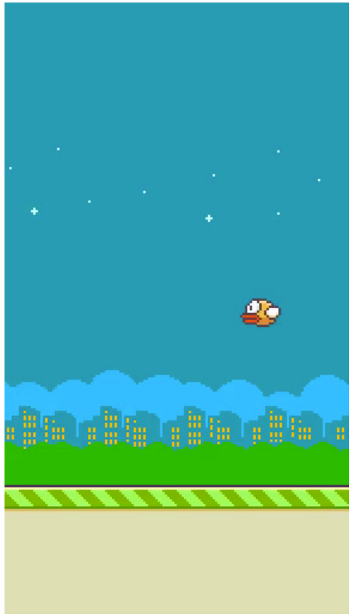
# Deep Reinforcement Learning with Double Q-learning

## Flappy Bird Hacking using Deep Reinforcement Learning

Shubham | Naman | Ziyu | Jianqiu | Zhenye


The best agent

## Introduction

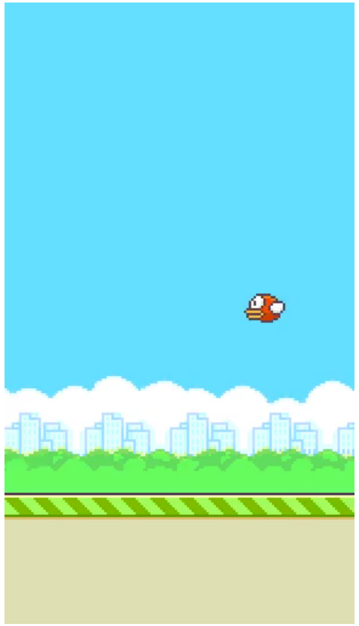
*Paper: Deep Reinforcement Learning with Double Q-learning*

**About this project**
We implement DDQN on PLE FlappyBird environment in PyTorch

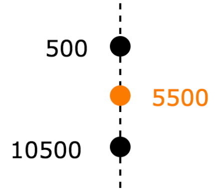
**Why DDQN?**
DDQN is proposed to solve the overestimation issue of Deep Q Learning (DQN).

Apply separate target network to choose action,
reducing the correlation of action selection and value evaluation.

*https://github.com/drl-dql/DQN-Flappy-Bird*



Video for
#Episodes



# Implementation: Environment

## Valid Actions

Up causes the bird to accelerate upwards.

## Terminal states (game_over)

If the bird makes contact with the ground, pipes or goes above the top of the screen the game is over.

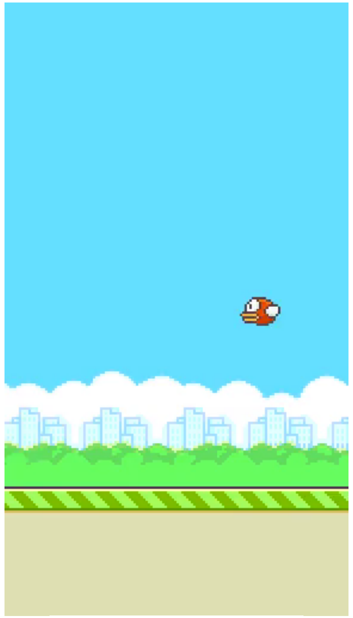
## Rewards

For each pipe it passes through it gains a positive reward of +1. For each frame the live bird receives reward of +0.1. Each time a terminal state is reached it receives a negative reward of -1.

# Experimental Setup: Hyperparameters

| | | | |
|---|---|---|---|
| Optimizer | Learning Rate | Discount Factor | Replay Buffer Size |
| Total Episodes | Target Frequency | Epsilon | Batch Size |
| Initial Observed Episodes | Epsilon Discount | Screen Width | Screen Height |

