Applying Modern Reinforcement Learning to Play Video Games

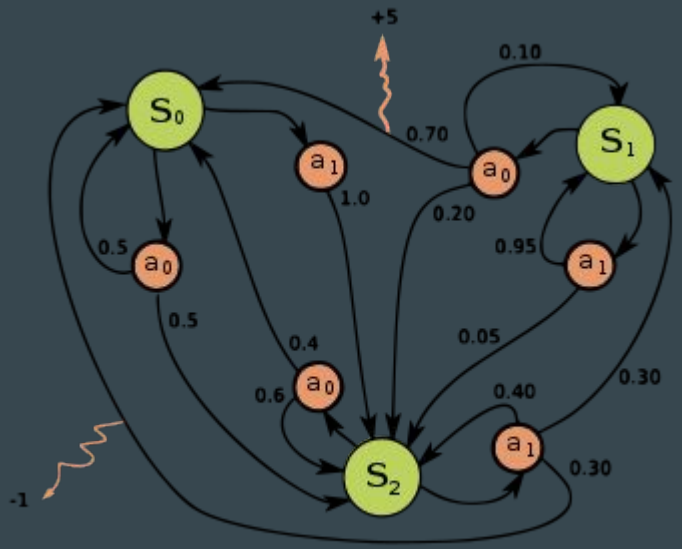
Computer Science & Engineering
Leung Man Ho
Supervisor: Prof. LYU Rung Tsong Michael

Outline

- Background
- Game Environment
- Vision
- System Design
- Methods
- Experiments & Results
- Future Work

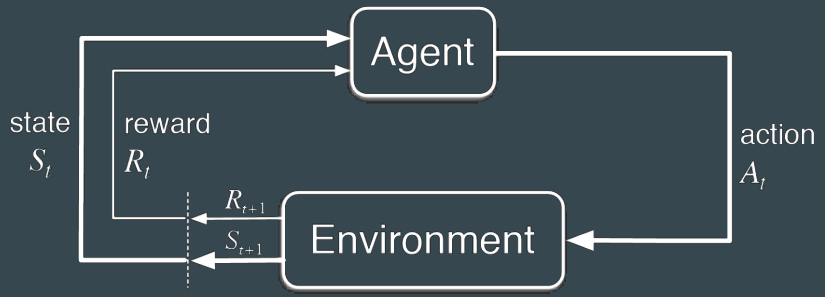
Background - Reinforcement Learning

- Reinforcement learning is learning what to do - Prof. Richard S. Sutton
- Often modelled as Markov Decision Processes
  - S: a finite set of states.
  - A: a finite set of actions.
  - T(s'|s, a): Transition model
  - R_a(s, s'): Reward model
  - γ: future discounted factor
- Objective
  - $\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})$,
  - Maximize discounted future reward



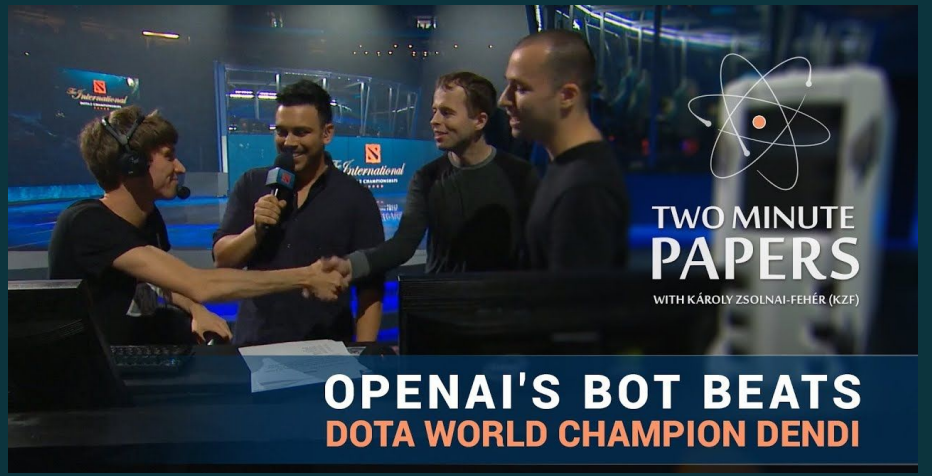
Background - Challenges

- Unknown transition and reward model
- Sparse and delayed reward
- Credit assignment problem
- Exploitation vs Exploration



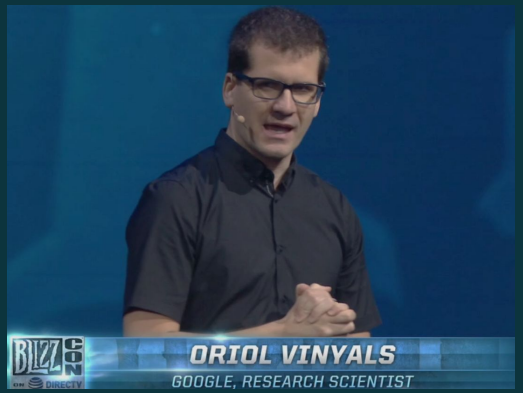
Background - Timeline

- 2014 - Deep Q Network
- 2016 - AlphaGo
- 2017 - OpenAI Dota Bot
- 2017 - AlphaGo Zero
- 2017 - And more!



Background - Motivation

- Explore the boundary of modern RL
- Select a challenging, unexplored and meaningful video game



Why video game? Why is it meaningful?

"At DeepMind, our mission is to solve intelligence and use that to solve complex real world problems, but in order to do that, we need to test our algorithmic ideas in challenging environments." - BlizzCon on DeepMind x Starcraft II

Game Environment - Discontinued Trials

Space Impact II

- Partially solved by our NEAT agent
- Limited skills required
- Not much left to be explored

Slither.io

- Interesting competitive environment
- Too much dependency
- Unstable and inconsistent training

Game Environment - Little Fighter 2

- LF2
  - Developed by CUHK Alumni
  - Visual fighting game
  - Very popular in HK
- Game
  - HP & MP
  - 7 keys, {up, down, left, right, attack, jump, defense}
  - Special abilities for each character, triggered by key sequences
  - Exploitable game objects

Game Environment - Why Little Fighter 2

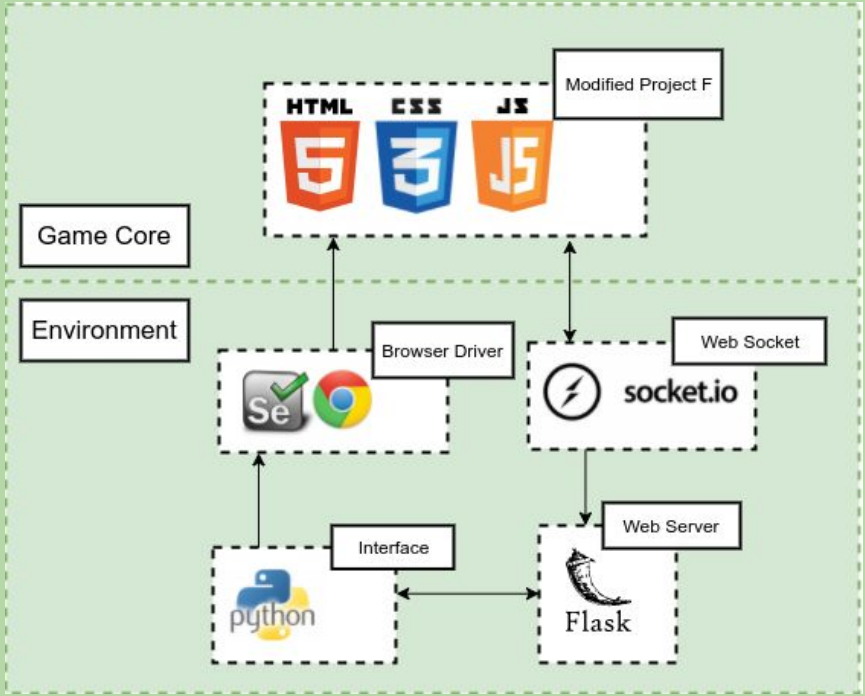
- Multi-agent competitive game
- Highly dynamic & skill intensive
- In-game AIs available as comparison
- Open-source (Initiated by Project F)

