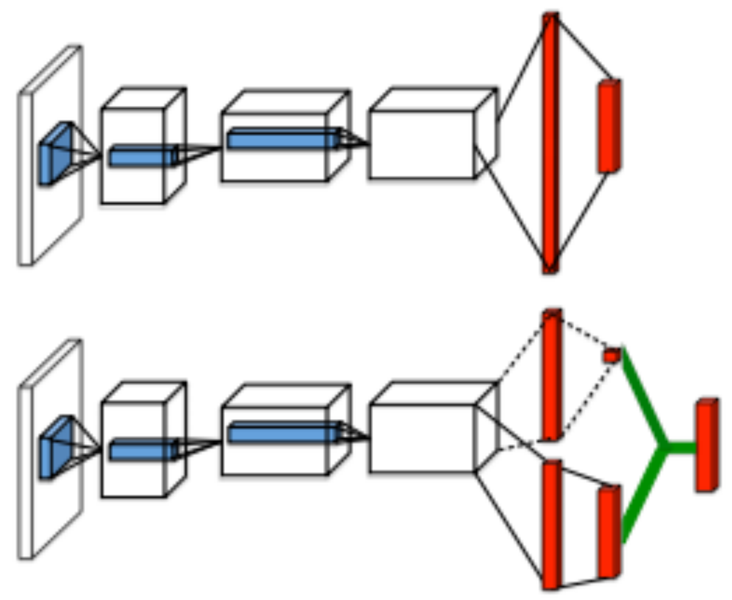
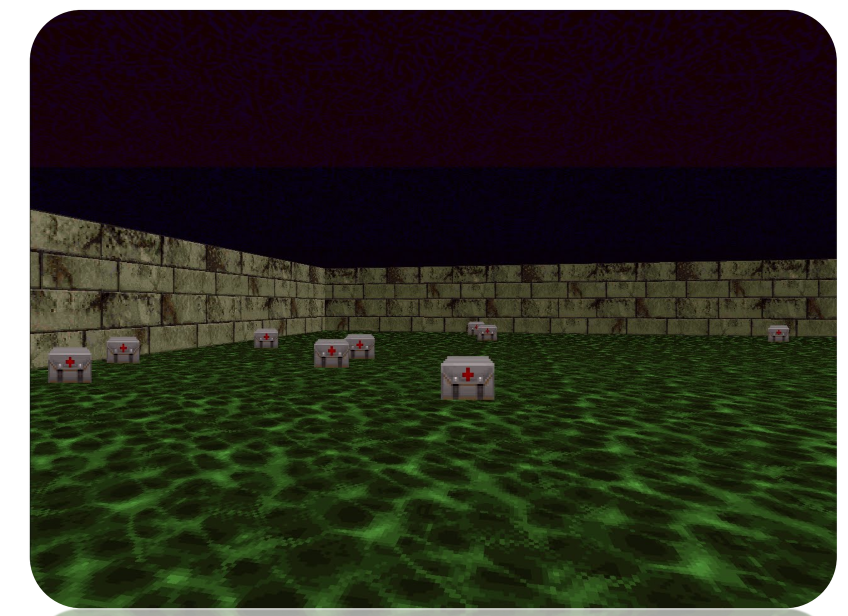
Playing DOOM using Deep Reinforcement Learning

{tdhoot, danikhan, bkonyi}@stanford.edu



Dueling DQN

Seek & Destroy Scenario



Health Gathering Scenario

Problem Statement

Teach agent to perform well in various scenarios in the video game classic, DOOM.

Maximize rewards in our training scenarios using only state that would be available to a human player (images, items, health)

Training the agent to perform well at "**seek-and-destroy**" and "**health gathering**" scenarios.

Network Variations

Double DQN: Split **action selection** network and **evaluation selection** to reduce overestimation:

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

Dueling DQN: Split the network into 2 streams: **value function** for state S, *V(S)* and **advantage function** for each action a in state S *A(S, a)*

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

Multi-Frame States: State consists of four frames instead of a single frame. This captures movement in time.

Dataset, Features & Training

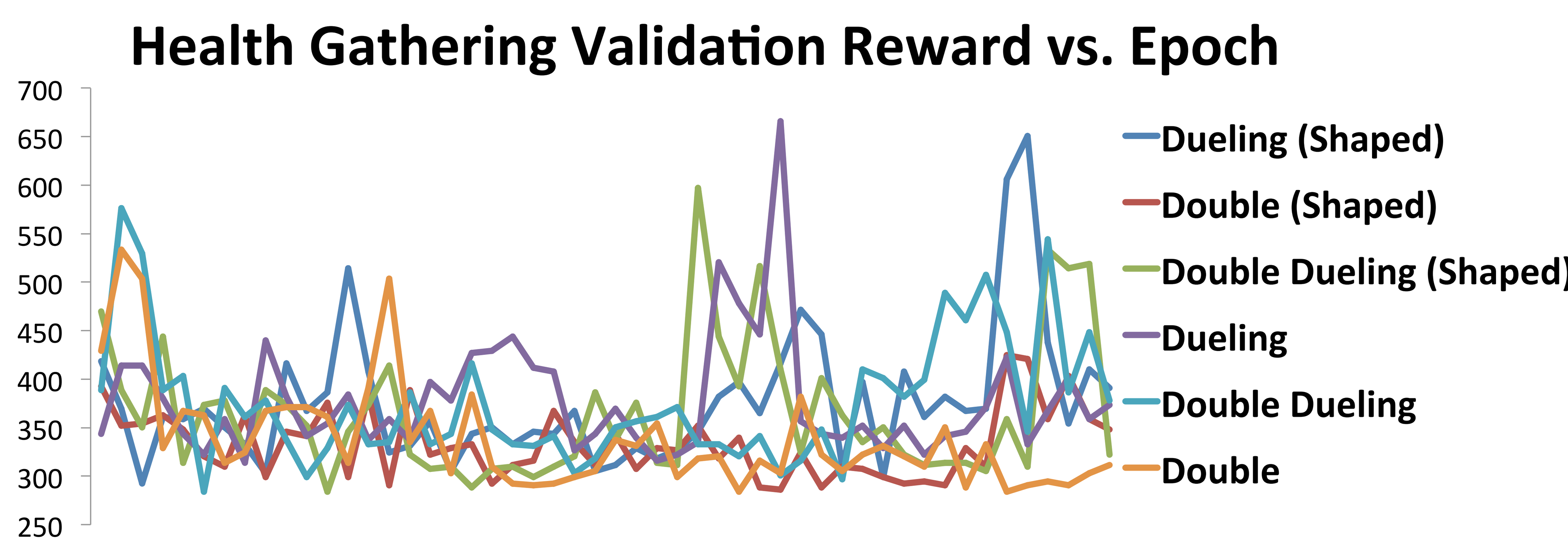
The data that was used to train our **DQN** raw pixel data, game state that would be readily available to a human player (current ammunition count, health).

