WillPressLeverForFood at DeepHack.RL

Alexander Guschin, Sergey Korolev, Sergey Ovcharenko,

Sergey Sviridov, Max Kharchenko

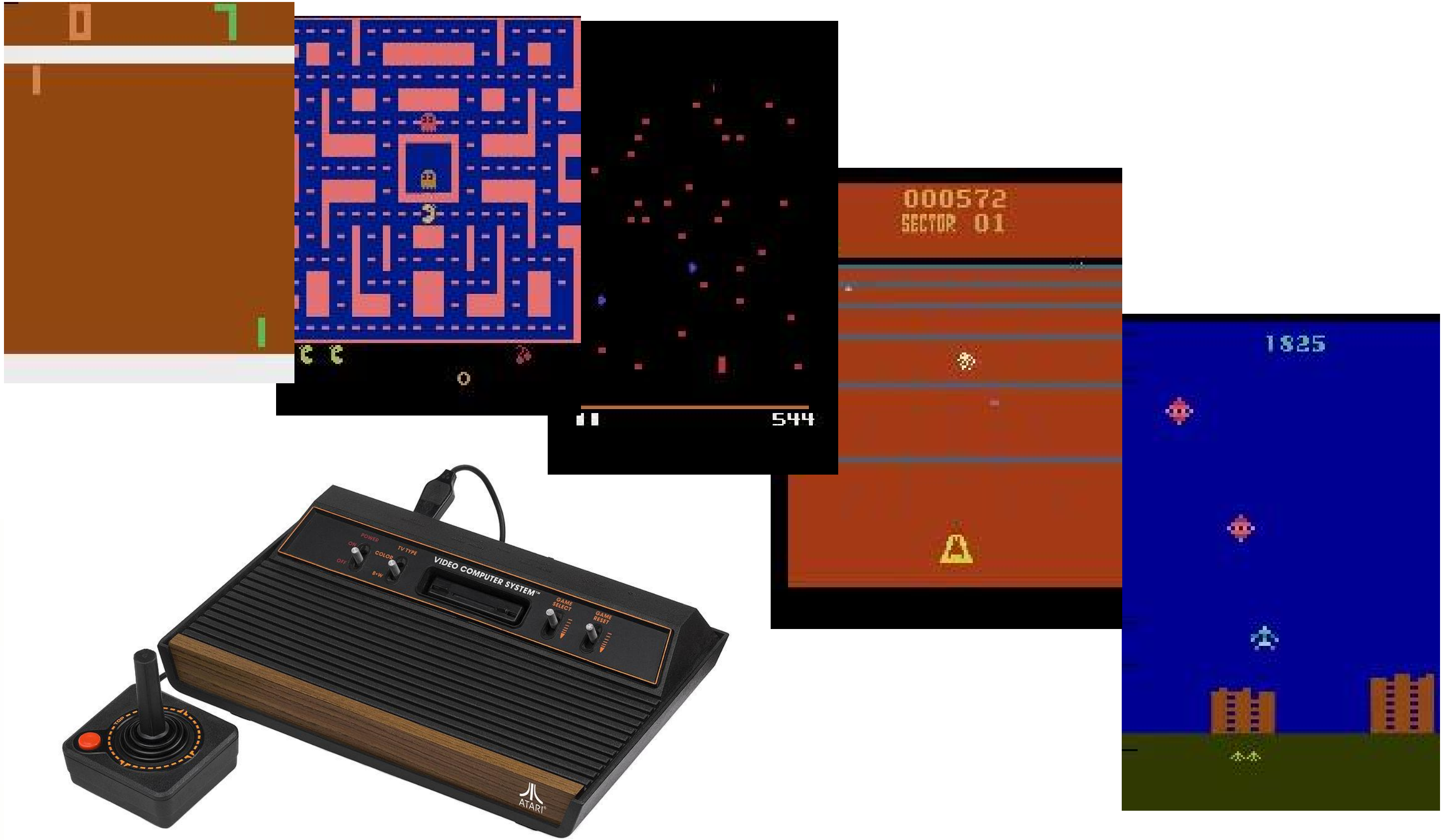
Moscow 2017

What is DeepHack

- State-of-art problem
- 7 days of programming and lectures from top notch researchers
- Bleeding edge technologies and developments
- And still hackathon and team challenge

