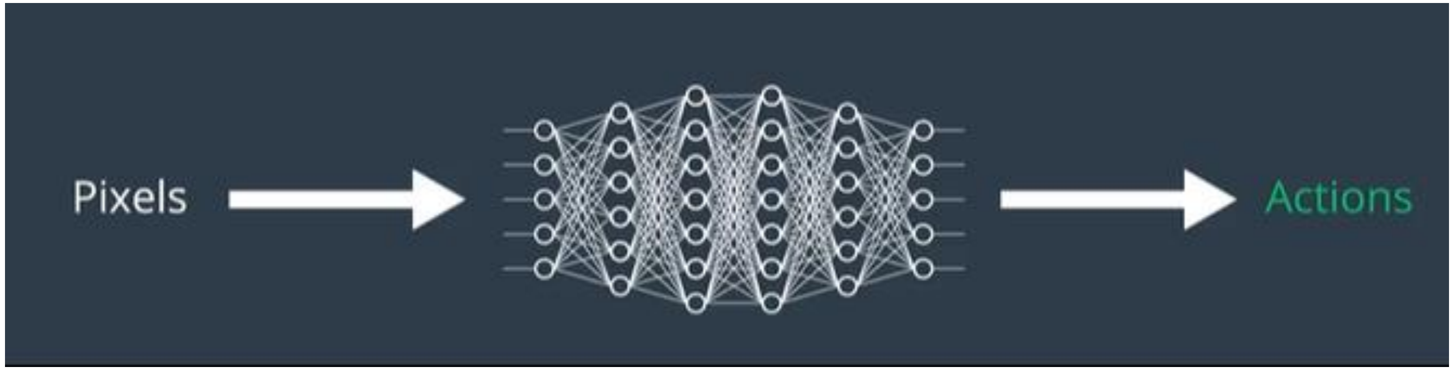
# Banana Navigation

# DQN Reinforcement Model

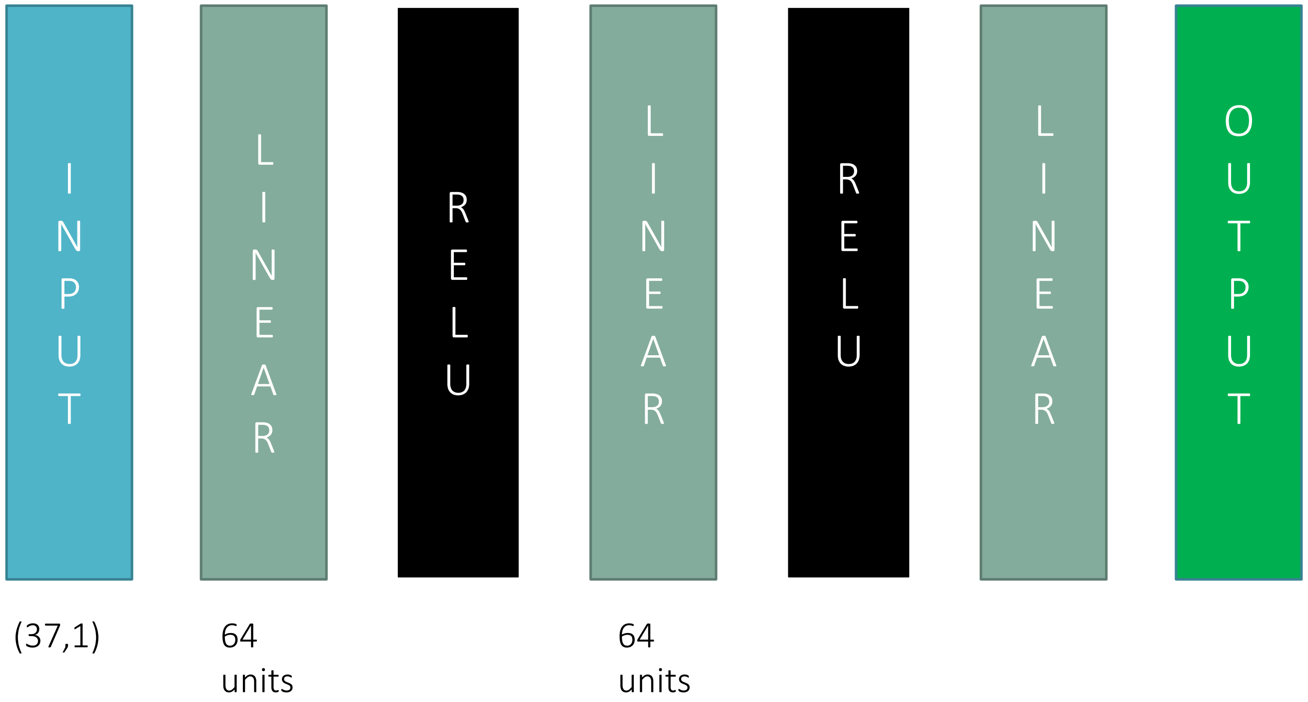
Because we have continuous state-space, we should discretize or use non-linear function Approximation

