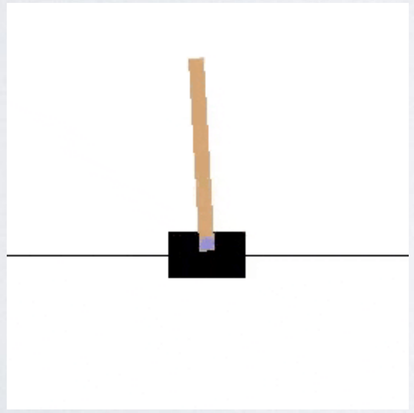
ENGINEERING REINFORCEMENT LEARNING

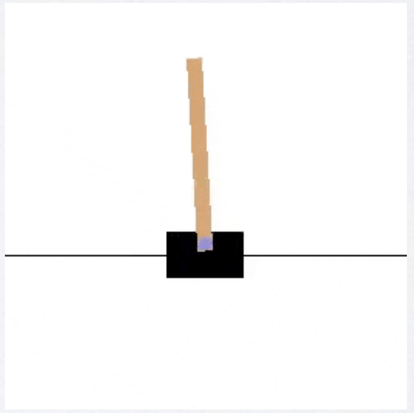
Deepest
2018. 7. 28 Hosting
Sungwon, Lyu
lyusungwon@dm.snu.ac.kr

REINFORCEMENT LEARNING

- **Value function approximation**
  - SARSA, DQN
  - Q(S, A) = Value
- **Policy gradient**
  - REINFORCE, A2C, DDPG, A3C
  - π(S) = Probability

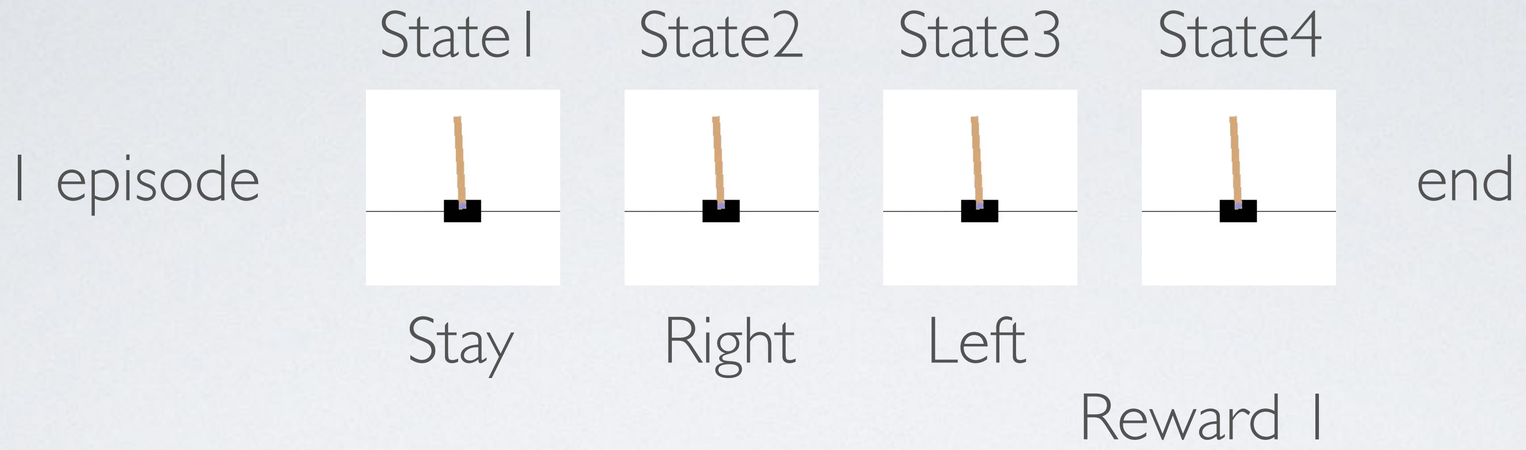


stay: 0.72
left: 0.86
right: 0.21



stay: 36%
left: 45%
right: 19%

REINFORCEMENT LEARNING



- **Temporal Difference**
  - Learn from incomplete episode
  - Save (S2, Right, Reward 0, S3, Left)
    - Q(State1, Stay) = 0 + 0.9 * Q(State2, A)
    - Q(State2, Right) = 0 + 0.9 * Q(State3, A)
    - Q(State3, Left) = 1
- **Monte-Carlo method**
  - Learn from complete episode
  - Save Entire episode
    - Q(State3, Left) = 1
    - Q(State2, Right) = 0.9
    - Q(State1, Stay) = 0.81

REINFORCEMENT LEARNING

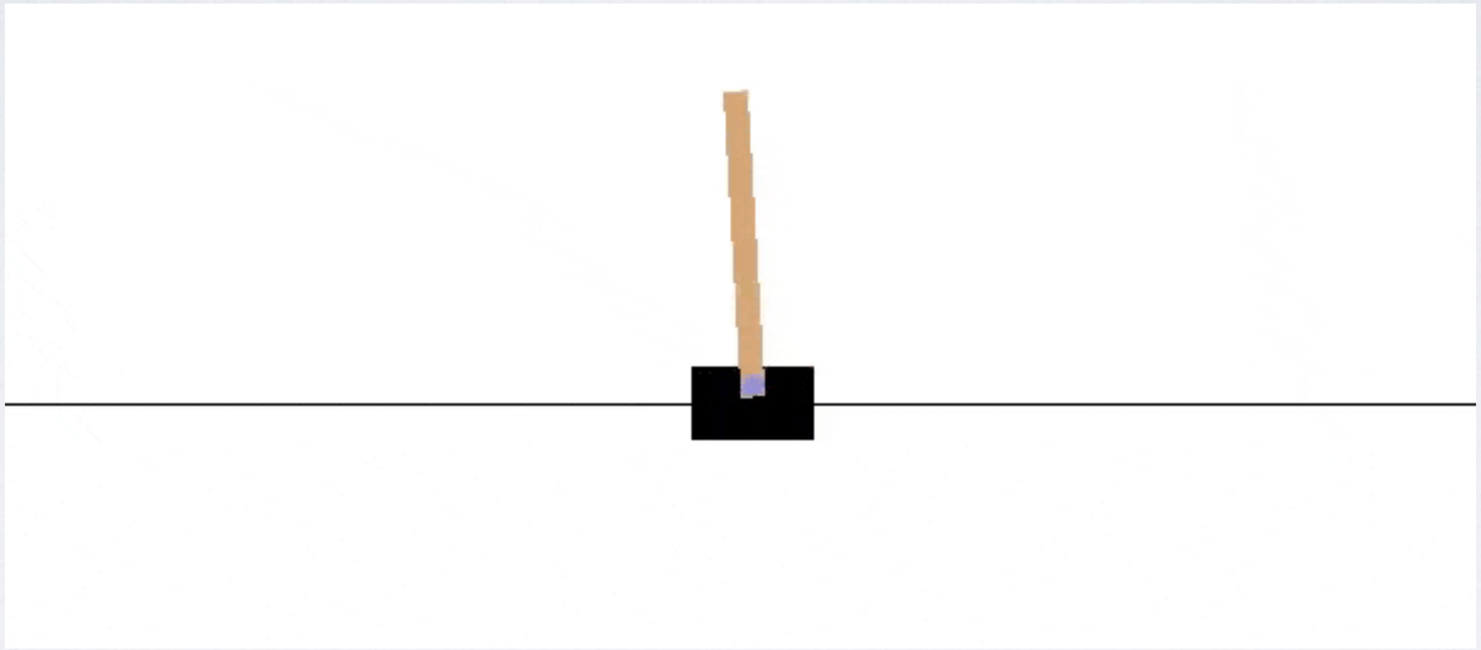
- **DQN**
  - Value function approximation & TD
  - Epsilon decay
  - Q(S,A) as neural network (CNN)
  - Off-policy (target network)
  - Replay Memory