Atari Games

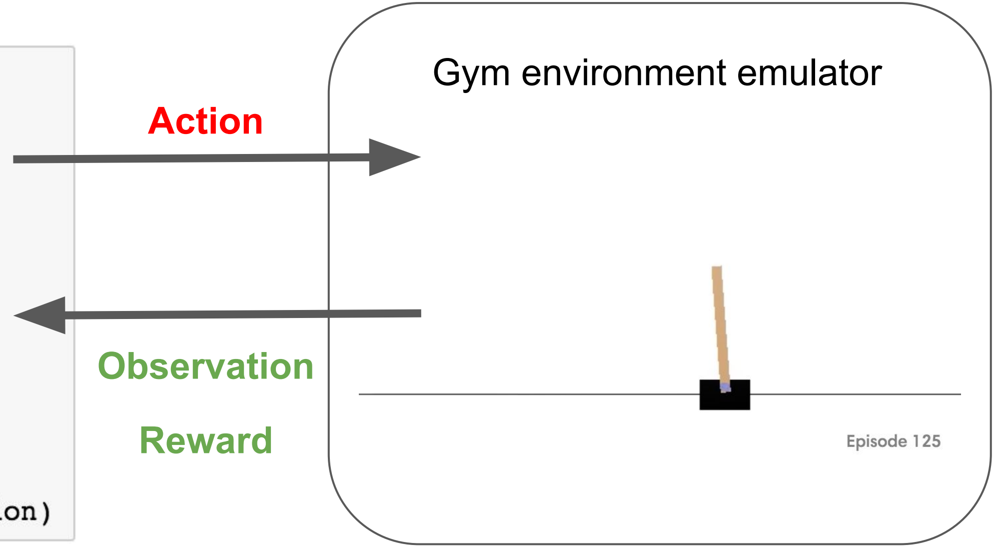
100+ games



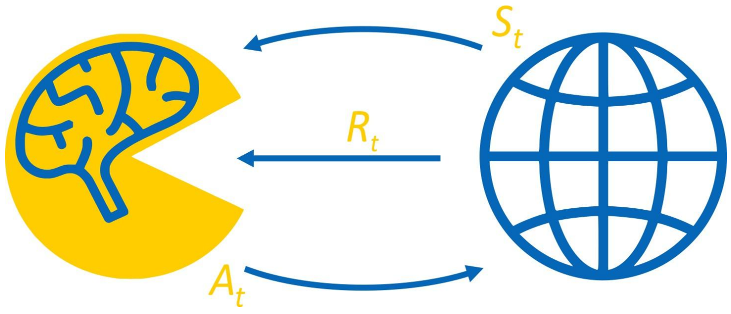
OpenAI Gym and Reinforcement learning

- Emulator of environments (including Atari 2600 games)

```python
import gym

# Make environment
env = gym.make('CartPole-v0')

# Reset environment
env.reset()
for _ in range(1000):

    # Take a random action
    action = env.action_space.sample()
    # Collect reward and new observation
    observation, reward, done, info = env.step(action)
```



Markov Decision Process



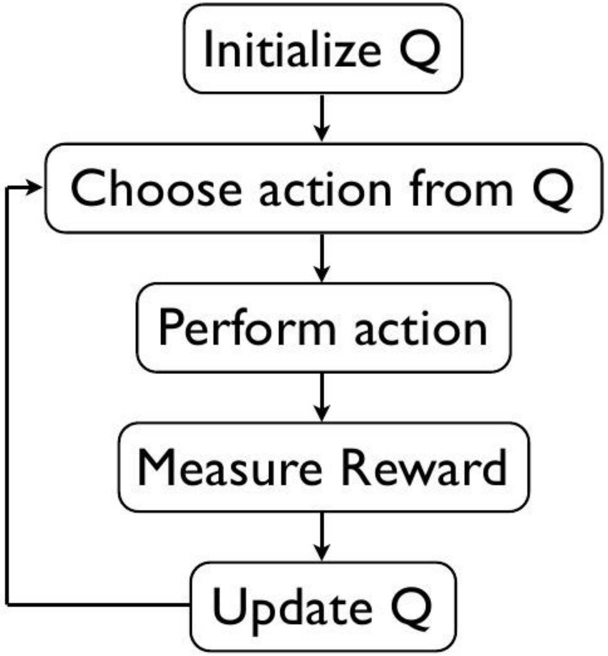
Setup

- Set of states: S
- Set of actions: A
- Immediate reward: $R_a(s, s')$
- Transitions: $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$
- Discount factor: $\gamma \in [0, 1]$