Vision

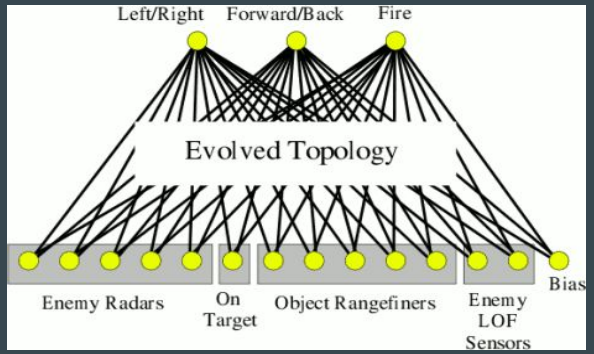
- Focus on 1v1 game
- Create RL agents that
  - Understand game objective and basic controls
  - Outperform in-game AIs
  - Demonstrate non-trivial playing style
  - Challenging for human
- Research importance
  - Explore the capabilities and limitations of RL
  - Proof of concept for game industry
    - Create dynamic AI with zero domain knowledge
    - Self improving
    - More amusing to play against

System Design - Overview

- Components
  - Game core
  - Environment interface
  - RL implementations
- Features
  - Separation of concerns
  - Fault tolerance
  - Efficient game simulation
  - Provide simple interface

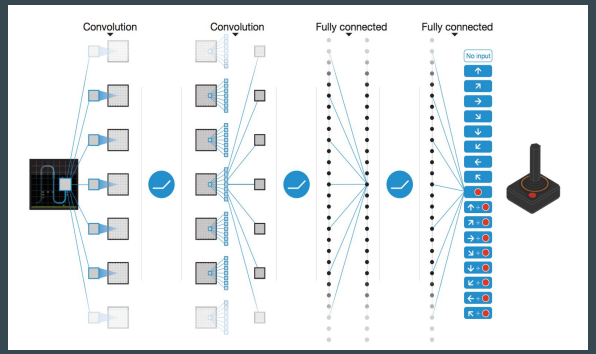


Methods - Frameworks



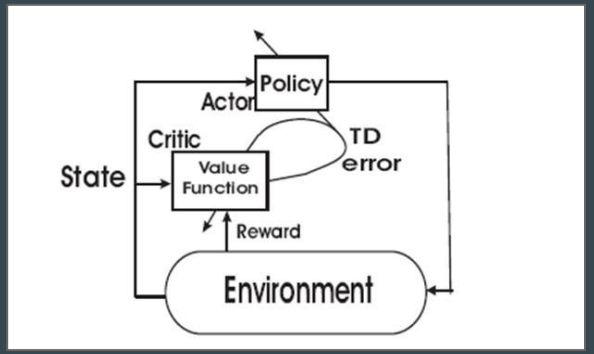
NeuroEvolution of Augmenting Topologies

- NEAT
- Proposed in 2002
- Evolutionary method



Deep Q-Network

- DQN
- Proposed in 2014
- Value iteration method

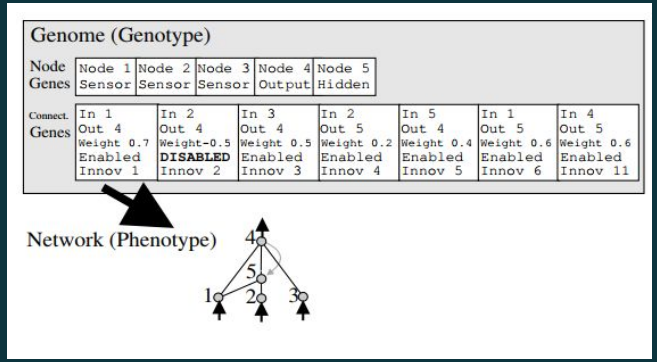


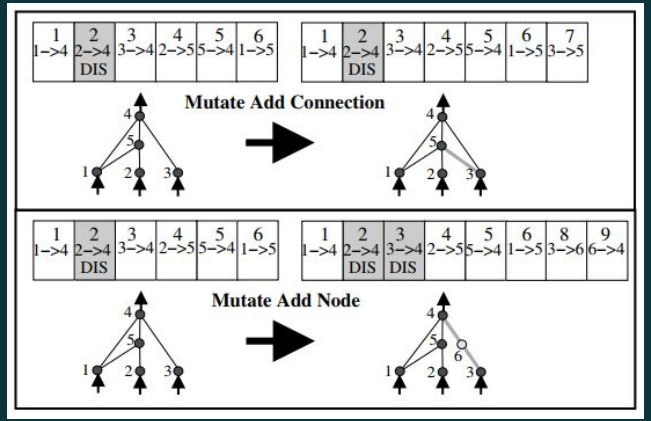
Actor Critic using Kronecker-Factored Trust Region

- ACKTR
- Proposed in 2017
- Actor critic method

Methods - NEAT

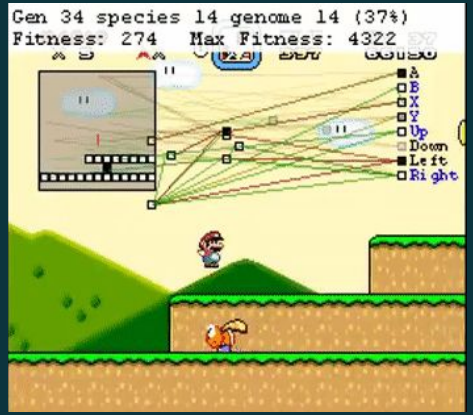
- Main idea
  - Genetic Algorithm
  - Evolve network weight and **topology**
  - Preserve innovation using speciation
  - Search from minimal structure
- High level details
  - Encode policy network into genomes
  - Separate genomes into species based on network similarity
  - Fitness normalized by species size
  - Reproduction within each species





Methods - NEAT

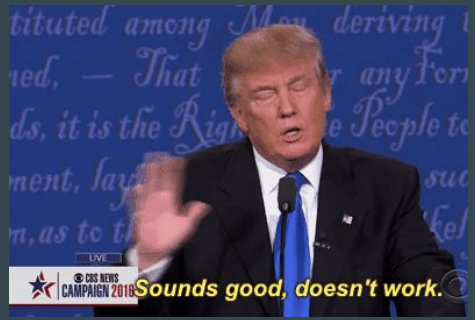
- Success story
  - SethBling Mario project in 2015
  - Our Space Impact trial
- Motivated us to apply it first on LF2