$$Y_t^{DQN} = R_{t+1} + \gamma max_a Q(S_{t+1}, a; \theta_t^-)$$

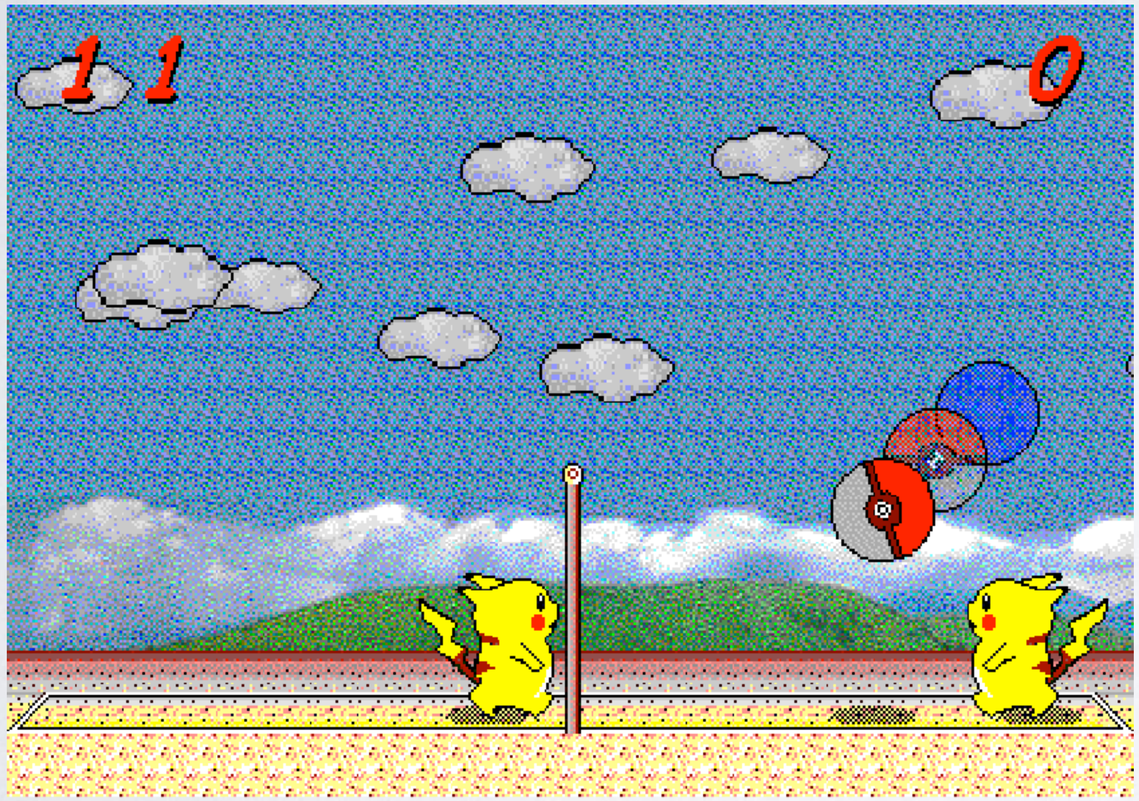
$$J(\theta) = (Q(S_t, a_t; \theta_t) - Y_t^{DQN})^2$$

Source: Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529.

MOTIVATION



PIKACHU - VOLLEYBALL



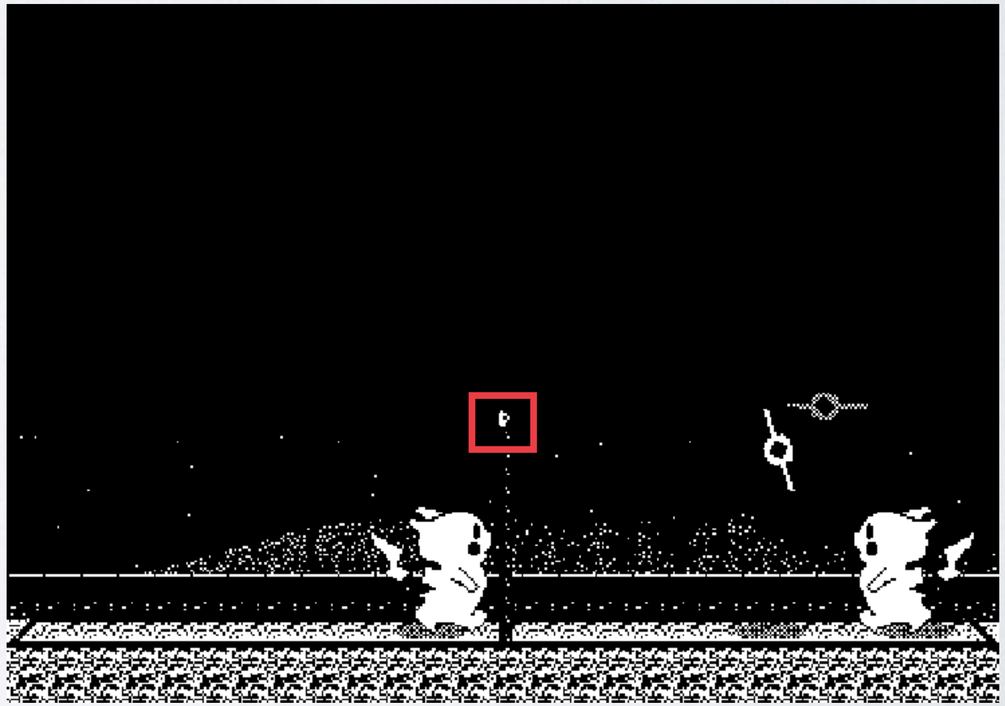
- Window only?
  - No Linux
  - No Server
- No environment toolkit
  - Screen, key-input, timing
  - Start, end, state, action, reward
- Noisy background
  - Clouds
  - Scores

VERSION 1.0

- Window only?
  - Wine

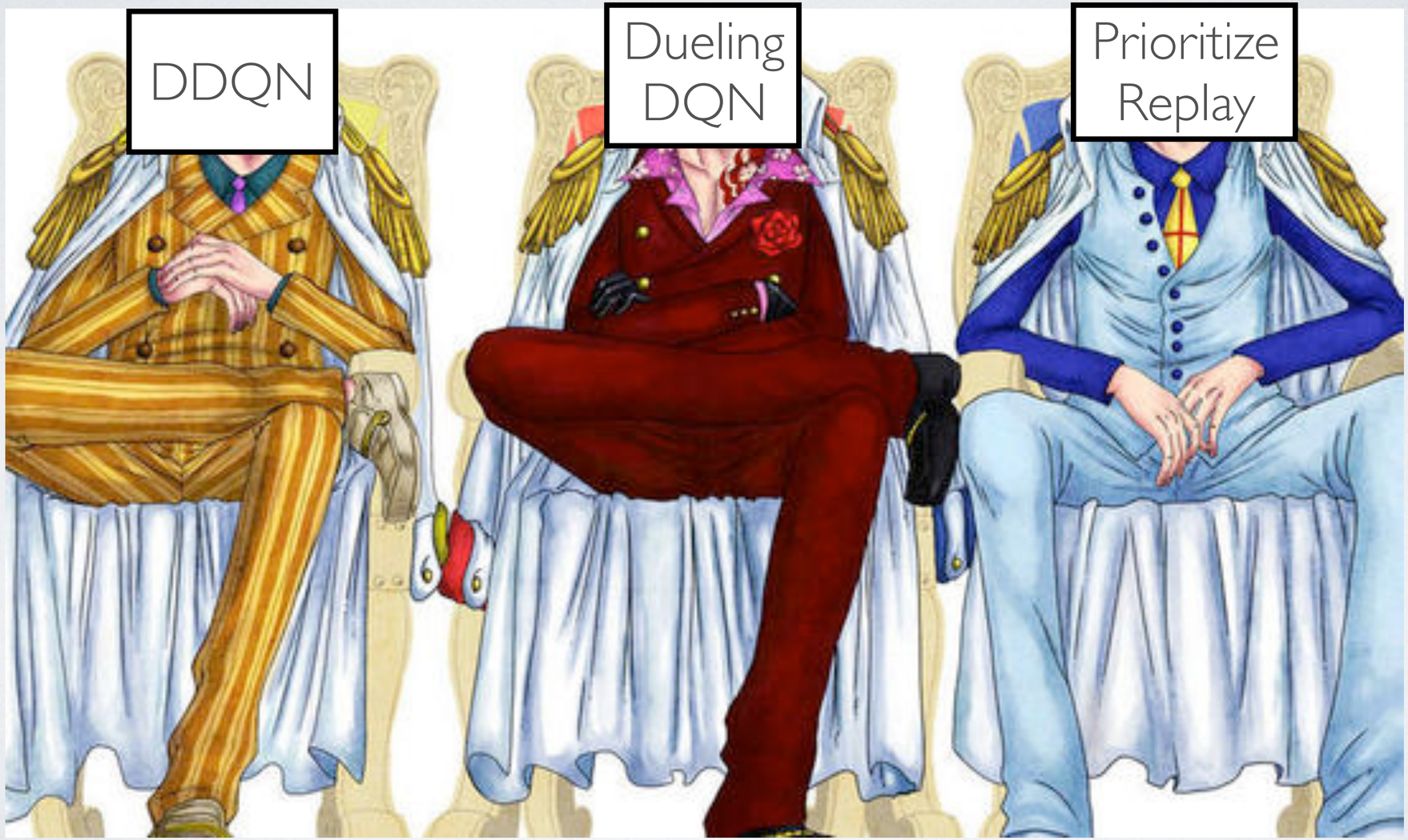
- No environment toolkit
  - Screenshot (mss package)
  - Template matching
  - Key input (pynput package)
  - Feature engineering (Start / end / restart)

