Comparative Study of Reinforcement Learning using Pacman Game

Santhiya Theanraj

Temple University

Introduction

This project focuses on developing a Pacman game using **Reinforcement Learning**. The environment contains four main components:

- **Pacman (Agent)**
- **Food (Dots)**
- **Ghosts (Enemies)**
- **Walls**

The objective is to eat all food dots in the maze without being caught by the ghost. Each state and action produces a specific reward signal:

Event	**Reward**	**Purpose**
Normal move	-1	Encourage efficiency
Eat dot	+10	Incentivize collecting food
Hit ghost	-500	Strong penalty for dying
Collect all dots	+500	Bonus for winning

Pacman Grid Design

Different Pacman grid sizes were used to study how the game's complexity increases as the state space grows.

- **Small Grid**: 6×6
- **Medium Grid**: 7×8
- **Small Classic Grid**: 12×6
- **Medium Classic Grid**: 23×9

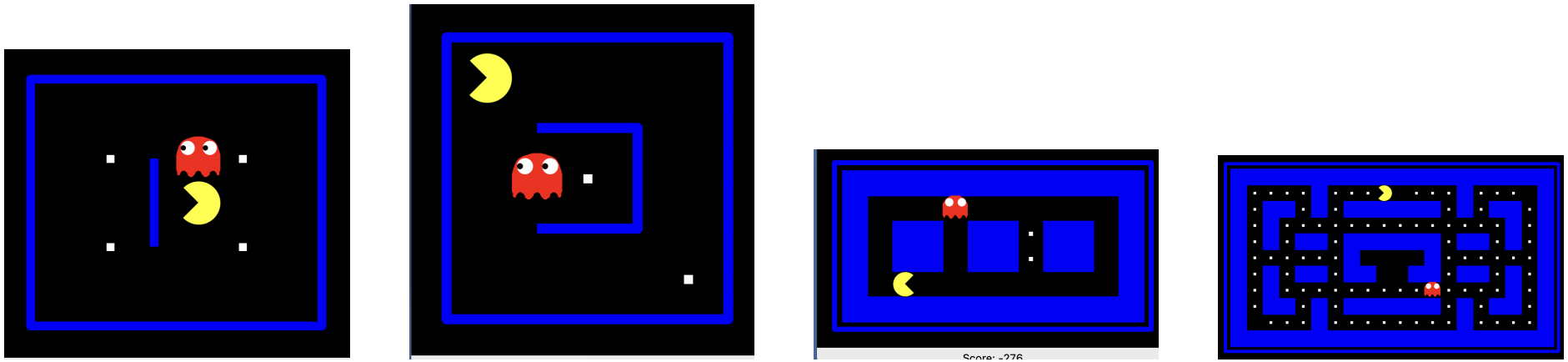


Figure: Visual representation of Pacman grids from small to super classic

Q-Learning Algorithm Overview

What is Q-Learning?

- Model-free reinforcement learning algorithm
- Learns optimal action-value function $Q(\text{state}, \text{action})$
- Finds best policy without knowing environment dynamics
- **Q-Table:** Stores quality values for state-action pairs
- **Bellman Equation:** Updates Q-values based on rewards
- **Epsilon-Greedy Policy:** Balances exploration vs. exploitation

Mathematical Foundation:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

Q-Learning Training on Small Grid

Experiment 1:

- Agent wins the game but has negative rewards during training.
- Exploration is limited.
- Parameters: `python pacman.py small -a q -t 2000 -p 5 -d 0.9 -r 0.5 -e 0.3`

