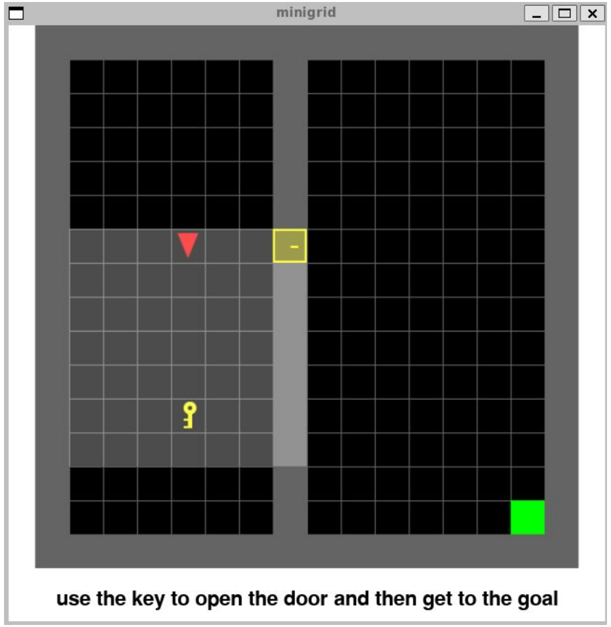
# The Algebraic Value-Editing Conjecture in Deep Reinforcement Learning

By: Eddie Wei
Mentor: Andrew Gritsevskiy

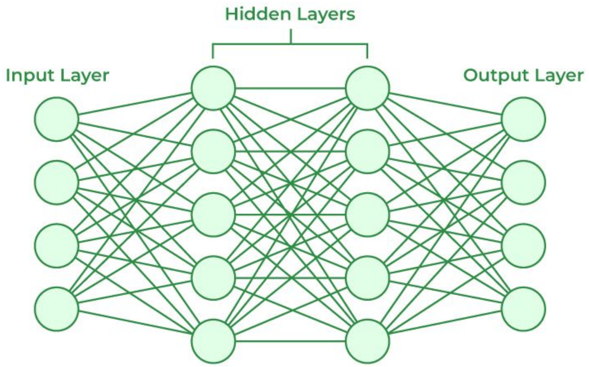
# What is Reinforcement Learning (RL)?

- Two main characters that interact:
  - **The agent**
  - **The environment**
- RL: Agents learn through trial + error
  - **State:** Entire description of the environment
  - Agent sees an **observation:** Partial description of state
  - Chooses an action by its **policy**
  - **Action Spaces:** All valid actions
  - **Return:** Cumulative “score” over a set of actions



# Deep Reinforcement Learning (Deep RL)

- **Deep** RL
  - = Deep learning + RL
  - Use **Parameterized Policies:** Determined by some complex function
- Neural networks
  - How to modify our policy so that it maximizes the expected return
  - Nodes contain an **Activation**
  - **Forward pass:** Weighted averages form the activations of the previous layers
  - **Gradient descent:** Tweaks made to the edges to optimize the policy
  - **Backpropagation:** Backward pass to compute gradient



# Background

- We’ve always wanted to understand the internal mechanisms of how the agent learns
- The math is way too complex for humans to understand: 100,000+ connections in typical models
- No intuitive concepts or patterns that were found yet...

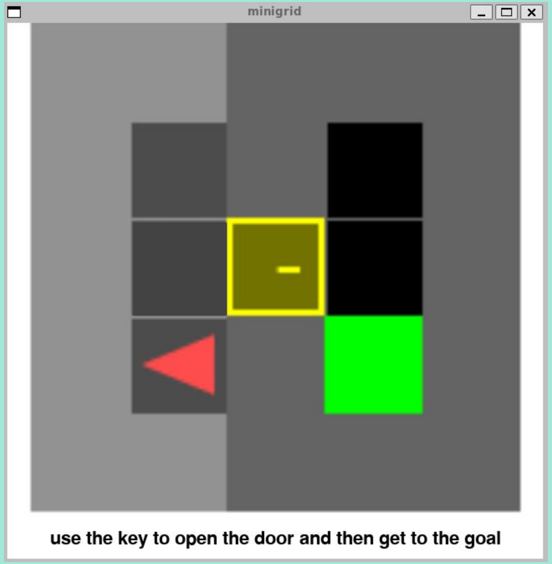
So this is a formula for calculating a single activation in a preceding layer...

