Final Year Project Lab book

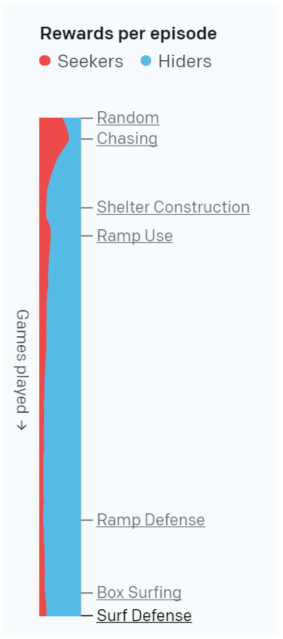
October 12th, 2021:

I had my first meeting with Prof. Kit and we discussed a couple of things regarding the project:

- The chosen game for the project should be at least a 2 vs 2 game, so that I can experiment multi-agent algorithms. Therefore, games like Chess and Connect 4 would not work.
- Students in the past have used various games such as: 2v2 football, Pacman (the monsters being the agents), as well as creating their own games.
- I am between two choices, either creating a simple game, or using a game that is already made. The advantage of creating one would be having full control of the game parameters and being able to change them when something goes wrong (e.g. when the agent breaks the game). The disadvantage is that it will be very time consuming to create a full game and might have a lot of issues compared to already tested games.
- My first task now is to look for games available for me to use and compare them, then choose the most fit game for my research.
- We agreed that there is no need for a physical lab book, since my research will be entirely on software. Therefore, this Word document will act as my lab book.
- I might be able to use some of UCL's resources such as their supercomputers, but it depends on a lot of factors. I will start using my own laptop and then see if I face issues.
- We will have a weekly meeting, next couple of weeks will be online on Zoom. Email the Prof if I don't get a zoom link to remind him.

October 13th, 2021:

- I watched this video https://www.youtube.com/watch?v=eYosANC7yeQ from Microsoft Research, about multi-agent learning for open-world games such as Minecraft, giving the public access to compete on these games by using multi-agent learning techniques.
- OpenAI's Multi-Agent Hide and Seek: https://openai.com/blog/emergent-tool-use/. Note the interesting presentation of the different stages:



- This Github repo has a list of game AI resources on **multi-agent** learning: https://github.com/datamllab/awesome-game-ai#texas-holdem-projects
- If stuck: I found a ucl module called Multi-agent Artificial Intelligence (COMP0124) taught by Professor Jun Wang. Email him if super stuck.
- Actually.. Making a simple game on python might not be a bad idea. I will start looking up how to do so.
- An interesting example: Run Forrest game https://www.youtube.com/watch?v=ZX2Hyu5WoFg

- What about 2v2 Chess???!!! I think It is an amazing Idea! Well.. yeah but you can just treat all the opponent pieces as one opponent so it's like 2v1? There is also the soccer stars game I might be able to make..

October 16th, 2021:

- I am more inclined to code my own game right now. This is because I don't know how to use already created games for my own and train ai agents within them (e.g StarCraft) as well as those games might take up too much memory or processing power to the point that they affect execution time.
- This is an interesting article on pac-man multi-agents: https://davideliu.com/2020/02/13/playing-pacman-with-multi-agents-adversarial-search/
- I am currently writing my project proposal. I don't know what to put for the specific goals and objectives in the long run (in the Gantt chart too).
- The biggest 3 ideas to code now are: Pacman, 2v2 chess, and Baloot (card game).
- I talked with Andrey as he has a similar project.

October 17th, 2021:

- I made a Gantt chart for the project proposal.

I sent an email update to Prof Kit. Content of the email: I am writing to update you on my progress and ask a couple of questions.

I have attached my initial project proposal draft to this email. Could you please give me your thoughts about it since we were not able to discuss it beforehand.

As for the project progress, I am more inclined now to create my own simple game. Mainly because I could not find open-source games suitable for multi-agent AI research, such as StarCraft (if you have any resources, I would be really grateful). In our last meeting, you mentioned a 2 vs 2 football game that was used for this type of research. Can you send me some resources related to that game?

