PACMAN + DQNs

Elijah Tai

Contents

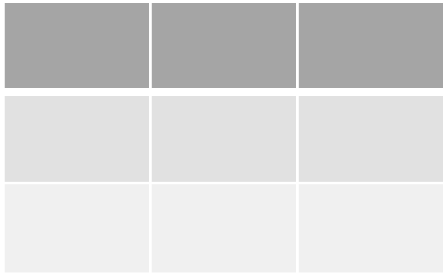
- Topic + Background
- Hypothesis + Importance
- Progress from Midterm Report
- Results + Code
- Issues + Solutions
- Next Steps

Contents

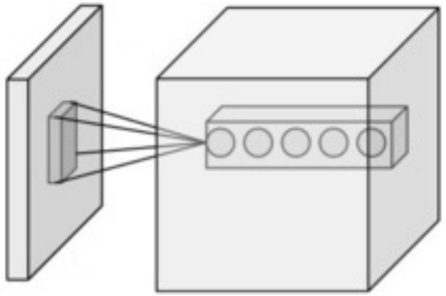
$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{current value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t, a_t)}_{\text{current value}} \bigg)}^{\text{temporal difference}}$$

Watkins, C.J.C.H., Dayan, P. *Q*-learning. *Mach Learn* 8, 279–292 (1992). https://doi.org/10.1007/BF00992698

$$Q^{new}(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{current value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\bigg(\underbrace{\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(s_t, a_t)}_{\text{current value}} \bigg)}^{\text{temporal difference}}$$