$$a_0^{(1)} = \overset{\text{Sigmoid}}{\overset{\downarrow}{\sigma}}\left( w_{0,0}\, a_0^{(0)} + w_{0,1}\, a_1^{(0)} + \cdots + w_{0,n}\, a_n^{(0)} + \underset{\text{Bias}}{\underset{\uparrow}{b_0}} \right)$$

# The Algebraic Value-Editing Conjecture (AVEC)

It's possible to deeply modify a range of alignment-relevant model properties, without retraining the model, via techniques as simple as "run forward passes on prompts which e.g. prompt the model to offer nice- and not-nice completions, and then take a 'niceness vector', and then add the niceness vector to future forward passes." [1]


−




**Observation with the key**

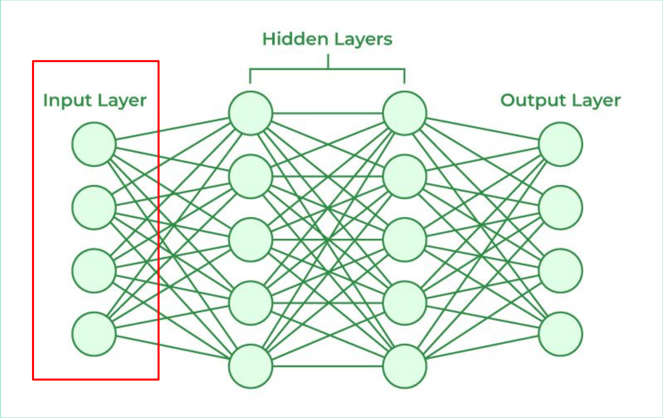
Hypothetical Activations: [3, 4, 1, 2]

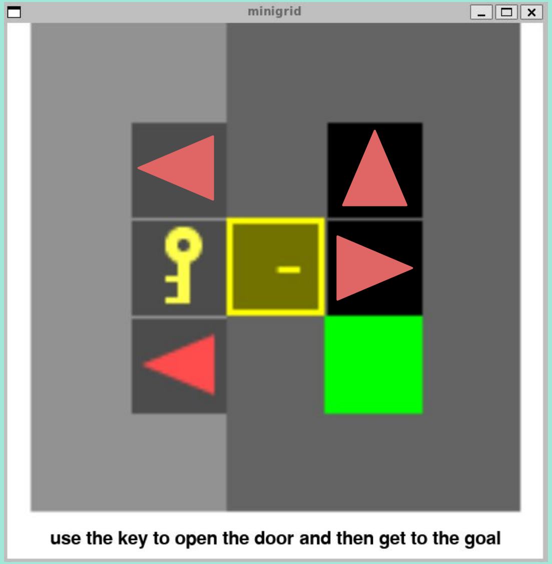
**Observation without the key**

Hypothetical Activations: [1, 1, 0, 2]

=

**Key Vector**: [2, 3, 1, 0]




The key vector **SHOULD** make the agent avoid the key at its starting position.

(Obs. with key) - (Obs. without key) = Key Activation

Obs. - Key Activation ≈ Avoid the Key

We can continue applying forward passes through the maze where our position is changed. But subtracting the key vector for our activations **still** makes the agent avoid the key. The math does not work anymore, but the relation holds!!

# Previous Papers

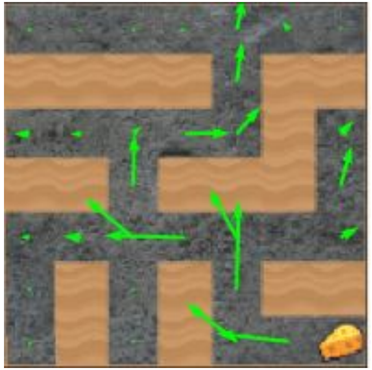
**Understanding and controlling a maze-solving policy network**

- Used the Maze environment in Procgen
- “Cheese vector” = Cheese Activations - No Cheese Activations
- Net probability vectors of the entire maze to show the effects of the cheese
- Adding cheese vector has no effect
    - Subtracting removes ability to see the cheese
    - Adding just increases “cheese perception” which is irrelevant

