

Open Program

Kaan G



Inhoud

Experimenting with Reinforcement Learning and CNNs	2
Training a CNN	2
Manually Labelling	2
Automatic Labelling	4
Reinforcement Learning.....	6
Extracting Game Information.....	7
Injecting Socket.....	7
Object Tracking	9
Colour Tracking	10
Results.....	11
What now?.....	13

Experimenting with Reinforcement Learning and CNNs

I Did research into RL by making smaller exercises and making use of RL. I experimented a lot with CartPole Game. All my notebooks are included for interested. The notebooks contain different approaches on training a CNN to play CartPole Game, and a notebook with RL.

Training a CNN

I thought this felt in the category Reinforcement Learning but it didn't. I tried to train a CNN to play CartPole Game without accessing the state variables. If this would work, I could try to train a CNN to play any game. I trained a CNN by collecting training data during play and train the CNN during play as well. I experimented using different amounts of frames in the frame stack that would be given to the CNN.

Manually Labelling

With a frame stack of 2 it wasn't too much effort to go through the data and label it myself. After labelling the data I tested it and it got a better average score than a random agent and a decent high score. When I inspected the gameplay of the agent I concluded that its gameplay would improve with a bigger frame stack, so I expanded the frame stack to 3

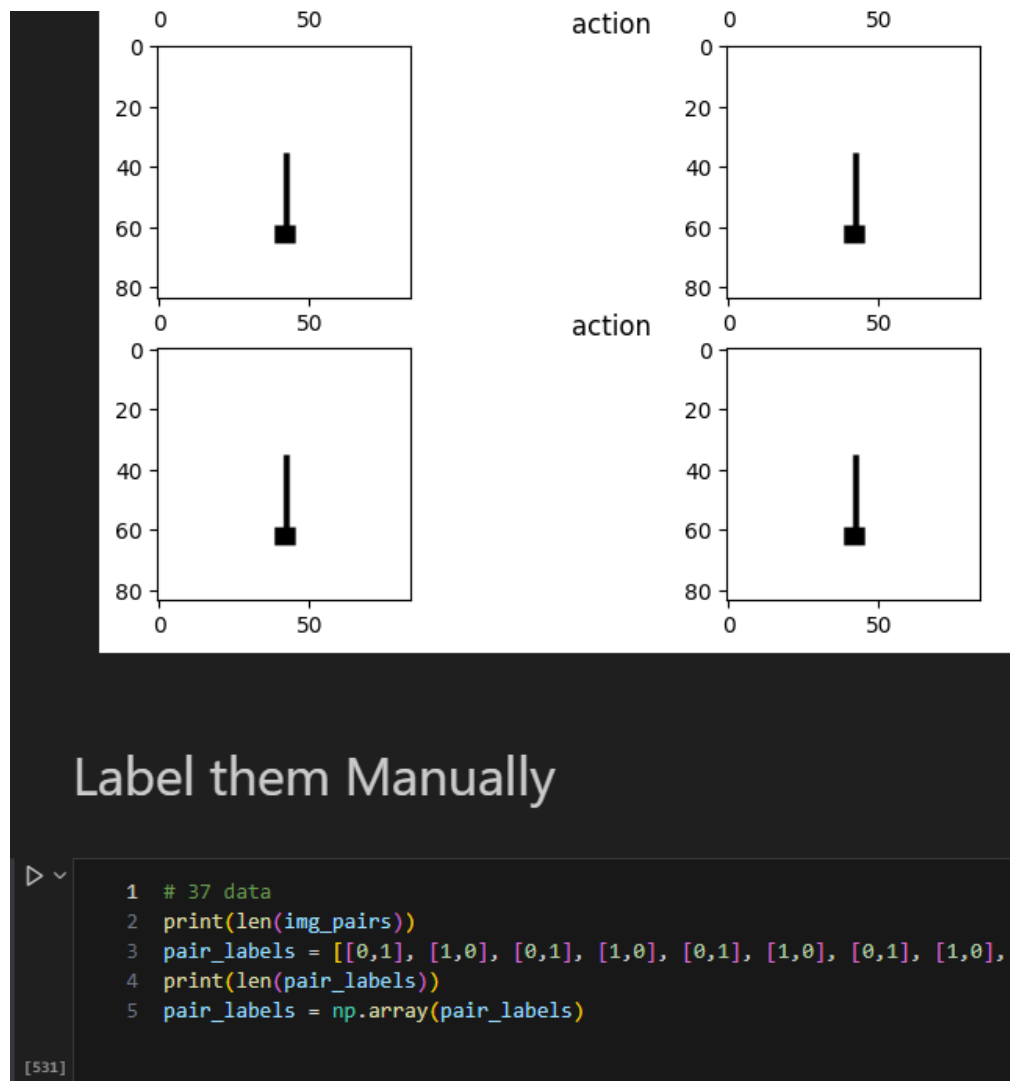


Figure 1: In the top we see a left image and right image. Left representing $t-1$ and right representing t . After inspecting the image I labelled it manually. You can see the labels in the bottom. the side the 1 is decided which direction it should go.