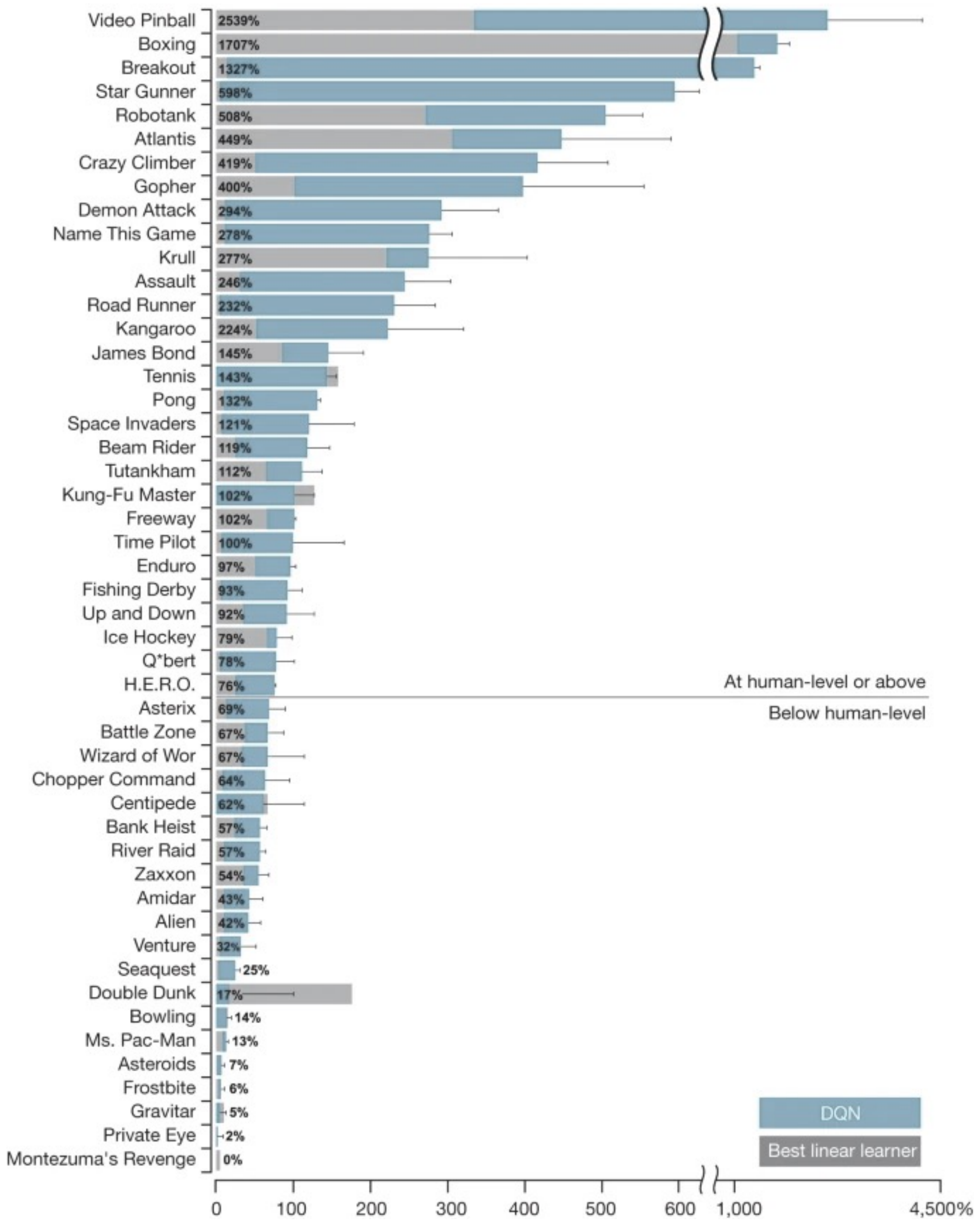


$$Q^A_{t+1}(s_t, a_t) = Q^A_t(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma Q^B_t\left(s_{t+1}, \arg\max_a Q^A_t(s_{t+1}, a)\right) - Q^A_t(s_t, a_t)\right)$$