Methods - DQN

- Proposed by DeepMind
- Q-learning
  - Q(s, a): maximum discounted reward the optimal policy can get by committing a under state s
  - Bellman optimality: $Q(s,a) = r + \gamma \max_{a'} Q(s',a')$,
  - Value iteration: $Q_i(s,a) = Q_{i-1}(s,a) + \alpha \underbrace{(r + \gamma \max_{a'} Q_{i-1}(s',a') - Q_{i-1}(s,a))}_{\text{TD-error}}$,
  - Converge when TD-error ~= 0
  - Follow epsilon-greedy exploration
- What if, state space and action space is huge?
- Idea:
  - Use neural network to approximate Q-function
  - Treat TD-error as loss function apply stochastic gradient descent!

$$L(\theta) = E_{s,a,r,s'}[((r + \gamma \max_{a'} Q(s',a';\theta')) - Q(s,a;\theta))^2],$$



Methods - DQN

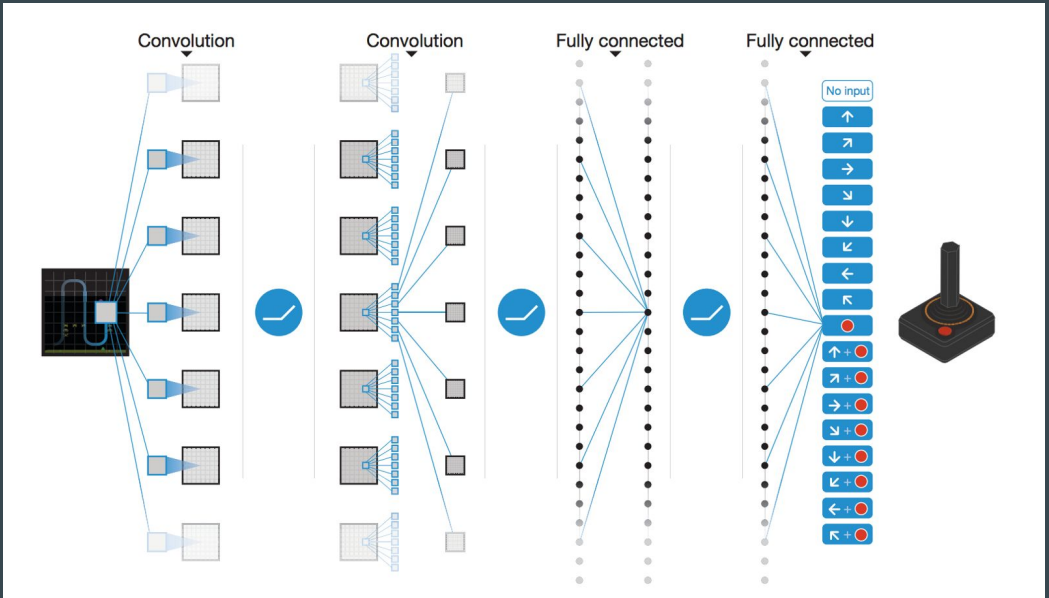
- Problem: Proven hard to converge
  - Consecutive states are highly correlated
  - Data distribution affected by current policy
  - Moving parts in loss function

$$L(\theta) = E_{s,a,r,s'}[((r + \gamma \max_{a'} Q(s', a'; \theta')) - Q(s, a; \theta))^2],$$

- DeepMind solutions
  - Experience replay
  - Freeze target network periodically

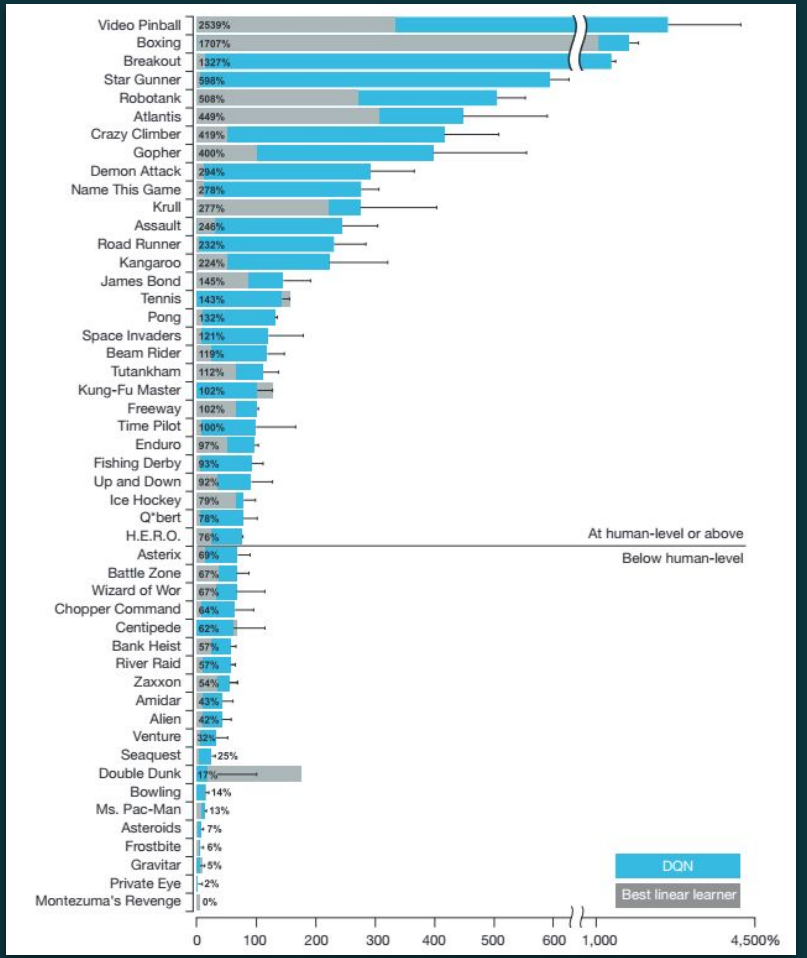
Methods - DQN structure

- Preprocess pixels into 84x84 grayscale
- Stacked 4 consecutive frames
- Output |A| Q-values



Methods - DQN

- Success story
  - Atari 2600 environments
- DQN received many upgrades
  - Double Q-learning
  - Prioritized replay
  - Dueling Network