# Experimental Results: Best



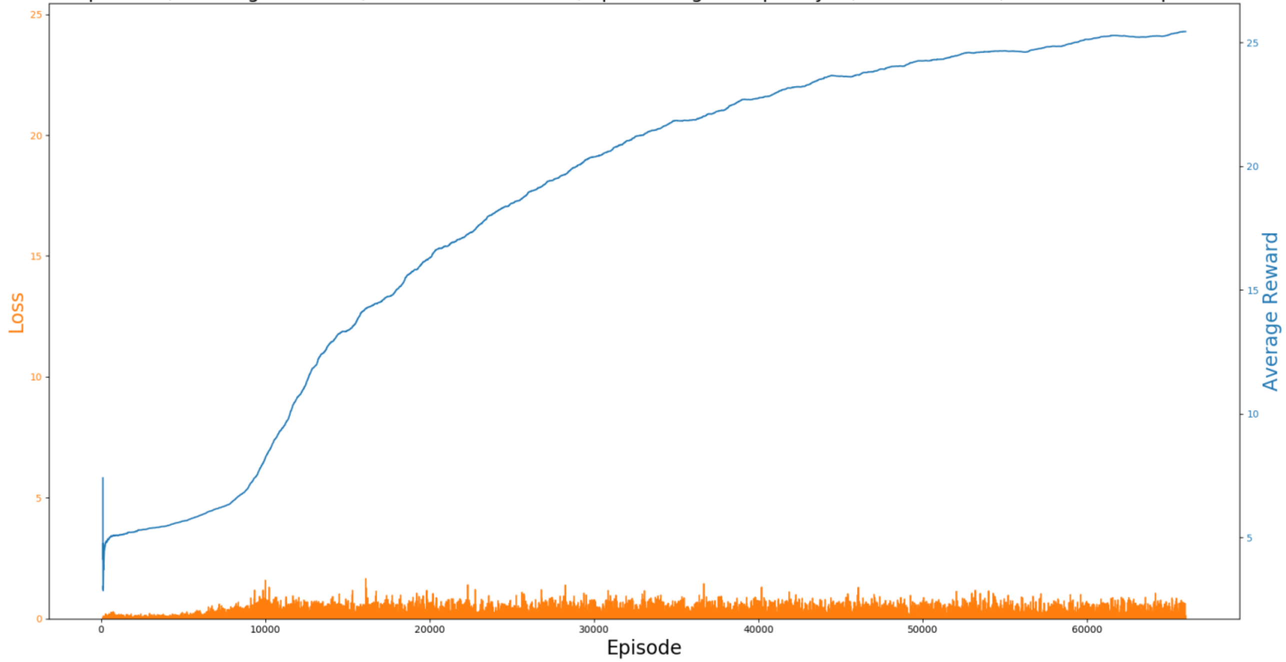
Video for #Episodes

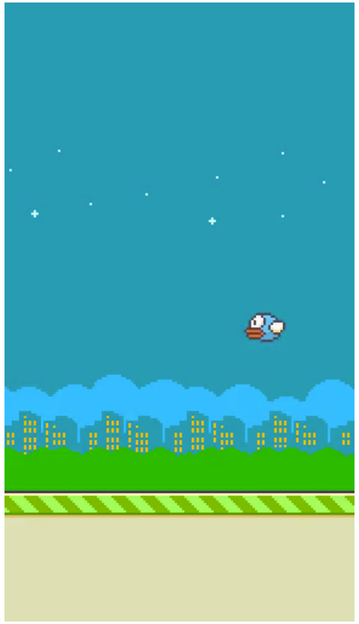
500

5500

10500

Adam optimizer, learning rate: 1e-4, Discount factor: 0.99, update target frequency: 3, batch size: 16, intial observe episode: 120





Video for
#Episodes

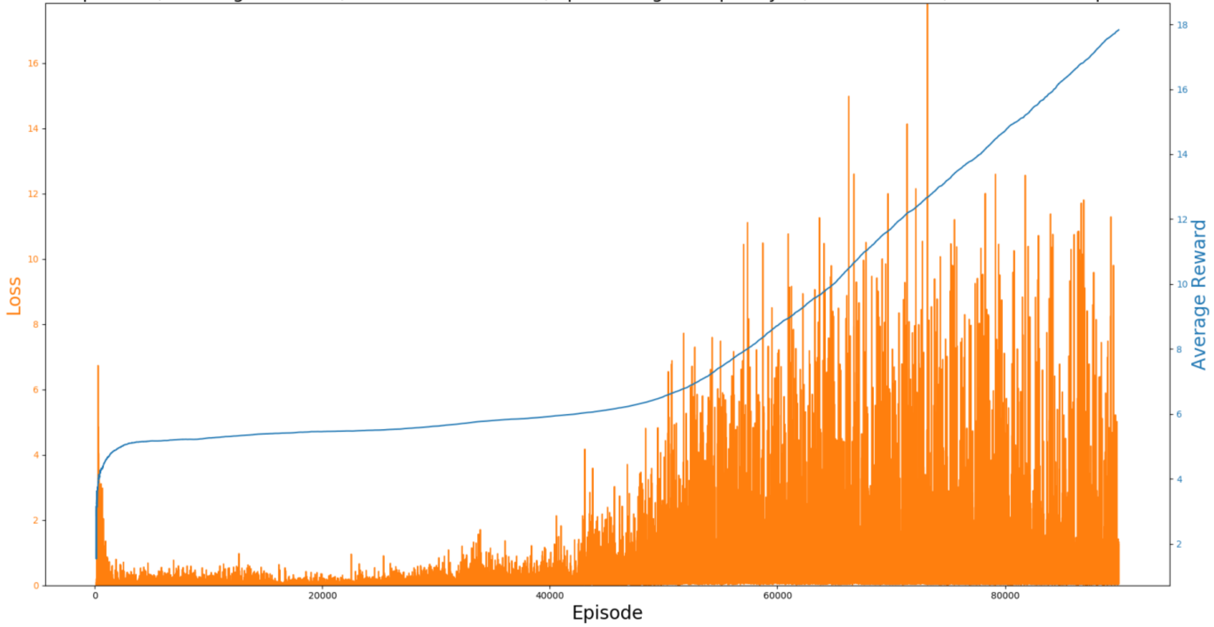
15500

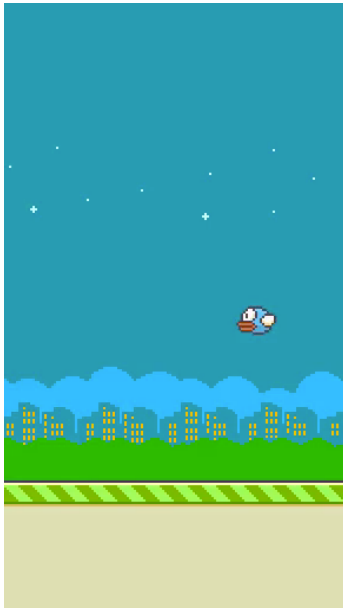
20500

25500

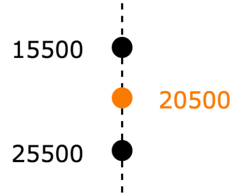
# Experimental Results: Worst

Adam optimizer, learning rate: 1e-4, Discount factor: 0.99, update target frequency: 3, batch size: 16, intial observe episode: 120