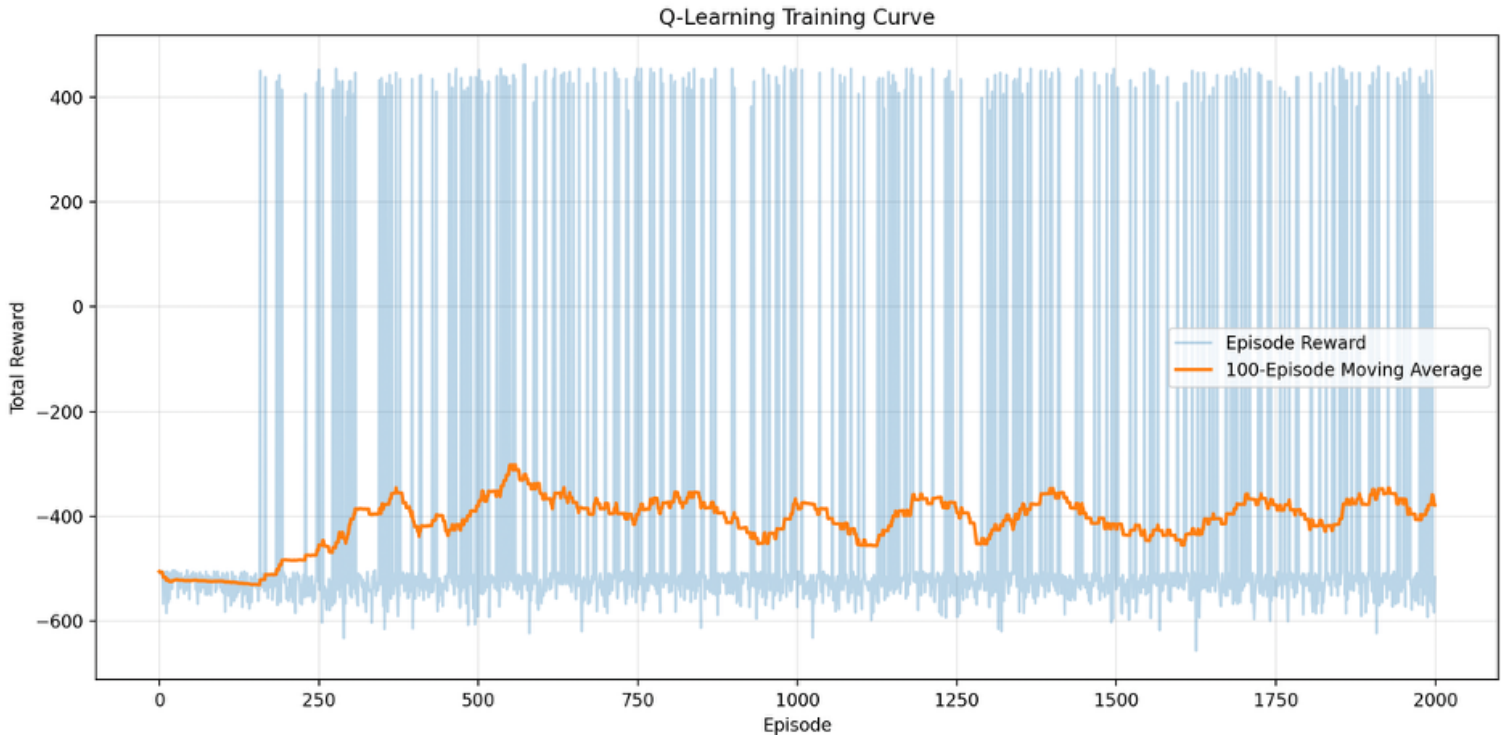


Figure: Q-Learning Training Curve (Episode 1)

Q-Learning Training on Small Grid

Experiment 2:

- Reduced exploration rate (epsilon) from 0.3 to 0.1 to encourage more exploration.
- Observed changes in reward trends and learning curve.