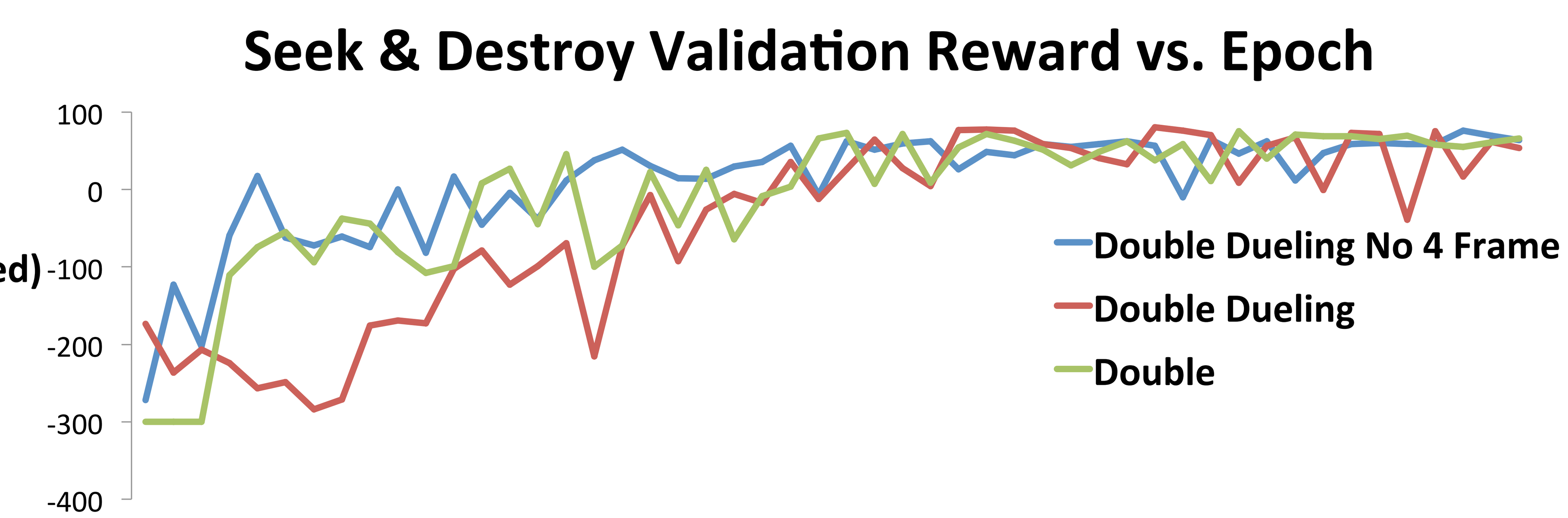
Frame Size: <640 x 480> scaled to down to **<160 x 120> (RGB & gray scale)**

Reward shaping: Encouraged movement by giving rewards based on distance moved

$$Loss = (Q(s,a) - (r + \gamma \max_a Q(s,a)))^2$$

Results





Discussion

After each training epoch (max 50), we did a validation run (**deterministic policy** using trained DQN).

The DQN performed very well on **seek & destroy,** approaching **>60 validation reward (**quickly killing the monster, +99 pts)

Results were mixed overall, for **health gathering** (reward shaping improved performance a bit) **double dueling** performed the best, **large variation** with reward shaping.

Training loss converged quickly (after a few epochs) but converged in half the time with **dueling network.**

References

[1] Deep Reinforcement Learning with Double Q-learning
Hado Hasselt-Arthur Guez-David Silver - https://arxiv.org/abs/1509.06461

[2] V. Mnih, "Playing Atari with Deep Reinforcement Learning." [Online]. Available: https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf.

[3] Dueling Network Architectures for Deep Reinforcement Learning Ziyu Wang-Tom Schaul-Matteo Hessel-Hado Hasselt-Marc Lanctot-Nando Freitas - https://arxiv.org/abs/1511.06581

DQN Architecture

First convolutional layer: 32 filters, 7x7 kernel, stride of 1, ReLU

Second convolutional layer: 32 filters, 5x5 kernel, stride of 1, ReLU

Max-pooling: 2x2 window, stride of 2 (on both convolutional layers)

First fully connected layer: 1024 output units and had 38,400 inputs from our processed images, in addition to our scalar states.

Second fully connected layer: 512 output units

Linear output layer: one output per valid action