- Noisy background
  - => Color filtering



Fail

DQN-EXTENSION



DQN-EXTENSION

- **Double DQN**
  - Prevent over-optimization

$$Y_t^{DQN} = R_{t+1} + \gamma max_a Q(S_{t+1}, a; \theta_t^-)$$

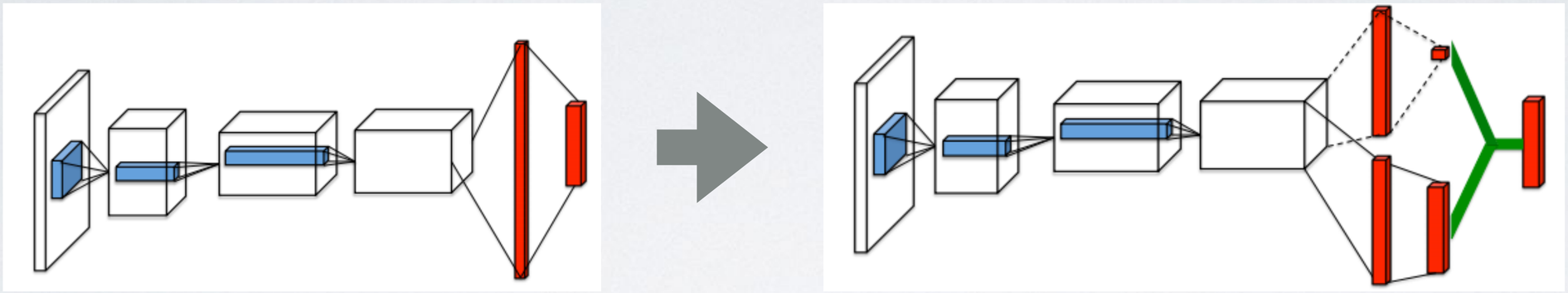
$$Y_t^{DoubleQ} = R_{t+1} + \gamma Q(S_{t+1}, argmax_a Q(S_{t+1}, a; \theta_t); \theta_t')$$

  - Implementation: 1min cut
  - => DDQN

Source: Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning." *AAAI*. Vol. 2. 2016

DQN-EXTENSION

- **Dueling DQN**
  - Action is not important for some state



$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + (A(s,a;\theta,\alpha) - \frac{1}{\|\mathcal{A}\|}\Sigma_{a'}A(s,a';\theta,\alpha))$$

  - Implementation: 5min cut
  - => Dueling DQN

Source: Wang, Ziyu, et al. "Dueling network architectures for deep reinforcement learning." *arXiv preprint arXiv:1511.06581* (2015).

DQN-EXTENSION

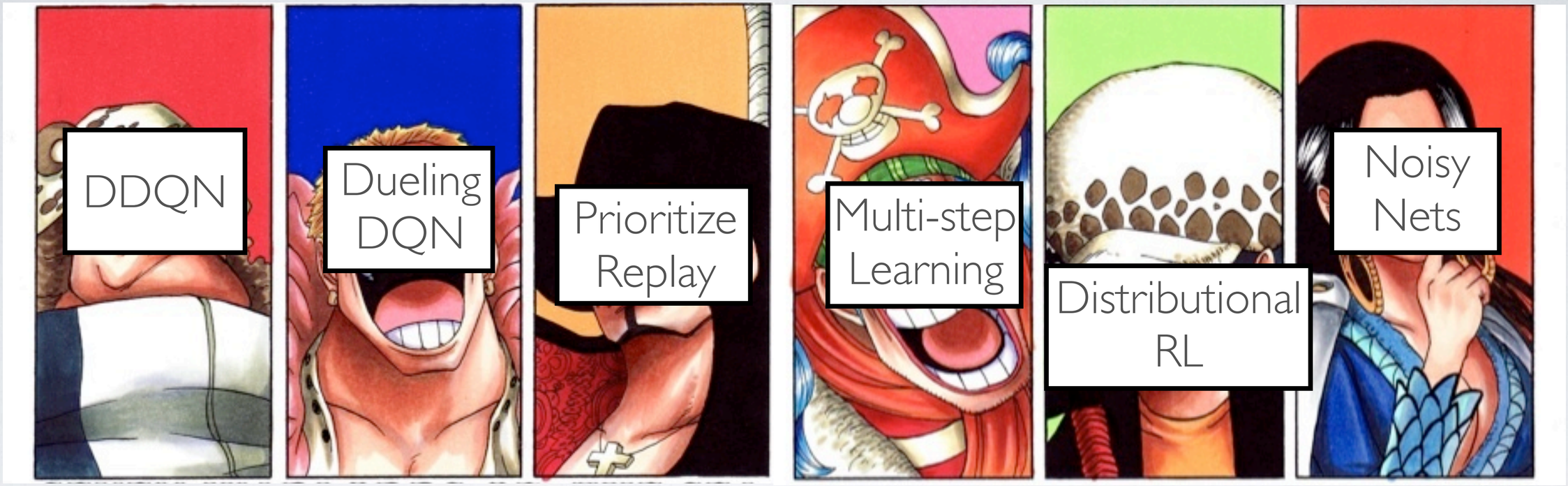
- **Prioritize replay**
  - Not all experience are equal
  - Weighted Sample according to ‘Surprise’

$$\delta_j = R_j + \gamma_j Q_{target}(S_j, argmax_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$$

  - Implementation: 30min cut
  - => Prioritized DQN

Source: Schaul, Tom, et al. "Prioritized experience replay." *arXiv preprint arXiv:1511.05952* (2015).