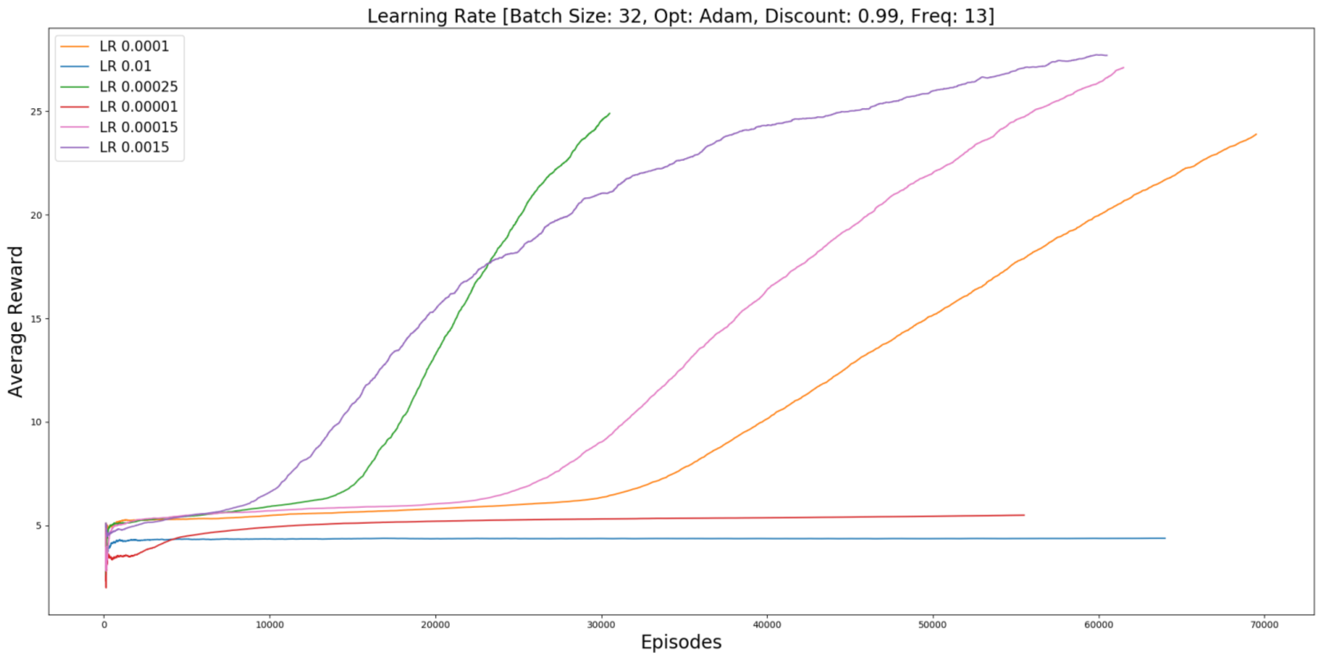


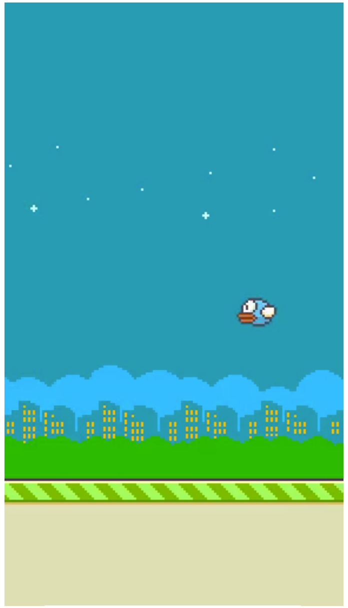


Video for
#Episodes



# Learning Rate Analysis





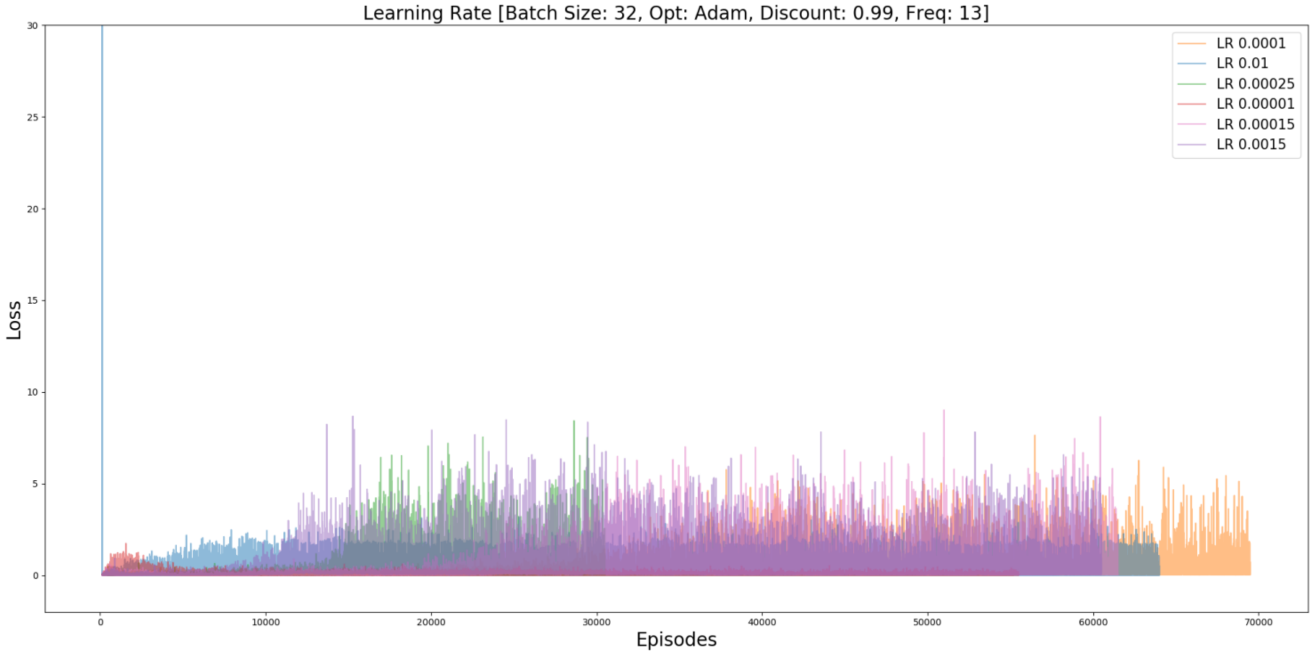
Video for
#Episodes

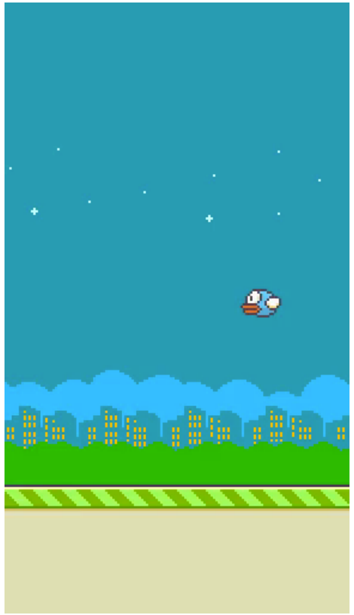
15500

20500

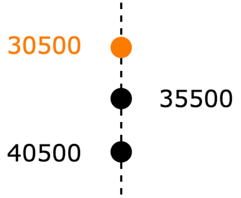
