# LEARNING TO PLAY PAC-MAN WITH REINFORCEMENT LEARNING

Machine Learning 2016/2017
Final Project

Davide Aurucci - David Langbroek
Albert Segarra - Fthi Arefayne

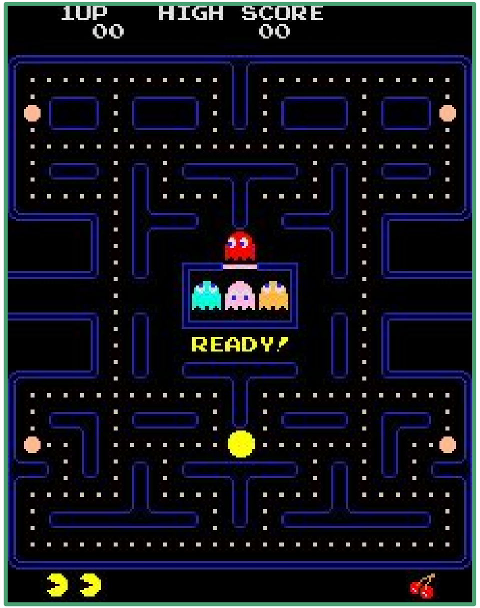
31 January 2017

# PAC-MAN

- Arcade video-game (1982)
- Simplicity of the game rules
- Complex strategies that are required to obtain a proper score

**Our Idea**

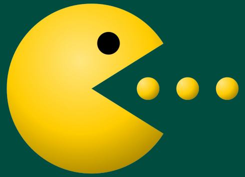
- Reinforcement Learning (**RL**)
- Neural Network that is trained using **Q-Learning**
- Development in **C++**



# GENERAL OVERVIEW

- Application
- Methods
- Rewards of the game
- Experiments & Results
- Demo

# Application



# SIMULATION OF THE GAME

**KEY POINT**: we created our **own simulation** of the game with **3 mazes and 1 level**

## Elements of the Game

### PAC-MAN

- ➔ 1 Life

### 4 GHOSTS

- ➔ Scatter/Chase
- ➔ Scared, dead or alive
- ➔ Variable speed

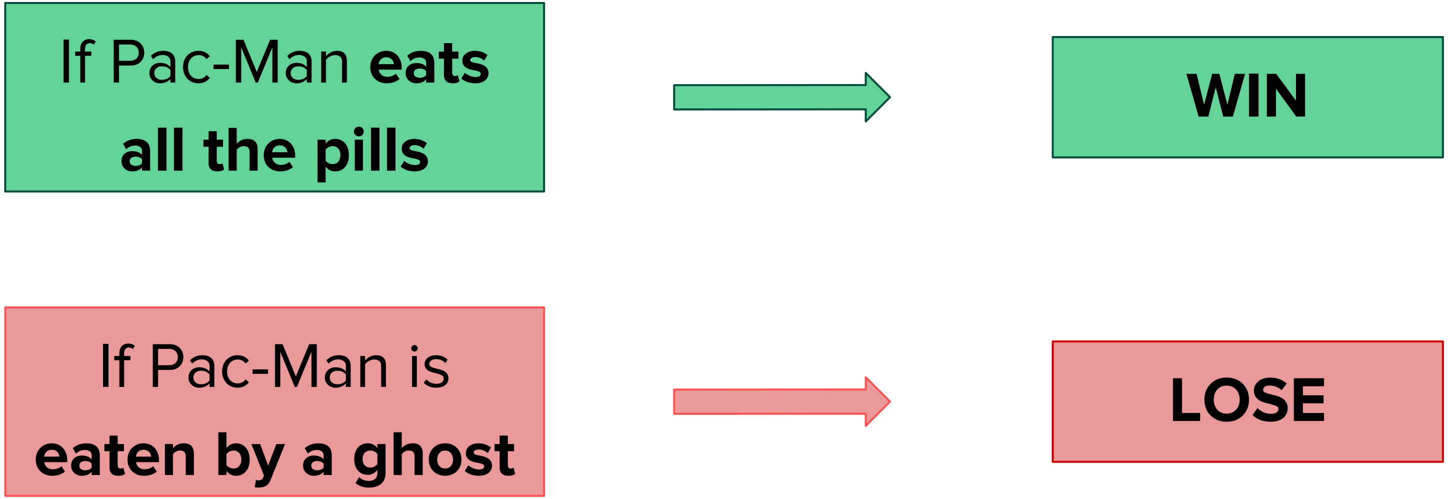
### PILLS

Needed to finish the game

### 4 POWER PILLS

If eaten, Pac-Man relative speed increases

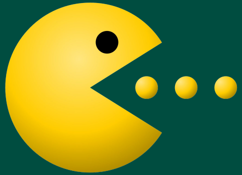
# SIMULATION OF THE GAME



If Pac-Man **eats all the pills** → **WIN**

If Pac-Man is **eaten by a ghost** → **LOSE**

# Methods



# METHODS

**GOAL** : let Pac-Man **quickly learn** how to win a game

**Neural Network**

**Reinforcement Learning**

**Q-Learning**

# REINFORCEMENT LEARNING

**KEY IDEA** : Reinforcement learning algorithms enable an agent to **optimize** its **behavior** from interacting with a specific environment

**Idea**

Reinforcement Learning is **used** to train the Neural Network on the task of playing the game of Ms. Pac-Man

# REINFORCEMENT LEARNING

ELEMENTS OF THE **RL** SYSTEM

**Model**

Simulation of the game

**Agent**

Neural Network

**Policy**

How states are mapped to actions?

**Reward Function**

**Value Function**

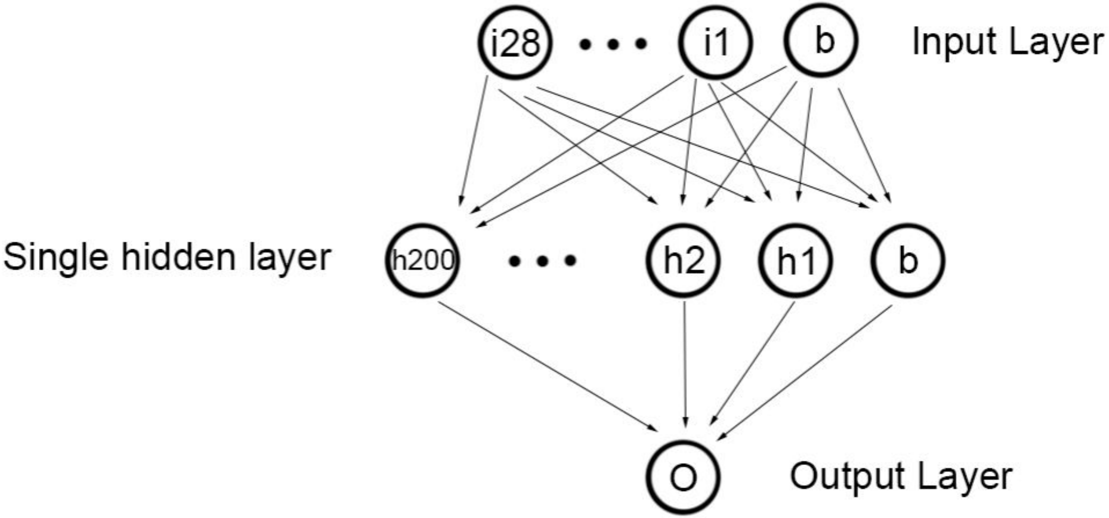
# Q-LEARNING

**KEY IDEA** : Q-Learning specifies the way in which **immediate rewards** should be used to **learn the optimal value of a state**

$$Q(s_t, a_t) \leftarrow \underset{\text{Old value}}{\boxed{Q(s_t, a_t)}} + \underbrace{\alpha}_{\text{Learning Rate}}(\underbrace{r_t}_{\text{Reward}} + \underbrace{\gamma}_{\text{Discount factor}} \max_a Q(s_{t+1}, a) - \underset{\text{Old value}}{\boxed{Q(s_t, a_t)}})$$