MORE DQN-EXTENSION

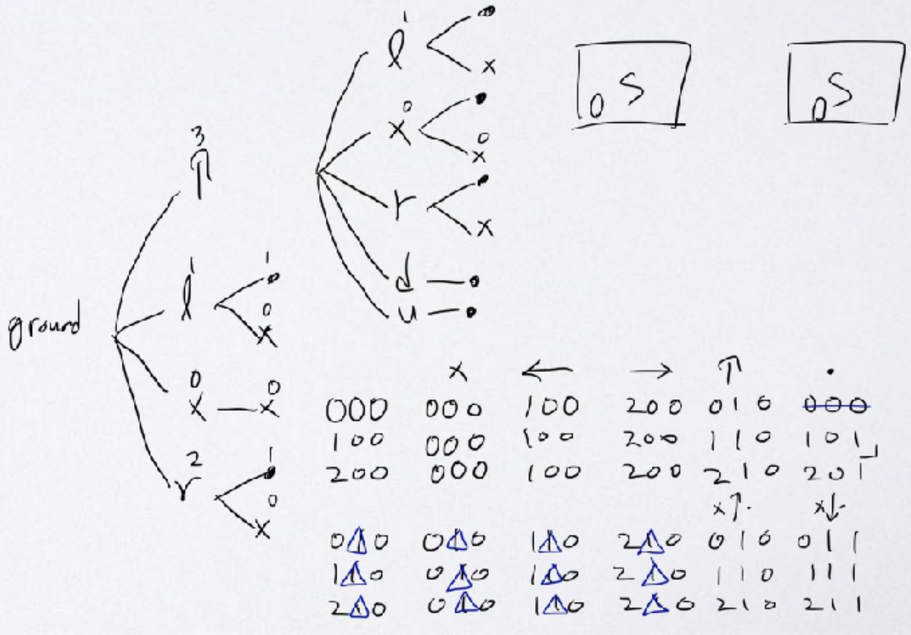


- => Rainbow

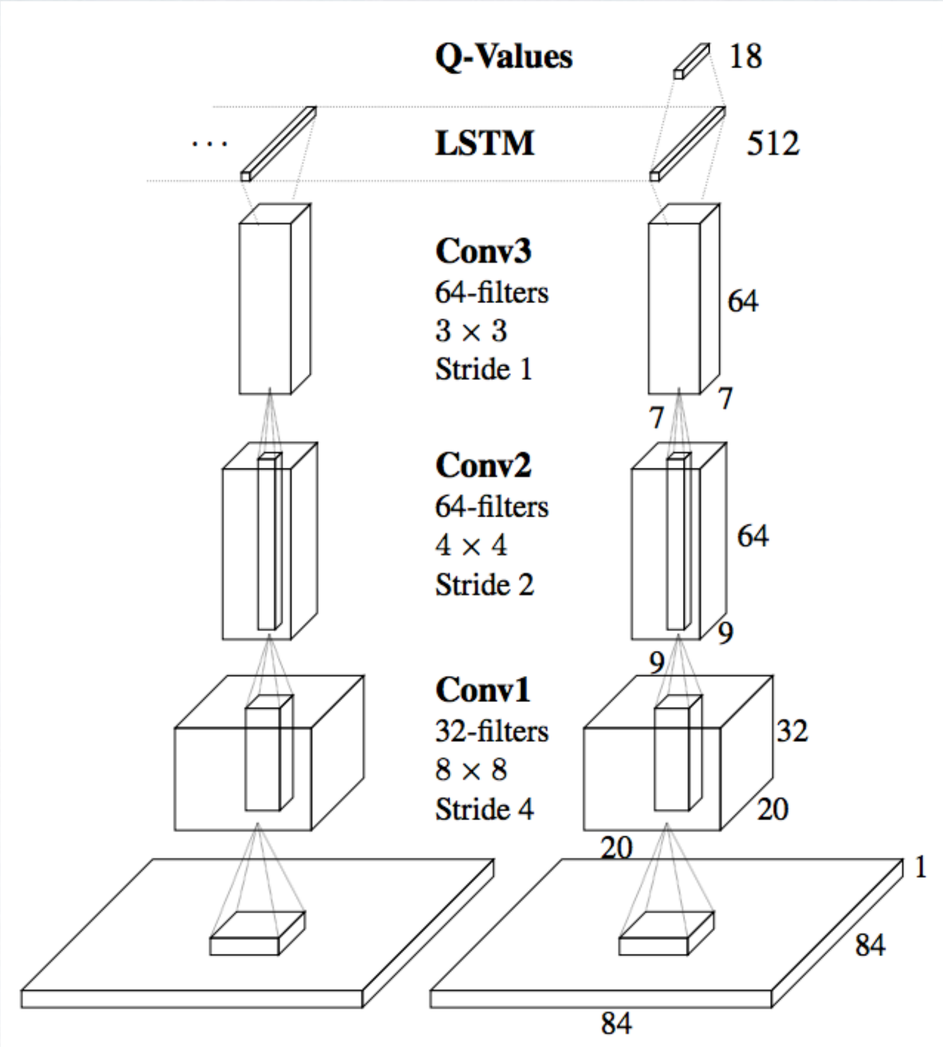
Source:
Bellemare, Marc G., Will Dabney, and Rémi Munos. "A distributional perspective on reinforcement learning." *arXiv preprint arXiv:1707.06887* (2017).
Fortunato, Meire, et al. "Noisy networks for exploration." *arXiv preprint arXiv:1706.10295* (2017).
Hessel, Matteo, et al. "Rainbow: Combining improvements in deep reinforcement learning." *arXiv preprint arXiv:1710.02298*(2017).

VERSION 2.0

- Pytorch
- Prioritized Dueling DDQN
- Model
  - c16(3,1)-c32(3,1)-c64(3,1)-c64(3,1)-c64(3,1)-c64(3,1) => c16(8,4)-c32(4,2)-c32(3,1)
- State: 4 * 140 * 215 => 4 * 84 * 129
- Action: 9 => 6
- Frame: 0.125 -> 0.2



DRQN



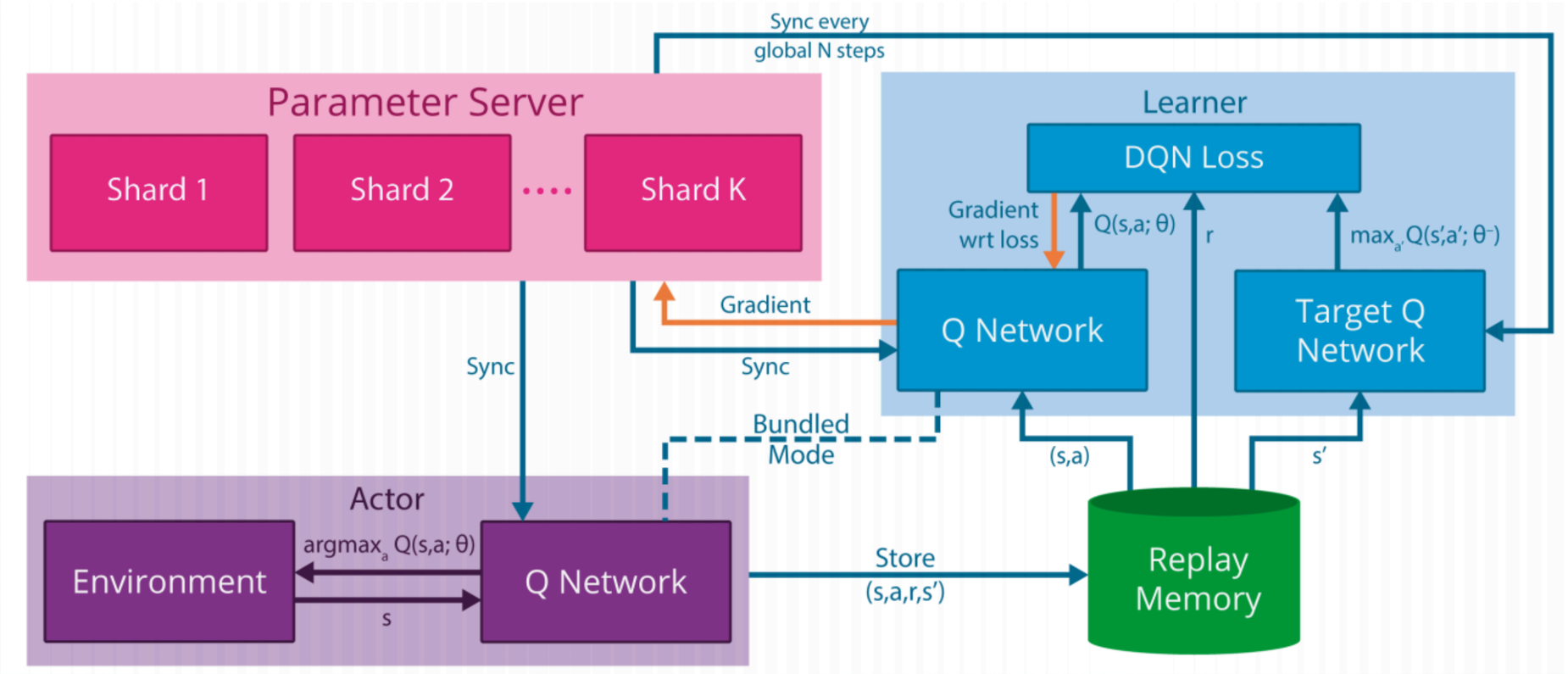
Source: Hausknecht, Matthew, and Peter Stone. "Deep recurrent q-learning for partially observable mdps." *CoRR, abs/1507.06527*7.1 (2015).