25500

## Learning Rate Analysis

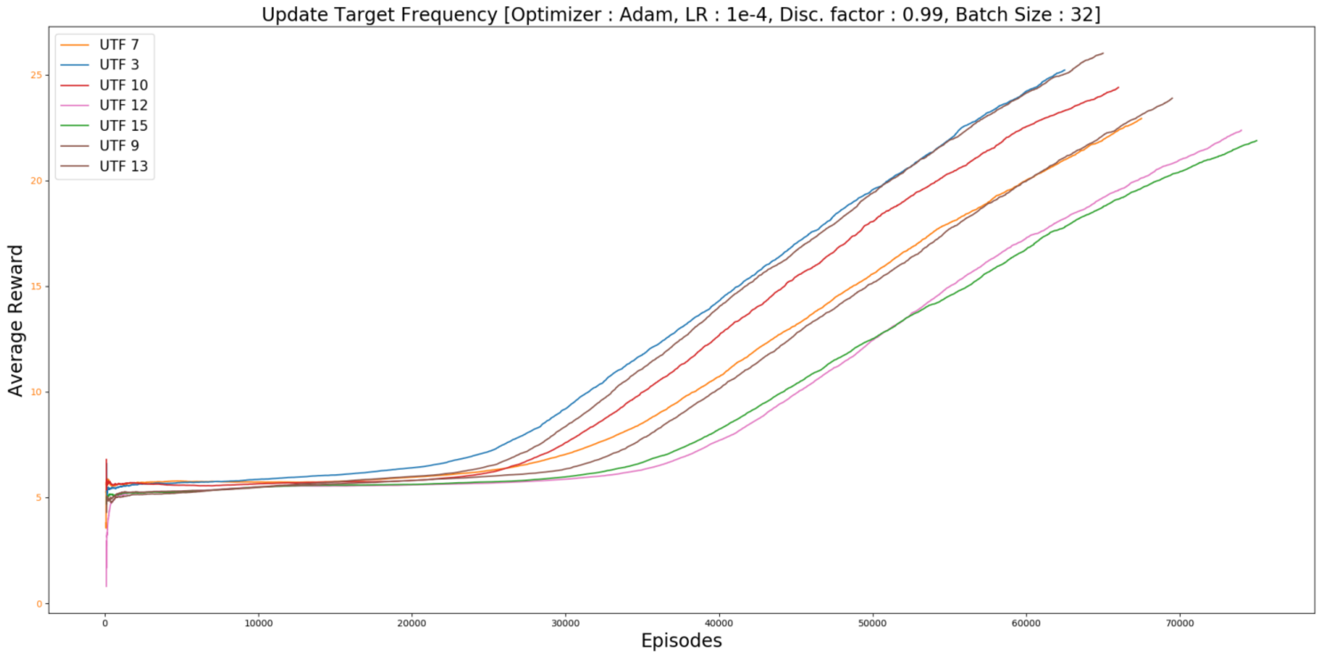




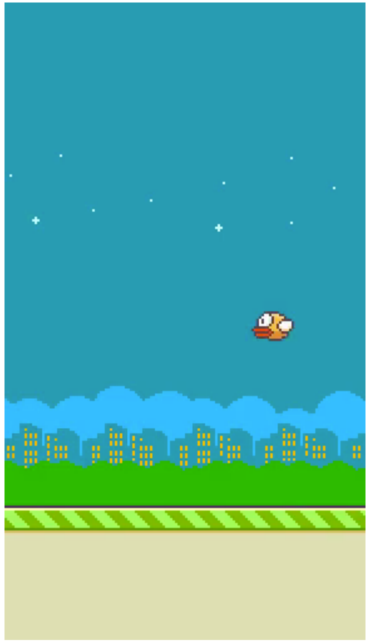
Video for
#Episodes



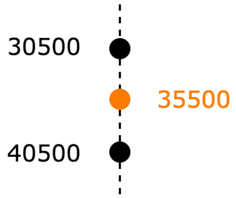
# Update Target Frequency Analysis

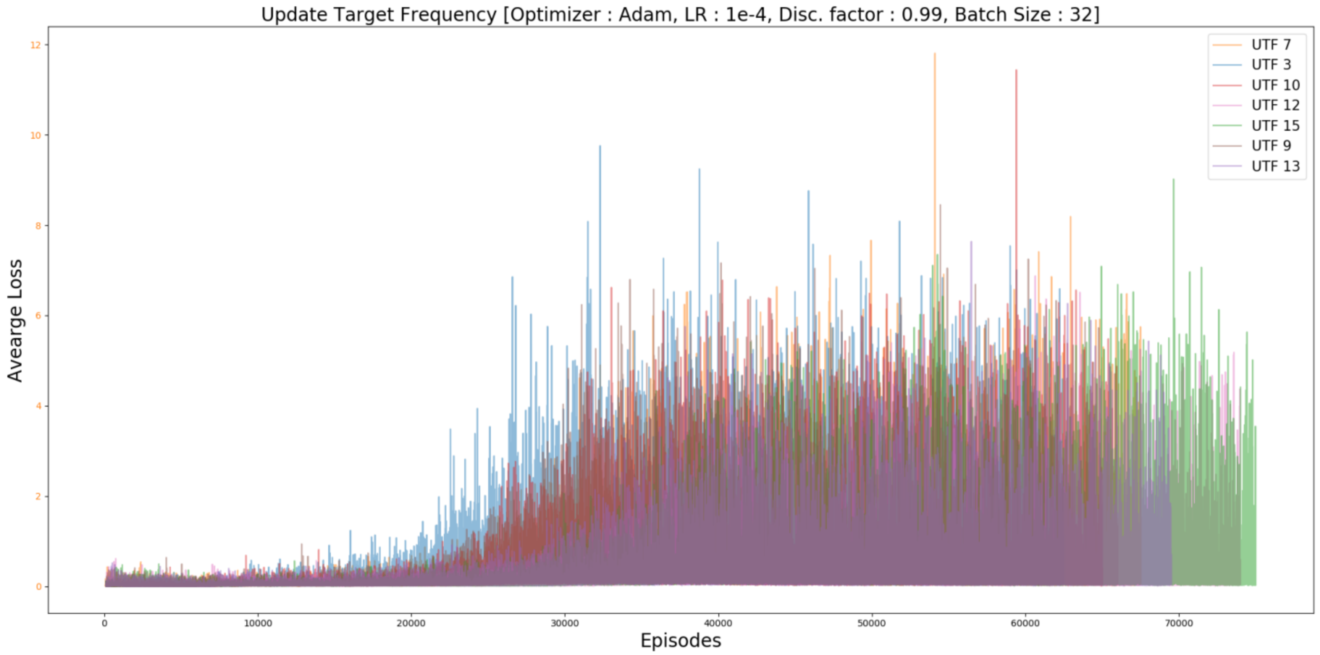


