Building Scalable Framework and Environment of Reinforcement Learning

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

Object Recognition
Medical
Translation
Speech Recognition

Personalization
Surveillance
Smart Design
Board game
What AI still needs to improve

- Very few supervised data
- Complicated environments
- Lots of Corner cases.

Question Answering

ChatBot

Common Sense

StarCraft

Exponential space to explore

Autonomous Driving

Home Robotics

Very few supervised data
Complicated environments
Lots of Corner cases.
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”
What AI still needs to improve

We need novel algorithms

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”

Performance
Efforts
Reinforcement Learning

Agent

Environment

State $S_t$

Reward $r_t$

Action $a_t$

$S_{t+1}$

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

Sequential Decision Making

- Bidding Agent
- Real-time ads bidding
- Recommendation

RL for Optimization

- 3D Bin-packing
- Travel Salesman problem
- Architecture Search
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

Algorithm is slow and data-inefficient

Require a lot of resources.

Abstract game to real-world

Hard to benchmark the progress
Game as a testbed of Reinforcement Learning

Algorithm is slow and data-inefficient

Abstract game to real-world

Require a lot of resources.

Hard to benchmark the progress

Better Algorithm/System

Better Environment
Our work

**Better Algorithm/System**

*Go Engine*
*(DarkForest, Y. Tian, Y. Zhu, ICLR16)*
*(ELF OpenGo, Y. Tian et al.)*

*Doom AI*
*(Yuxin Wu, Yuandong Tian, ICLR17)*

**Better Environment**

*ELF: Extensive Lightweight and Flexible Framework*
*(Yuandong Tian et al, NIPS17)*

*House3D: An interactive 3D environment for navigation*
*(Yi Wu, Georgia Gkioxari, Yuxin Wu, Yuandong Tian)*
A Framework for Deep Reinforcement Learning

Design Choices:

CPU, GPU?
Simulation, Replays
Concurrency

State $S_t$
Reward $r_t$
Action $a_t$

[State $S_{t+1}$]

[R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction]
A Framework for Deep Reinforcement Learning

- **CPU**
  - Game simulation

- **GPU**
  - Neural network

- **Single thread**
  - Shared memory

- **Single process, multiple threads**

- **Multi-process**

- **Distributed System**
  - Scalable, high-latency, less robust
**ELF**: Extensive, Lightweight and Flexible Framework for Game Research

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

**ELF**: A simple for-loop

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

Game Threads (C++)

0
1
2
3
4
5
6
7

Batch Batch Batch Batch Batch

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.
Extensibility

Go

Pong

ALE

ELF

Breakout

RTS Engine

MiniRTS

…
Lightweight

**KFPS per CPU core for Pong (Atari)**

- 1 core
- 2 cores
- 4 cores
- 8 cores
- 16 cores

- OpenAI Gym

<table>
<thead>
<tr>
<th>Threads</th>
<th>64 threads</th>
<th>128 threads</th>
<th>256 threads</th>
<th>512 threads</th>
<th>1024 threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>KFPS</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
Lightweight

KFPS per CPU core for Pong (Atari)
while True:
    batched = GameContext.Wait()
    replies = model(batched)
    GameContext.Steps(replies)
while True:
    ...
    if batch["type"] == "actor":
        ...
    elif batch["type"] == "train":
        ...

Flexibility
Flexibility

while True:
    ...
    if batch["type"] == "actor0":
        ...
    elif batch["type"] == "actor1":
        ...

Self-play
Flexibility

```python
while True:
    ...
    for i in range(n):
        if batch["type"] == "actor%d" % i:
            ...
```
Monte Carlo Tree Search

Aggregate win rates, and search towards the good nodes.

\[ a_t = \arg \max_a (Q(s_t, a) + u(s_t, a)) \]

\[ u(s, a) \propto \frac{P(s, a)}{1 + N(s, a)} \]

Tree policy

Default policy
Monte-Carlo Tree Search
How Game AI works

Even with a super-super computer, it is not possible to search the entire space.
How Game AI works

Even with a super-super computer, it is not possible to search the entire space.

Current game situation

Lufei Ruan vs. Yifan Hou (2010)
How Game AI works

How many action do you have per step?

