Applied Machine Learning in HOVENSTAR

Michael Yuen,
Zhewen Miao,
Yifan Meng,
Chengxiang Duan,
Mengchen Tan,
Ye Xiao

HOVENSTAR is an interstellar tactical role-playing game about resolving time travel crimes.

Hovenstar

- Developing Turn based SLG game similar to Into the Breach, Fire Emblem, and Mutant Year Zero.
- Features include two boards to fight over, Units evolving during the fight, Units with different passive abilities, campaign with different scenarios and maps.

Problem Specification

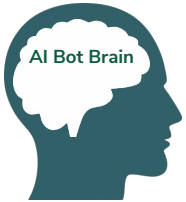
- **Reinforcement Learning**
- Focused on building a stronger, generalized AI to play against
- Similar to AlphaGo, does not use computer vision methods to train

- **Generative Adversarial Networks**
- Focused on generating new game assets for randomly generating scenarios
- Similar to Deep Fakes, generate fake images

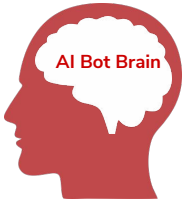
First Goal

Create an AI Bot based on reinforcement learning

- Trained by fighting against each other
- Use ML-Agents' library



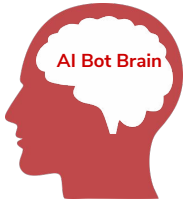
AI vs. AI for training



Training methods

- Use player inputs to train the AI.
- Use Proximal Policy Optimization.

Human Vs. AI with Different Levels



Development Pipeline

Setup Environment

AI Bot Scenario

- Build a 2 Team AI Bot scene
- Train by PPO

Multi AI Brains

- Two Regiments have different goals
-Trained by separate AI brains

Environment Features

- Add tiles of mountain, water, etc.
- different type of tiles have different buff or debuff features

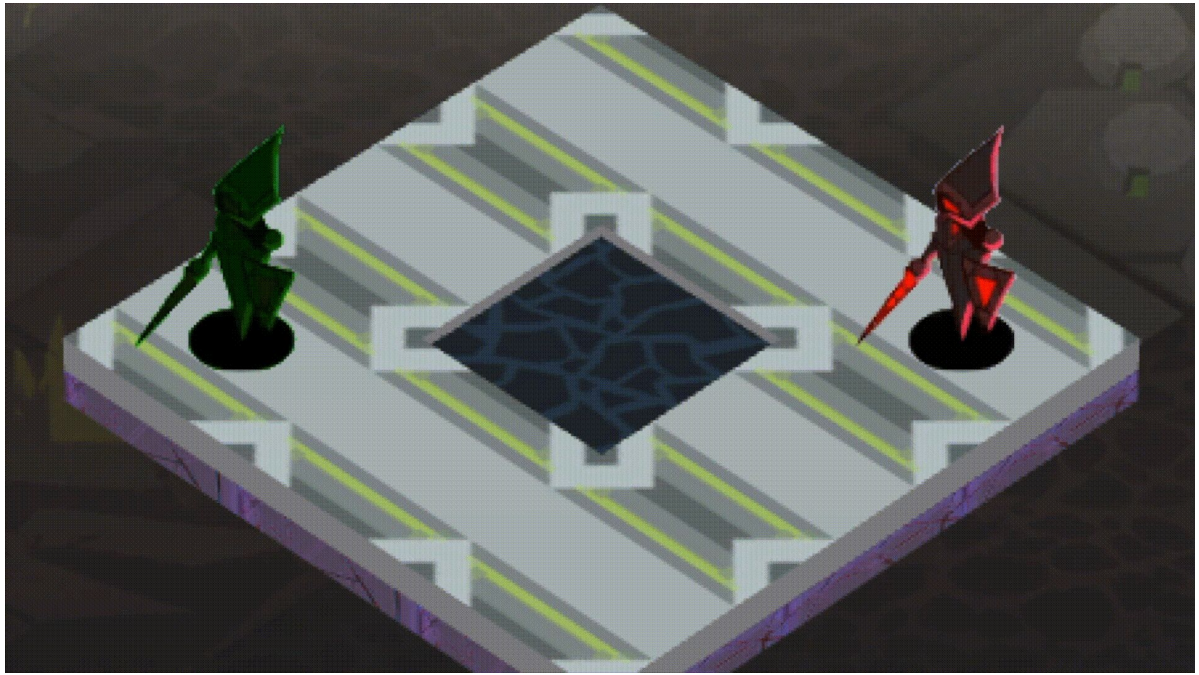
Play as a team

- Explore whether one regiment could make team decisions

Evaluate

AI Player
1 vs. 1 Scene

- One fights against the other
- Randomly move and attack, training based on PPO



Rewards

- Trained 3 AI Reward Strategies:
- Basic Strategy Rewards: +1 for win, -1 for loss
- Offensive Strategy Rewards: reward for attacking, reward for killing enemy units, punishment for taking too many turns
- Defensive Strategy Rewards: reward for not taking damage,

