpenGo

MiniRTS

House3D
**ELF**: Extensive, Lightweight and Flexible Framework for Game Research

Yuandong Tian, Qucheng Gong, Wendy Shang, Yuxin Wu, Larry Zitnick (NIPS 2017 Oral)

- **Extensive**
  - Any games with C++ interfaces can be incorporated.

- **Lightweight**
  - Fast. Mini-RTS (40K FPS per core)
  - Minimal resource usage (1GPU+several CPUs)
  - Fast training (a couple of hours for a RTS game)

- **Flexible**
  - Environment-Actor topology
  - Parametrized game environments.
  - Choice of different RL methods.

**ELF**: A simple for-loop

C++

Python

```python
while True:
    batched_states = GameContext.Wait()
    replies = model(batched_states)
    GameContext.Steps(replies)
```
How ELF works

Game Threads (C++)

Batch Batch Batch Batch Batch

Python

```python
while True:
    batched_states = GameContext.Wait()
    replies = model(batched_states)
    GameContext.Steps(replies)
```
ELF Characteristics

**Extensive**
Any games with C++ interfaces can be incorporated.

**Lightweight**
Fast. Mini-RTS (40K FPS per core)
Minimal resource usage (1GPU+several CPUs)
Fast training (half a day for a RTS game)

**Flexible**
Environment-Actor topology
Parametrized game environments.
Choice of different RL methods.
Flexibility

while True:
    batched = GameContext.Wait()
    replies = model(batched)
    GameContext.steps(replies)
Flexibility

while True:
    ...
    if batch["type"] == "actor":
        ...
    elif batch["type"] == "train":
        ...

Training
Flexibility

```
while True:
    ...
    if batch["type"] == "actor0":
        ...
    elif batch["type"] == "actor1":
        ...
Self-play
```
Flexibility

while True:
    ...
    for i in range(n):
        if batch["type"] == "actor%d" % i:
            ...

Multi-agent
Flexibility

Monte-Carlo Tree Search

```python
while True:
    batched = GameContext.Wait()
    replies = model(batched)
    GameContext.Steps(replies)
```
Reimplementation of AlphaGo Zero

Generate Training data

Update Models

Zero-human knowledge

$\theta_i$

Self-Replays

[Silver et al, Mastering the game of Go without human knowledge, Nature 2017]
AlphaGo Zero Strength

- 3 days version
  - 4.9M Games, 1600 rollouts/move
  - 20 block ResNet
  - Defeat AlphaGo Lee.

- 40 days version
  - 29M Games, 1600 rollouts/move
  - 40 blocks ResNet.
  - Defeat AlphaGo Master by 89:11
Demystifying AlphaGoZero/AlphaZero

• Amazing performance but no code available.
  • Huge computational cost (15.5 years to generate 4.9M selfplays with 1 GPU)
  • Sophisticated (distributed) systems.

• Lack of ablation analysis
  • What factor is critical for the performance?
  • Is the algorithm robust to random initialization and changes of hyper parameters?
  • How the ladder issue is solved?

• Lots of mysteries
  • Is the proposed algorithm really universal?
  • Is the bot almighty? Is there any weakness in the trained bot?
ELF OpenGo

- System can be trained with 2000 GPUs in 2 weeks (20 block version)
- Superhuman performance against professional players and strong bots.
- Abundant ablation analysis
- Decoupled design, code reusable for other games.

We open source the code and the pre-trained model for the Go and ML community.
ELF OpenGo Performance

Vs top professional players

<table>
<thead>
<tr>
<th>Name (rank)</th>
<th>ELO (world rank)</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kim Ji-seok</td>
<td>3590 (#3)</td>
<td>5-0</td>
</tr>
<tr>
<td>Shin Jin-seo</td>
<td>3570 (#5)</td>
<td>5-0</td>
</tr>
<tr>
<td>Park Yeonghun</td>
<td>3481 (#23)</td>
<td>5-0</td>
</tr>
<tr>
<td>Choi Cheolhan</td>
<td>3466 (#30)</td>
<td>5-0</td>
</tr>
</tbody>
</table>

Single GPU, 80k rollouts, 50 seconds
Offer unlimited thinking time for the players

Vs professional players

Single GPU, 2k rollouts, 27-0 against Taiwanese pros.