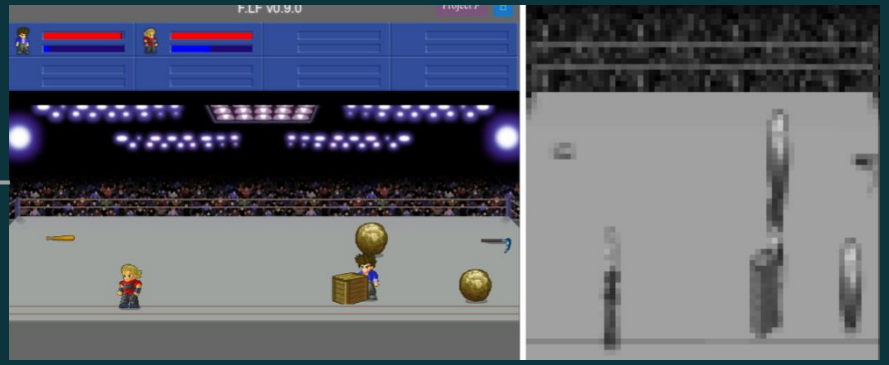
Methods - ACKTR

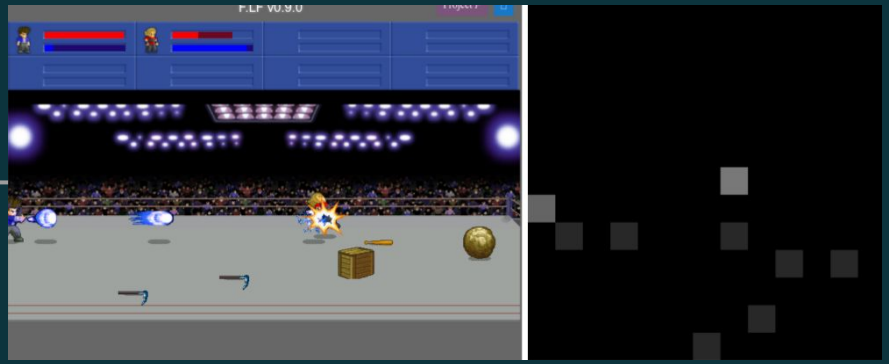
- Proposed by OpenAI (Aug 2017)
- Actor Critic
  - Directly estimate optimal policy
    - Output a distribution $\pi(a|s)$
    - Optimize expected reward $J(\theta) = E_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)],$
    - Reward gradient: $\nabla_\theta J(\theta) = E_\pi[\sum_{t=0}^{\infty} \psi^t \nabla_\theta \log \pi_\theta(a_t|s_t)],$
  - Value function needed for ψ^t
    - Advantage function: $A^\pi(s_t, a_t) = r(s_{t+1}, a_{t+1}) + \gamma V(s_{t+1}) - V(s_t),$
    - Can be computed similar as DQN
- Contribution
  - Not using Gradient Descent
  - Using Kronecker-factored approximate curvature (K-FAC) with trust region
- Proven more sample efficient than DQN

Methods - Feature Types**

- Pixel
- Sprite-based map
- Handcrafted
  - Two tuples of game information
  - 3D coordinates
  - 3D velocities
  - Facing direction
  - Action states

- Scale all feature reasonably





Methods - Reward Shaping**

- Challenges
  - Natural reward is too sparse
  - Optimal strategy must remain an invariant
  - Positive reward cycle issue
- Potential-based reward function $\Phi(s)$
  - Assign potential value for states
  - Reward is the difference in potential function
  - Resolve positive reward cycle issue

- All reward are normalized by a constant

Methods - Reward Shaping**

- Basic reward form
  - $\Phi(s) = ally\_hp(s) + k_1 \times num\_ally(s) - enemy\_hp(s) - k_2 \times num\_enemy(s)$
- Distance reward form
  - $\Phi(s) = ally\_hp(s) + k_1 \times num\_ally(s) - enemy\_hp(s) - k_2 \times num\_enemy(s) - k_3 \times dist\_enemy(s)$
  - Reward intermediate behaviors - exploration reward
  - Tune to be small relative to the whole term

Experiments & Results - Overview

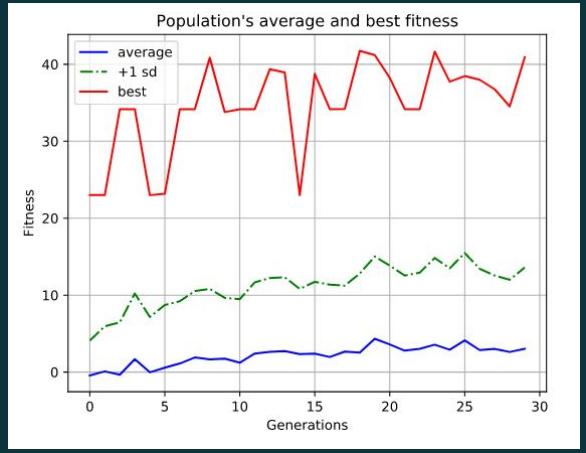
- Phase 1: Static agent task
- Phase 2: In-game AI
- Phase 3: Self play

Experiments & Results - Static agent task

- Motivation
  - Make sure the agent understand game objective
- Very important task
- Hard for random agent to solve it

Static agent task - NEAT

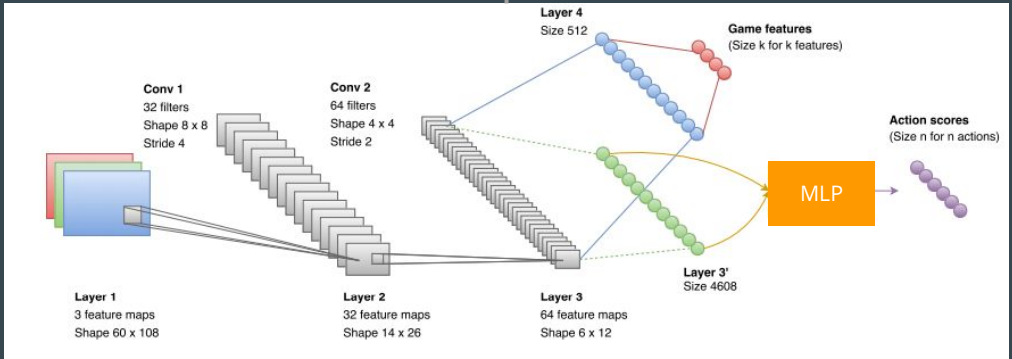
- Failed under multiple setups
  - Sprite-based vs handcrafted features
  - Basic vs Distance reward
- Problems
  - Hard to tune
  - Low sample efficiency (> 10M frames)
  - Only learning trivial strategy
  - Structure too simple
  - Weak against random perturbation
- Discontinued

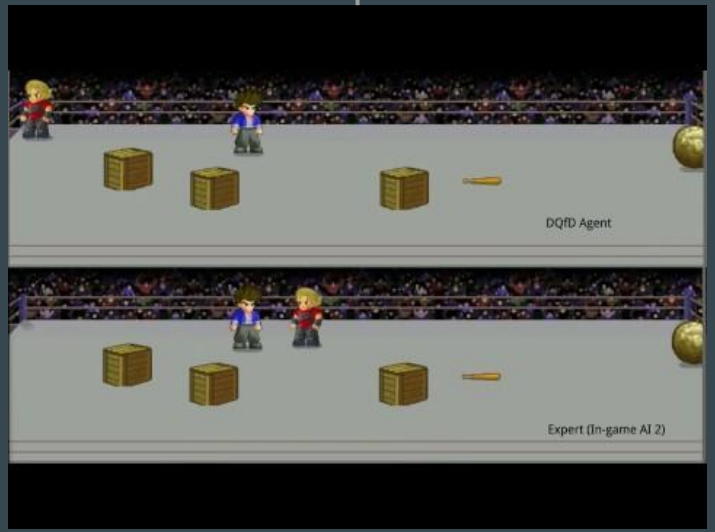


Static agent task - CNN-based DQN/ACKTR

Failed attempts:

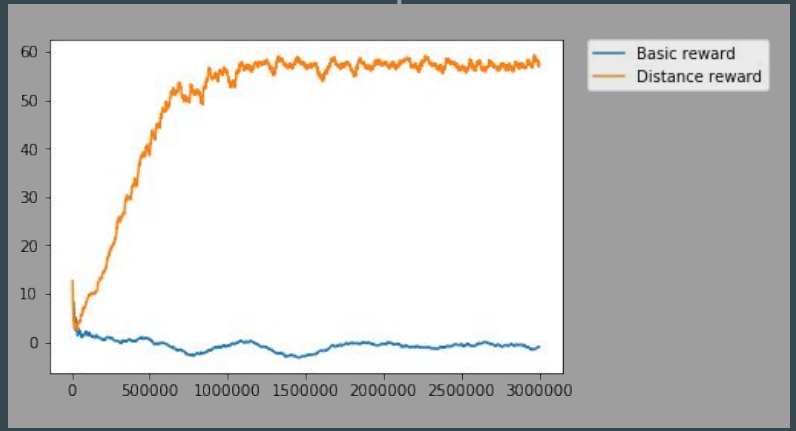
- Vanilla DQN
- Deep Q-learning from Demonstrations (DQfD)
- DQN with game feature augmentation
- CNN-ACKTR





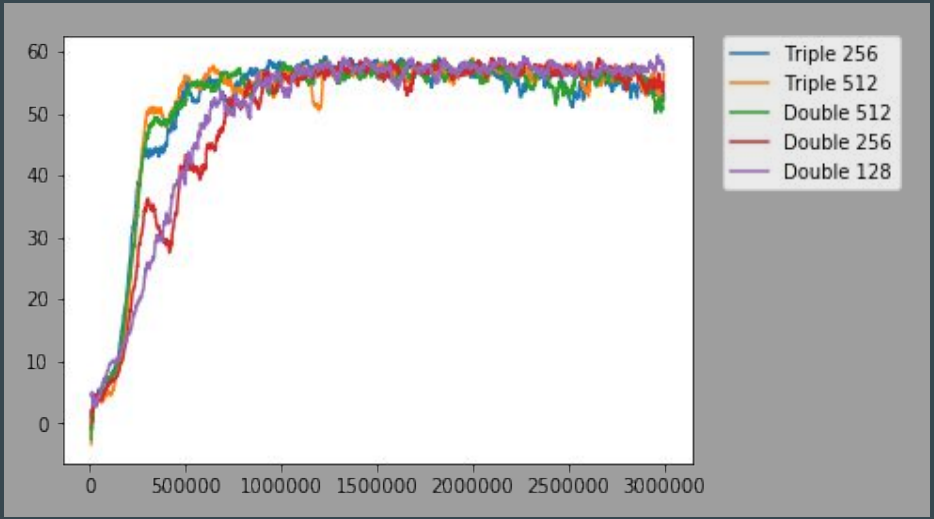
Static agent task - MLP-based DQN

- Formed by removing CNN layers
- First and most important milestone
- Important observations
  - Handcrafted feature is crucial
  - Distance reward is crucial



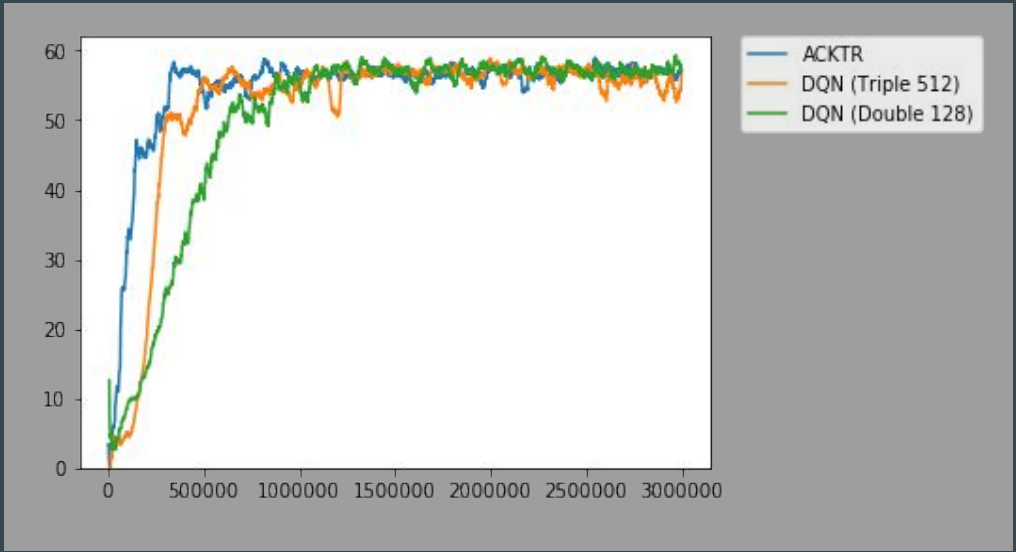
Static agent task - MLP-based DQN

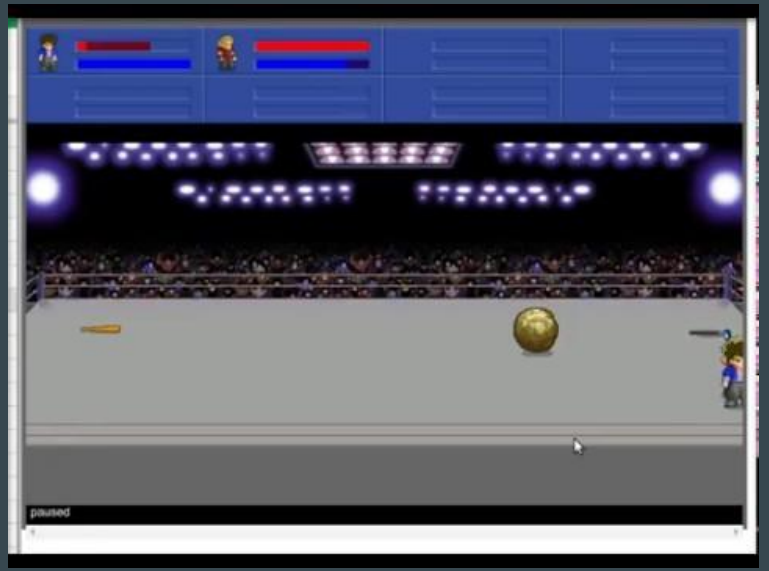
- Simpler network perform better when inspected
- Complex network train faster, but more unstable
- Stuck issue was sometimes observed, especially for complex network



Static agent task - MLP-based ACKTR

- Converge in 30 minutes under 6 parallel agents
- No stuck issue at all!