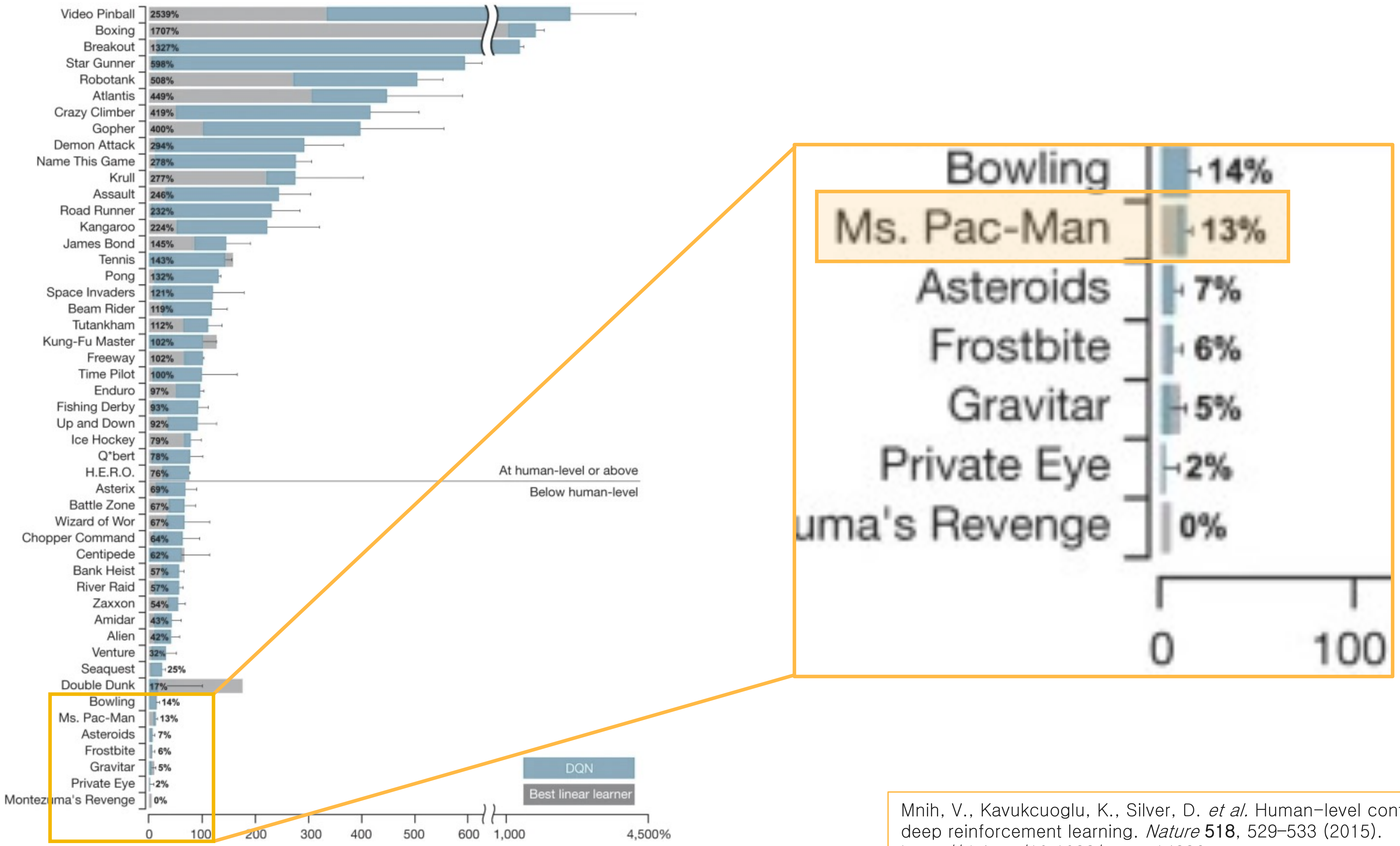


Watkins, C.J.C.H., Dayan, P. *Q*-learning. *Mach Learn* **8**, 279–292 (1992). https://doi.org/10.1007/BF00992698

Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. *arXiv preprint arXiv:1312.5602* (2013).



Mnih, V., Kavukcuoglu, K., Silver, D. *et al.* Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015). https://doi.org/10.1038/nature14236



Mnih, V., Kavukcuoglu, K., Silver, D. *et al.* Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015). https://doi.org/10.1038/nature14236

Contents

Hypothesis

- DQN algorithm struggles in MS PACMAN because of the power-up pellets. Suddenly, the best strategy is to chase ghosts instead of fleeing. This change confuses the network.

Removing the reward from eating ghosts during training paradoxically increases rewards.	Initializing powered-up MS PACMAN to do the opposite movement increases rewards.	Discretely context switching between two different networks will increase rewards.	Valid combinations of the other optimizations will further increase rewards.

Importance

- While the optimizations are specific to MS PACMAN, this generally demonstrates tactics to improve performance, especially within settings of low computational resources when the neural network may not have a chance to stably converge.

Contents

Progress

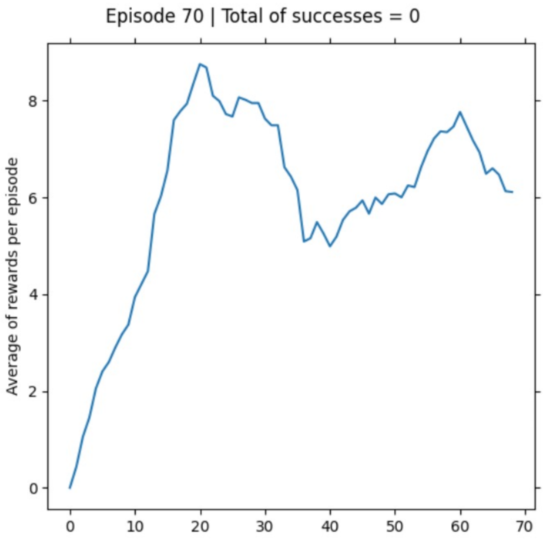
- Previously, I got an existing DQN running from: github.com/bourbonut/dqn-pacman

- Learned about DQNs and fixed bugs in the original code.
- Wrote a script to determine the invincibility frames of ms pacman after eating a power-pellet
- Coded different behaviors triggered by certain score rewards are triggers.
- Ran 10k frames ~65 epochs on three of the four specific hypotheses
- Generated plots for visualization