XVFB

- XVFN(X Virtual Frame Buffer)
  - Display server: operate graphics on memory
  - How?
    - sudo apt-get install xvfb
    - xvfb :99 -ac -screen 0 1280x1024x24 > /dev/null &
    - echo "export DISPLAY=:99" >> ~/.bashrc
    - DISPLAY=:99 python actor.py
- xdotool
  - sudo apt-get install xdotool
  - DISPLAY=:99 xdotool search --name “PIKA"
  - DISPLAY=:99 xdotool key --window 12582913 KP_Enter

GORILA

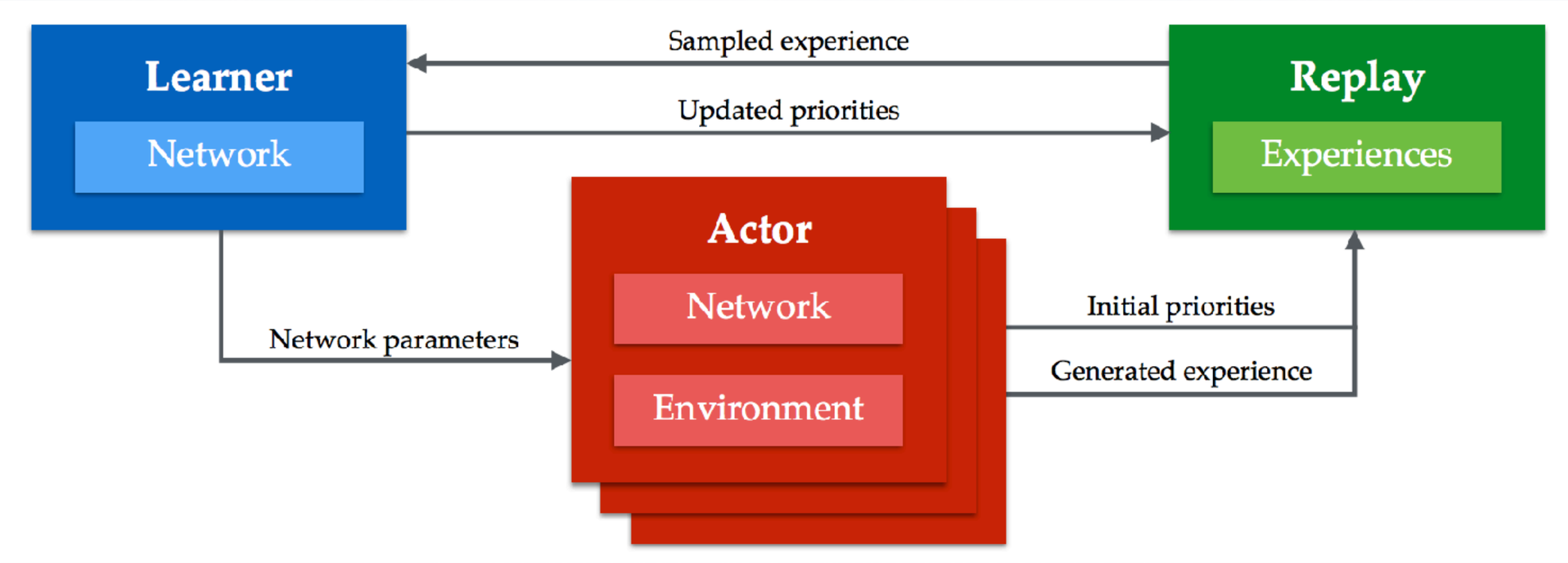
- Gorila (General Reinforcement Learning Architecture)



Source: Nair, Arun, et al. "Massively parallel methods for deep reinforcement learning." *arXiv preprint arXiv:1507.04296* (2015).

APE-X

- Ape-X DQN / DPG



Source: Horgan, Dan, et al. "Distributed prioritized experience replay." *arXiv preprint arXiv:1803.00933* (2018).

STATE-OF-THE-ART BENCHMARK

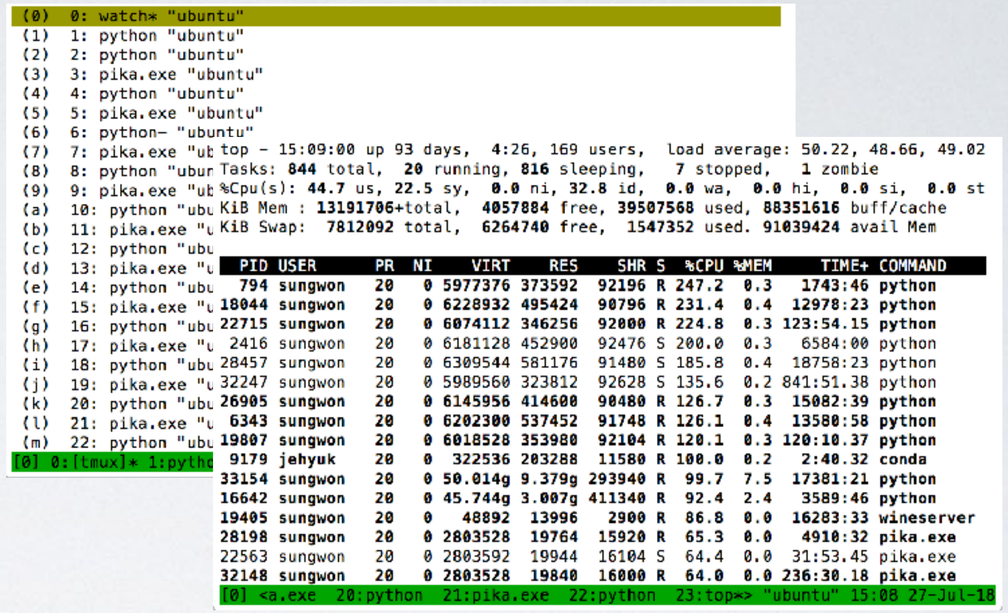
Algorithm	Frames	Bowling		Venture	
		%	raw	%	raw
RUDDER	12M	**62.10**	**108.55**	96.55	1,147
rainbow	200M	5.01	30	0.46	5.5
Prioritized DDQN	200M	28.71	62.6	72.67	863
Noisy DQN	200M	39.39	77.3	0	0
Dueling DDQN	200M	30.81	65.5	41.85	497
DQN	200M	19.84	50.4	13.73	163
Distributional DQN	200M	37.06	74.1	93.22	1,107
DDQN	200M	32.7	68.1	8.25	98
Ape-X DQN	22,800M	-17.6	4	**152.67**	**1,813**
Random	–	0	23.1	0	0
Human	–	100	160.7	100	1,187

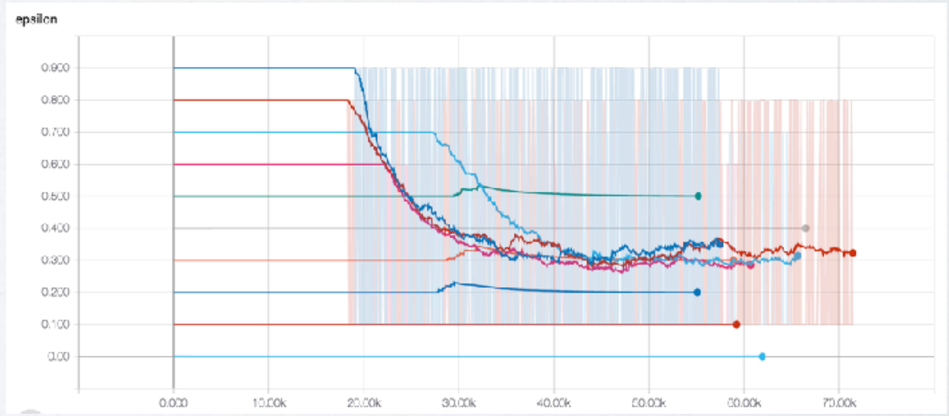
Table 1: Results of RUDDER and other methods when learning the Atari games Bowling and Venture. Normalized human-percentage and raw scores over 200 testing-games with no-op starting condition: A3C scores are not reported, as not available for no-op starting condition. Scores for other methods were taken from previous publications [8, 39]. The RUDDER model is chosen based only on its training loss over 12M frames.