Static agent task - Summary

- Perception while learning is hard in this domain
- Simple network perform better
- ACKTR perform generally better than DQN

In-game AI task - Provided targets

In-game AI 0

- Uses all special abilities
- Good at close and long range
- Unfair comparison
- Challenging to mid level player

In-game AI 1

- Move away from target
- Launch jump kicks from angles
- Challenging to mid level player

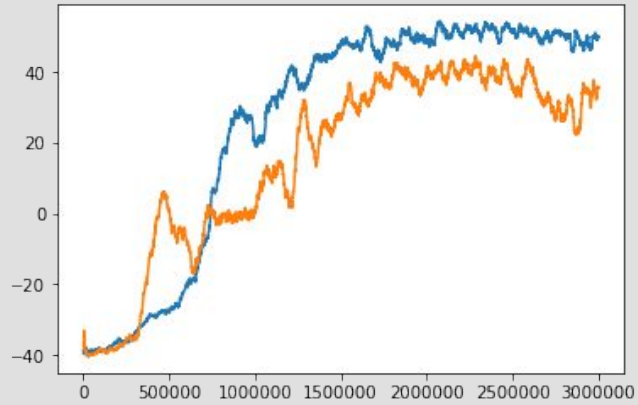
In-game AI 2

- Mainly close range
- Move back and forth and attack
- Challenging to amatuer level player

In-game AI task - Operation mode

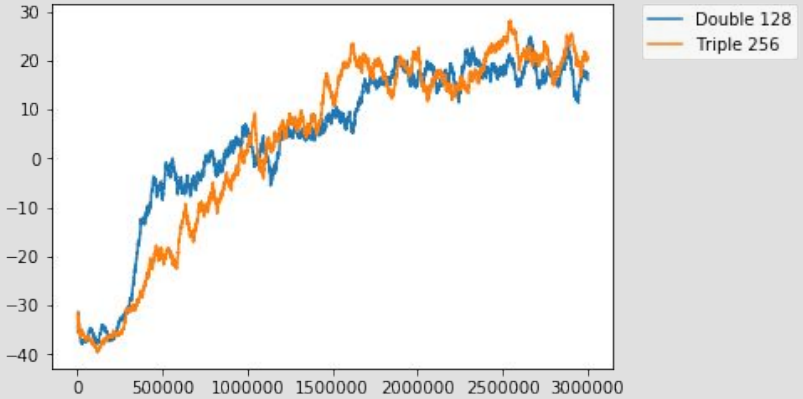
- Fixed opponent mode
  - Study whether model is able to outperform each in-game AI
- Random opponent mode
  - Force agent to learn more general strategy

Fixed opponent mode - MLP-based DQN



In-game AI 1

- 100% win rate



In-game AI 2

- 85% win rate

Fixed opponent mode - MLP-based DQN

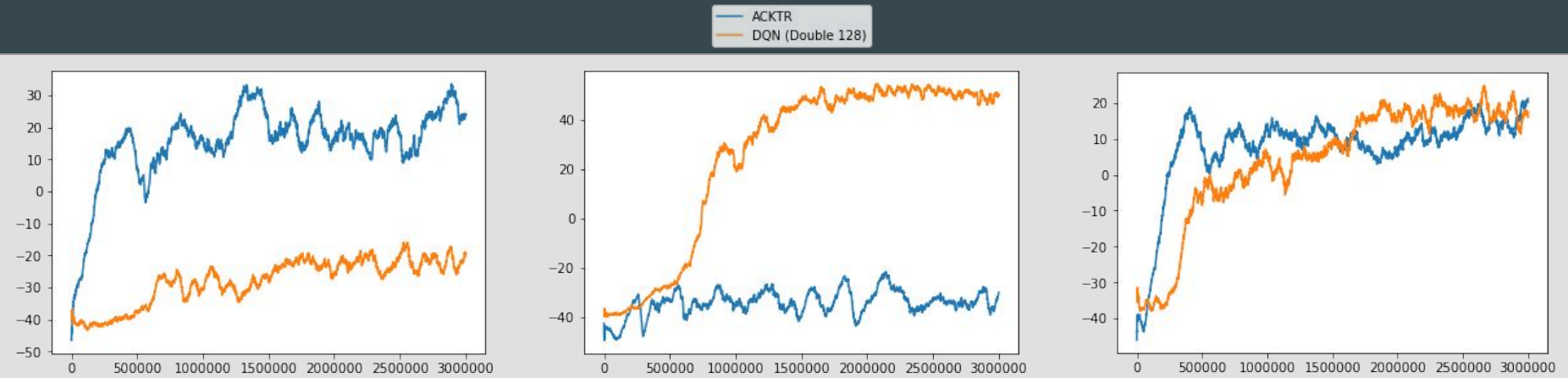
In-game AI 1

- Keep chasing target
- Discovered run -> attack combo
- Target cannot launch its signature move

In-game AI 2

- General playing style
- Smart defensive tactics **

Fixed opponent mode - MLP-based ACKTR



In-game AI 0

- 80% win rate
- The only setup that outperform AI 0

In-game AI 1

- 5% win rate
- Cannot converge

In-game AI 2

- 100% win rate
- Converge much faster

Fixed opponent mode - MLP-based ACKTR

In-game AI 0

- Exploited a specific design flaw
- The only agent that uses fire blast
- Useless against other agents

In-game AI 1

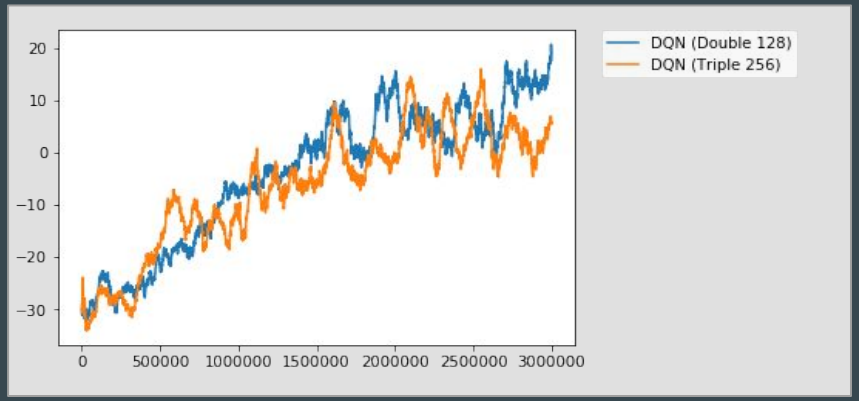
- Not doing anything sensible

In-game AI 2

- Exploited a specific design flaw
- Trivial and boring strategy
- Useless against other agents

Random opponent mode - MLP-based DQN

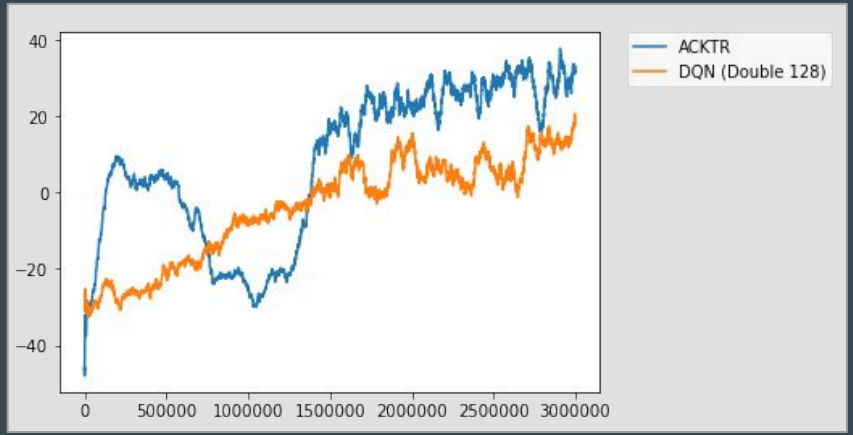
- More general performance
- Improve performance vs hardest AI
- Still, unable to dominate all
- Often stuck in static task