Contents

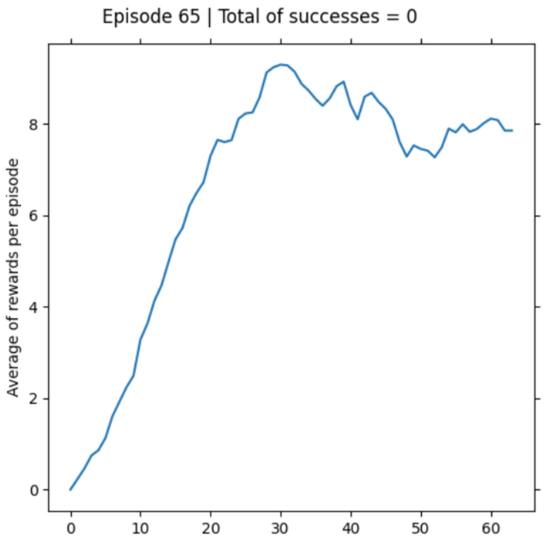
Results: 10k steps

Neither



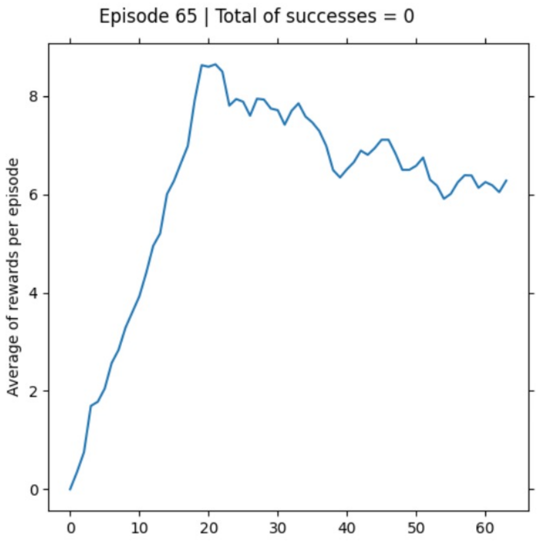
6.0

No Ghost Reward



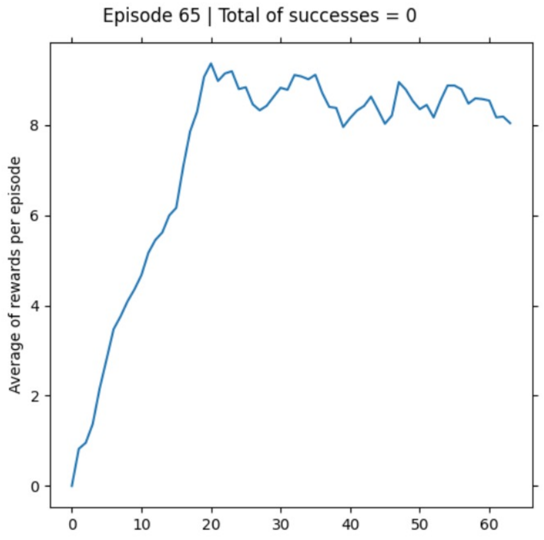
7.8

Opposite Initialization



6.5

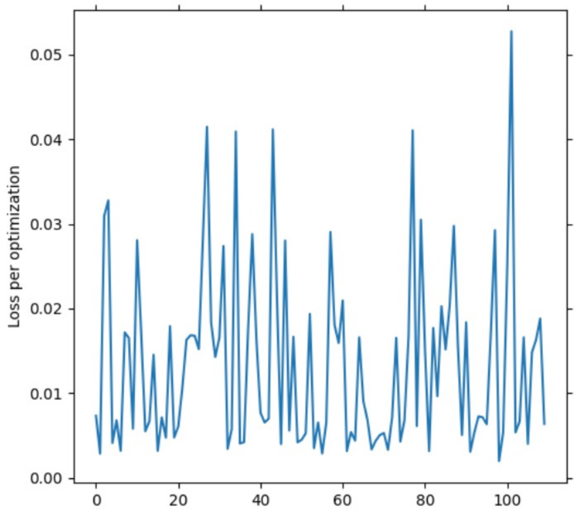
Both



8.1

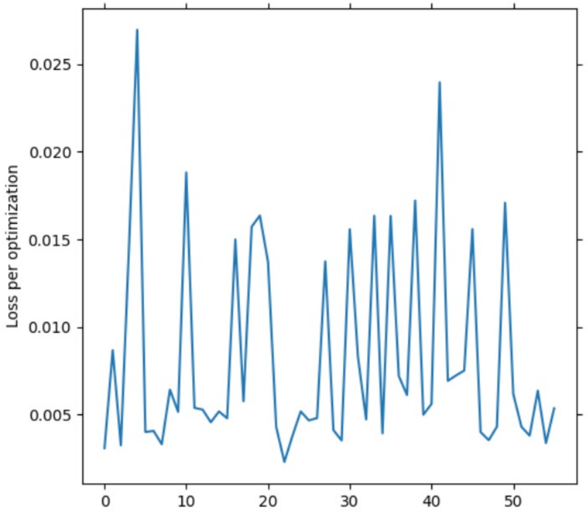
Results: 10k steps

Neither



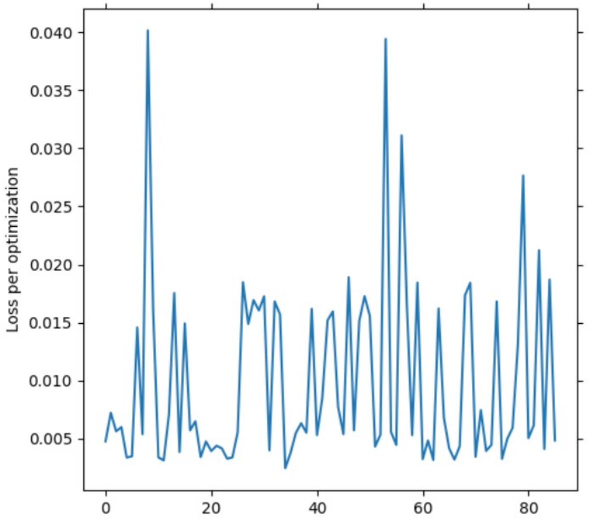
0.054

No Ghost Reward



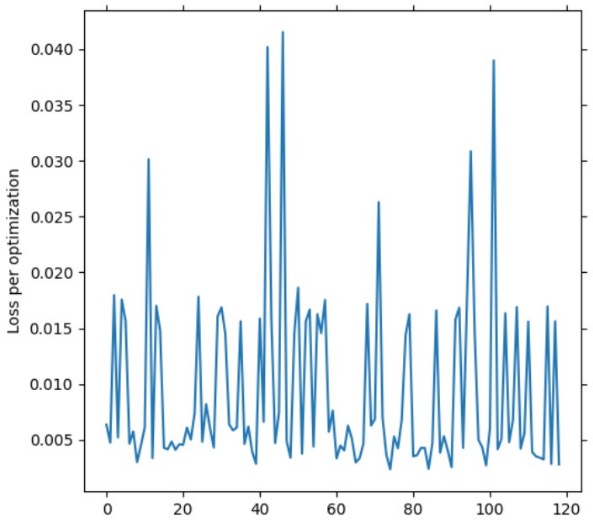
0.027

Opposite Initialization



0.040

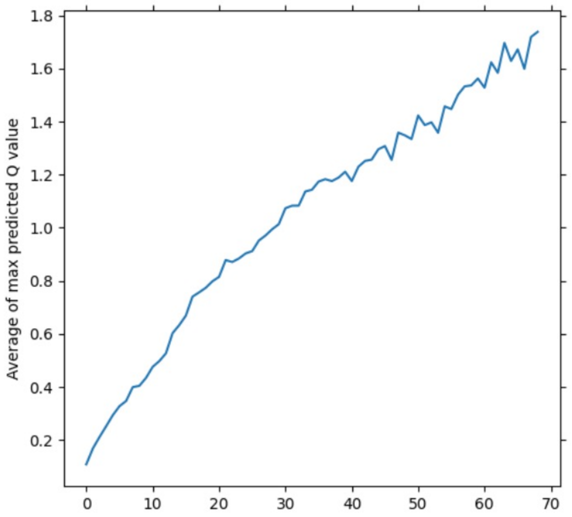
Both



0.042

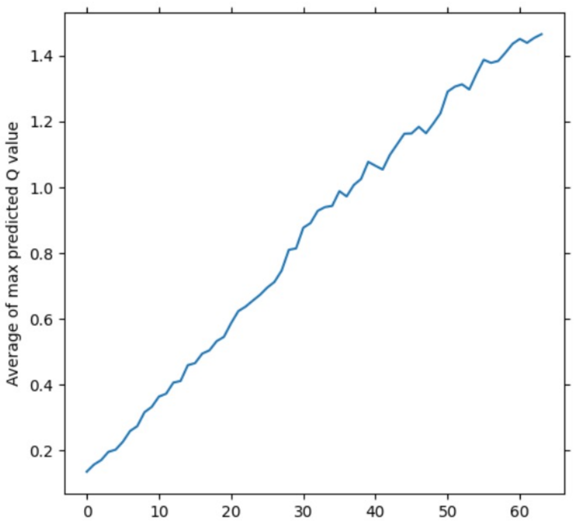
Results: 10k steps

Neither