Final RL Model Demo

Multi AI Player 2 vs. 3 Scene

- More Units, Controlled By One Brain Each Regiment
- Larger Map Size
- Update More Observations
- Update Reward Strategies

Multi AI Player 2 vs. 3 Scene

- More Units, Controlled By One Brain Each Regiment
- Larger Map Size
- Update More Observations
- Update Reward Strategies

Another Multi AI Player Scene

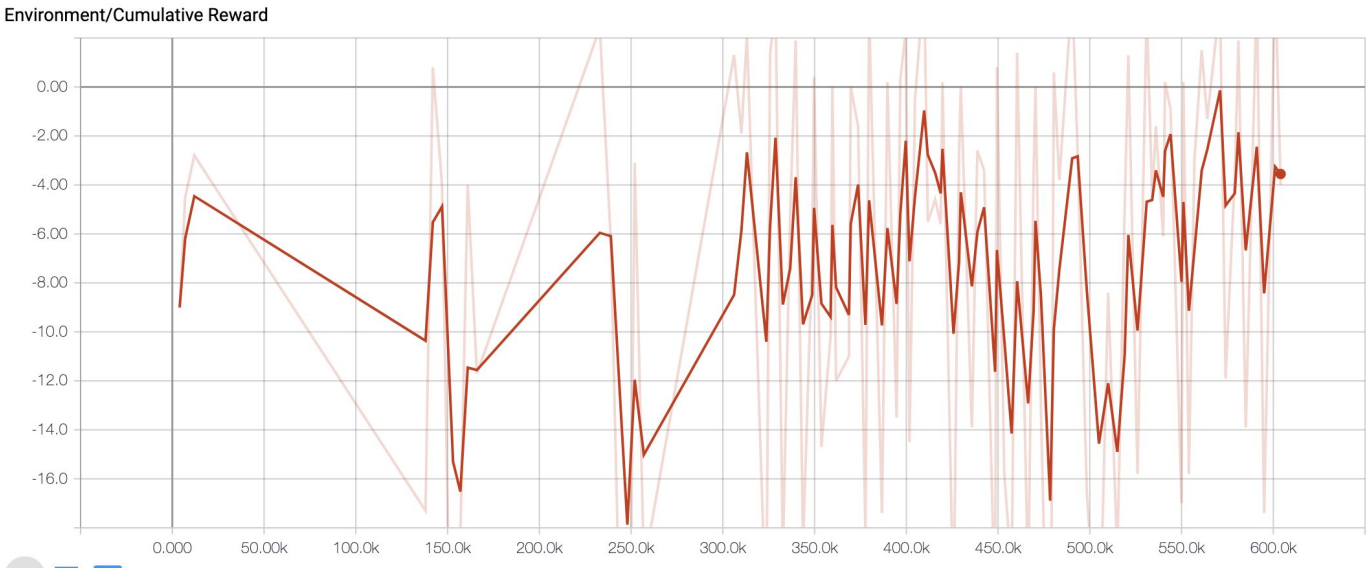
- 3 vs. 2
- Different Type of Unit in One Regiment

3 different types vs 2 Training Demo


Green
8
6
6
8
8

Statistics

Name	Smoothed	Value	Step	Time	Relative
Hovenstar-3_HovenstarRegimentBrain	-3.553	-4.000	604.0k	Tue Oct 15, 12:34:20	3h 32m 2s



Evaluation

- Compared AIs among each other and found trained AIs to typically **constantly make moves**, yet fail to finish games/kill enemies
- A unit from a regiment may attack its own regiment. After the training they **still then to kill each other** much too often, may due to lack of tuning rewards.
- No big difference among each strategy, each team and strategy has **nearly equal chance to win**
- Model knows how to move and attack but still **doesn't know an optimal way** to play the game.

Problems Encountered

- Too many dependencies, ML-Agents, Python side, and Hovenstar merged together have **bugs and code issues**
- ML-Agents and RL is better oriented for a continuous action space and real time games
- Setting up new scenarios and environments for training takes too long
- **Not enough** training or computational power
- Training the models take too long with **a vague goal**; limited numerical evaluation and large and changing action space limits progress of training