Network \ Target	Static Agent	In-game AI 0	In-game AI 1	In-game AI 2
Double 128 (Random mode)	40	45	90	20
Triple 256 (Random mode)	25	5	40	20
Double 128 (Fixed mode)	15	20	100	85

Random opponent mode - MLP-based ACKTR

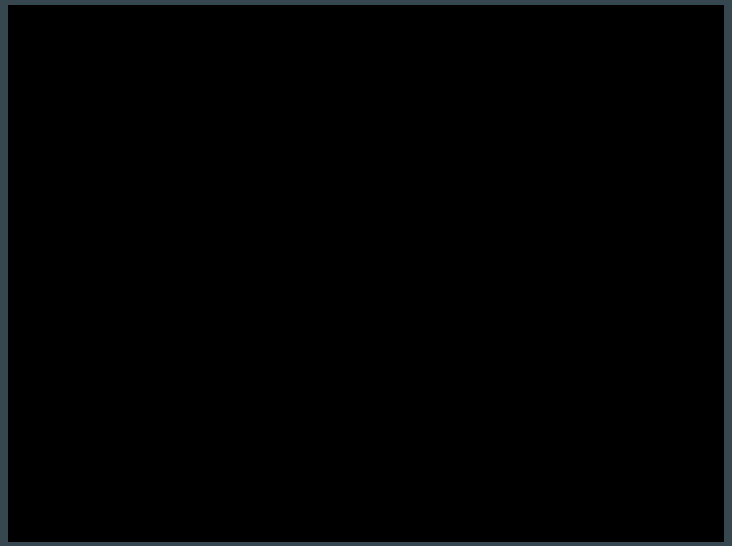
- Significantly improve generality
- Better avg. reward/win rate than DQN
- Never stuck in static task



Network \ Target	Static Agent	In-game AI 0	In-game AI 1	In-game AI 2
ACKTR (Random mode)	100	15	100	45
DQN Double 128 (Random mode)	40	45	90	20
DQN Triple 256 (Random mode)	25	5	40	20

Random opponent mode - MLP-based ACKTR

- More general performance
- Improve performance vs hardest AI
- Still, unable to dominate all
- Often stuck in static task
- Why ACKTR learn stronger skills **



Network \ Target	Static Agent	In-game AI 0	In-game AI 1	In-game AI 2
ACKTR (Random mode)	100	15	100	45
DQN Double 128 (Random mode)	40	45	90	20
DQN Triple 256 (Random mode)	25	5	40	20

In-game AI task - Summary

- For each in-game AI, obtained that agent dominate it
  - Some strategies are highly specific
- Obtained agents that have decent win rate against all opponents
  - More general game play
  - Challenging to human**

Self Play

- Motivation
  - Experience not constrained by in-game AI
  - Continuously pressured to improve
  - Hope to induce more general game play
- Self play curriculum
  - Start from completely random play
  - Always compete against latest network

Self Play - MLP-based DQN

- Able to defeat some unseen AIs!
- Simplest network work poorly
- Defeated best DQN agent trained against random in-game AIs!
  - Win rate 60% (10% draw game)

Network \ Target	Static Agent	In-game AI 0	In-game AI 1	In-game AI 2
DQN (Double 128)	0	10	75	5
DQN (Double 256)	70	5	80	30
DQN (Triple 256)	20	5	95	40
DQN (Triple 512)	5	5	70	60

Self Play - MLP-based ACKTR

- Able to defeat some unseen AIs, but better!
- Defeated best ACKTR agent trained against random in-game AIs!
  - Win rate 100%

Network \ Target	Static Agent	In-game AI 0	In-game AI 1	In-game AI 2
ACKTR	85	15	55	70
DQN (Double 128)	0	10	75	5
DQN (Double 256)	70	5	80	30
DQN (Triple 256)	20	5	95	40
DQN (Triple 512)	5	5	70	60

Self Play - Summary

- ACKTR self play agent has more general skill against DQN self play agent
- Self play agents defeated random opponent trained agents
  - Although not as strong against unseen in-game AI
  - Showing that self play > random targets in some aspects

Self play curriculum

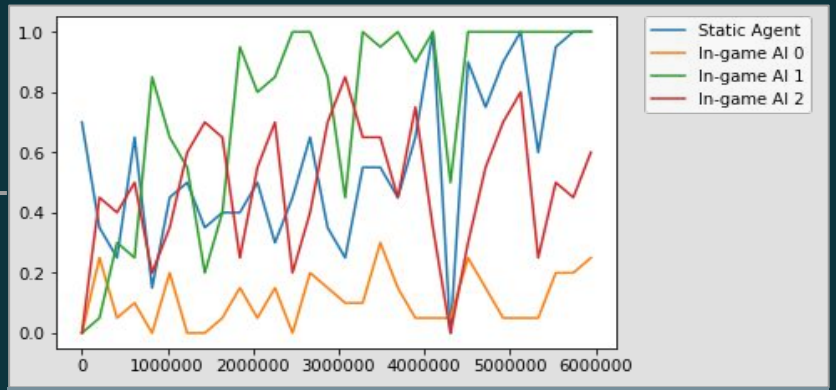
Network \ Target	Static Agent	In-game AI 0	In-game AI 1	In-game AI 2
ACKTR	85	15	55	70
DQN (Double 128)	0	10	75	5
DQN (Double 256)	70	5	80	30
DQN (Triple 256)	20	5	95	40
DQN (Triple 512)	5	5	70	60

Random in-game AIs curriculum

Network \ Target	Static Agent	In-game AI 0	In-game AI 1	In-game AI 2
ACKTR (Random mode)	100	15	100	45
DQN Double 128 (Random mode)	40	45	90	20
DQN Triple 256 (Random mode)	25	5	40	20

Future Work - Improve self play

- Motivation
  - Time wasted in symmetric configuration
  - Catastrophic forgetting
- Ideas
  - AlphaGo style self play: Opponent sampling
  - Hybrid self play curriculum



A separate ACKTR self play experiment

Future Work - Diversify play style

- Motivation
  - Agents doesn't use special abilities (except one trained ACKTR agent)
  - No information in features regarding special abilities
  - Limited dynamics
- Ideas
  - Feature engineering
  - Encode action history into feature
  - Deep Recurrent Q-Network (DRQN)

Future Work - Online AI evaluation platform

- Motivation
  - Cannot objectively measure AI skills
  - Cannot systematically discover AI weakness
- Idea: Online platform for human to interact with the RL agent
  - Receive objective match metrics
  - Receive feedback on AI skillfulness
  - Double blind test on which AI is most enjoyable (in-game AI vs RL agent)