Goal

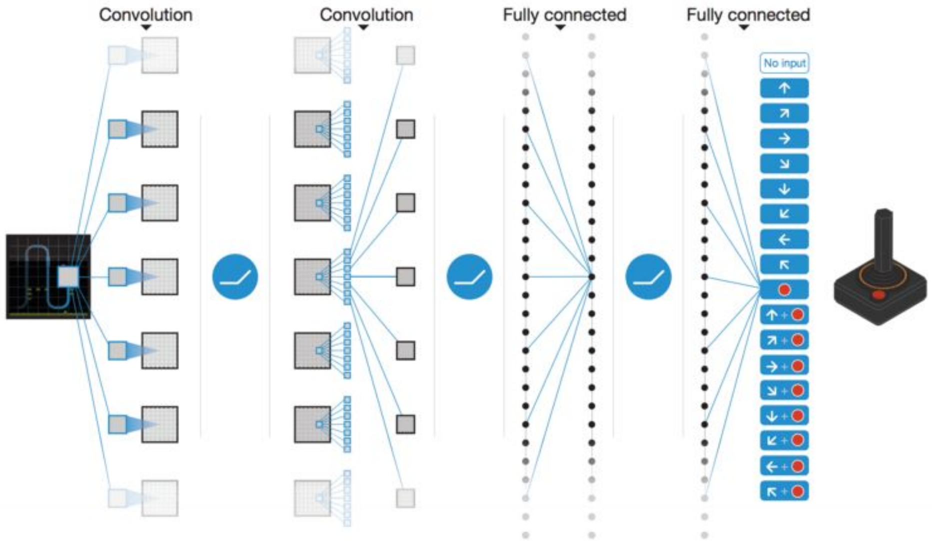
- Find a policy that maximizes the expected future rewards

Q-learning



$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma \max_{a'} Q(S',a') - Q(S,A) \right)$$

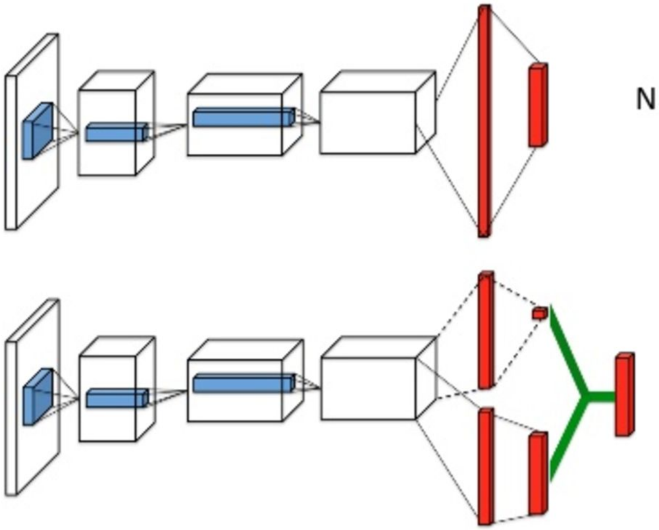
Deep Q-Networks



$$Q^*(s,a) \approx Q(s,a;\theta)$$

$$L_i(\theta_i) = \mathbb{E}\left(r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) - Q(s,a;\theta_i)\right)^2$$