# NEURAL NETWORK

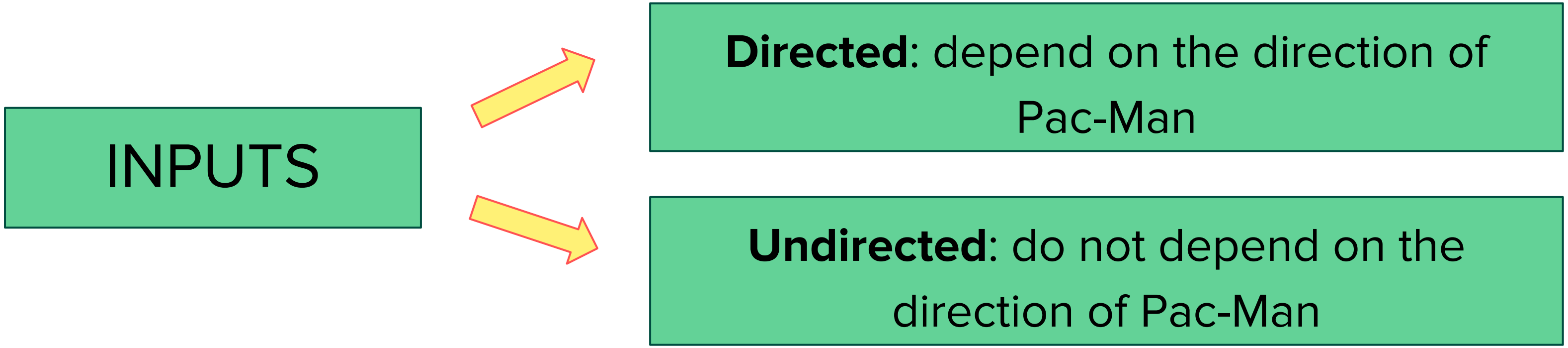
**KEY IDEA** : The Neural Network **approximates Q(s, a)**. Here **a** is the given direction of movement for pacman and **s** is the game state representation given by 28 inputs.



**200** Hidden units

**Sigmoid** activation

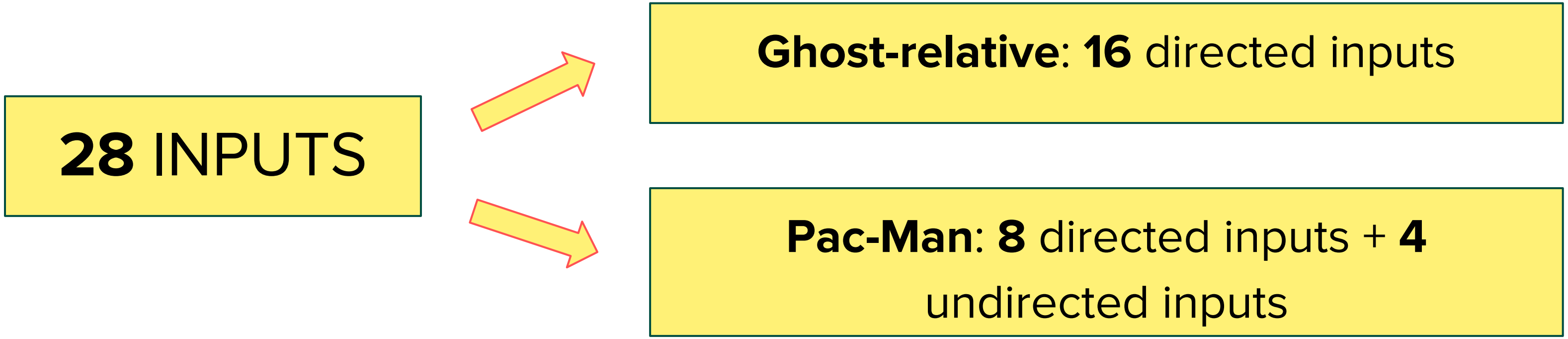
# ABOUT THE INPUTS



INPUTS

**Directed**: depend on the direction of Pac-Man

**Undirected**: do not depend on the direction of Pac-Man



**28** INPUTS

**Ghost-relative**: **16** directed inputs

**Pac-Man**: **8** directed inputs + **4** undirected inputs

# ABOUT THE INPUTS

## INPUTS OF THE GHOSTS (**4 for each ghost**)

1) Distance PacMan - Ghost (not scared & not dead)
2) Distance PacMan - Ghost (scared & not dead)
3) Pac-Man in the same direction of the Ghost
4) Number of intersections between Pac-Man and the Ghost in the shortest path

# ABOUT THE INPUTS

## **DIRECTED** INPUTS OF PAC-MAN

**1)** Pac-Main going in the same direction of previous round

**2)** Distance to the closest pill (any)

**3)** Distance to the closest power pill

**4)** Pac-Man cannot reach any intersection of distance 1

**5)** Danger level of the ghost

**6,7,8)** Number of intersections of Graph Distance 1,2,3 that Pac-Man can reach before any ghost

# ABOUT THE INPUTS

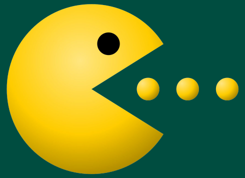
## **UNDIRECTED** INPUTS OF PAC-MAN

1) Proportion of eaten pills
2) Proportion of eaten power pills
3) Percentage of time left until the end of the effect of the power pill
4) Proportion of scared ghosts over all the ghosts

# ABOUT THE AGENT

- Obtain Q values from Neural Network in all possible directions
- Exploration Rule: Annealing exploration
- Otherwise Best Action

# Rewards



# REWARDS

| EVENT | REWARD | DESCRIPTION |
|---|---|---|
| Pill | + 9 | Pac-Man has eaten a pill |
| Power Pill | + 1 | Pac-Man has eaten a Power Pill |
| Kill a Ghost | + 2 | Pac-Man and a scared ghost have collided |
| Win | + 100 | Pac-Man has eaten all pills and power pills |
| Lose | - 400 | Pac Man and a not scared ghost have collided |
| Change direction | - 0.5 | Pac-Man reversed on his path |
| Step | - 5 | Pac-Man performed a move |