# Update Target Frequency Analysis

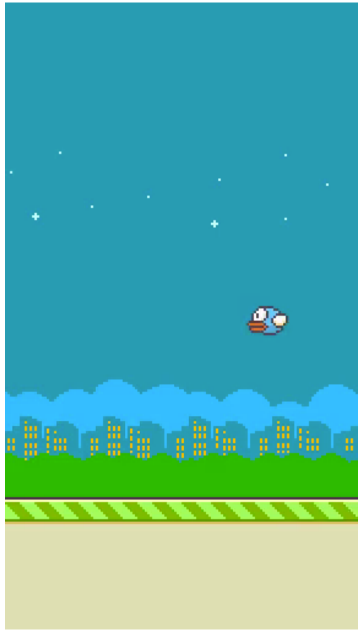


Video for
#Episodes





# Batch Size Analysis

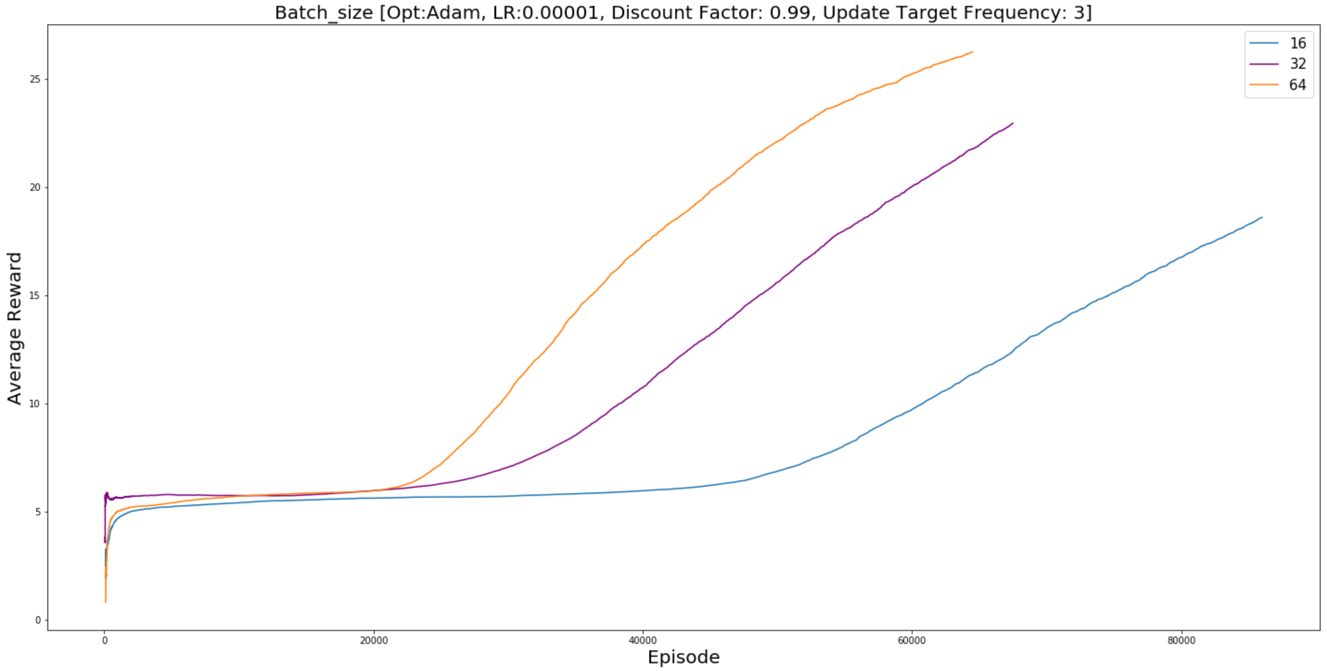


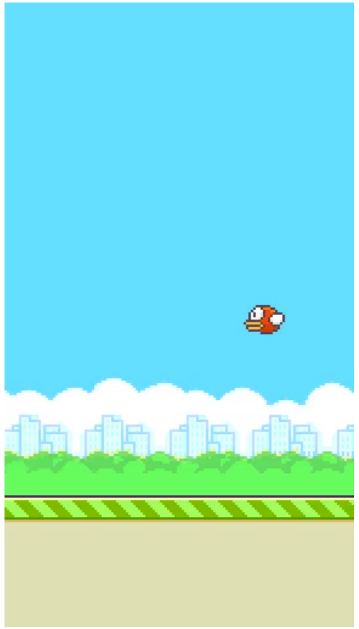
Video for
#Episodes

30500

35500

40500





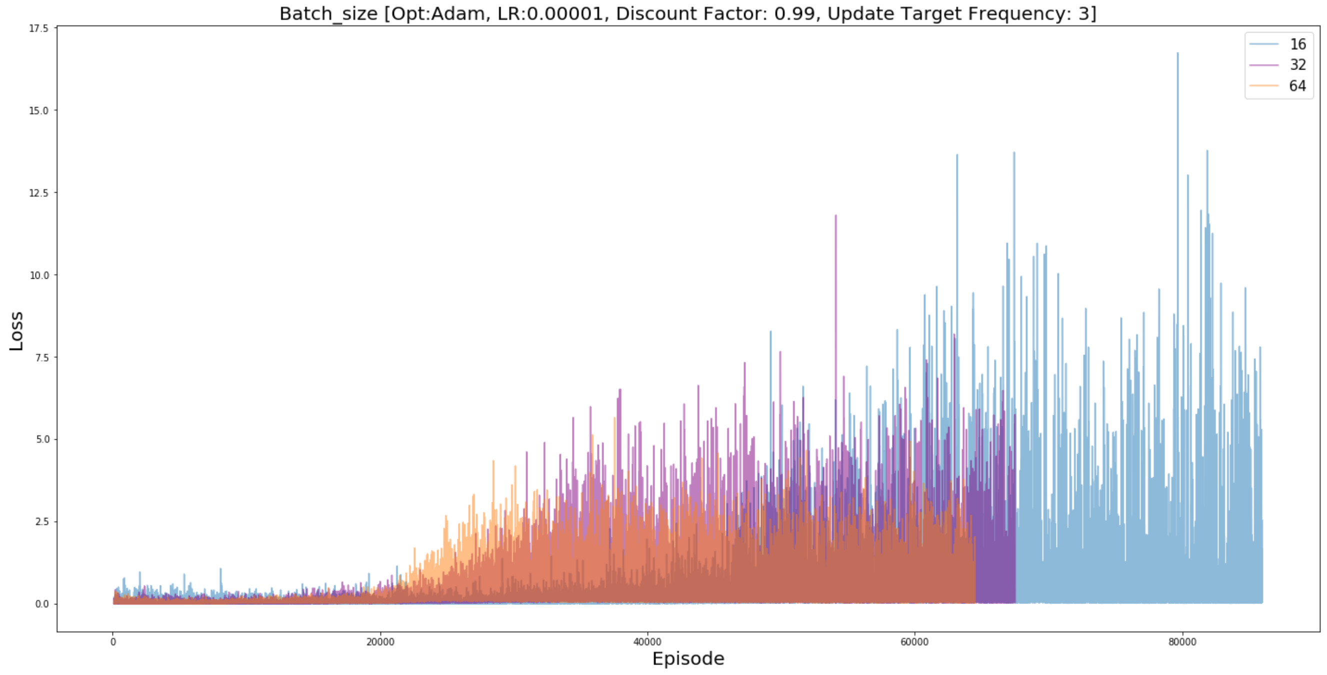
Video for