Vs strong bot (LeelaZero)

[158603eb, 192x15, Apr. 25, 2018]: 980 wins, 18 losses (98.2%)
ELF OpenGo Sample Game
Distributed ELF

Putting AlphaGoZero and AlphaZero into the same framework

- AlphaGoZero (more synchronization)
- AlphaZero (less synchronization)

Server controls synchronization
Server also does training.
**MiniRTS**: A miniature RTS engine

![Map](image)

<table>
<thead>
<tr>
<th>Platform</th>
<th>Frame per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALE</td>
<td>6,000</td>
</tr>
<tr>
<td>Open AI Universe</td>
<td>60</td>
</tr>
<tr>
<td>Malmo</td>
<td>120</td>
</tr>
<tr>
<td>DeepMind Lab</td>
<td>287*/866**</td>
</tr>
<tr>
<td>VizDoom</td>
<td>7,000</td>
</tr>
<tr>
<td>TorchCraft</td>
<td>2,000</td>
</tr>
<tr>
<td>MiniRTS</td>
<td>40,000</td>
</tr>
</tbody>
</table>

* Using CPU only
** Using CPUs and GPU
MiniRTS

Base
Build workers and collect resources.

Resource
Contains 1000 minerals.

Barracks
Build melee attacker and range attacker.

Worker
Build barracks and gather resource.
Low speed in movement and low attack damage.

Melee Tank
High HP, medium movement speed, short attack range, high attack damage.

Range Tank
Low HP, high movement speed, long attack range and medium attack damage.
## 9 Discrete Strategic Actions

<table>
<thead>
<tr>
<th>No.</th>
<th>Action name</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IDLE</td>
<td>Do nothing</td>
</tr>
<tr>
<td>2</td>
<td>BUILD WORKER</td>
<td>If the base is idle, build a worker</td>
</tr>
<tr>
<td>3</td>
<td>BUILD BARRACK</td>
<td>Move a worker (gathering or idle) to an empty place and build a barrack.</td>
</tr>
<tr>
<td>4</td>
<td>BUILD MELEE ATTACKER</td>
<td>If we have an idle barrack, build an melee attacker.</td>
</tr>
<tr>
<td>5</td>
<td>BUILD RANGE ATTACKER</td>
<td>If we have an idle barrack, build a range attacker.</td>
</tr>
<tr>
<td>6</td>
<td>HIT AND RUN</td>
<td>If we have range attackers, move towards opponent base and attack.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Take advantage of their long attack range and high movement speed to hit and run if enemy counter-attack.</td>
</tr>
<tr>
<td>7</td>
<td>ATTACK</td>
<td>All melee and range attackers attack the opponent’s base.</td>
</tr>
<tr>
<td>8</td>
<td>ATTACK IN RANGE</td>
<td>All melee and range attackers attack enemies in sight.</td>
</tr>
<tr>
<td>9</td>
<td>ALL DEFEND</td>
<td>All troops attack enemy troops near the base and resource.</td>
</tr>
</tbody>
</table>
Rule-based AIs

AI_SIMPLE
Build 5 tanks and attack

AI_HIT_AND_RUN
Build 2 tanks and harass

MiniRTS trains with a single GPU and 6 CPUs in half a day.
Using Internal Game data and Off-policy Actor-Critic Methods. Reward is only available once the game is over.
Trained AI

Trained with a single machine with GPU in a few hours
Trained AI
Comparison between different models

<table>
<thead>
<tr>
<th>Method</th>
<th>Vanilla</th>
<th>Vanilla (hist=4)</th>
<th>RNN</th>
<th>BuildHistory</th>
<th>PrevSeen</th>
<th>Complete Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win rate</td>
<td>72.9±1.8</td>
<td>79.8±0.7</td>
<td>79.7±1.3</td>
<td>80.8±1.7</td>
<td>81.4±0.8</td>
<td>81.7±0.7</td>
</tr>
</tbody>
</table>
MiniRTS v2

More units
- Lancer
- Swordman
- Knight
- Catapult
- Dragon
- Archer
- Tower

Rock-Paper-Scissor dynamics

Hengyuan Hu*
Denis Yarats*
Qucheng Gong
Yuandong Tian
Michael Lewis
Language-driven Actions

\[ \pi_\theta(a|s, \text{"gather enough resources and build a barrack"}) \]

\[ \pi_\theta(a|s, \text{"take the catapult and destroy towers from the distance"}) \]
Instructor: Only gives language descriptions

Executor: Turn language description into unit-level actions
Data Collection

Ask Turkers to play the game in pairs.