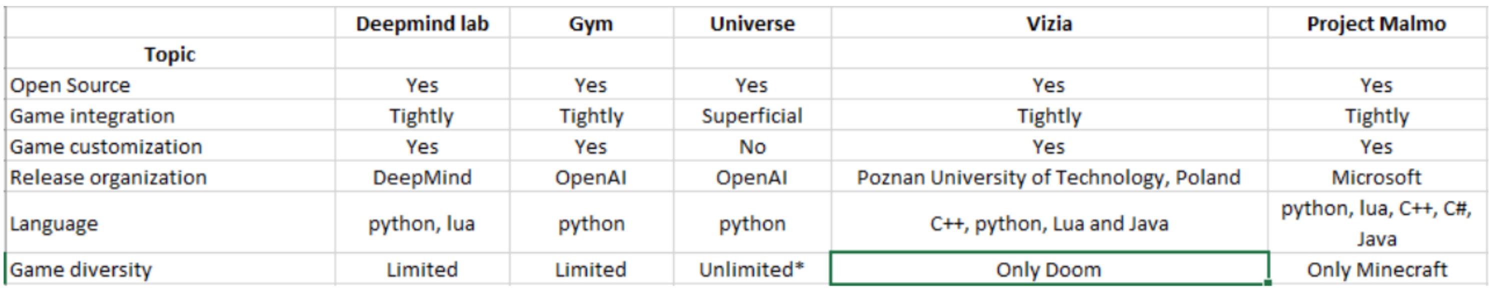
As for my own games ideas, I currently have three games in mind:

1. 2 vs 2 Chess: The issue with this game is that it might be very difficult to interpret the agents' moves and prove if they cooperate or not, since it is even more complicated than regular chess.
2. Baloot: It is a popular Arabic 2 vs 2 card game. The issue could also be that it is difficult to show agent cooperation to someone who is not familiar with the game.
3. Pac-man: Using the ghosts as agents. The issue is that it has been used before by other people so it might not be original?

October 18th, 2021:

- Prof Kit responded with: StarCraft would be way too complex for us to handle given the limited computing resources we have for the project.
  Don't worry too much about originality at the moment as novelty can be introduced at a later stage.
  Pac man and soccer games seem to be good choices although I am sure there can be many other equally suitable games.
  In previous years, some students used the multi-agent environment developed by Google DeepMind using MuJoCo.
- Interesting video on MuJoCo: read the paper associated with it.
  https://www.youtube.com/watch?v=KHMwq9pv7mg&t=0s
- I found this OpenAI "gym": https://gym.openai.com/ "We provide the environment; you provide the algorithm."
- A list of opensource AI RL platforms: https://www.analyticsvidhya.com/blog/2016/12/getting-ready-for-ai-based-gaming-agents-overview-of-open-source-reinforcement-learning-platforms/
  So the main five platforms are (the rest are notable mentions):
  1. DeepMind Lab
  2. OpenAI Gym
  3. OpenAI Universe
  4. Project Malmo
  5. VizDoom

6. RL-Glue
7. CommAI
8. Burlap
9. rlenvs



Topic	Deepmind lab	Gym	Universe	Vizia	Project Malmo
Open Source	Yes	Yes	Yes	Yes	Yes
Game integration	Tightly	Tightly	Superficial	Tightly	Tightly
Game customization	Yes	Yes	No	Yes	Yes
Release organization	DeepMind	OpenAI	OpenAI	Poznan University of Technology, Poland	Microsoft
Language	python, lua	python	python	C++, python, Lua and Java	python, lua, C++, C#, Java
Game diversity	Limited	Limited	Unlimited*	Only Doom	Only Minecraft

October 20th, 2021:

- Since VizDoom and Project Malmo only play single games, they will not be considered. This is because their games are not suitable for the project. We need a multiplayer game that is simple enough for our computers to handle. Such as the football game from DeepMind Lab.
- DeepMind Lab open-source repo: https://github.com/deepmind/lab "Disclaimer: This is not an official Google product."
- OpenAI Gym open-source repo: https://github.com/openai/gym
- Getting Started with Gym: https://gym.openai.com/docs/ This is a really good starting point to try out OpenAI Gym.
- OpenAI Universe is essentially an extension to OpenAI gym, with support for literally "anything" you can do on a computer. Its repo: https://github.com/openai/universe
- Did old video games use AI? I always ask myself this: https://en.wikipedia.org/wiki/Artificial_intelligence_in_video_games#
  "Many experts complain that the "AI" in the term "game AI" overstates its worth, as game AI is not about intelligence, and shares few of the objectives of the academic field of AI. Whereas "real AI" addresses fields of machine learning, decision making based on arbitrary data input, and even the ultimate goal of strong AI that can reason, "game AI" often consists of a half-dozen rules of thumb, or heuristics, that are just enough to give a good gameplay experience.[citation needed] Historically, academic game-AI projects have been relatively separate from commercial products because the academic approaches tended to be simple and non-scalable. Commercial game AI has developed its own set of tools, which have been sufficient to give good performance in many cases.[2]