1 Frame results

- Trained for 40 epochs (different data split each 10 epochs)
- Tested on 100 episodes
- Average Score: 24.89
- Max Score: 50

Frame stack of 2 results

First Attempt

- Trained for 40 epochs (different data split each 10 epochs)
- Tested on 100 episodes
- Average Score: 18.65
- Max Score: 56

Second Attempt

- Trained for 60 epochs (different data split each 10 epochs)
- Tested on 100 episodes
- Average Score: 20.81
- Max Score: 55

Third Attempt

- Trained for 80 epochs (different data split each 10 epochs)
- Tested on 100 episodes
- Average Score: 29.24
- Max Score: 54

Automatic Labelling

I expanded the frame stack to 5 and let the labelling go automatically. But for some reason it didn't work. The data that got labelled automatically was good data, but during testing it would always just hold a direction and reach the minimum score. I also tried this with a frame stack of 4 but got the same results

Here is the frame stack of 5 data that got labelled automatically. The data that got labelled automatically looks perfect, but whenever I tested it, it kept getting a minimum score.

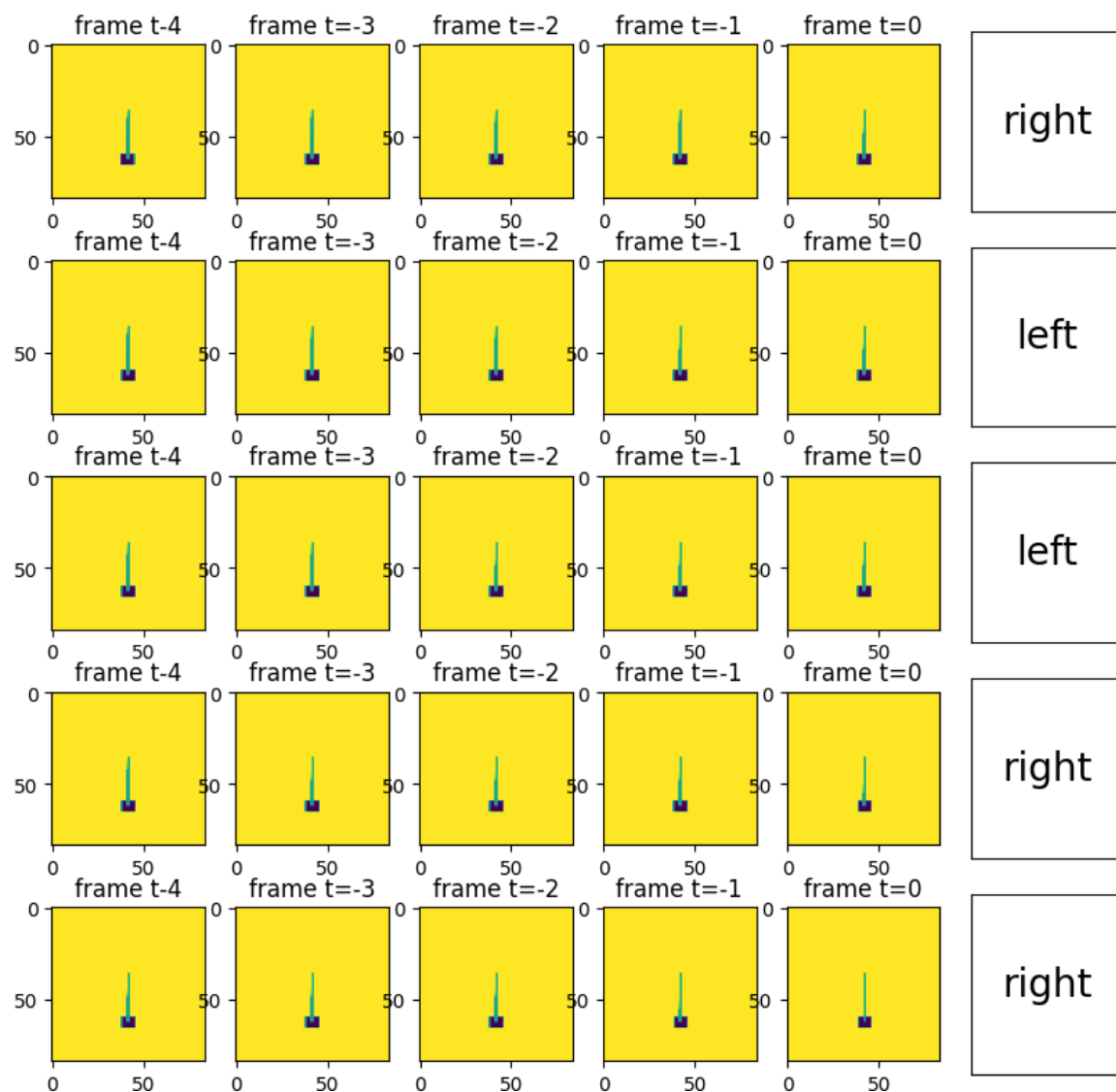