# Experiments and Results

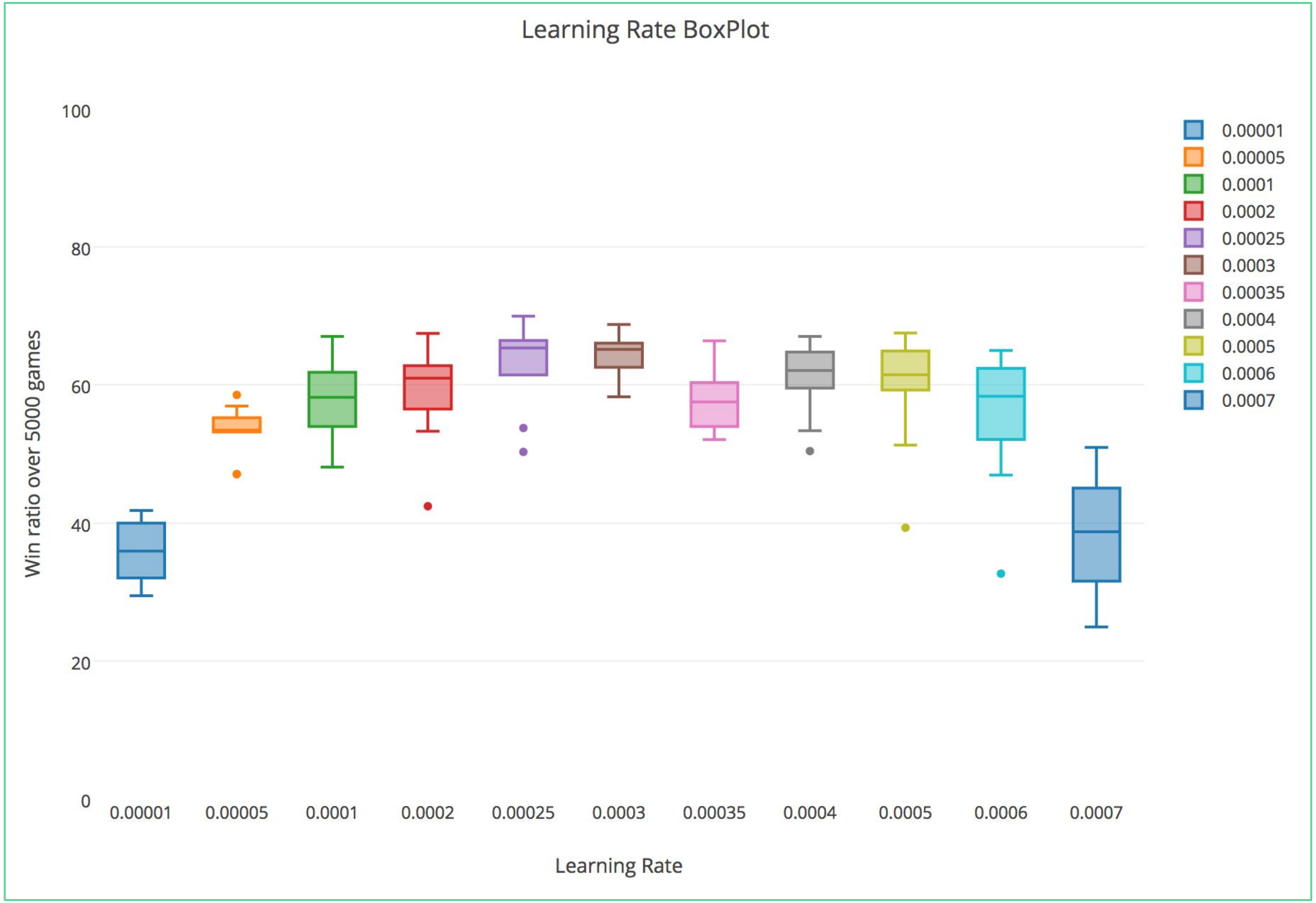
# NEURAL NETWORK TRAINING

- Training on **15000 games**

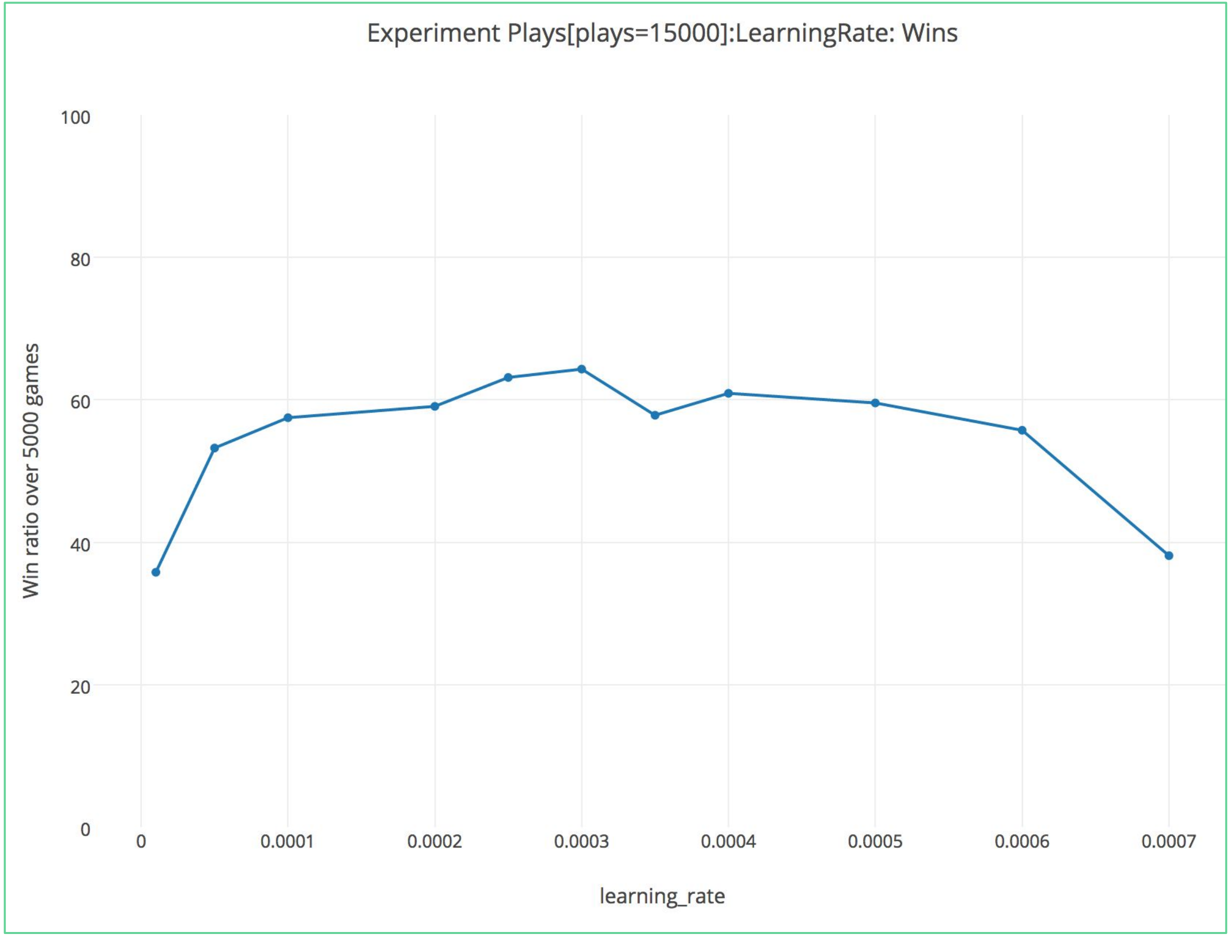
- Testing on **5000 games**

- Experiments with different settings:
  - Learning Rate
  - Discount Factor
  - Number of hidden layers/neurons
  - Reward to kill a ghost
  - Agents/Mazes

