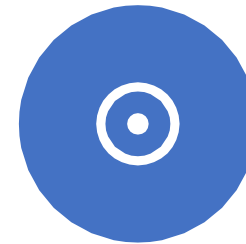


# Banana Navigation

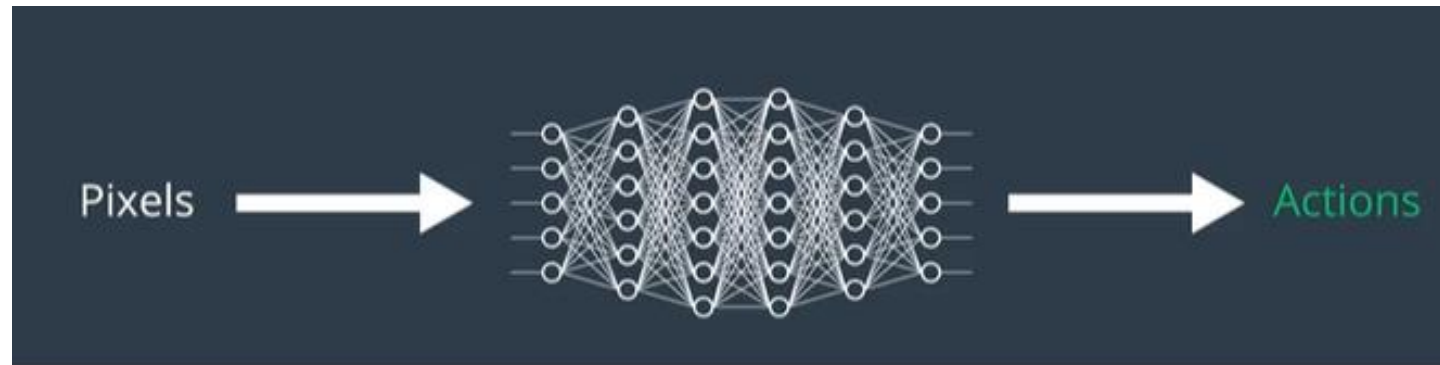
# DQN Reinforcement Model



Because we have continuous state-space, we should discretize or use non-linear function Approximation