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



# Neural Network architecture

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



# Parameters

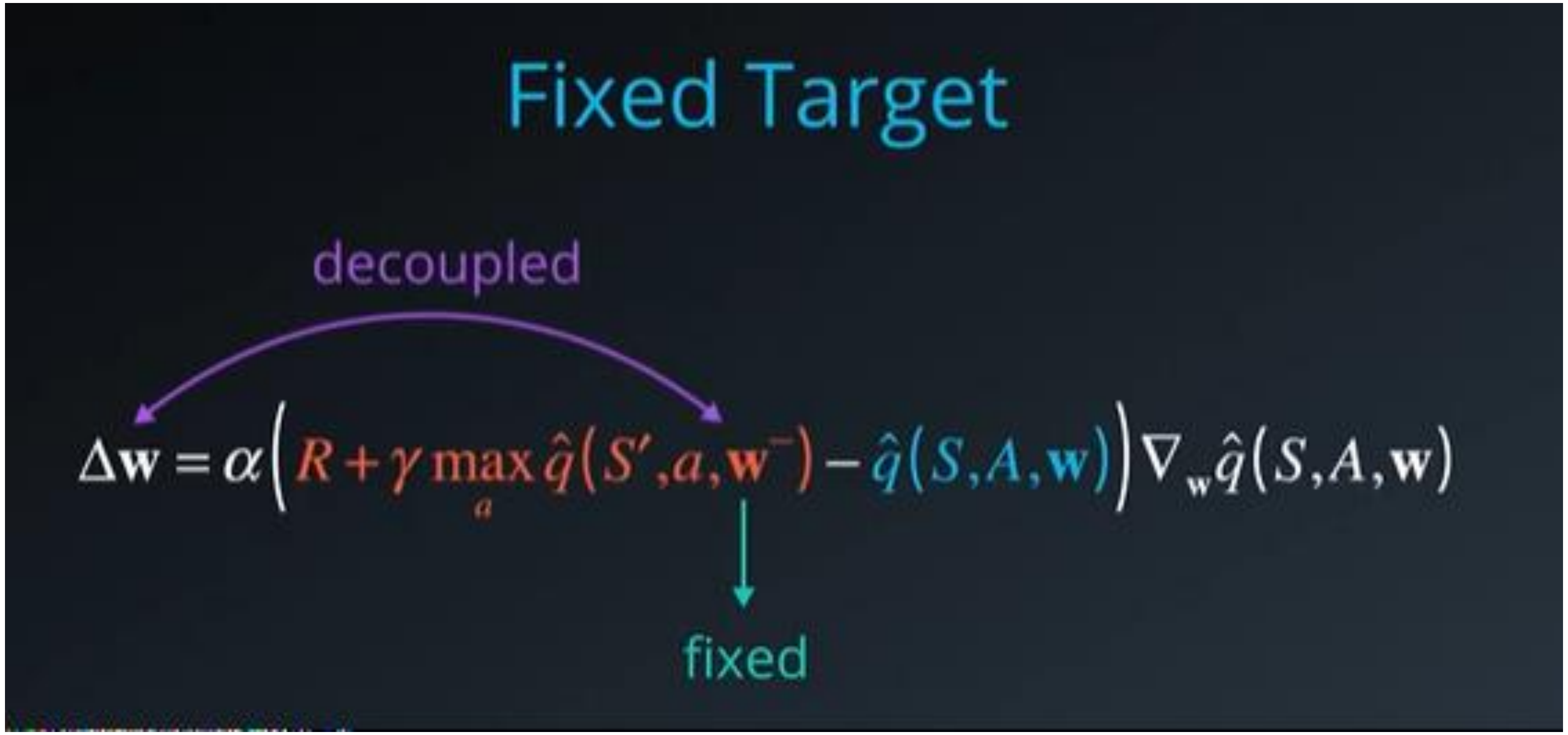
- **GAMMA** (discounted rate) = 0.99 ➔ this specify we are interested in our future reward
- **eps** (Epsilon) ➔ Has starting and ending [1.0, 0.01] and its decay over every episode by factor = 0.995
- Tau (soft-update) ➔ help prevent the variance
- **BATCH_SIZE** (Replay Buffer) = 64 ➔ more about Replay 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