**Improvements**

- No addition vector found yet
- Not 100% accurate yet

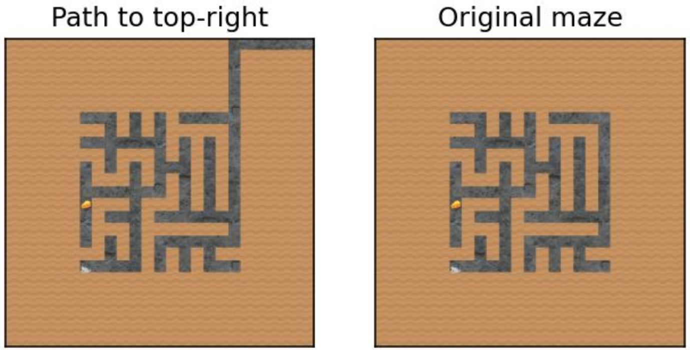

Maze Environment


Patched V-field

# Previous Papers (cont.)

**Understanding and controlling a maze-solving policy network**

- Adding a “Top-right vector”
  - Subtracting has no effect
- Effects of scaling the vector
- “Top-right vector” transfers across mazes!
- Applies to other applications other than “cheese” and top-right

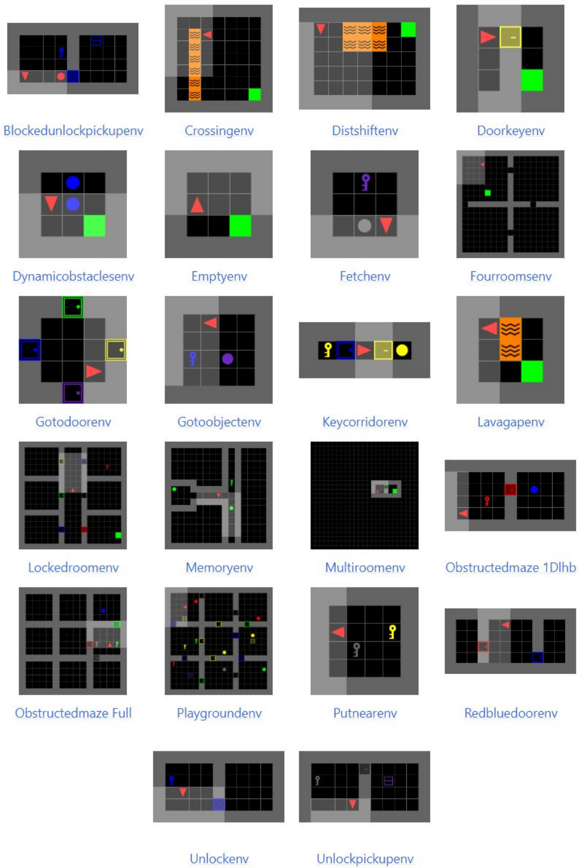



**Improvements**

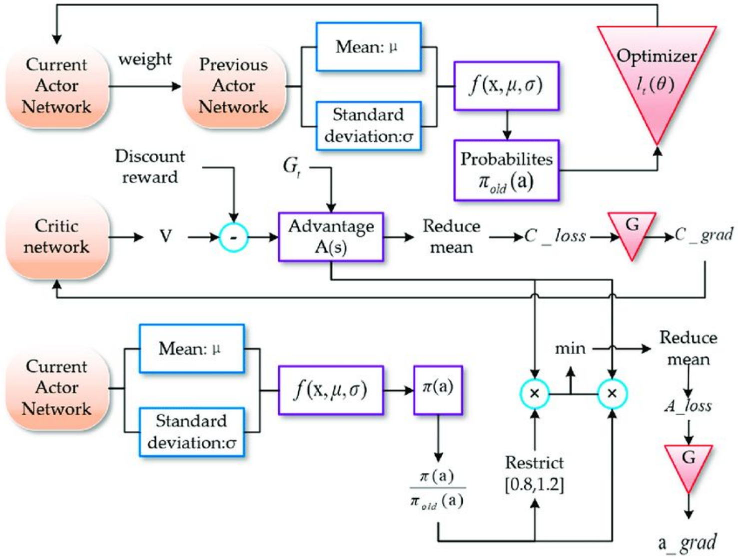
- Unsure about effects of scaling

# The Plan

- Train a PPO model on the Minigrid environment:
  - Changing map sizes
  - Easily customizable mazes
  - More complex
- Replicate the results of the conjecture...