Figure 2: Frame stacks labelled automatically. This table is generated to check if my automatically labelled data is labelled correctly.

When I started on the stack or 4 frames approach I asked my RL teacher for help. You will see that on the result page of the 4 frame stack there is a lot going on, but still minimum scores.

Frame stack of 5 results

Second Attempt

- Trained for 682 epochs (episode data first 2 epochs per episode, later on 1 epoch per episode)
- Tested on 100 episodes
- Average Score: 9.3
- Max Score: 11

Third Attempt

- Trained for 1576 epochs (episode data first 2 epochs per episode, later on 1 epoch per episode)
- Tested on 100 episodes
- Average Score: 9.4
- Max Score: 11

Frame stack of 4 results

First Attempt

- Trained for 91 epochs (episode data + all earlier episodes data -> 1 epoch after every episode)
- learning_rate = 0.001
- batch_size = 32
- Tested on 100 episodes
- Average Score: 9.3
- Max Score: 11

Second Attempt

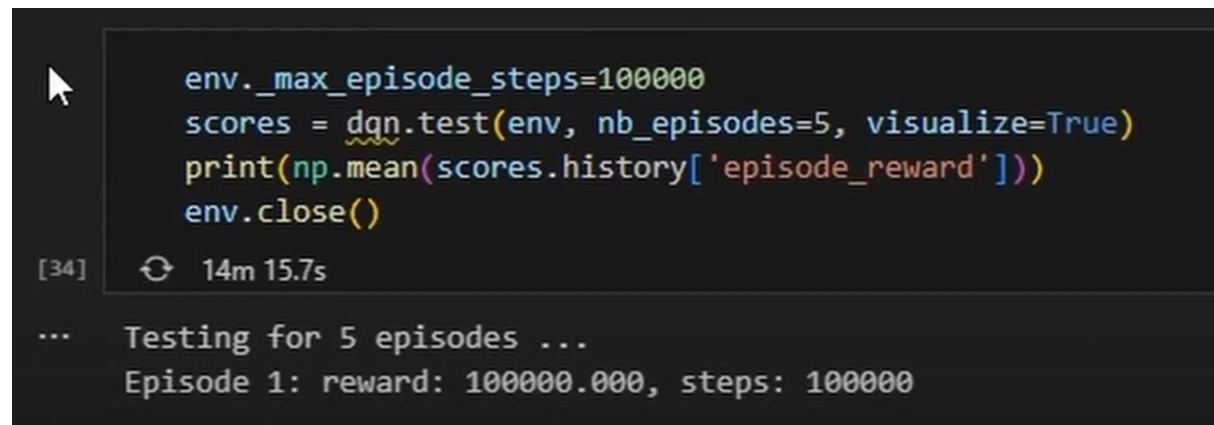
- Trained for 25 epochs (episode data + all earlier episodes data, but we take random stacks of data using train test split -> 1 epoch after every episode)
- learning_rate = 0.001
- batch_size = 32
- Tested on 100 episodes
- Average Score: 9.2
- Max Score: 11
- *Note: more complex model*