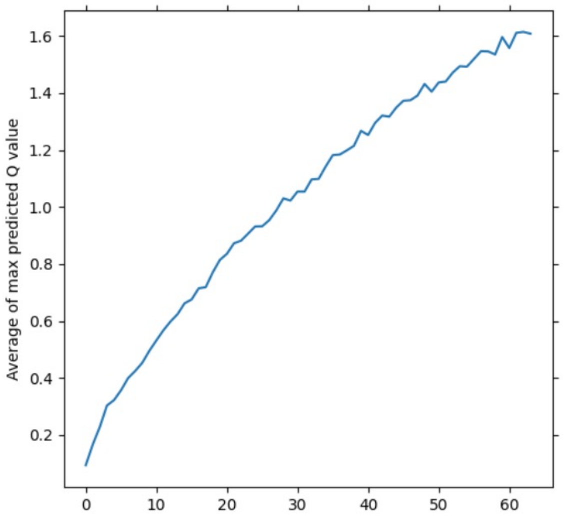
1.72

No Ghost Reward



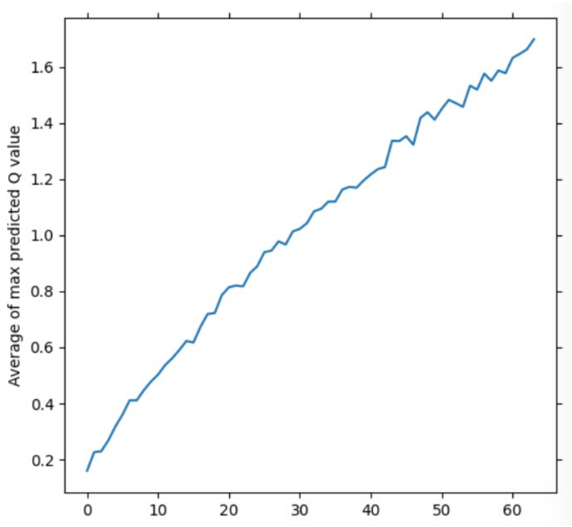
1.47

Opposite Initialization



1.62

Both



1.70

Main Code Alterations

Invincibility Framing

```python
import gym
import numpy as np
import matplotlib.pyplot as plt
import torch
import random as r

env = gym.make("MsPacman-v4", render_mode='human')
env.reset()

stepmode = False
phase = 0
counter = 0
for _ in range(10000):
    # env.action_space.n |=> 9
    # env.get_action_meanings() |=> ['NOOP', 'UP', 'RIGHT', 'LEFT', 'DO

    if phase == 0:
        next_state, reward, terminated, truncated, info = env.step(4)
        counter += 1
        if counter == 120:
            phase += 1
            counter = 0
    if phase == 1:
        next_state, reward, terminated, truncated, info = env.step(3)
        counter += 1
        if counter >= 80:
            phase += 1
            counter = 0
    if phase == 2:
        counter += 1
        next_state, reward, terminated, truncated, info = env.step(1)
        if reward == 50:
            phase += 1
            counter = 0
    if phase == 3:
        counter += 1
        input(f'continue {counter}')

        next_state, reward, terminated, truncated, info = env.step(0)

    env.render()
    print(phase)

env.close()
```