One Turker plays as a coach:
- High-level strategy
- Communicate via language

The other Turker plays as a player:
- Clicks through high-level commands
- Only makes local decisions

ParlAI

https://github.com/facebookresearch/ParlAI
The Dataset

<table>
<thead>
<tr>
<th>Statistics</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Games</td>
<td>5392</td>
</tr>
<tr>
<td>Human win rate</td>
<td>58.67%</td>
</tr>
<tr>
<td>#instructions</td>
<td>76045</td>
</tr>
<tr>
<td>#Unique instructions</td>
<td>39598</td>
</tr>
<tr>
<td>#words</td>
<td>307162</td>
</tr>
<tr>
<td>#Vocabulary</td>
<td>2851</td>
</tr>
<tr>
<td>#words / instruction</td>
<td>7.76</td>
</tr>
<tr>
<td>#instruction / game</td>
<td>13.09</td>
</tr>
<tr>
<td>#actions / instruction</td>
<td>7.18</td>
</tr>
</tbody>
</table>

Linguistic Phenomena

<table>
<thead>
<tr>
<th>Phenomenon</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counting</td>
<td>Build 3 dragons.</td>
</tr>
<tr>
<td>Spatial Reference</td>
<td>Send him to the choke point behind the tower.</td>
</tr>
<tr>
<td>Composed Actions</td>
<td>Attack archers, then peasants.</td>
</tr>
<tr>
<td>Cross-instruction anaphora</td>
<td>Use it as a lure to kill them.</td>
</tr>
</tbody>
</table>

Each game has a complete sequence of user actions.
How to plan the trajectory in unknown environments?
House3D

SUNCG dataset, 45K scenes, all objects are fully labeled.

https://github.com/facebookresearch/House3D
Build a semantic model

Find “oven”

incomplete model of the environment

[Y. Wu et al, Learning and Planning with a Semantic Model, submitted to ICLR 2019]
Build a semantic model

Bayesian Inference $P(z|Y)$

Learning experience $Y$

- car
- sofa
- outdoor
- living room
- dining room
- kitchen
- oven
- chair

Next step “kitchen”
LEAPS
LEArning and Planning with a Semantic model
Learning the Prior between Different Rooms

- Dining Room
  - garage: 0.05
  - kitchen: 0.7
  - living room: 0.56
  - outdoor: 0.08
  - office: 0.16

- Bedroom
  - garage: 0.03
  - bathroom: 0.2
  - outdoor: 0.05
  - Dining room: 0.13

- Outdoor
  - bedroom: 0.05
  - bathroom: 0.06
  - Living room: 0.12
  - garage: 0.28
Test Performance on ConceptNav

RoomNav $H = 500$ (#plan)

- blue: random
- green: pure $\mu(\theta)$
- red: LEAPS

Relative Improvements $H = 500$

- blue: random
- green: pure $\mu(\theta)$
Case Study

• A case study
• Go to “outdoor”
Case Study

- A case study
- Go to "outdoor"