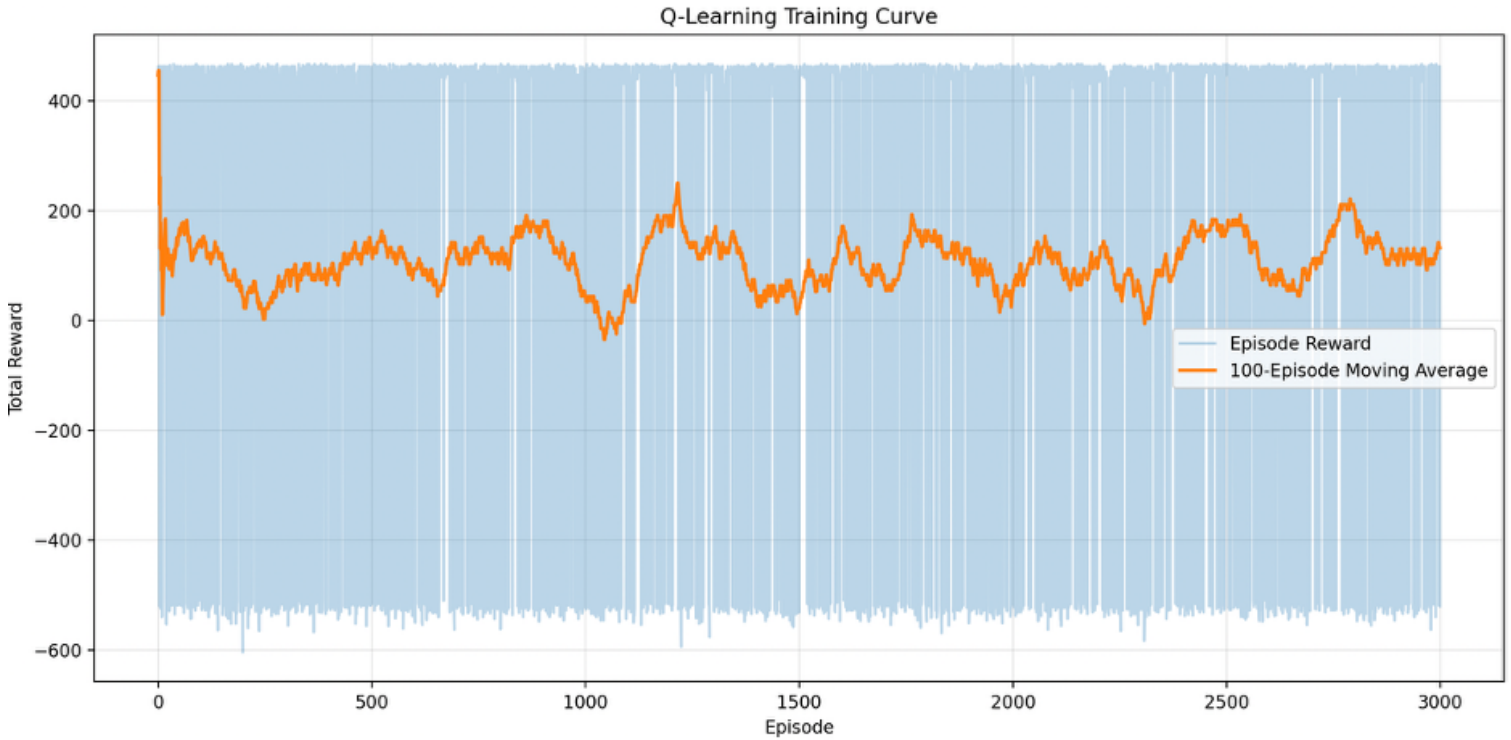


Figure: Q-Learning Training Curve (Episode 2)

Approx Q Learning with 6 features

1. **Ghost Distance** – How far is the ghost? (continuous $[0, 1]$)
2. **Ghost Danger** – Is ghost within 2 steps? (binary $\{0, 1\}$)
3. **Eats Food** – Does this action collect food? (binary $\{0, 1\}$)
4. **Food Distance** – How close to nearest food? (continuous $[0, 1]$)
5. **Dots Remaining** – How much food is left? (continuous $[0, 1]$)
6. **Bias** – Baseline constant (always 1.0)

These features capture the *essence* of any game state, allowing generalization to unseen situations.