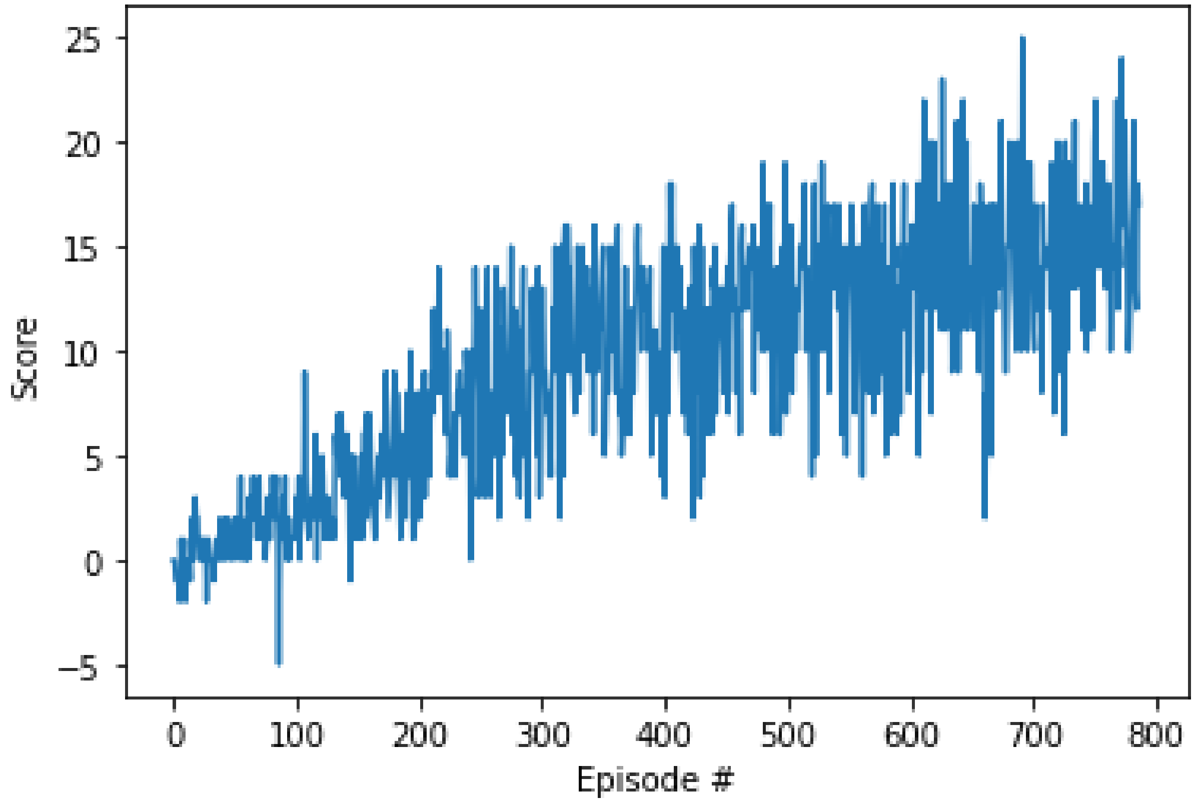
# Fixed Q-Targets

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



# Result

The Agent reaches **score>15.0** after 787 episodes




```
Episode 100     Average Score: 0.97
Episode 200     Average Score: 3.76
Episode 300     Average Score: 7.93
Episode 400     Average Score: 10.49
Episode 500     Average Score: 11.78
Episode 600     Average Score: 12.50
Episode 700     Average Score: 14.31
Episode 787     Average Score: 15.03
```

# 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