Double DQN and Dueling DQN



Natural DQN

Dueling Network:
decomposing state value V and
Action Advantage value A

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s).$$

Policy gradient methods

- Represent policy by deep network with weights $\mathbf{u}$

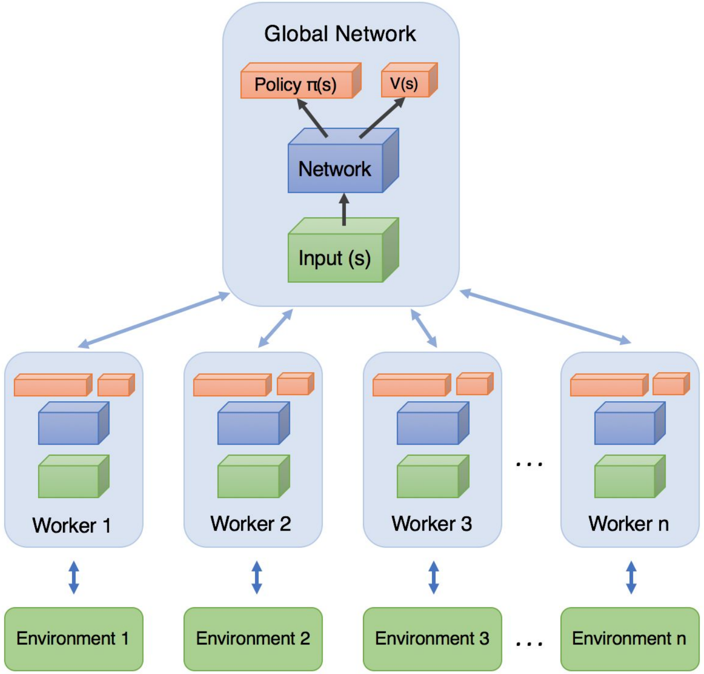
$$a = \pi(a|s, \mathbf{u}) \text{ or } a = \pi(s, \mathbf{u})$$

- Define objective function as total discounted reward

$$L(\mathbf{u}) = \mathbb{E}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots \mid \pi(\cdot, \mathbf{u})\right]$$

- Optimise objective end-to-end by SGD
- i.e. Adjust policy parameters $\mathbf{u}$ to achieve more reward

Actor-critic and a3c



- Actor is updated towards target

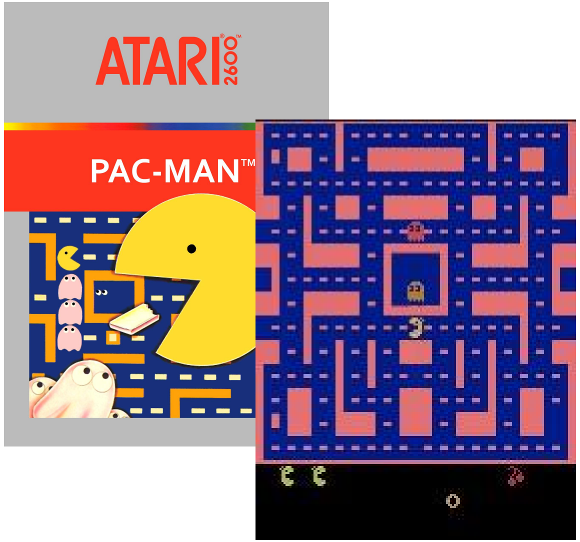
$$\frac{\partial l_u}{\partial \mathbf{u}} = \frac{\partial \log \pi(a_t|s_t,\mathbf{u})}{\partial \mathbf{u}}(q_t - V(s_t,\mathbf{v}))$$

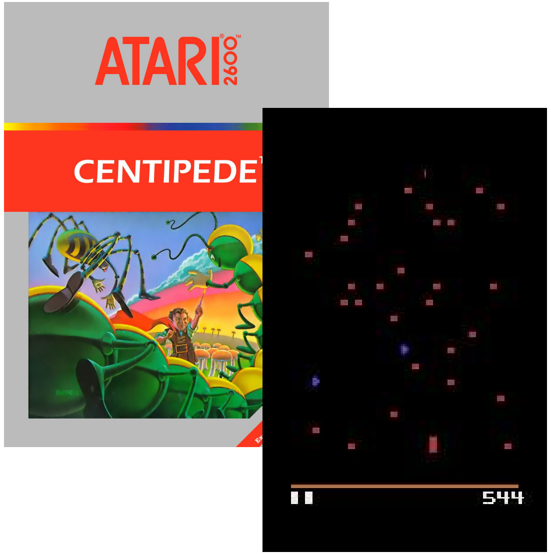
- Critic is updated to minimise MSE w.r.t. target

$$l_v = (q_t - V(s_t,\mathbf{v}))^2$$

- Each worker has own copy of environment
- During rollout worker accumulate gradients
- After rollout worker update global network