Third Attempt

- Trained for 90 epochs (episode data + all earlier episodes data, but we take random stacks of data using train test split -> 5 epochs after every episode)
- Extra data collectiong without learning
- Trained for 35 epochs (episode data + all earlier episodes data, but we take random stacks of data using train test split -> 10 epochs after every episode (basically 1k+ framestacks per epoch))
- learning_rate = 0.01
- batch_size = 32
- Tested on 100 episodes
- Average Score: 9.5
- Max Score: 11
- *Note: more complex model*

Reinforcement Learning

In the end I managed to train a DQN agent to play CartPole Game using the state variables which can get an infinite score, in other words play perfectly. This could mean that if I get my hands on the Brawlhalla state variables I could make a perfect agent for Brawlhalla as well.



```
env._max_episode_steps=100000
scores = dqn.test(env, nb_episodes=5, visualize=True)
print(np.mean(scores.history['episode_reward']))
env.close()
```

[34] 14m 15.7s

... Testing for 5 episodes ...
Episode 1: reward: 100000.000, steps: 100000

Figure 3: My CartPole Agent getting a score of 100000

Extracting Game Information

I tried to extract game information from the game in multiple ways.

Cheat Engine

I experimented with cheat engine and managed to monitor the variables I was looking for. The only problem is that I couldn't get them out of cheat engine. If I could build a socket in between cheat engine and my RL agent I could get some results.

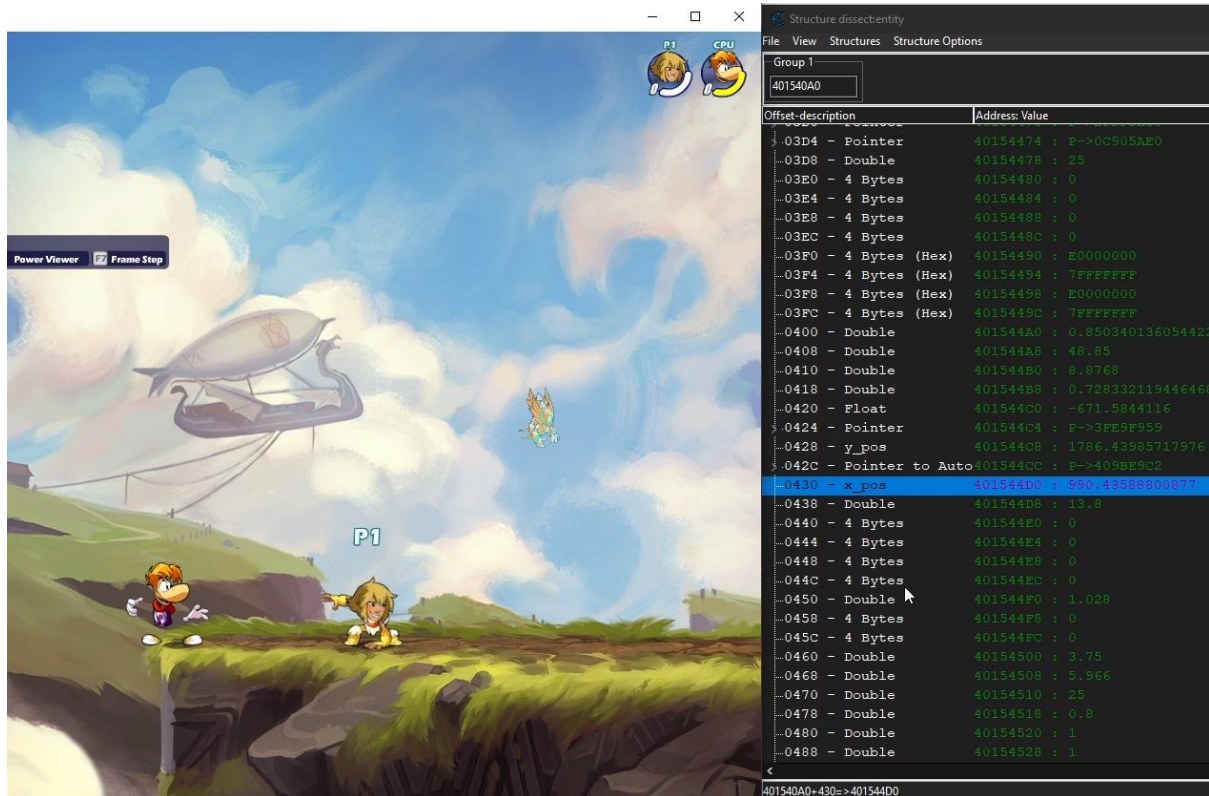


Figure 4: Marked in blue we can see the `x_pos`. This is the X position of the yellow female-like legend on the left. the `x_pos` variable updates 4 times a second.