env.st

No Ghost Reward

```python
parser.add_argument(
    "--ghostbonus",
    action="store_true",
    dest="ghostbonus",
    help="no reward for eating a ghost or a strawberry",
)
```

```python
            # Elijah: Change the reward to ignore eating a ghost
            display_reward = reward_
            if args.ghostbonus and (reward_ == 200 or reward_ % 400 == 0):
                display_reward = 0
```

```python
            # Elijah: We want to display the true reward, but we wa
            if args.ghostbonus:
                display.data.rewards.append(display_reward)
            else:
                display.data.rewards.append(reward)
            reward = torch.tensor([reward], device=device)

            old_action = action_
            if reward != 0:
                dmaker.old_action = action.item()

            next_state = preprocess_observation(observations, obs)
```

```python
    def _save(self, image=False):
        """Save data in `pickle` file and an image"""
        if image:
            PATH_PLOTS = self.data.path / '..' / 'plots'
            self.update_axis()
            self.fig.tight_layout()
            plt.savefig(PATH_PLOTS / f"episode-{self.data.ep}.png")
            print(f"Figure {self.data.ep} saved.")
            for axis in self.axis:
                axis.cla()
        self.data.save()
```

Opposite Initialization

```python
parser.add_argument(
    "--ghostrev",
    action="store_true",
    dest="ghostrev",
    help="train a single NN, reversing actions under power pellet"
)
```

```python
            # Elijah: Reverse the action that we do
            if args.ghostrev and rev_counter > 0:
                action = [0, 4, 3, 2, 1, 9, 7, 6, 5][action]
                rev_counter -= 1

            action_ = ACTIONS[old_action][action.item()] # Elijah: This sort of softens the new action.

            obs, reward_, done, info = env.step(action_)
```

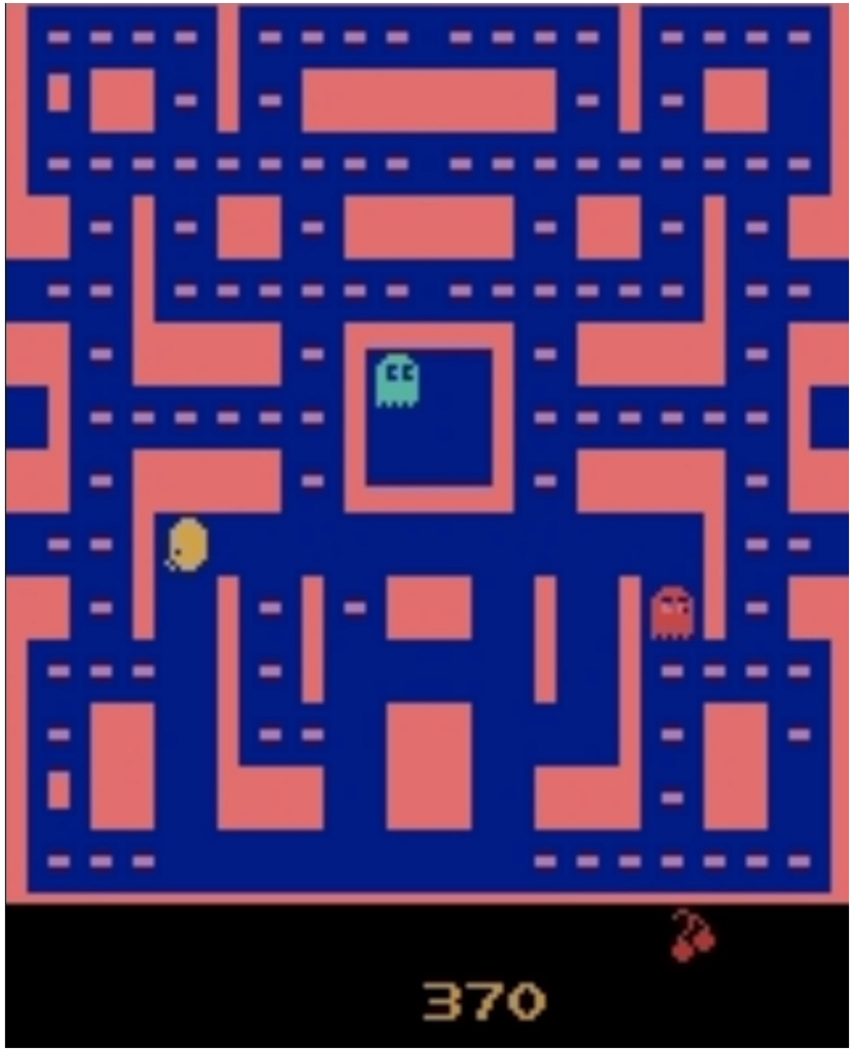
```python
REVERSED = {0: 1, 1: 0, 2: 3, 3: 2}
isreversed = (
    lambda last_action, action: "default" if REVERSED[action] - last_action else "reverse"
)

ACTIONS = {
    1: [1, 4, 6, 5],
    2: [5, 7, 3, 2],
    3: [6, 8, 3, 2],
    4: [1, 4, 8, 7],
    5: [1, 4, 3, 2],
    6: [1, 4, 3, 2],
    7: [1, 4, 3, 2],
    8: [1, 4, 3, 2],
}
```

```python
TARGET_UPDATE = 400  # here, Elijah changed from 8_
REPLAY_MEMORY_SIZE = 3 * 1200 #here, Elijah changed

# Environment constants
N_ACTIONS = 4
AVOIDED_STEPS = 80  # At the beginning, there is a
DEAD_STEPS = 36  # frames to avoid when the agent d
K_FRAME = 2

# Optimizer parameters
LEARNING_RATE = 2.5e-4
# DECAY_RATE = 0.99
MOMENTUM = 0.95

# Algorithm constant
MAX_FRAMES = 20_000 #Elijah changed from 2,000,000
SAVE_MODEL = 5 # Elijah changed from 20
```