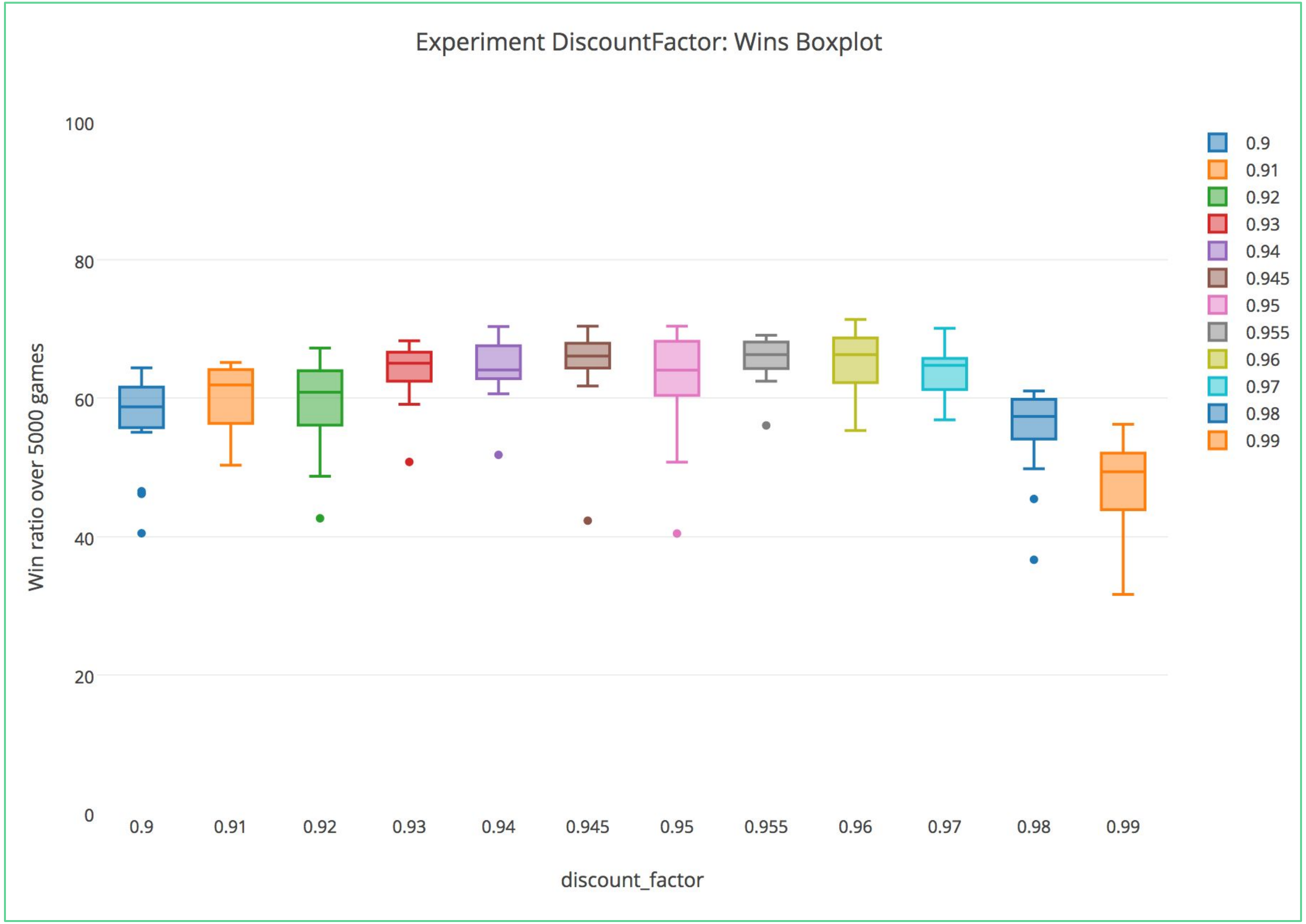
# LEARNING RATE



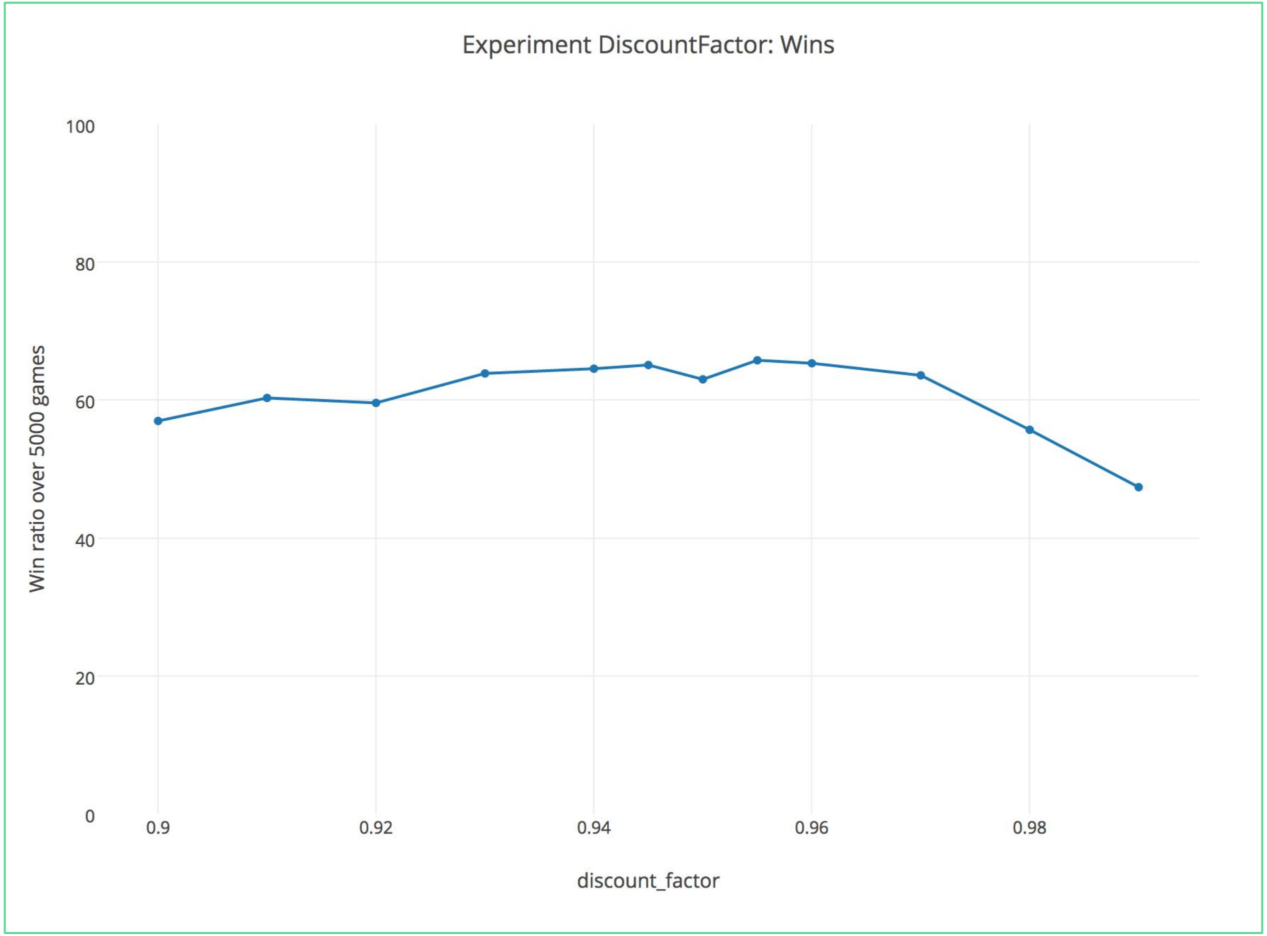
# LEARNING RATE



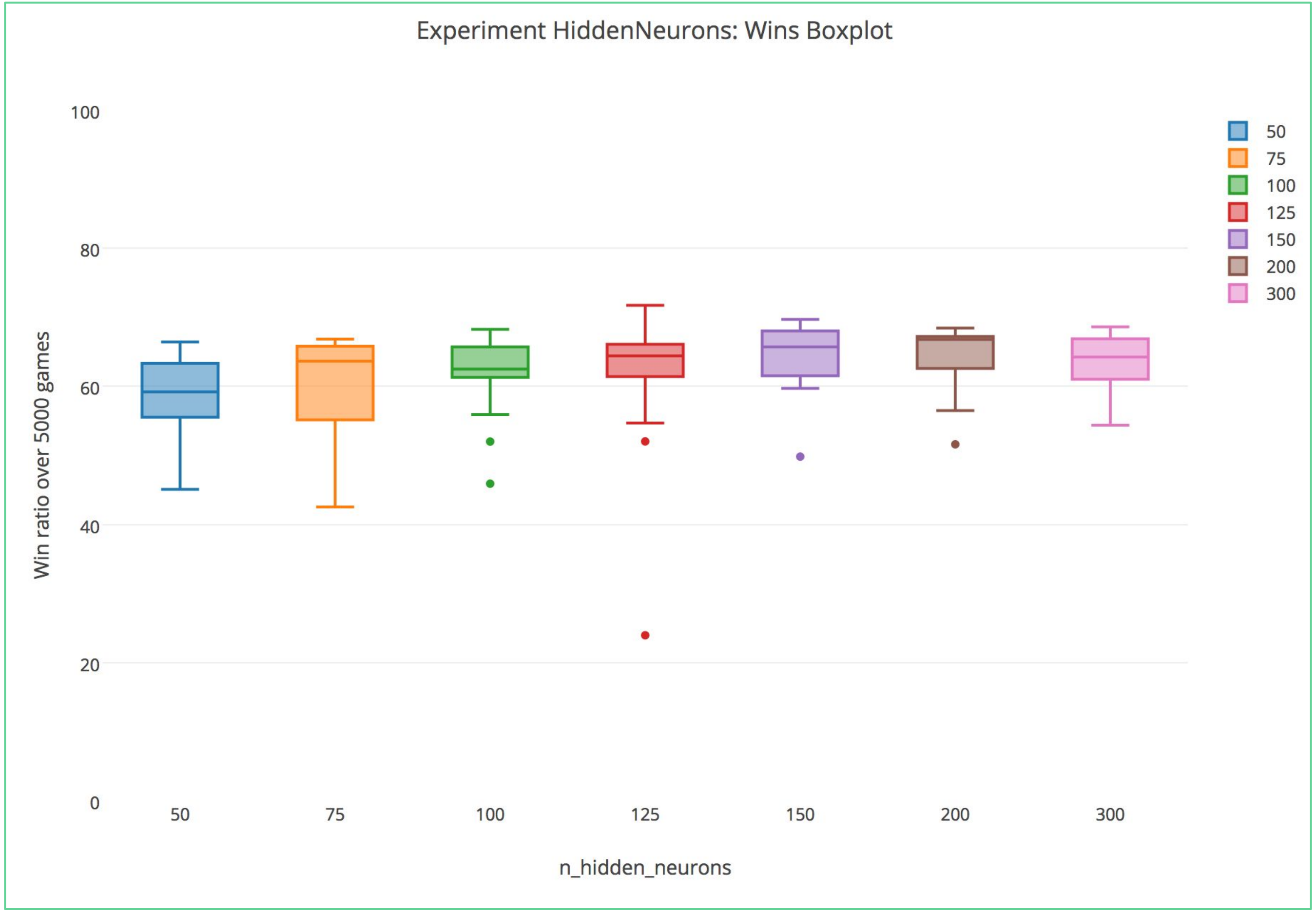
# DISCOUNT FACTOR



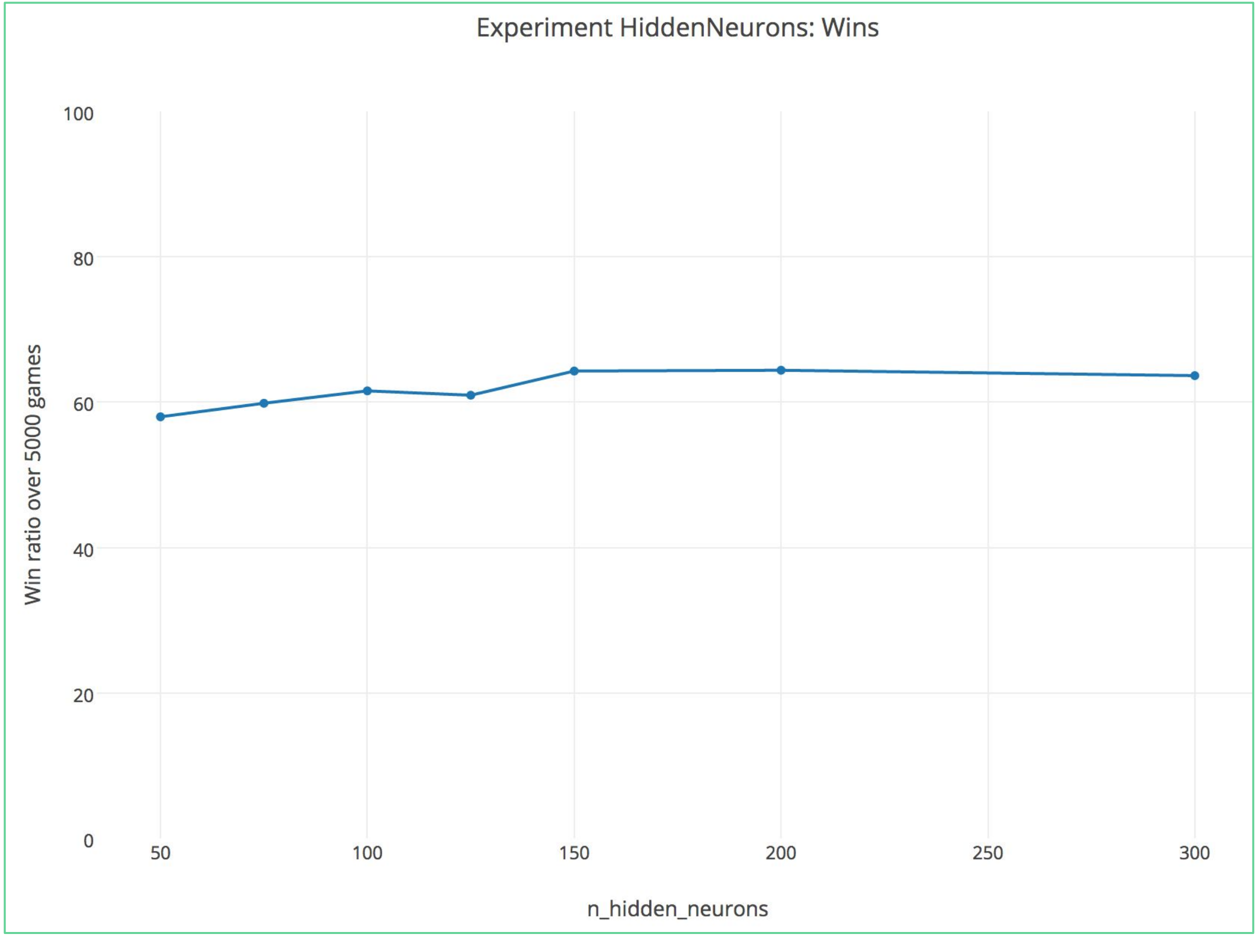
# DISCOUNT FACTOR



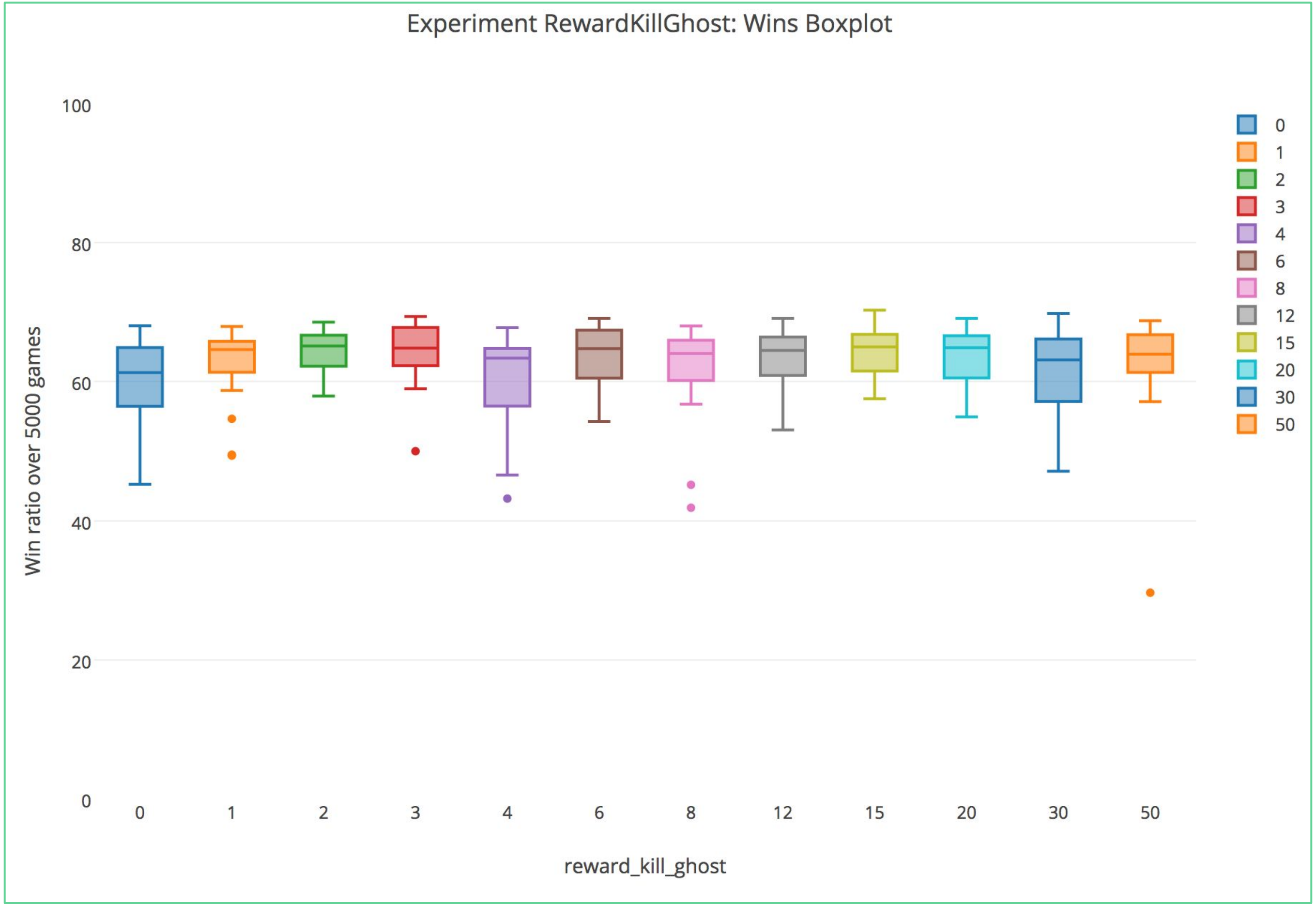
# NUMBER OF HIDDEN NEURONS



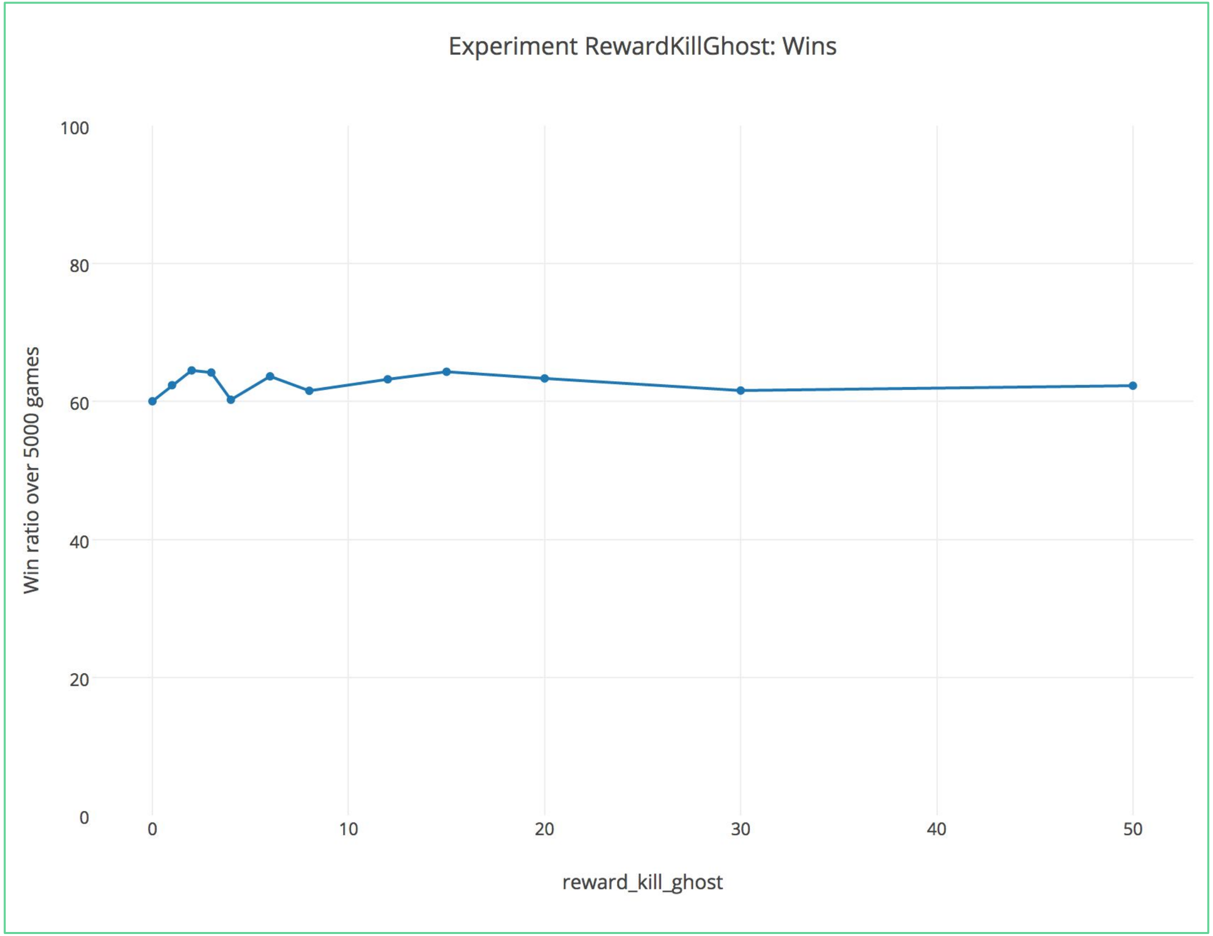