Injecting Socket

I managed to contact one of the best *Brawlhalla* modders in the community. He managed to create a socket between Brawlhalla and a python server. After he made that he quit the project saying it would be too much effort. It was hard for me to contribute to all of this since the code was obfuscated, it was in ActionScript and I have no experience with sockets. It would be a fun project if I would have more time to work on it but it's very unrealistic to learn all of it and make it as my open program due to time constraints.



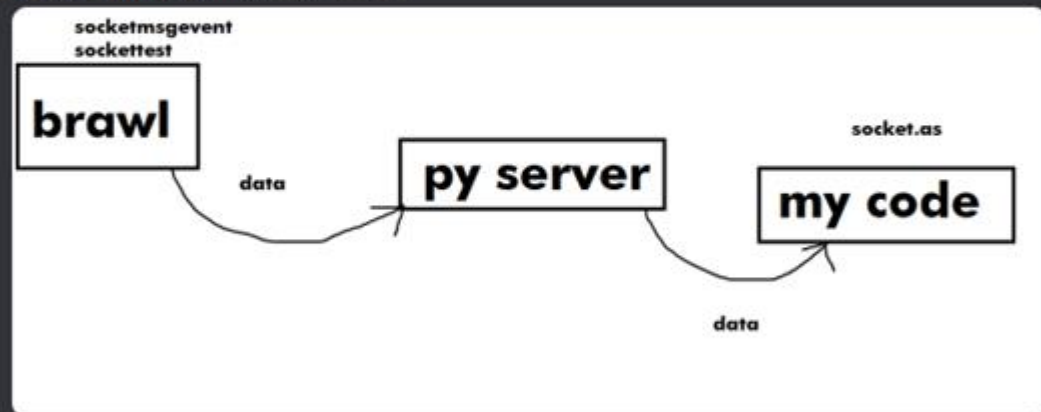
CrossyChainsaw 29/10/2023 12:09

ohhhhhhhhhhhhhhhhhhh i think i start to get it
let me try to explain to you to make sure

Ok



CrossyChainsaw 29/10/2023 12:11



and i suppose each frame data gets send
and i dont really get how my python program will receive data with socket.as
like where does it go or do i have to write a script
should i learn more about tcp (i know nothing about it) (edited)

Figure 5: The Theoretical solution to extract game variables in real time. Building a socket between a python server and the game Brawlhalla.