Source: Arjona-Medina, Jose A., et al. "RUDDER: Return Decomposition for Delayed Rewards." *arXiv preprint arXiv:1806.07857* (2018).

VERSION 3.0

- Multi-thread => Multi-process
  - Communicate through file
  - 10 actors and 1 learner
  - Different epsilon per actor
- Run on server
  - Learner on gpu / actors on cpu
- Memory allocation
  - 40000 experience replay for learner
  - 4000 per actor

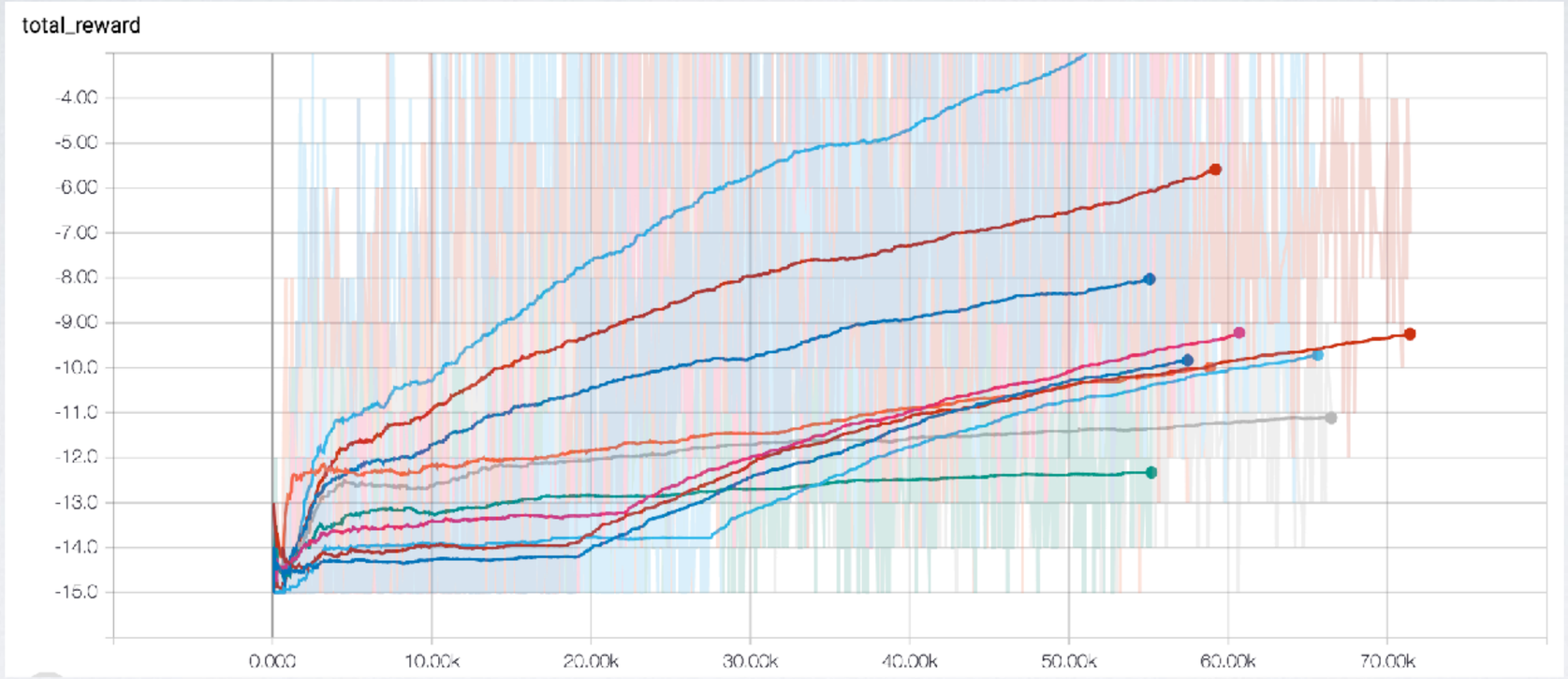




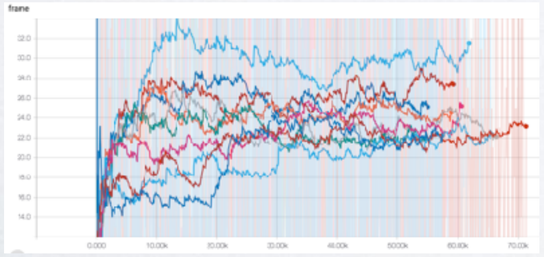
Hsu, Wei-Ning, Yu Zhang, and James Glass. "Unsupervised Learning of Disentangled and Interpretable Representations from Sequential Data." *Advances in neural information processing systems*. 2017.

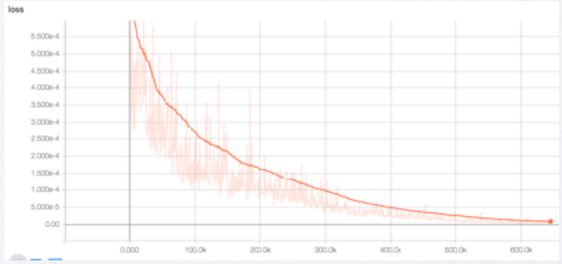
RESULT 1

- Total reward (15 - Score difference)



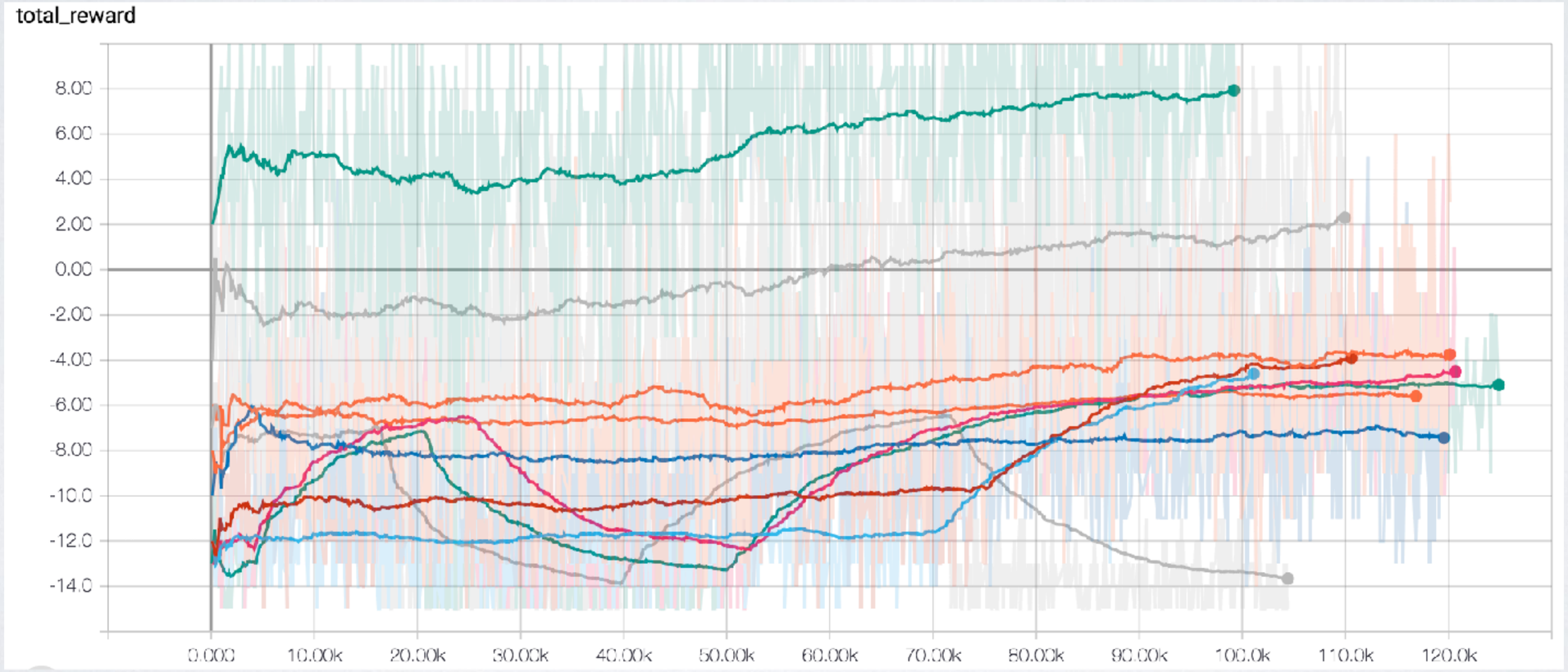
- Frame, loss





RESULT 2

- Total reward (15 - Score difference)