Deephack Games





Preliminary stage

Skiing-v0

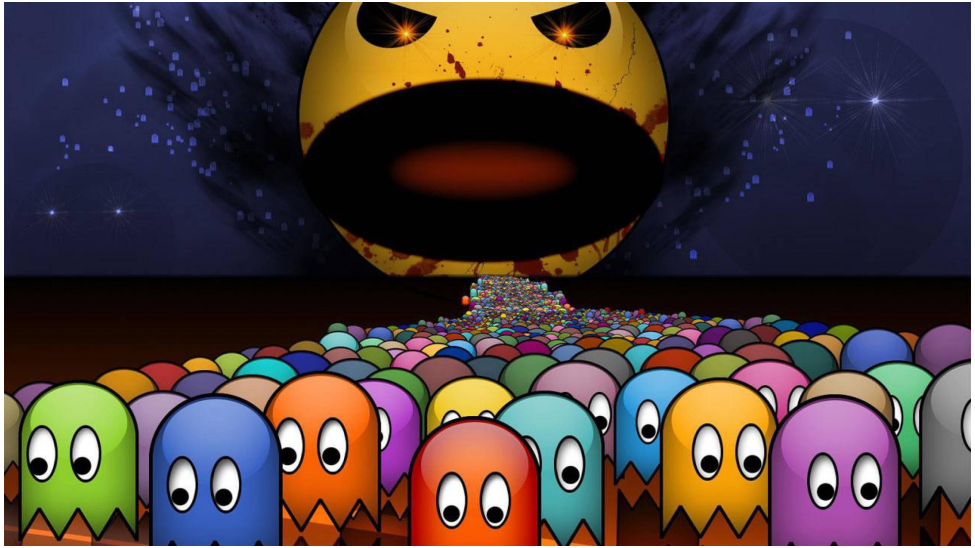
Main stage

MsPacman-v0

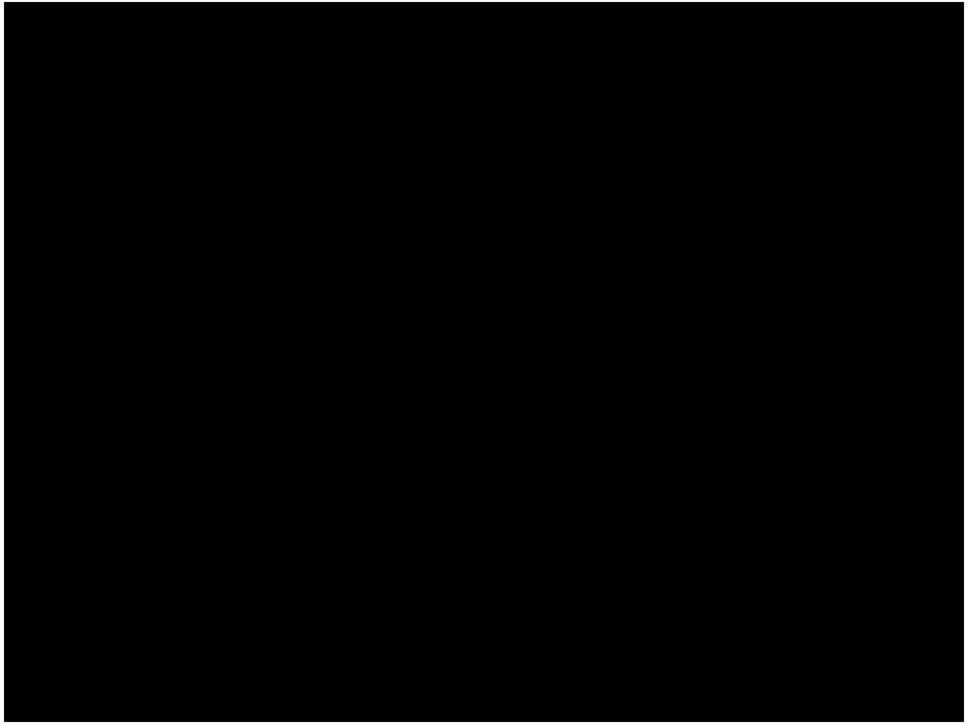
Code Freeze stage

Centipede-v0

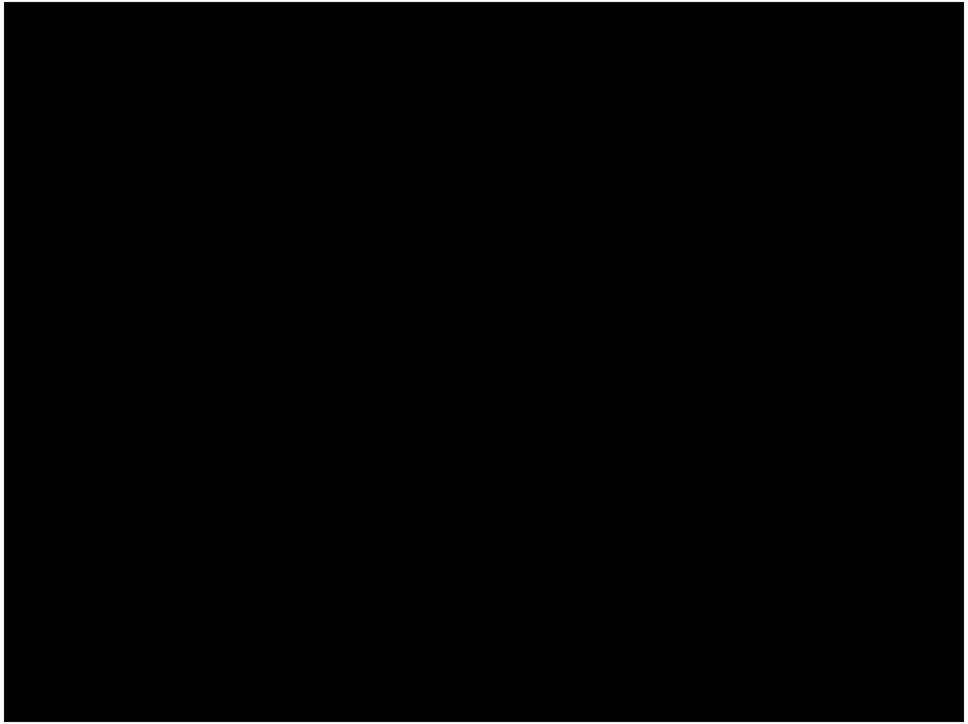
Expected behaviour



Reality:Skiing