# NUMBER OF HIDDEN NEURONS



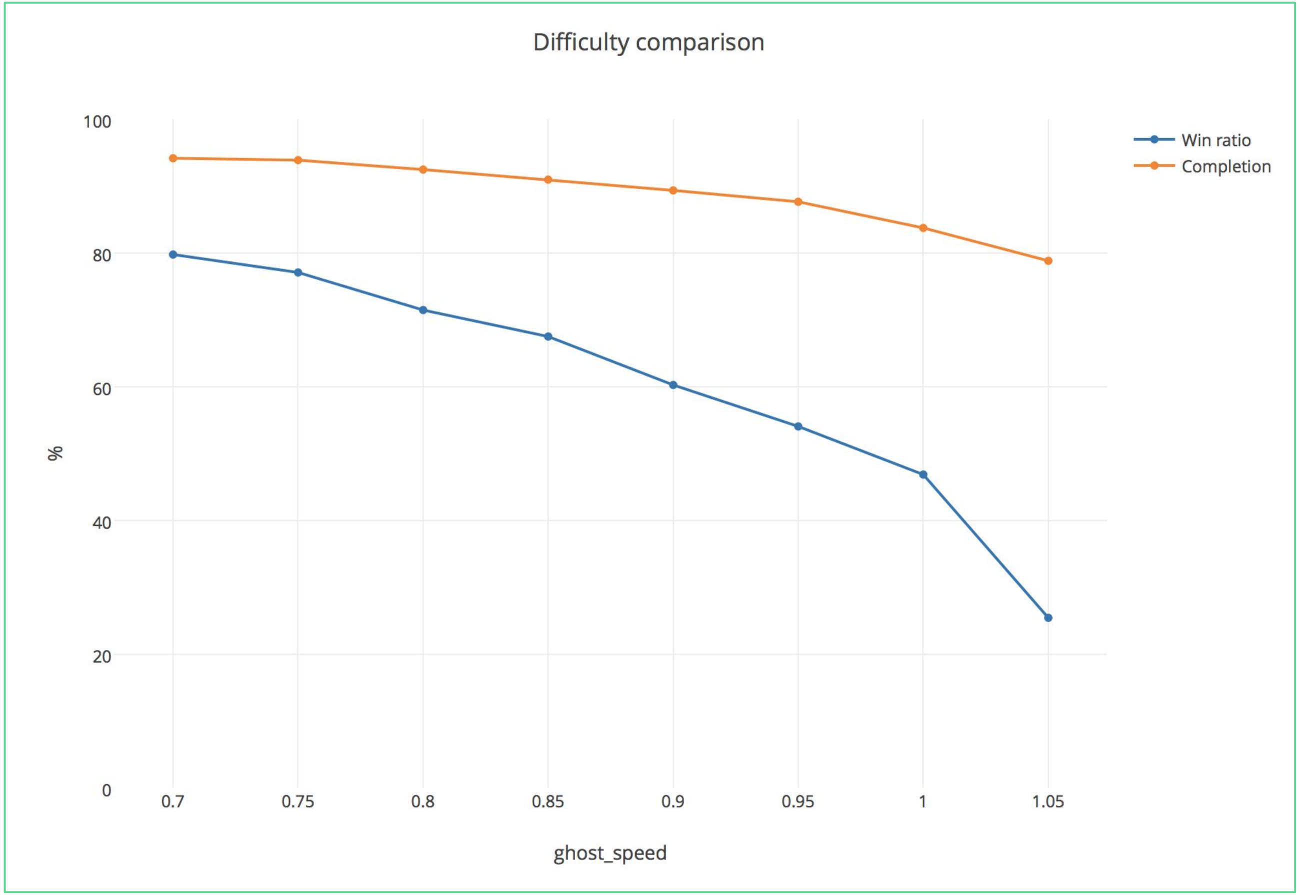
# REWARD TO KILL A GHOST



# REWARD TO KILL A GHOST

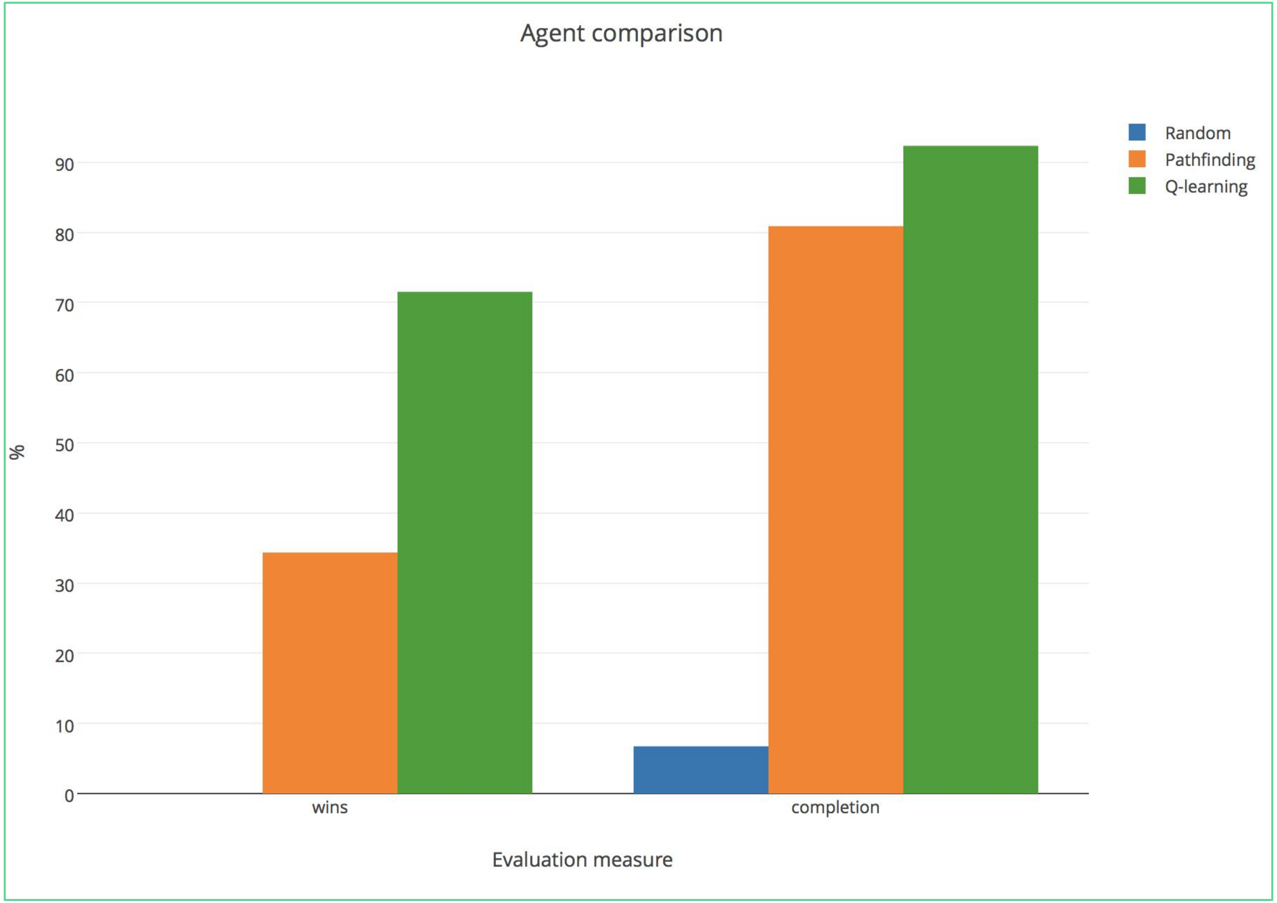


# CHANGING THE DIFFICULTY



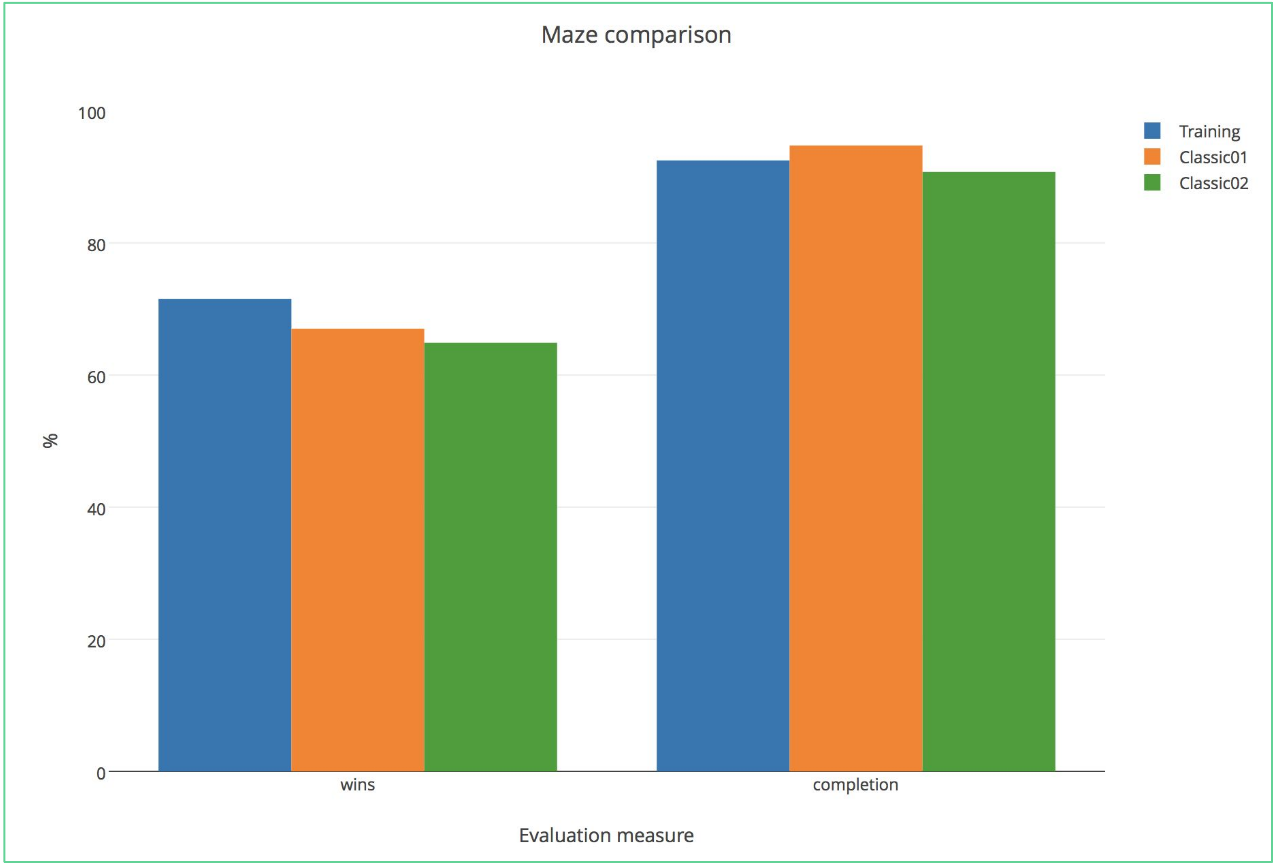
The agent **generalizes well on different difficulties**

# COMPARISON WITH OTHER AGENTS



The agent performs **better than handcrafted algorithms**

# PLAYING WITH DIFFERENT MAZES



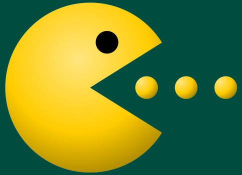
The agent **generalizes well to other mazes**

# SOME NUMBERS

Training in **less than 3 minutes**

We implemented everything by ourselves: **2900 Lines of C++ code**

# References and Repository



# REFERENCES

- Bom, Luuk, Ruud Henken, and Marco Wiering. "Reinforcement learning to train Ms. Pac-Man using higher-order action-relative inputs." *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2013 IEEE Symposium on*. IEEE, 2013.

- S. van den Dries and M.A. Wiering. Neural-Fitted TD-Leaf learning for Playing Othello with Structured Neural Networks. IEEE Journal of Transactions on Neural Networks and Learning Systems, Volume 23(11), pages: 1701-1713, 2012.

# CODE AND GAMEPLAY VIDEO

**GitHub Repository**: *http://bit.ly/2pacmancode*

**Gameplay**:

*https://www.youtube.com/watch?v=Wcs1dmcgvKY*

# Q & A

# Thank You