Building Scalable Framework and Environment of Reinforcement Learning

Yuandong Tian
Facebook AI Research
AI works in a lot of situations

Object Recognition  Medical  Translation

Personalization  Surveillance  Speech Recognition
What AI still needs to improve

Initial Enthusiasm
“It really works! All in AI!”

A scary trend of slowing down
“Man, we need more data”

Trying all possible hacks
“How can that be...”

Despair
“No way, it doesn’t work”

Human level

Performance

Efforts
What AI still needs to improve

Performance

Human level

We need novel algorithms

Trying all possible hacks
“How can that be...”

A scary trend of slowing down
“Man, we need more data”

Initial Enthusiasm
“It really works! All in AI!”

Despair
“No way, it doesn’t work”
What AI still needs to improve

- Question Answering
- ChatBot
- Autonomous Driving
- Home Robotics

Common Sense

- Few supervised data
- Complicated environments
- Lots of Corner cases.

High-order Reasoning

- Program Induction
- Text Generation
Reinforcement Learning

State $s_t$

Reward $r_t$

Action $a_t$

Environment

$S_{t+1}$

$R$. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction
Supervised Learning v.s Reinforcement Learning

**Supervised learning**
- The boss decides what you will learn
- You work hard to get them right

**Reinforcement learning**
- Explore the space to find a good solution
- You decide what data you want to learn
- More data hungry
- More computational resources
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

Go

Shogi

StarCraft II

Chess

Quake 3

Dota 2
Game as a testbed of Reinforcement Learning

Need good simulator

Require a lot of data/resources.

Sim2real issue

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

[https://github.com/facebookresearch/ELF](https://github.com/facebookresearch/ELF)

**ELF**: A simple for-loop

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

Game Threads (C++)

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

Generate Training data

Update Models

Self-Replays

Zero-human knowledge

\[ \theta_i \]

[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</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 (version 1)

Selfplay 1 \rightarrow \text{Selfplay data} \rightarrow \text{Train} \rightarrow \text{Model Zoo} \rightarrow \text{Win rate} > 55\% \rightarrow \text{Update best model and next candidate}

Selfplay 2 \rightarrow \text{Selfplay data} \rightarrow \text{Train} \rightarrow \text{Model Zoo} \rightarrow \text{Win rate} > 55\% \rightarrow \text{Update best model and next candidate}

\cdots \cdots \cdots

Selfplay n \rightarrow \text{Selfplay data} \rightarrow \text{Train} \rightarrow \text{Model Zoo} \rightarrow \text{Win rate} > 55\% \rightarrow \text{Update best model and next candidate}

\text{Evaluation Server} \rightarrow \text{Evaluation 1} \rightarrow \text{Evaluation 2} \rightarrow \cdots \rightarrow \text{Evaluation m}
Distributed System (version 1)

- Selfplay 1
- Selfplay 2
- Selfplay n

Open a port
Receive selfplay data via ZeroMQ

Training procedure

Current trained model

Model Zoo

8 GPUs

Evaluation 1
Evaluation 2
Evaluation m

Evaluation Server
Distributed System (version 1)

Selfplay 1  Selfplay 2  \( \cdots \)  Selfplay n  300-2k GPUs

Training procedure (8 GPUs)

Pick the best model and keep selfplaying

Current best model

Each selfplay client batches 32 parallel games in a batch size of 128

Evaluation 1

Evaluation 2

Evaluation m

Model Zoo

Model Zoo

Model Zoo
Distributed System (version 1)

Selfplay 1 → Training procedure (8 GPUs) → Model Zoo

Selfplay 2

... Selfplay n

Model Zoo

Evaluation Server

Evaluation 1

Evaluation 2

Evaluation m

No GPU needed

Update best model and next candidate

Win rate > 55%
Distributed System (version 1)

Selfplay 1 → Training procedure (8 GPUs) → Model Zoo

Selfplay 2 → Model Zoo

Selfplay n → Model Zoo

Send the current model pairs to evaluate

Evaluation 1
Evaluation 2
Evaluation m

Each evaluation client batches 2 parallel games
Distributed ELF (v2)

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

<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
# 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>
Trained AI

Trained with a single machine with GPU In a few hours
Trained AI
First Person Shooter (FPS) Game

Yuxin Wu, Yuandong Tian, ICLR 2017

Play the game from the raw image!
VizDoom AI Competition 2016 (Track1)

We won the first place!