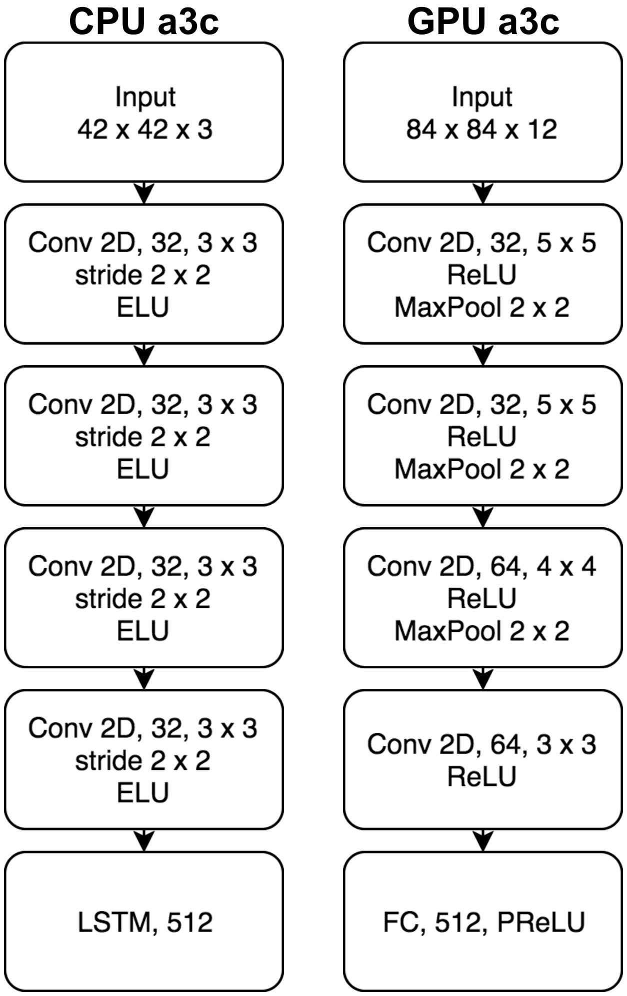


Reality: Pacman



Models selection

- DQN-networks were bad :(
- Two a3c models
  - GPU implementations w/o LSTM
  - CPU implementation with LSTM
- Random actor as backup strategy
- Metaestimator for model selection (by max score)