#Episodes

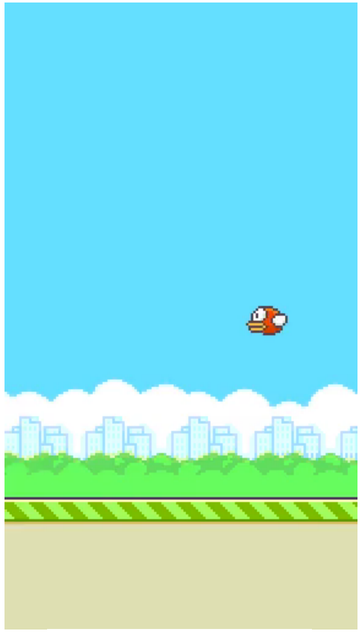
45500

50500

55500

## Batch Size Analysis





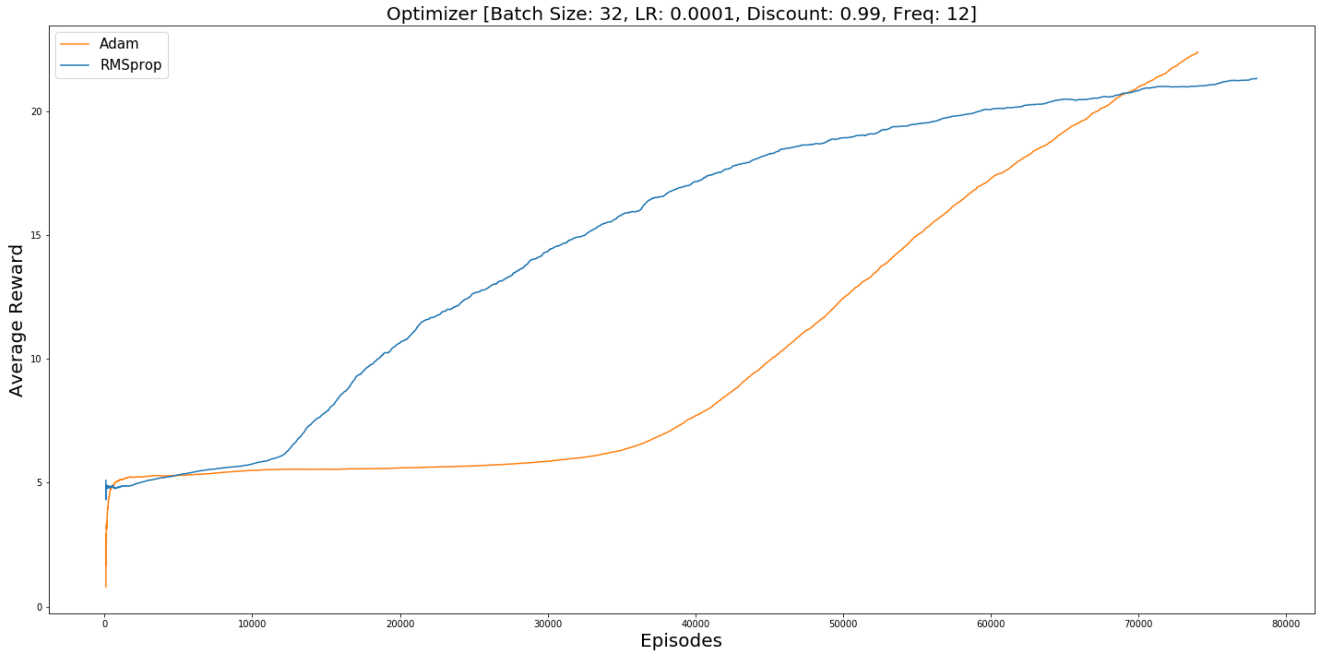
Video for
#Episodes

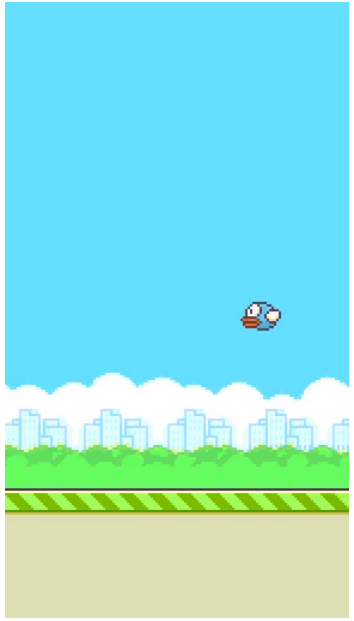
45500 ●

● 50500

55500 ●

# Optimizer Analysis





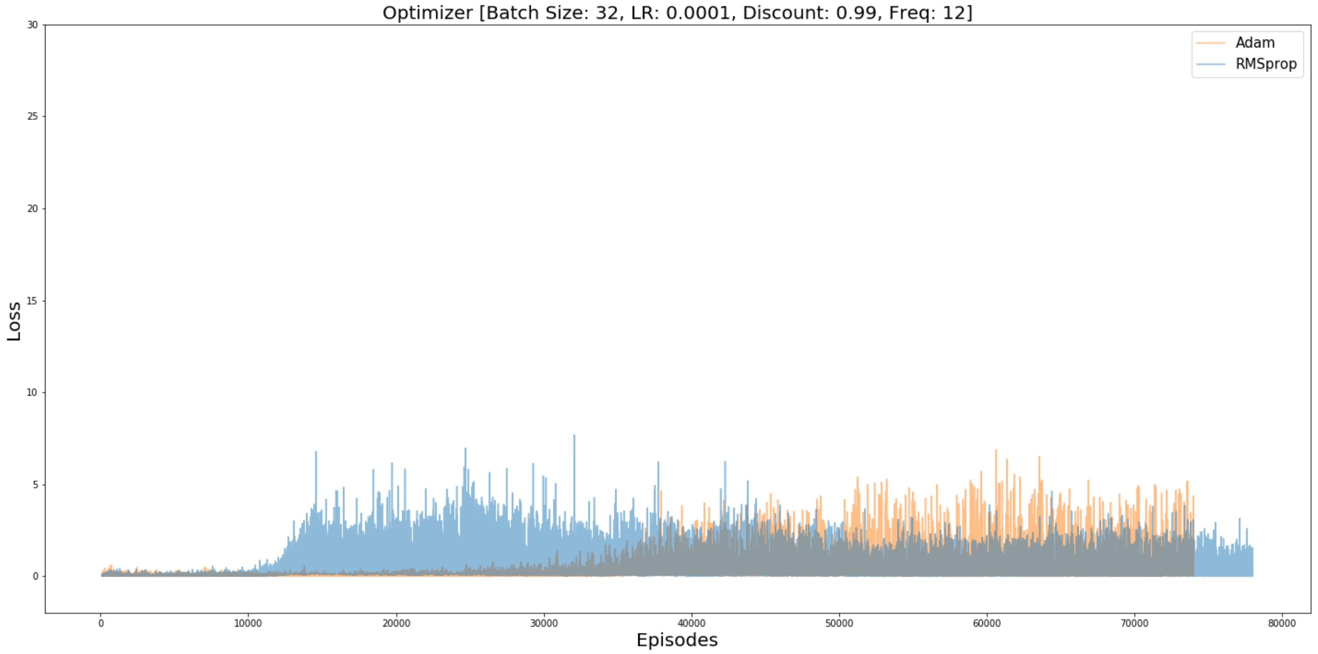