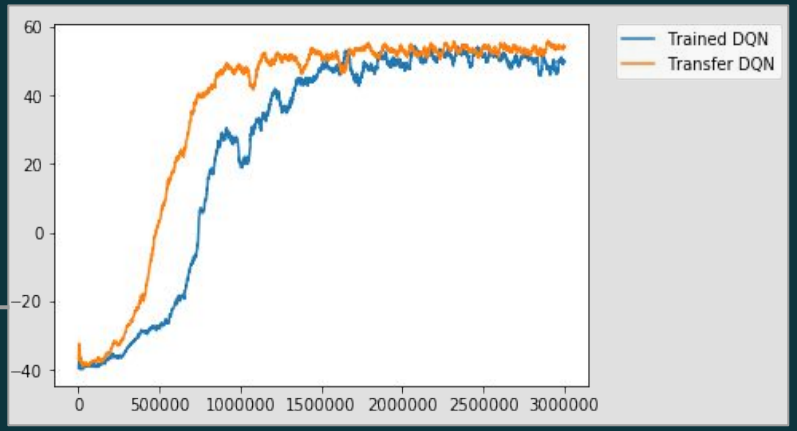
Appendix

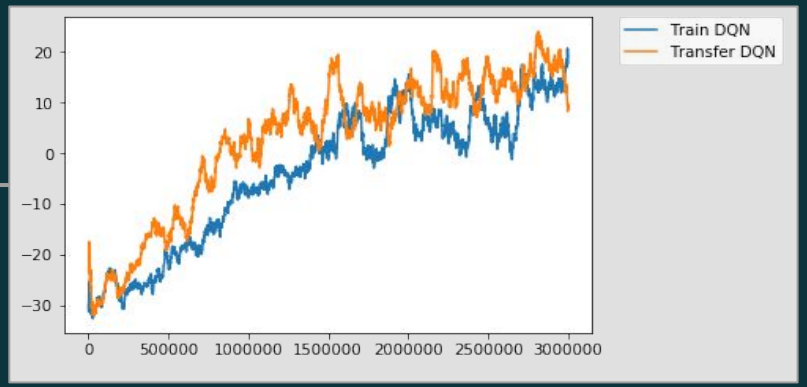
- Transfer learning
- Best agents against human
- Why ACKTR policy is smoother than DQN?
- What is smart defensive tactic?
- More interesting fights
- Contd: System Design

Appendix: Transfer learning

- Uses trained network as base network
  - Trained agent on static task
- Improved training on different tasks
  - In-game AI 1
  - In-game AI 2
- Skill transferred

Network \ Target	Static Agent	In-game AI 0	In-game AI 1	In-game AI 2
Trained DQN (Double 128)	0	10	75	5
Transfer DQN (Double 128)	30	5	90	25





Appendix: Best agent against human

- No obvious exploitable flaws
- Hard pressed to win (if not using special attack for fairness)

ACKTR from random mode

- Keep chasing target
- Discovered run attack
- Target cannot launch its signature move

ACKTR from self play

- General playing style
- Defensive smart tactics

Appendix: Why ACKTR policy is smoother than DQN

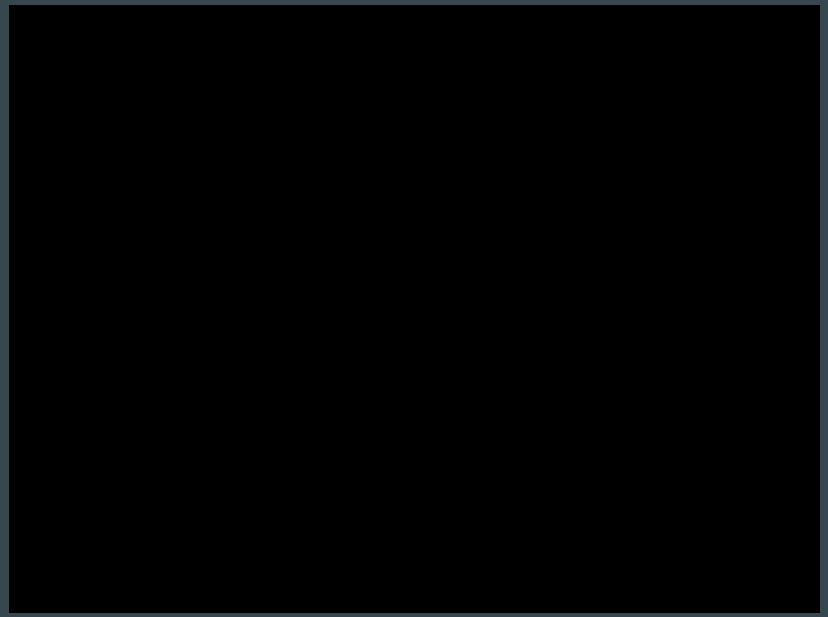
- Q-learning
  - Nearly deterministic (except epsilon)
  - Trappy state cycle
  - Confusion in training (fight or flight) under same state
- Policy gradient method
  - Stochastic policy
  - Learn an optimal action probability distribution under same state

Appendix: What is smart defensive tactic?

- Learned by DQN in fixed mode against in-game 2
- We called it Mexican-standoff
- Why is it smart (at least against in-game AI 2)

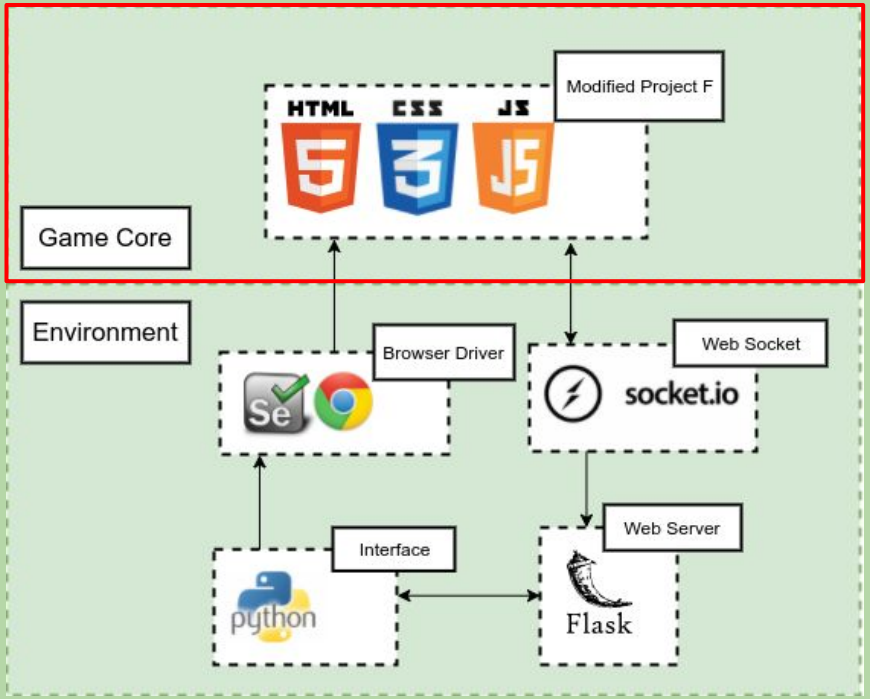
Appendix: More interesting fights

- ACKTR self play vs all in-game AIs



System Design - Game Core

- Modified to pause on each frame
- Preprocess game information
  - State
  - Reward
  - Terminal signal
  - Additional information
- Send to Python server
- Wait for feedback



System Design - Environment

- Inspired by OpenAI Gym Env
- Constructor
  - Configure and start the game
- Reset
  - Reset a game episode
- Step
  - Commit action on frame
  - Obtain info. on next frame
- Render
  - Visualize pixel-like state