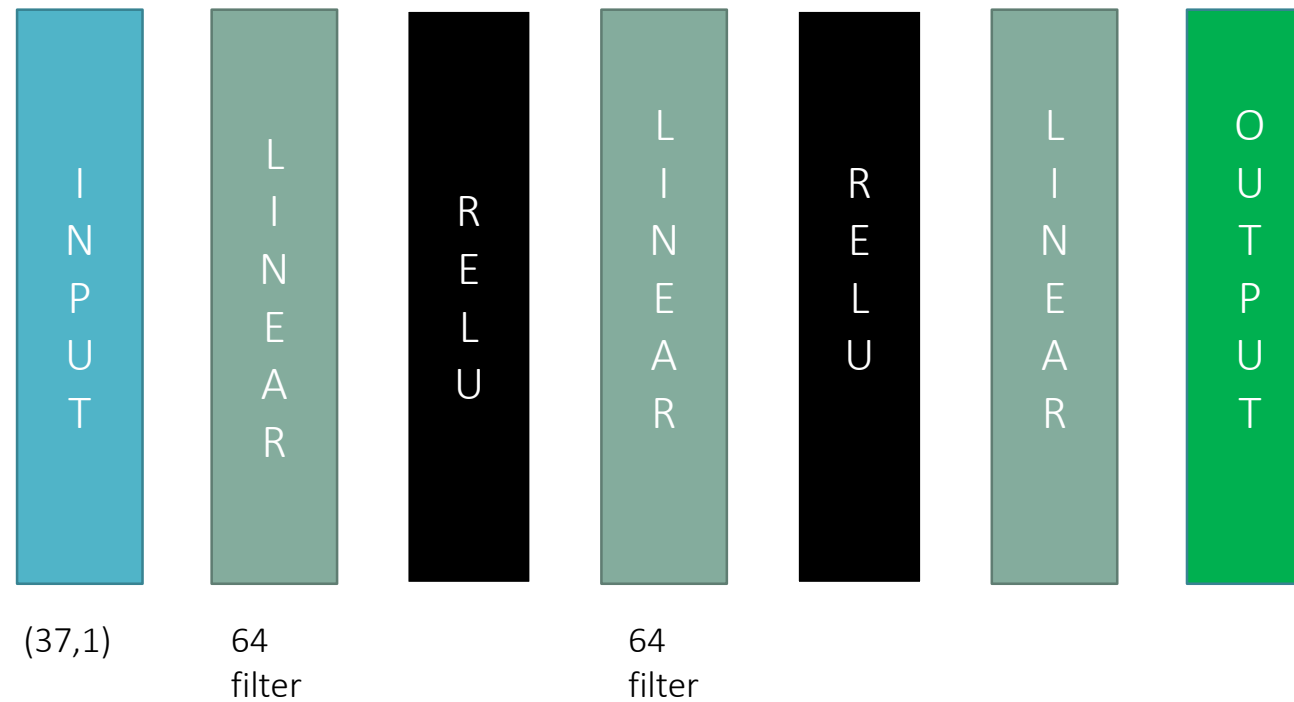


We are going to use DQN (pixel to Action) where we map an image (pixels) to Actions



# CNN architecture

We are going to build network to train our model (updating the weights) using epochs (iteration) of forward & backpropagation



# Parameters

- **GAMMA** (discounted rate) = 0.99 → this specify we are interested in our future reward
- **eps** (Epsilon) → Has starting and ending [1.0, 0.01] and it's decay over every episode by factor = 0.995
- **BATCH\_SIZE** (Replay Buffer) = 64 → more about Replay in the next slide
- **UPDATE\_EVERY** (Fixed Q-Target) = 4 → update target with latest parameter after 4 iteration (more in the next slide)

# Replay Buffer

Isolate the agent from the training process, selecting random action and store the collective reward (score) aside then uses the stored Experience randomly when training the model



# Fixed Q-Targets

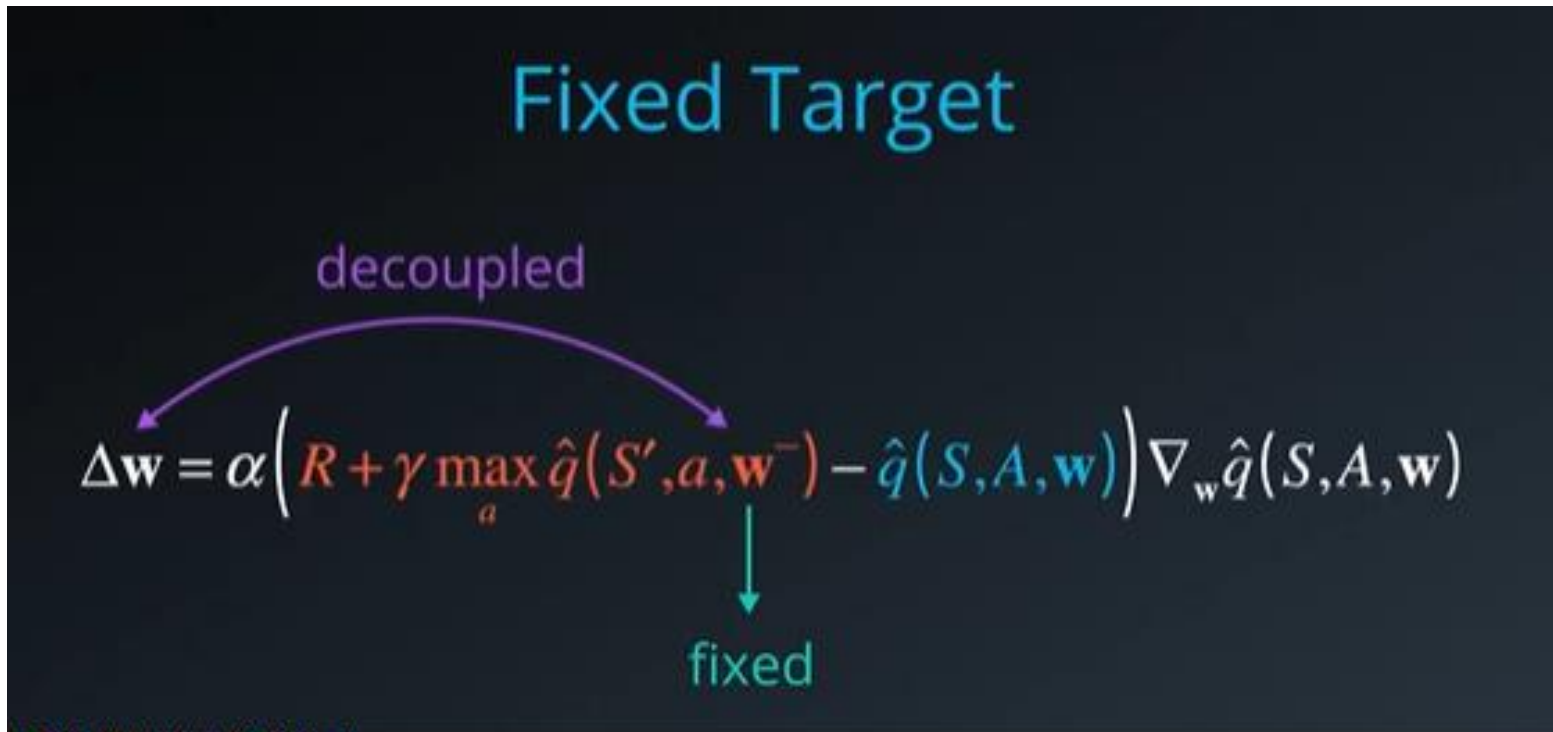
Decoupled the target from the parameters, make the training more stable, we only going to update  $W^-$  with the latest  $W$  in fixed interval