Potential Improvements

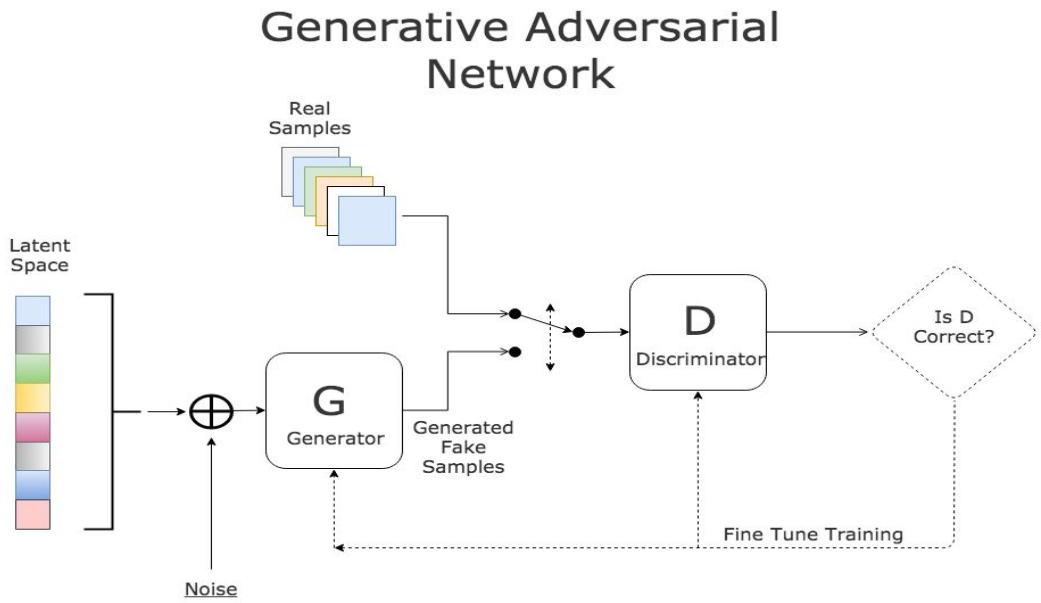
- Tune, train, and add more rewards during playing the game.

- Create curriculum learning.
Learning easy things first (valid actions, avoid friendly fire), then continue to learn harder ones.

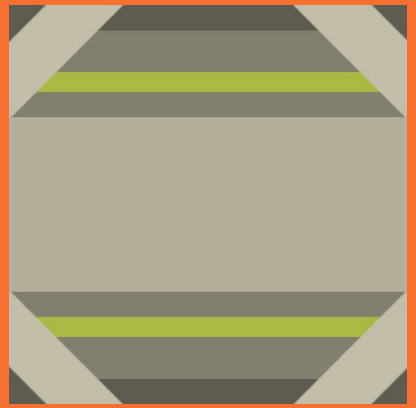
Tile Creation

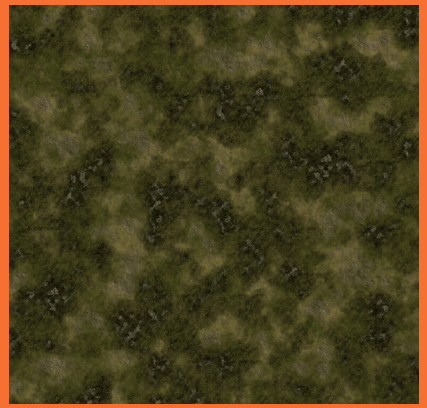
Tile Creation

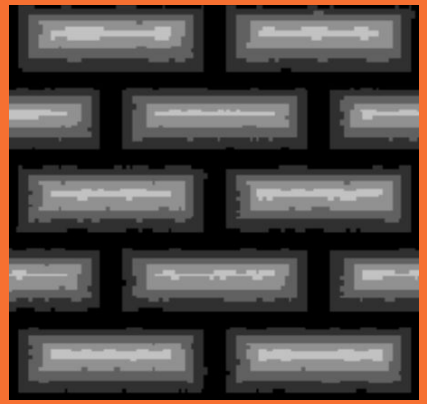
- Generate new tiles off of the tiles in Hovenstar, along with tiles from rpgmaker tilesets and unity tilesets
- Use a GAN: which has two neural networks, one for generating the image and one for discriminating real and fake tiles



Tile Data



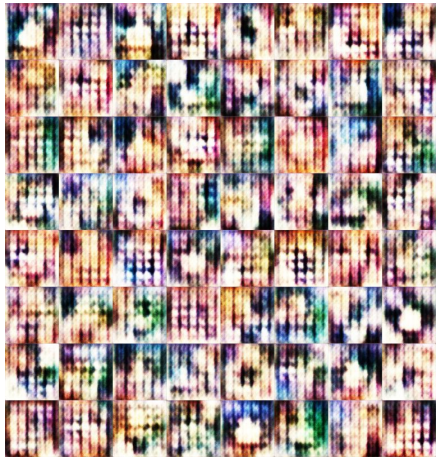




Initial Attempts

- Use Anime Character dataset for now (since tile data is not enough yet)
- Created basic Generator and Discriminator with simple NN structure (2-3 layers each)
- Ran training for 1 day

1000 iterations



5000 iterations

20000 iterations

What's Next?

- Data Gathering and Cleansing on tilesets and tile images
- Train more iterations with different GANs on the tile dataset
- Tile Stitching: try to put generated tiles together to create a usable board configuration
- Try creating units and backgrounds to use on the game
- If time permits: try creating full fledged randomly generated scenarios from GANs

Thank You!