Q-Learning Comparison (SmallClassic 12×6)

Environment: SmallClassic Grid (12×6)

Metric	Approx Q (Better)	Approx Q (Simple)	Q-Learning
Episodes	1500	1500	5000
Wins	911	51	1
Losses	589	1449	4999
Win Rate (%)	60.73	3.40	0.02
Last 100 Win (%)	61.00	2.00	0.00
Avg Reward	-200.24	-715.12	-769.29
Last 100 Avg	-231.35	-744.06	-856.26
Best Reward	209.00	164.00	167.00
Worst Reward	-2730.00	-1415.00	-3258.00

- Approx Q-Learning with better features significantly outperforms all other methods.
- Simple features reduce performance drastically.
- Vanilla Q-Learning fails on this grid (near-zero win rate), highlighting the need for feature engineering.

Deep Q-Network (DQN): High-Level Overview

DQN replaces the Q-table with a deep neural network that learns Q-values directly from raw grid states. It automatically extracts features through convolutional layers, removing the need for hand-crafted features used in Approximate Q-Learning.

Key Ideas:

- Uses a **5-channel grid input** (Pacman, Ghost, Food, Ghost Direction, Walls)
- Learns Q-values using **3 convolutional layers + 2 fully connected layers**
- Stabilized with:
  - **Experience Replay** (100K transitions)
  - **Target Network** (updated every 100 steps)
- Trains end-to-end: no manual features needed

DQN Architecture and Training Pipeline

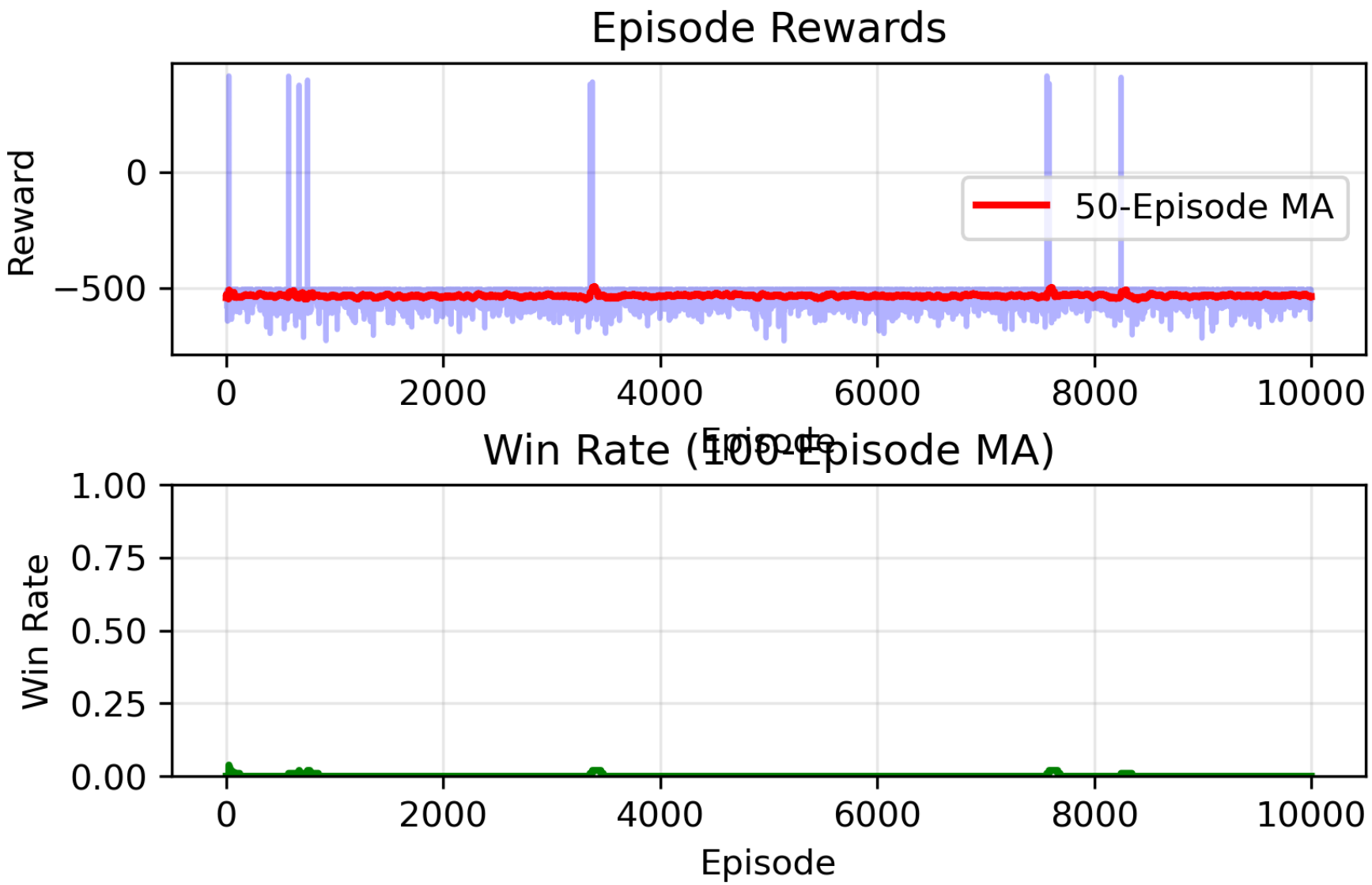
Neural Network Architecture:

- **Input:** 5-channel grid (Pacman, Ghost, Food, Direction, Walls)
- **Conv Layers:**
  - Conv1: 8 filters, 3×3
  - Conv2: 16 filters, 3×3
  - Conv3: 32 filters, 3×3
- **FC Layers:**
  - FC1: 256 neurons
  - FC2: 4 outputs (Q-values for Up/Down/Left/Right)
- **Total Parameters:** $\sim$ 794K

Training Mechanisms:

- **Experience Replay:** stores 100K transitions, samples random minibatches
- **Target Network:** updated every 100 steps for stable targets
- **Epsilon-Greedy:** $1.0 \rightarrow 0.1$ over 4000 episodes

DQN Performance on Medium Classic Grid (19×22)



DQN summary

- **Outcome:** 0 wins out of 10,000 episodes.
- Reward curve stays around $-500 \rightarrow$ agent dies very early.
- 50-episode moving average (red) shows no improvement.
- Win rate remains at 0% throughout training.

Why it failed:

- Very large state space for CNN to learn from.
- Sparse rewards + early deaths $\rightarrow$ unstable learning.
- Replay memory dominated by negative transitions.
- DQN underpowered for complex ghost dynamics.

Conclusion and Future Work

DQN Training Status on Classic-Medium Grid (19×22)

- Still refining the CNN architecture to improve DQN stability and performance.
- Training remains computationally expensive:
  - Over 4 hours required for just 5,000 episodes on limited hardware.
- Switched to **Kaggle TPU** to accelerate experiments and reduce training time.
- Continuing to:
  - Optimize CNN layers for better feature extraction.
  - Tune hyperparameters (learning rate, replay buffer size, batch size).
  - Experiment with deeper convolutional stacks and residual connections.

Work in progress — aiming for a DQN agent that can win consistently in the 19×22 Classic-Medium Grid.

References I


Berkeley Pacman Code.
`http://ai.berkeley.edu/projectoverview.html`


Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller.
Playing Atari with Deep Reinforcement Learning.
arXiv:1312.5602, 2013.


Volodymyr Mnih et al.
Human-level Control through Deep Reinforcement Learning.
Nature, 2015. doi:10.1038/nature14236.


Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver.
Prioritized Experience Replay.
arXiv:1511.05952, 2016.