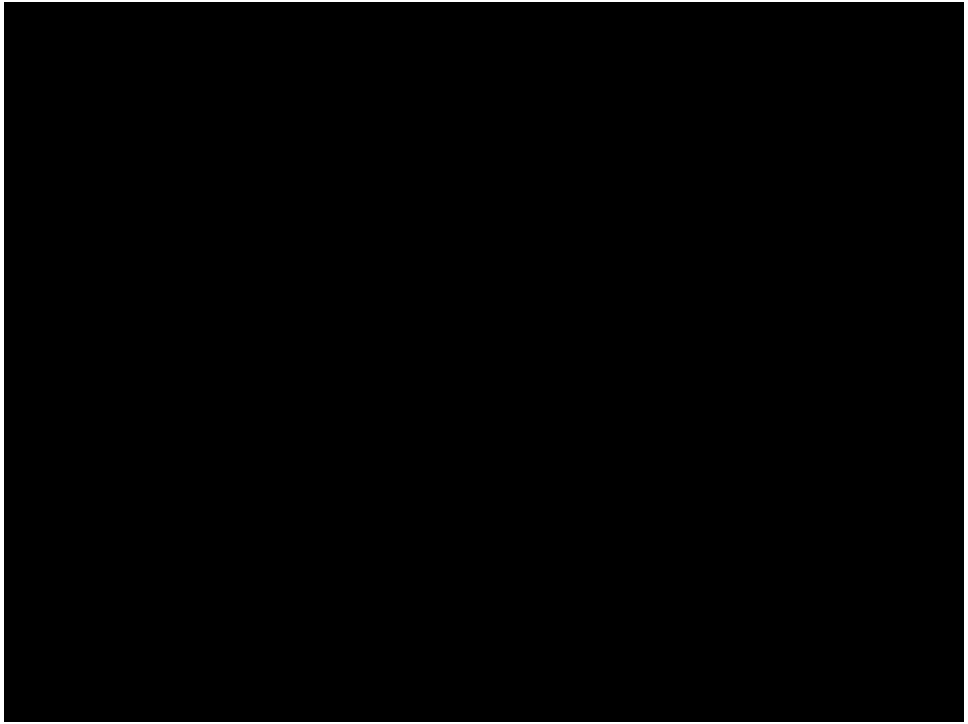


Hyperparameters optimization

- Preprocessing tweaking
  - A3C preprocessed images to grayscale - made it RGB
  - A3C cropped images and deleted some parts on labyrinth in Pacman
- Gamma tuning $\gamma \in [0, 1]$

Nagibator style: Skiing

Nagibator style: Pacman



Results

ЛУЧШИЙ РЕЙТИНГ ПО 3-м ИГРАМ

	Skiing	Ms. Pac-man	Centepide	Avrg.Rating
WPLFF	**-3493**	**5512**	7512	0.790
5vision	-4489	2716	17130	0.753
w0rms	-8493	2311	**20328**	0.680
bad_skiers_evolved	-8387	2352	17294	0.636
ModestKoalaGrad	-9013	1715	17130	0.579
sw1sh	-16699	2522	5684	0.246
Generation Gap	-9013	-	-	0.194
miptcap	-9013	-	-	0.194
State of the Art	-10853	**7534**	6297	
	Prioritized DQN	**PGQ**	**Gorila**	

Summary

- Try simple methods first
- Don’t be afraid to use already existing solutions
- But don’t think that these solutions have no mistakes
- Try to dig deeper into model and understand why it performs one way or another

References

- https://github.com/libfun/deephack3 - our solution
- https://universe.openai.com/envs - OpenAI environments
- https://github.com/openai/universe-starter-agent - CPU a3c agent with LSTM
- https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-0-q-learning-with-tables-and-neural-networks-d195264329d0#.4rukzup4m - RL tutorials with TensorFlow
- https://github.com/ppwwyyxx/tensorpack/tree/master/examples/A3C-Gym - GPU a3c without LSTM
- https://arxiv.org/pdf/1602.01783.pdf - Asynchronous Methods for Deep Reinforcement Learning