DEMO

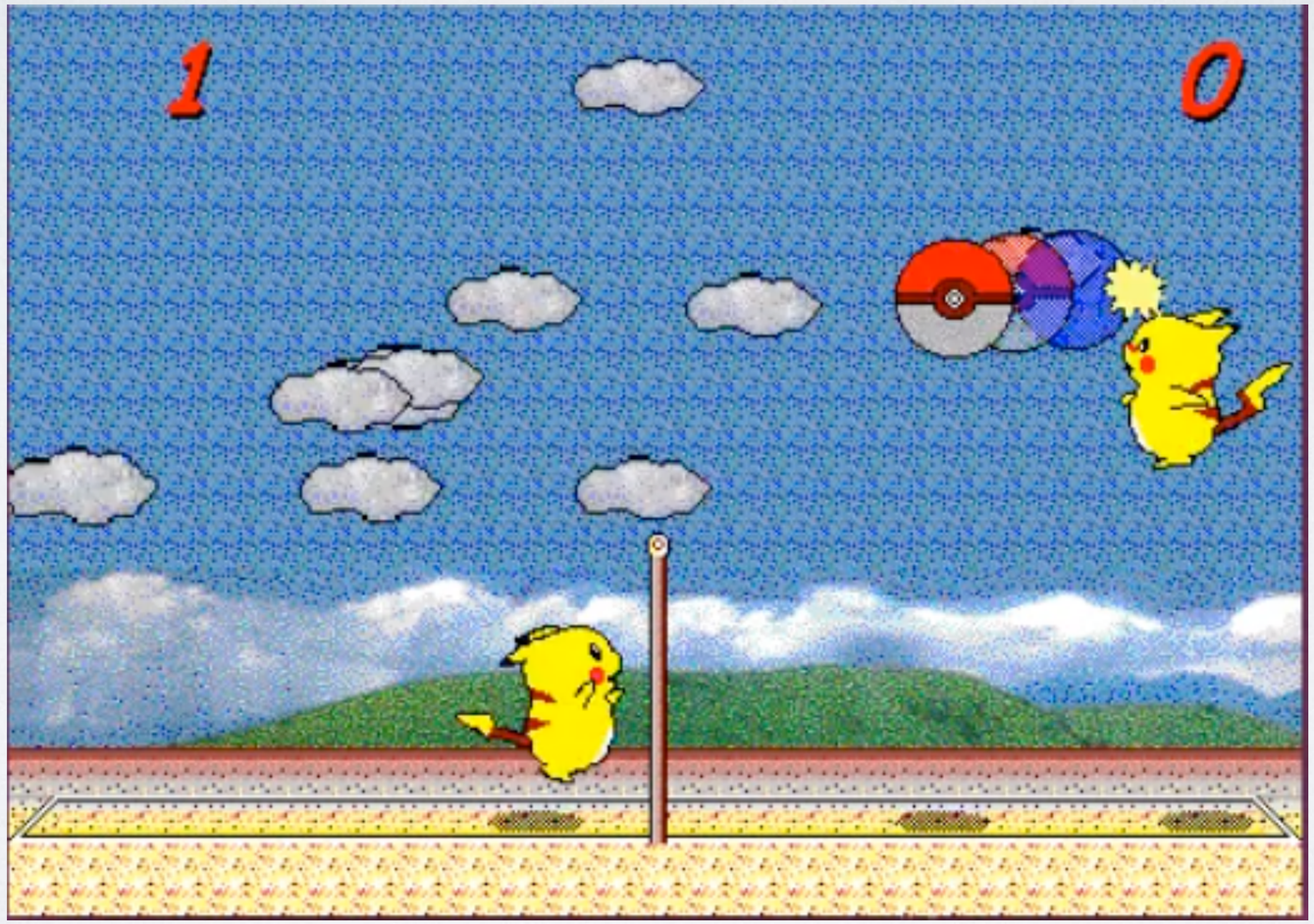
Hsu, Wei-Ning, Yu Zhang, and James Glass. "Unsupervised Learning of Disentangled and Interpretable Representations from Sequential Data." *Advances in neural information processing systems*. 2017.