- Checker: a few possible moves
- Poker: a few possible moves
- Chess: 30-40 possible moves
- Go: 100-200 possible moves
- StarCraft: 50^100 possible moves

- Alpha-beta pruning + Iterative deepening [Major Chess engine]
- Counterfactual Regret Minimization [Libratus, DeepStack]
- Monte-Carlo Tree Search + UCB exploration [Major Go engine]

Current game situation

Actor

Extensive search
Evaluate
Consequence
How Game AI works

How complicated is the game situation? How deep is the game?

- Chess: Rule-based
- Go: Linear function for situation evaluation [Stockfish]
- Poker: End game database
- StarCraft: Random game play with simple rules [Zen, CrazyStone, DarkForest]
- Deep Value network [AlphaGo, DeepStack]

Current game situation

Extensive search

Evaluate

Consequence

Critic

Black wins
White wins
Black wins
White wins
Black wins
How to model Policy/Value function?

Non-smooth + high-dimensional
Sensitive to situations. One stone changes in Go leads to different game.

Traditional approach

• Many manual steps
• Conflicting parameters, not scalable.
• Need strong domain knowledge.

Deep Learning

• End-to-End training
  • Lots of data, less tuning.
• Minimal domain knowledge.
• Amazing performance
Our computer Go player: DarkForest

- Top amateur level
- Release 3 month before AlphaGo, < 1% GPUs
Reimplementation of AlphaGo Zero

Generate Training data

\[ \theta_i \]

Update Models

Self-Replays

Zero-human knowledge

[Silver et al, Mastering the game of Go without human knowledge, Nature 2017]
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?
Distributed ELF

Putting AlphaGoZero and AlphaZero into the same framework

AlphaGoZero (more synchronization)
AlphaZero (less synchronization)
ELF OpenGo

• System can be trained with 2000 GPUs in 2 weeks.
• Decent performance against professional players and strong bots.
• Abundant ablation analysis
• Decoupled design, code highly reusable for other games.

We open source the code and the pre-trained model for the Go and ML community

http://github.com/pytorch/elf
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 strong bot (LeelaZero)

[158603eb, 192x15, Apr. 25, 2018]: 980 wins, 18 losses (98.2%)
Sample games versus Kim Jiseok (world #3)
Open Source

https://github.com/pytorch/ELF
**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><strong>MiniRTS</strong></td>
<td><strong>40,000</strong></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.
Training AI

Game internal data (respecting fog of war)

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

Win rate versus iterations

<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>
First Person Shooter (FPS) Game

Yuxin Wu, Yuandong Tian, ICLR 2017

Play the game from the raw image!
Curriculum Training

From simple to complicated

FlatMap

CIGTrack1
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
Visualization of Value functions

Best 4 frames (agent is about to shoot the enemy)

Worst 4 frames (agent missed the shoot and is out of ammo)
House3D: A rich and realistic 3D environment

Yi Wu, Georgia Gkioxari, Yuxin Wu

[Yi Wu et al, Building Generalizable Agents with a Realistic and Rich 3D Environment, ICLR 2018 workshop]
SUNCG dataset, 45K scenes, all objects are fully labeled.
House3D

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

https://github.com/facebookresearch/House3D
## Comparison

<table>
<thead>
<tr>
<th>Environment</th>
<th>3D</th>
<th>Realistic</th>
<th>Large-scale</th>
<th>Fast-speed</th>
<th>Customizable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atari (Bellemare et al., 2013)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenAI Universe (Shi et al., 2017)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Malmo (Johnson et al., 2016)</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>DeepMind Lab (Beattie et al., 2016)</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>VizDoom (Kempka et al., 2016)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AI2-THOR (Zhu et al., 2017)</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>House3D</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
</tr>
</tbody>
</table>
Architectures
Generalization capability

Training Success Rate on the Large Set

Generalization Success Rate on the Test Set
(trained on the large set)
Target: Bathroom
Target: Kitchen
Target: Dining Room
Future Directions

Hierarchical RL

Multi-Agent

RL applications

RL Systems

Model-based RL

RL for Optimization

Admiral (General)  
Captain  
Lieutenant

\[ \pi_\theta(a|s) \]

\[ s' = f_\phi(s, a) \]

Policy Optimization  
Model Estimation
How to do well in Reinforcement Learning?

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

Strong math skills

Experience on (distributed) systems

Parameter tuning skills

Strong coding skills
Thanks!