Video for
#Episodes

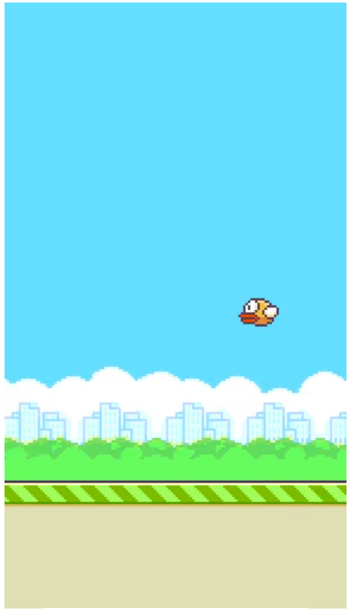
45500

50500

55500

# Optimizer Analysis





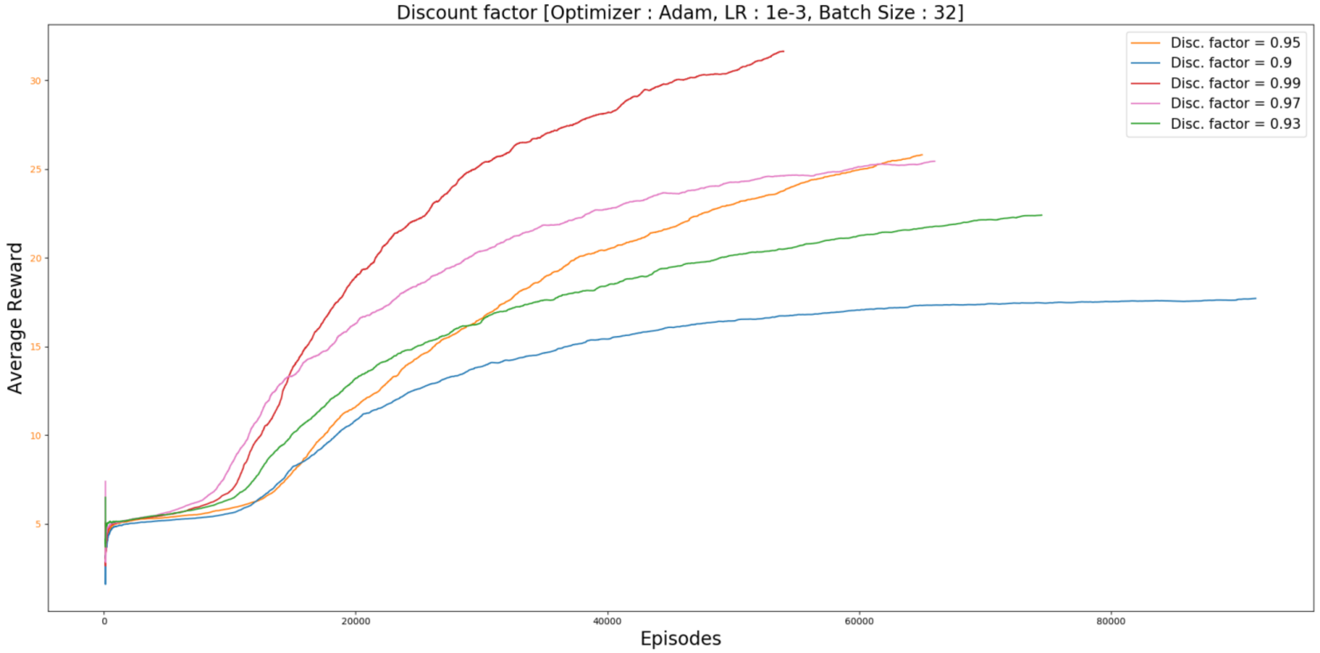
Video for
#Episodes

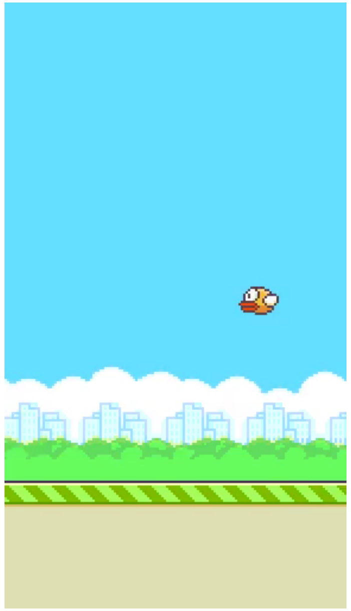
60500

65500

70500

## Discount factor Analysis





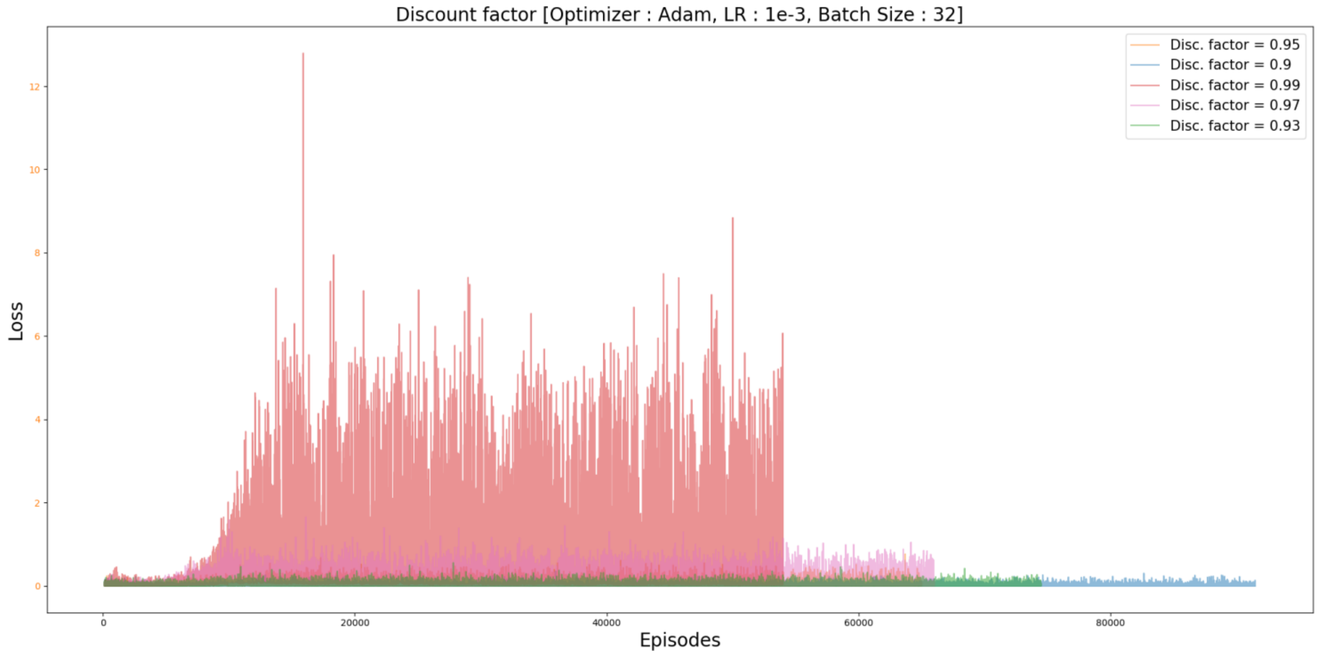
Video for
#Episodes

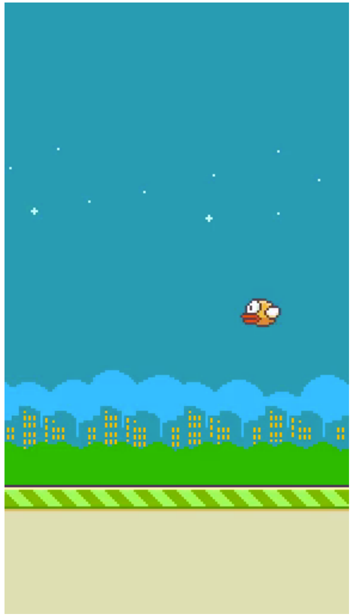
60500

65500

70500

## Discount factor Analysis





## Questions?

**Resources Used**

1. Blue Waters ~ 1300 Computation Hours
2. Google Colab (Free GPU) ~ 30 Hours

*https://github.com/drl-dql/DQN-Flappy-Bird*