FEATURES

- Infinite loop

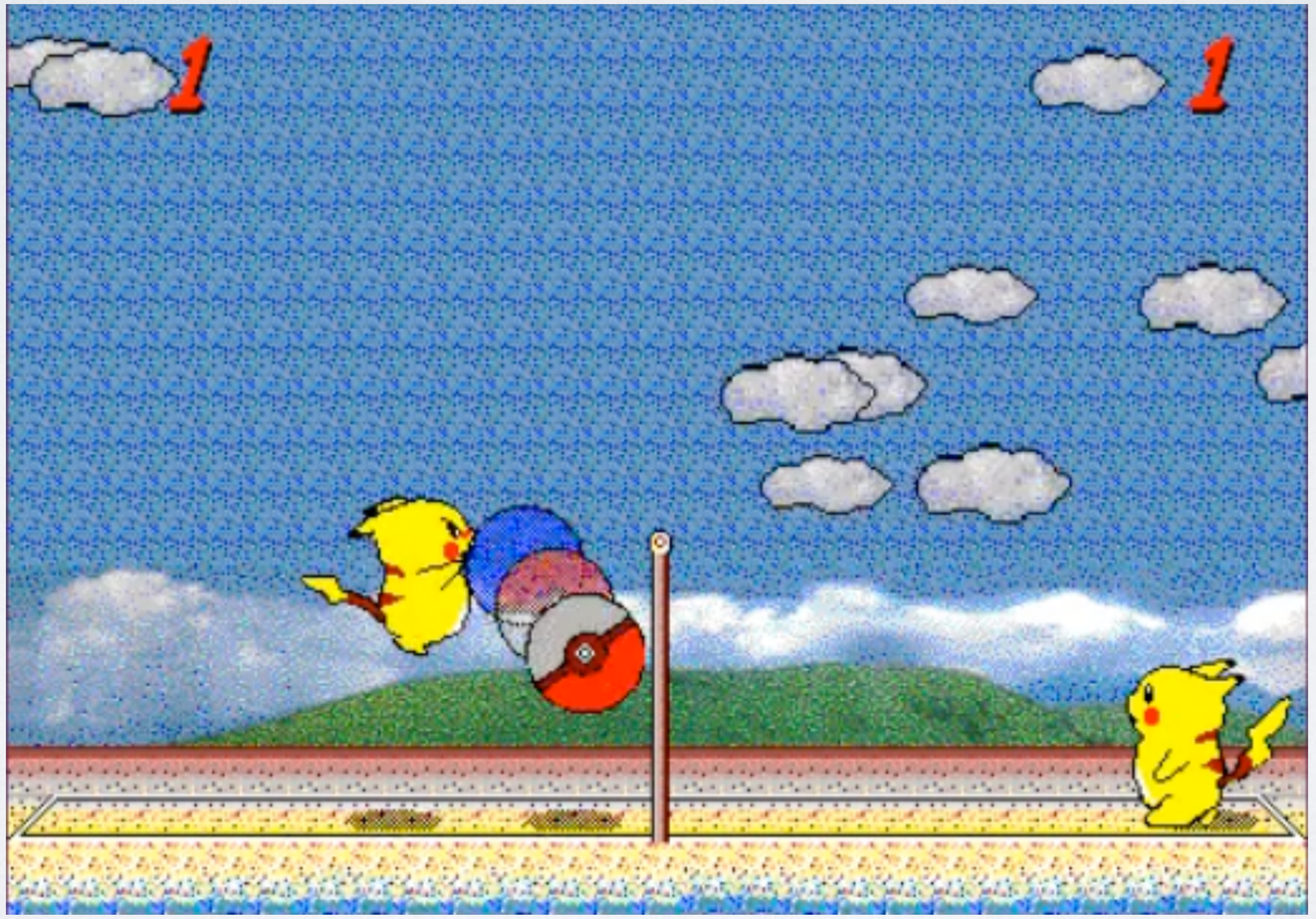
FEATURES

- Mode



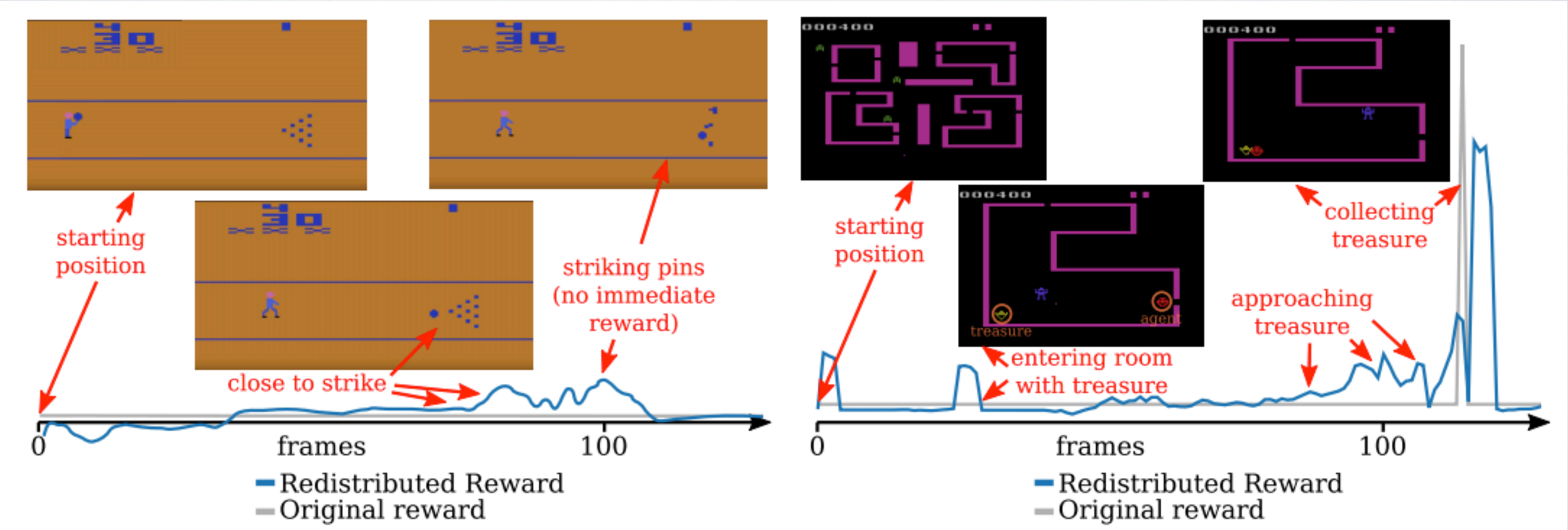
FEATURES

- Suicide



RUDDER

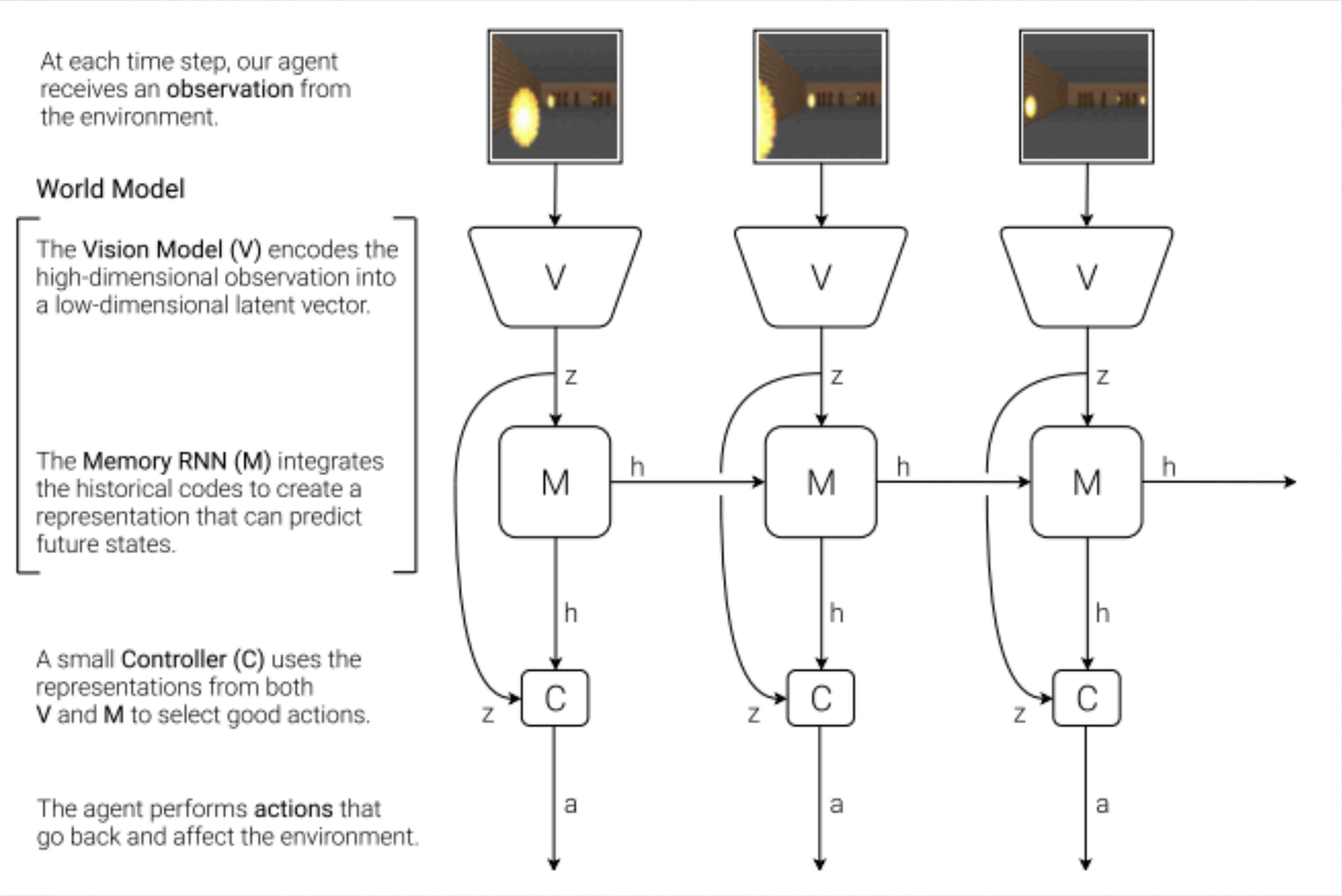
- RUDDER: Return Decomposition for Delayed Rewards



Source: Arjona-Medina, Jose A., et al. "RUDDER: Return Decomposition for Delayed Rewards." *arXiv preprint arXiv:1806.07857* (2018).

WORLD MODEL

- World Model



Source: Ha, David, and Jürgen Schmidhuber. "World Models." *arXiv preprint arXiv:1803.10122* (2018).

CONCLUSION

- Reinforcement learning can be very inefficient
  - Do everything before training
  - Keep the states / action size small as possible
- Parallel training is necessary
  - Model size matters
  - Hardware matters
- Lots of drawbacks in reinforcement learning
  - Opportunity?
- Why?

SUMMARY

- Reinforcement learning basics
  - Value approximation function
  - Policy gradient
  - Temporal difference
  - Monte Carlo method
  - DQN
- DQN extensions
  - Double DQN
  - Dueling DQN
  - Prioritized experience replay
- Parallel training
  - Gorila
  - Ape-X
- Recent models
  - RUDDER
  - World model
- Engineering Tips