# Proximal Policy Optimization (PPO) Model



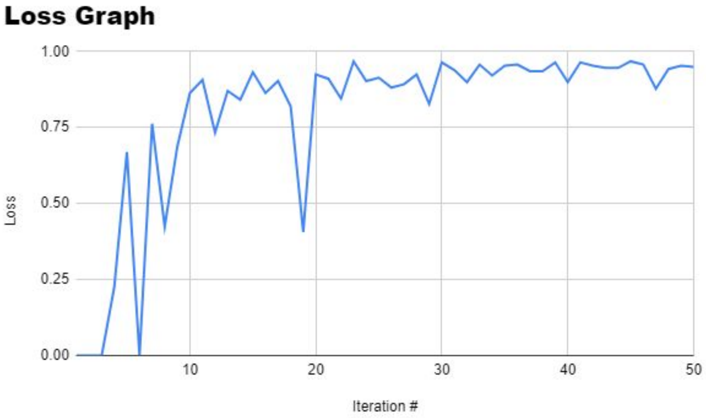

- State of the art
- Estimation of the advantage function
  - Calculates the "benefit" of particular action to average action
  - The decision maker for the agent
- Updating the policy:
  - Measures the difference between updated and old policies
  - Sampled across many small batches of **trajectories** (sequences of states & actions)
  - Stochastic gradient descent
  - Clipping mechanism

# Our Model

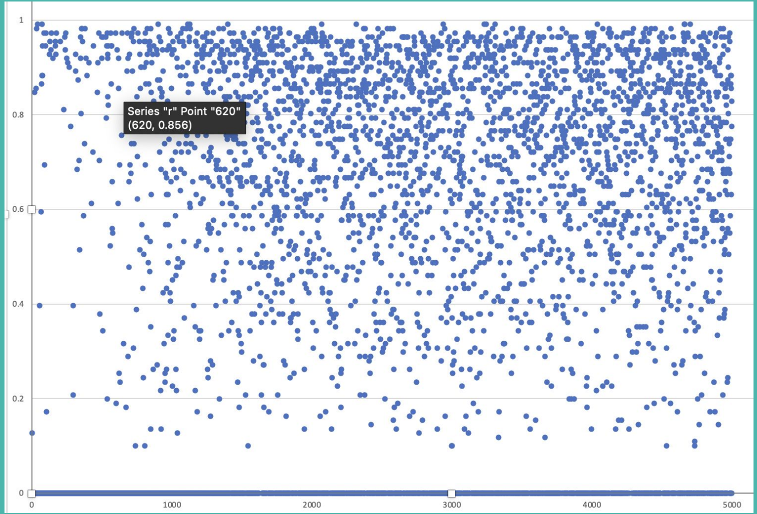
- Trained PPO model on 5x5 Minigrid DoorKey and MiniGrid-Four-Rooms environment
- Differences:
    - Limited vision
    - Key + Lock
    - Different buttons for interacting with key & lock

# Results

5x5 Grid




Four-Rooms



# Conclusion

- There are still many questions about this conjecture
- **Scaling:**
  - Big factors (>10) of the vector mess up the results
  - Small factors don't have great impacts
- **Cannot Add & Subtract the same vector:**
  - Adding the cheese vector and subtracting the top-right vector have no effect
- **The results do not generalize perfectly**:
  - Smaller seeds or complex ones tend to have different results
  - Why would this vector generalize at all anyways?

# Analysis

What if this conjecture is actually **true?**

- First insight into mechanics of neural networks and deep learning
- Massive training time save
- Applications to neuroscience: “Subtracting brain states”

Even through the internal complexities of neural network, a concept as simple as A - (A - B) = B still often seems to work!!

# Future Work

- Still need to replicate the results on our new environment
- Try different models:
  - Deep Q-Network (DQN)
  - Deep Deterministic Policy Gradient (DDPG)
  - Soft Actor-Critic (SAC)
- Test out other types of vectors other than just a key vector
  - Color vector
  - Goal vector
  - Our own “top-right” vector
- Other sub-environments in Minigrid

# Acknowledgements

I would like to thank:

- The professors of MIT PRIMES
- My mentor Andrew Gritsevskiy
- My family

# Citations


[1] Alex Turner et al. “Understanding and controlling a maze-solving policy net-work”. (2023).

[2] Alex Turner et al. “Understanding and controlling a maze-solving policy net-work”. (2023).

[3] Antonin Raffin. RL Baselines3 Zoo. https://github.com/DLR-RM/rl-baselines3-zoo.2020.