<table>
<thead>
<tr>
<th>Rank</th>
<th>Bot</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>Total frags</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>F1</td>
<td>56</td>
<td>62</td>
<td>n/a</td>
<td>54</td>
<td>47</td>
<td>43</td>
<td>47</td>
<td>55</td>
<td>50</td>
<td>48</td>
<td>50</td>
<td>559</td>
</tr>
<tr>
<td>2</td>
<td>Arnold</td>
<td>36</td>
<td>34</td>
<td>42</td>
<td>36</td>
<td>36</td>
<td>45</td>
<td>36</td>
<td>39</td>
<td>n/a</td>
<td>33</td>
<td>36</td>
<td>413</td>
</tr>
<tr>
<td>3</td>
<td>CLYDE</td>
<td>37</td>
<td>n/a</td>
<td>38</td>
<td>32</td>
<td>37</td>
<td>30</td>
<td>46</td>
<td>42</td>
<td>33</td>
<td>24</td>
<td>44</td>
<td>393</td>
</tr>
</tbody>
</table>

Videos:

https://www.youtube.com/watch?v=94EPSjQH38Y
https://www.youtube.com/watch?v=Qv4esGWOg7w&t=394s
What’s Beyond Game for RL?
Game as a testbed of Reinforcement Learning

Need good simulator

Require a lot of data/resources.

Sim2real issue

Applications?
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

Local Rewriting Framework

A learned “gradient descent” that

starts from a feasible solution
iteratively converges to a good solution

How to learn it?
Local Rewriting Framework

Current State (i.e. Solution)

\[ s_t \rightarrow \omega_t \sim \pi_\omega(\cdot | s_t) \rightarrow u_t \sim \pi_u(\cdot | s_t[\omega_t]) \]

\[ s_{t+1} = f(s_t, \omega_t, u_t) \]

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

\[ \omega_t^* = \arg\max_{\pi_\omega(\cdot, s_t)} \]

\[ (a) \leq (b) \]

\[ \leq \]

\[ \text{Constant Reduction} \]

\[ \omega_t^* = \arg\max \pi_\omega(\cdot, s_t) \]
Online Job Scheduling

Jobs

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

Resources

- Resource 1
- Resource 2

Scheduling 1

Scheduling 2

Graph representation 1

Graph representation 2
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

**Average slowdown**

- EJF
- SJF
- SJFS
- DeepRM
- OR-tools
- SJF-offline
- NeuRewriter

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

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

Next step “kitchen”

Learning experience $Y$

Bayesian Inference $P(z|Y)$
LEAPS
LEArning and Planning with a Semantic model

Planning the trajectory and more exploration

Dining room

kitchen

P(z_{dining,living room})

P(z_{kitchen,living room})

P(z_{sofa,living room})

living room

chair

sofa

Planning the trajectory and more exploration

Dining room

kitchen

P(z_{dining,living room})

P(z_{kitchen,living room})

P(z_{sofa,living room})

living room

chair

sofa

P(z_{sofa,living room} | Y) = 1
Learning the Prior between Different Rooms
Test Performance on ConceptNav

RoomNav $H = 500$ (#plan)

Relative Improvements $H = 500$
Case Study

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

- A case study
- Go to “outdoor”

Posterior: $P(z|E)$
Case Study

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

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

• A case study
• Go to “outdoor”

Posterior: $P(z|E)$

- Living room: 0.99
- Outdoor: 0.99
- Birth: 0.01
- Garage: 0.28

Sub-Goal: Outdoor
Success
<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></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>random</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>pure $\mu(\theta)$</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>aug.$\mu_S(\theta_s)$</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>RNN control.</td>
<td>53.4 / 58.4</td>
<td>15.6 / 31.5</td>
<td>4.5 / 12.5</td>
<td>3.6 / 6.6</td>
<td>7.0 / 18.0</td>
<td>16.4 / 27.9</td>
</tr>
<tr>
<td>LEAPS</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></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>random</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>pure $\mu(\theta)$</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>aug.$\mu_S(\theta_s)$</td>
<td>57.4 / 43.8</td>
<td>20.2 / 28.0</td>
<td>7.2 / 14.6</td>
<td>4.2 / 8.0</td>
<td>9.0 / 16.0</td>
<td>19.9 / 24.6</td>
</tr>
<tr>
<td>RNN control.</td>
<td>57.2 / 61.9</td>
<td>21.5 / 34.4</td>
<td>10.0 / 14.8</td>
<td>6.4 / 11.6</td>
<td>12.0 / 23.5</td>
<td>21.6 / 31.1</td>
</tr>
<tr>
<td>LEAPS</td>
<td>54.8 / 58.4</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></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>66.7 / 49.0</td>
<td>30.1 / 31.5</td>
<td>13.8 / 15.4</td>
<td>9.0 / 10.0</td>
<td>14.0 / 20.8</td>
<td>28.2 / 27.7</td>
</tr>
<tr>
<td>LEAPS</td>
<td>66.4 / 58.4</td>
<td>31.9 / 40.5</td>
<td>15.0 / 18.3</td>
<td>11.4 / 17.0</td>
<td>15.4 / 27.1</td>
<td>29.7 / 35.2</td>
</tr>
</tbody>
</table>
Future Directions

Hierarchical RL

Multi-Agent

RL applications

RL Systems

Model-based RL

RL for Optimization
How to do well in Reinforcement Learning?

\[ Q(s, a), V^\pi(s), V(s), \pi(a|s), Q^\pi(s, a) \]

Strong math skills

Experience on (distributed) systems

Parameter tuning skills

Strong coding skills
Thanks!