Fixed Target

decoupled

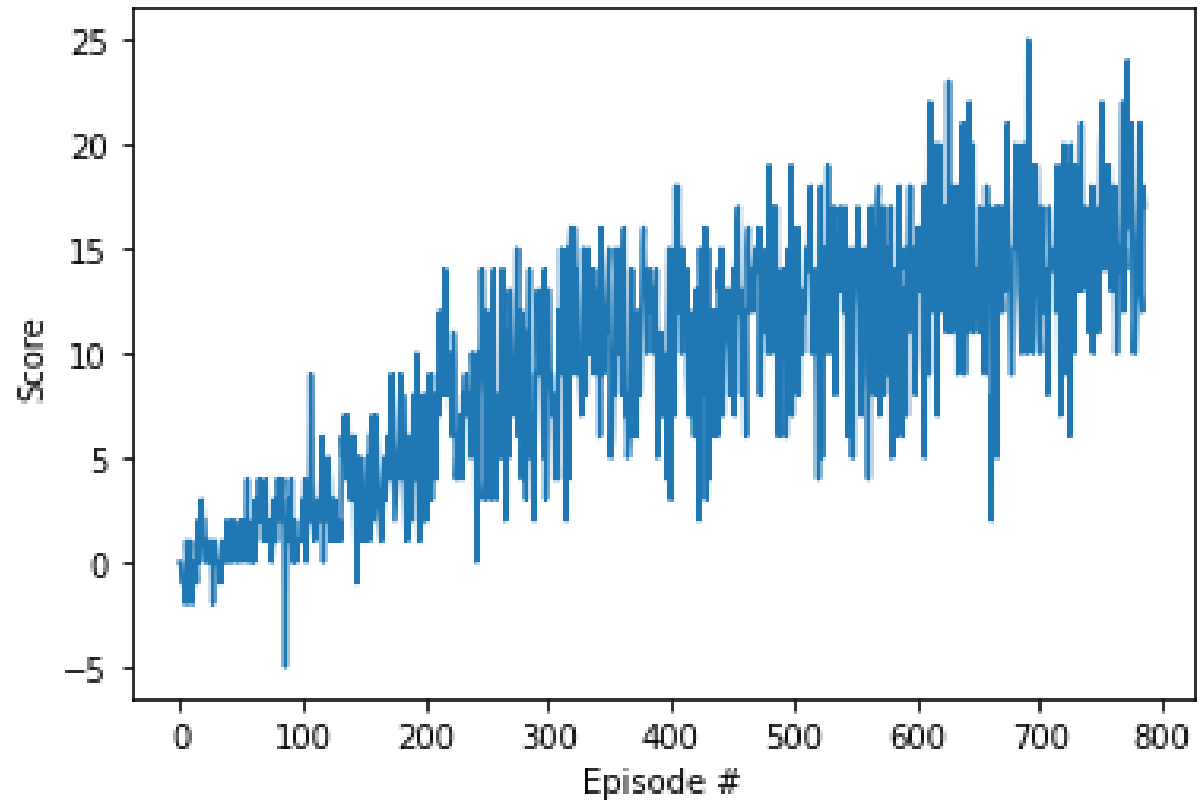
$$\Delta \mathbf{w} = \alpha \left( R + \gamma \max_a \hat{q}(S', a, \mathbf{w}^-) - \hat{q}(S, A, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{q}(S, A, \mathbf{w})$$

fixed



# Result

The Agent reaches  $\text{score} > 15.0$   
after 687 episodes



# Future Work

Trying to get higher score with fewer number of episodes (solve the environment earlier)

- Implement Agent model with Double DQN
- Implement Agent model with Rainbow