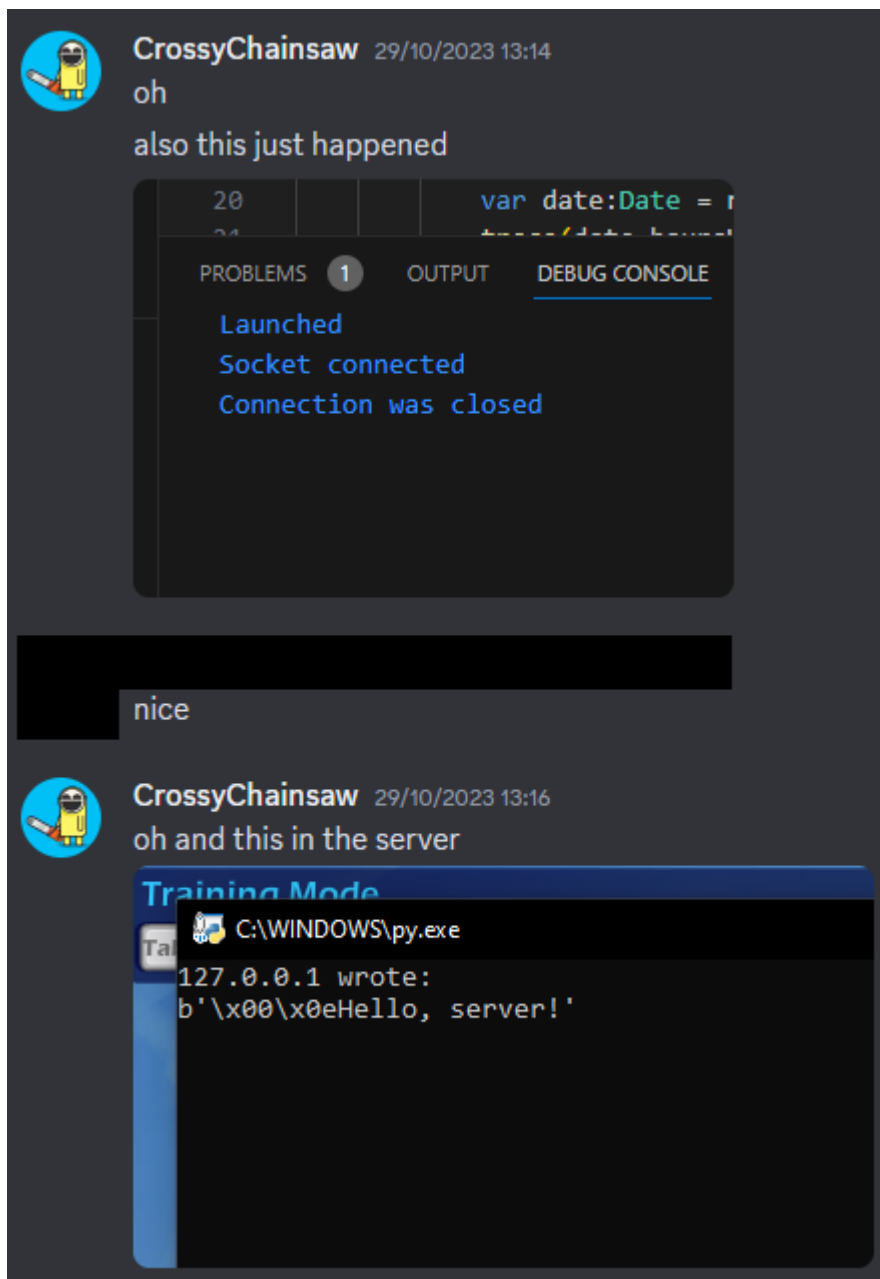


Figure 6: Running the code and connecting the Socket.

Object Tracking

After the modder dipped, I tried to continue within my capabilities. I managed to generate some sort of game variables with object tracking. In the image we see bounding boxes around everything that moves. The coordinates of these bounding boxes are printed. Theoretically we can also give these box coordinates to an RL agent, and see what it figures out.



Figure 7: Object Tracker tracking everything that moves (and has a minimum size of x). Below the gameplay you can see the coordinates of the bounding boxes.

Colour Tracking

On the Object Tracking approach I had a lot of bounding boxes. To reduce the amount of bounding boxes I came up with the idea to track a certain colour instead of everything that moves. In the image we can see the Red legend on the left of the map and a red bounding box around him. Knowing where the legend is we are playing ourselves is a small step into separating all data on screen.



Figure 8: We track the colour red and some tints of red close to plain red. On top of Orion we can see a red bounding box.

Results

I kept working on the colour tracking approach and managed to beat a Medium bot. I did this like the following. In this image we can see an easy guide the bot follows. We detect the location and depending on its location it will perform an action. Since we don't know where the enemy is, we try to stay in the centre and try to attack whenever we can.



Figure 9: The guide my bot follows

In the end my bot beat the medium bot, I made a video about my process. A lot of the code is not shown for ethical reasons.

Video: <https://www.youtube.com/watch?v=4esckiZkZu8&t=789s>

What now?

Here is the list of goals from my Open Program proposal. I put the data of completion next to it.

1. Beat an *Easy-Bot* in a 1v1 match in the game Brawlhalla using. – (30-12-2021)
2. Beat a *Medium-Bot* in a 1v1 match in the game Brawlhalla using AI – (06-10-2023)
3. Beat a *Hard-Bot* in a 1v1 match in the game Brawlhalla using AI.
4. Beat an *Expert-Bot* in a 1v1 match in the game Brawlhalla using AI.
5. Beat a *Chosen-Bot* in a 1v1 match in the game Brawlhalla using AI.
6. Beat a *Low-Diamond-Player* in a 1v1 match in the game Brawlhalla using AI.
7. Beat a *High-Diamond-Player* in a 1v1 match in the game Brawlhalla using AI.
8. Beat a *Valhallan-Player* in a 1v1 match in the game Brawlhalla using AI.
9. Beat a *Semi-Professional-Player* in a 1v1 match in the game Brawlhalla using AI.
10. Beat a *Professional-Player* in a 1v1 match in the game Brawlhalla using AI.

As you can see there is still a long way to go. Also there is still some stuff I can try. For example I didn't try building an RL Agent with plain screen input. Also, building a Socket to Brawlhalla has in my opinion potential to beat a professional player. But it will be tough.