- ghostbonusDQN
  - models
    - policy-model-5.pt
    - policy-model-10.pt
    - policy-model-15.pt
    - policy-model-20.pt
    - policy-model-25.pt
    - policy-model-30.pt
    - policy-model-35.pt
    - policy-model-40.pt
    - policy-model-45.pt
    - policy-model-50.pt
    - policy-model-55.pt
    - policy-model-60.pt
    - policy-model-65.pt
    - policy-model-final.pt
    - target-model-5.pt
    - target-model-10.pt
    - target-model-15.pt
    - target-model-20.pt
    - target-model-25.pt
    - target-model-30.pt
    - target-model-35.pt
    - target-model-40.pt
    - target-model-45.pt
    - target-model-50.pt
    - target-model-55.pt
    - target-model-60.pt
    - target-model-65.pt
    - target-model-final.pt

- plots
  - episode-5.png
  - episode-10.png
  - episode-15.png
  - episode-20.png
  - episode-25.png
  - episode-30.png
  - episode-35.png
  - episode-40.png
  - episode-45.png
  - episode-50.png
  - episode-55.png
  - episode-60.png
  - episode-65.png
- recorded-data
  - episode-5.pkl
  - episode-10.pkl
  - episode-15.pkl
  - episode-20.pkl
  - episode-25.pkl
  - episode-30.pkl
  - episode-35.pkl
  - episode-40.pkl
  - episode-45.pkl
  - episode-50.pkl
  - episode-55.pkl
  - episode-60.pkl
  - episode-65.pkl

Demo



Contents

Issues + Solutions

- Need variance metric for comparing the learning curves
  - Need to run the code multiple times to make better comparison
- Only have results for short time frames
  - Plan to let it run for ~1000 episodes
- Still need to add in the context switching (double) double DQN
  - Requires a new class and code structure changes
- This will add more quantitative rigor

Contents

Planning for Next Steps

- Running more times for longer
- Implementing the Double-Double DQN (the third hypothesis)
- Generating more quantitative metrics for learning curve comparison
- Extra: Implementing Dueling DQN or soft steps (Polyak averaging)
- Extra: Comparing greyscale versus color to understand ghost personalities

Thank you!