Posterior: $P(z|E)$

Living room

Outdoor

Birth

Garage

Sub-Goal: Outdoor

Failed!
Case Study

- A case study
- Go to “outdoor”
Case Study

- A case study
- Go to “outdoor”
Case Study

- A case study
- Go to “outdoor”
<table>
<thead>
<tr>
<th>opt. plan-steps</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizon $H = 300$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random</td>
<td>20.5 / 15.9</td>
<td>6.9 / 16.7</td>
<td>3.8 / 10.7</td>
<td>1.6 / 4.2</td>
<td>3.0 / 8.8</td>
<td>7.2 / 13.6</td>
</tr>
<tr>
<td>pure $\mu(\theta)$</td>
<td>49.4 / 47.6</td>
<td>11.8 / 27.6</td>
<td>2.0 / 4.8</td>
<td>2.6 / 10.8</td>
<td>4.2 / 13.2</td>
<td>13.1 / 22.9</td>
</tr>
<tr>
<td>aug. $\mu_S(\theta_s)$</td>
<td>47.8 / 45.3</td>
<td>11.4 / 23.1</td>
<td>3.0 / 7.8</td>
<td>3.4 / 8.1</td>
<td>4.4 / 11.2</td>
<td>13.0 / 20.5</td>
</tr>
<tr>
<td>RNN control.</td>
<td>52.7 / 45.2</td>
<td>13.6 / 23.6</td>
<td>3.4 / 9.6</td>
<td>3.4 / 10.2</td>
<td>6.0 / 17.6</td>
<td>14.9 / 21.9</td>
</tr>
<tr>
<td>LEAPS</td>
<td><strong>53.4 / 58.4</strong></td>
<td><strong>15.6 / 31.5</strong></td>
<td><strong>4.5 / 12.5</strong></td>
<td><strong>3.6 / 6.6</strong></td>
<td><strong>7.0 / 18.0</strong></td>
<td><strong>16.4 / 27.9</strong></td>
</tr>
<tr>
<td>Horizon $H = 500$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random</td>
<td>21.9 / 16.9</td>
<td>9.3 / 18.3</td>
<td>5.2 / 12.1</td>
<td>3.6 / 6.1</td>
<td>4.2 / 9.9</td>
<td>9.1 / 15.1</td>
</tr>
<tr>
<td>pure $\mu(\theta)$</td>
<td>54.0 / 57.5</td>
<td>15.9 / 25.6</td>
<td>3.8 / 7.7</td>
<td>2.8 / 6.4</td>
<td>4.8 / 8.6</td>
<td>16.2 / 22.9</td>
</tr>
<tr>
<td>aug. $\mu_S(\theta_s)$</td>
<td>54.1 / 51.8</td>
<td>15.5 / 26.5</td>
<td>4.6 / 8.2</td>
<td>3.0 / 11.8</td>
<td>4.6 / 12.5</td>
<td>16.1 / 23.5</td>
</tr>
<tr>
<td>RNN control.</td>
<td><strong>57.4 / 43.8</strong></td>
<td><strong>20.2 / 28.0</strong></td>
<td><strong>7.2 / 14.6</strong></td>
<td><strong>4.2 / 8.0</strong></td>
<td><strong>9.0 / 16.0</strong></td>
<td><strong>19.9 / 24.6</strong></td>
</tr>
<tr>
<td>LEAPS</td>
<td>57.2 / <strong>61.9</strong></td>
<td><strong>21.5 / 34.4</strong></td>
<td><strong>10.0 / 14.8</strong></td>
<td><strong>6.4 / 11.6</strong></td>
<td><strong>12.0 / 23.5</strong></td>
<td><strong>21.6 / 31.1</strong></td>
</tr>
<tr>
<td>Horizon $H = 1000$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random</td>
<td>24.3 / 17.6</td>
<td>13.5 / 20.3</td>
<td>9.1 / 14.3</td>
<td>8.0 / 9.3</td>
<td>7.0 / 11.5</td>
<td>13.0 / 17.0</td>
</tr>
<tr>
<td>pure $\mu(\theta)$</td>
<td>60.8 / <strong>58.4</strong></td>
<td>23.3 / 29.5</td>
<td>7.6 / 8.8</td>
<td>8.2 / 12.9</td>
<td>11.0 / 17.2</td>
<td>22.5 / 26.5</td>
</tr>
<tr>
<td>aug. $\mu_S(\theta_s)$</td>
<td>61.3 / 50.1</td>
<td>23.0 / 26.2</td>
<td>9.4 / 12.0</td>
<td>5.8 / 9.6</td>
<td>9.0 / 13.6</td>
<td>22.4 / 23.8</td>
</tr>
<tr>
<td>RNN control.</td>
<td><strong>66.7 / 49.0</strong></td>
<td><strong>30.1 / 31.5</strong></td>
<td><strong>13.8 / 15.4</strong></td>
<td><strong>9.0 / 10.0</strong></td>
<td><strong>14.0 / 20.8</strong></td>
<td><strong>28.2 / 27.7</strong></td>
</tr>
<tr>
<td>LEAPS</td>
<td>66.4 / <strong>58.4</strong></td>
<td><strong>31.9 / 40.5</strong></td>
<td><strong>15.0 / 18.3</strong></td>
<td><strong>11.4 / 17.0</strong></td>
<td><strong>15.4 / 27.1</strong></td>
<td><strong>29.7 / 35.2</strong></td>
</tr>
</tbody>
</table>
What’s Beyond Game for RL?
RL for optimization

Travel Salesman Problem

Job Scheduling

Vehicle Routing

Bin Packing

Protein Folding

Model-Search
Non-differentiability

• Direct predicting combinatorial solutions.

[O. Vinyals. et al, Pointer Networks, NIPS 2015]

Convex hull

Seq2seq model

[Resource Management with Deep Reinforcement Learning, ACM Workshop on Hot Topics in Networks, 2016]
Local Rewriting Framework

A learned “gradient descent” that starts from a feasible solution iteratively converges to a good solution

How to learn it?

Xinyun Chen  Yuandong Tian
Local Rewriting Framework

Current State (i.e. Solution)

\[ s_t \quad \rightarrow \quad \omega_t \sim \pi_\omega(\cdot | s_t) \quad \rightarrow \quad u_t \sim \pi_u(\cdot | s_t[\omega_t]) \quad \rightarrow \quad s_{t+1} = f(s_t, \omega_t, u_t) \]

Q-Actor-Critic Training of two policies \( \pi_\omega(\cdot | s_t) \) and \( \pi_u(\cdot | s_t[\omega_t]) \)

\[ \pi_\omega(\cdot | s_t) : \text{Q-learning with soft policy} \quad \pi_\omega(\omega_t | s_t; \theta) = \frac{\exp(Q(s_t, \omega_t; \theta))}{\sum_{\omega_t} \exp(Q(s_t, \omega_t; \theta))} \]

\[ \pi_u(\cdot | s_t[\omega_t]) : \text{Actor-Critic with learned Q} \quad L_u(\phi) = - \sum_{t=0}^{T-1} \Delta(s_t, (\omega_t, u_t)) \log \pi_u(u_t | s_t[\omega_t]; \phi) \]
Applications

(a) Online Job Scheduling

(b) Expression Simplification

\[
\omega_t = \arg\max_{\pi_{\omega}(, s_t)} \leq u_t
\]
Online Job Scheduling

Jobs

- **Job 1**: $T = 2, A = 1$
- **Job 2**: $T = 3, A = 2$
- **Job 3**: $T = 1, A = 3$

Resource 1

Resource 2

Scheduling 1

Scheduling 2

Graph representation
Online Job Scheduling

Baselines:
- Earliest Job First (EJF)
- Shortest Job First (SJF)
- Shortest First Search (SJFS)
- DeepRM

Offline baselines:
- Google OR-tools (OR-tools)
- SJF-offline

$D$: Number of resources
Online Job Scheduling: Ablation Study

The learned model can generalize to different job distributions.
Expression Simplification

```
<= 5 + max
   max v0 3

<= 5 max
   + 3
   +
   v0 3 3 3

||
<= 5 +
   +
   v0 3

||
<= 1
```

- Min/Max
- Distribution
- Min/Max Expansion
- 3 + 3 \rightarrow 6
- 5 \leq 6 \rightarrow 1
Expression Simplification

Baselines:
- Z3-simplify
- Z3-ctx-solver-simplify
- Heuristic Search
- Halide rules

Z3 is a state-of-the-art theorem prover.
Expression Simplification

Transfer learning still works well.

A model trained with expression length $\leq 50$ has good performance on test set with expression length $\geq 100$, and better than Z3.
Future Directions

Hierarchical RL

Multi-Agent

RL applications

RL Systems

Model-based RL

Policy Optimization

Model Estimation

$\pi_\theta(a|s)$

$s' = f_\phi(s, a)$

RL for Optimization
Thanks!