October 23rd, 2021:

- Ask Prof. Kit: Feedback on project proposal and the progress reports so far. Should I change how I write them?
- There is a big issue. Deepmind Lab and OpenAI Gym do not support windows! I can either choose them and use the Linux computers in the lab, or find another platform, or somehow use it on windows. OR, I could use dual-booting? Which means having both Linux and windows on my laptop?
- "**Always install Linux after Windows**".
- I think the best way to go is to install Linux and continue working on my laptop.

October 26th, 2021:

- Now that my SSD has arrived, I can back up my laptop before installing Linux.

October 27th, 2021:

- I installed Linux on my laptop alongside Windows. I am getting familiar with it as it is my first time using it.

October 29th, 2021:

- This is the official page for Mojoco football and it has links to Github and its paper. https://deepmind.com/research/open-source/mujoco-soccer-environment
- Found another game! Google Research football: https://github.com/google-research/football
  I found it through this 2 hour video: https://www.youtube.com/watch?v=8TMT-gHlj_Q
- Google game: "the Football Engine is out of the box compatible with the widely used OpenAI Gym API."
- Wow: https://www.youtube.com/watch?v=F8DcgFDT9sc&t=43s you can play against your own agent.

October 31st, 2021:

- I had an issue with running python from the command prompt, so I fixed it in order to pull repositories from Github and try Google Research football.
- I faced issues loading Google Research football game into Windows. Therefore, I will be using Linux.
- The game works on Linux!
- So far, this seems like the most supported game. However, is it suitable for my research? It is 11 vs 11 football, does that mean the multi-agent RL algorithms might be too complicated for my own research?
- Bad sound quality presentation on Google Research Football:
  https://www.youtube.com/watch?v=lsN5y2frNig
  Same presentation but shorter and better quality: https://www.youtube.com/watch?v=Va5dIxejqx0
- I tried to run Mojoco Soccer game on python.

November 1st, 2021:

- I had a meeting with Prof. Kit. Updated him on progress so far. "11 vs 11 might be a bit too much to train AI agents, as it might take weeks or months. They usually have hundreds of computers to train those agents. The most important aspect of a game is how much control you have over it." If I can change the number of players, which I think I can, then it will be ok!
- Two minute paper video on GRF: https://www.youtube.com/watch?v=Uk9p4Kk98_g most important thing to note: in order to train AI to play against the easy difficulty, you can use a single machine. But against the hard difficulty, arrays of machines were used! Which I think is a red sign for me. I either have to just train against the easy option, or use resources in UCL (their computers) or choose another game!
- Youtube playlist on GRF (tutorial):
  https://www.youtube.com/watch?v=SWllbdcrKLI&list=PL3YDJVV_1t7sKR_ZrO1Ursa_c41I0ErzC
  blog version: https://towardsdatascience.com/proximal-policy-optimization-tutorial-part-1-actor-critic-method-d53f9afffbf6
  Github: https://github.com/ChintanTrivedi/rl-bot-football

November 2nd, 2021:

- I found that you can pick different scenarios for the GRF game, meaning that it doesn't have to be a normal 11 vs 11 match. Here is a link to the scenarios available:
  https://github.com/google-research/football/blob/master/gfootball/doc/scenarios.md

  "You can add your own scenarios by adding a new file to the gfootball/scenarios/ directory. Have a look at existing scenarios for example."
- Blog listing various RL algorithm:
  https://jonathan-hui.medium.com/rl-deep-reinforcement-learning-series-833319a95530
- Blog on training RL algorithm in GRF:
  https://towardsdatascience.com/google-football-environment-installation-and-training-rl-agent-using-a3c-d058a44f0fad

November 4th, 2021:

- Best Benchmarks for Reinforcement Learning: The Ultimate List:
  https://neptune.ai/blog/best-benchmarks-for-reinforcement-learning
- RL benchmarks with multi-agent:
  1. Google Research Football – Multi-task; Single-/Multi-agent; Creating environments;
  2. Meta-World – Meta-RL; Multi-task;

3. Multiagent emergence environments – Multi-agent; Creating environments; Emergence behavior; e.g Hide & Seek: https://github.com/openai/multi-agent-emergence-environments
4. OpenSpiel – Classic board games; Search and planning; Single-/Multi-agent;
5. RLCard – Classic card games; Search and planning; Single-/Multi-agent;
6. StarCraft II Learning Environment – Rich action and observation spaces; Multi-agent; Multi-task;
7. The Unity Machine Learning Agents Toolkit (ML-Agents) – Create environments; Curriculum learning; Single-/Multi-agent; Imitation learning;

- in case I cannot work on GRF on my laptop, can't I use computing clusters at UCL? Would that be helpful?
- Slides about Multi-agent: https://www.karltuyls.net/wp-content/uploads/2020/06/MA-DM-ICML-ACAI.pdf

November 5th, 2021:

- I tried so many times to run Mujoco soccer but I had issues with importing libraries and library paths, so I will not spend more time on it and I will continue working on GRF.
- Currently facing issues with Library paths in Linux. It does not let me import libraries properly and I cannot progress on either Mujoco or GRF!

November 6th, 2021:

- I have an idea for the scenarios: Check out the scenarios folder, try to understand how they are made. Then check the scenario of the normal 11 vs 11 game, (if it exists as a scenario), then just try to change the number of players!
- Try to run GRF on Windows today! This is to avoid the Library Path issue in Linux which wasted me some time.

November 10th, 2021:

- GRF still doesn't work due to tensorflow! I get so many error messages to the point that I cannot make any progress.
- it finally works! After so many hours of troubleshooting.

November 11th, 2021:

- I emailed UCL to ask if I am eligible to use the computer clusters for research.
- Very good presentation on GRF! https://www.youtube.com/watch?v=esQvSg2qeS0
  (min 5): you don't have to solve 11 vs 11 setup. You can do 4 vs 4!!
  (min 9): controller is used instead of keyboard.
  (min 25): students used 5 vs 5 for multi agent! This is their blog post!! Good place to start:
  https://sites.google.com/view/rl-football/zpp
- With this info, I can confirm now that GRF is suitable for my research! Even if it is difficult, I am here to learn and I want to push myself to do the best possible.

November 12th, 2021:

- I have to resize my Linux partition/disk as it is almost full.
- GRF Kaggle competition 3rd place presentation:

https://www.youtube.com/watch?v=MmVcdfeRYD8&t=2625s
(min 43) shows links to other competitor's solutions.

November 13th, 2021:

- I re-installed Linux as some issues appeared, and to resize the partition.

November 16th, 2021:

- I am installing Pycharm and the required packages again.
- I think my laptop's GPU is not suitable! It says that it is not CUDA-capable, which means I can't use my laptop? It is weird because I can run the game.
- My GPU is not from NVIDIA that is why, I think.
- There is a solution! I can use UCL EEE's GPU resources which are from NVIDIA

- UCL EEE email content:

There are a number of resources you can use for compute (CPU only) simulations:

https://intranet.ee.ucl.ac.uk/it/servers/compute

For GPU based simulations you can use the resources listed at:

https://intranet.ee.ucl.ac.uk/it/servers/gpu

I've had a quick look at the google football research pages and it looks like this will probably require the GPU servers. I recommend that you create a python virtual environment to allow you to install/update python packages as required. You can view information on how to do this at:

https://intranet.ee.ucl.ac.uk/it/faq/deep-machine-learning

Please get in touch if you require any assistance with this.

November 17th, 2021:

- I am trying to edit my ~/.bashrc file in order to use UCL EEE's remote GPU.
- I tried to access UCL's GPU by ssh but I got this message:
  "kex_exchange_identification: Connection closed by remote host".
- I figured out how to connect to UCL's servers. Like this: ssh -X zceesa@london.ee.ucl.ac.uk
- I don't know how to run my python script after connecting to the remote server.
- IF GRF does not work with me. I will try Unity's environment (there is 2vs2 soccer and dungeon escape):
  https://github.com/Unity-Technologies/ml-agents/blob/main/docs/Learning-Environment-Examples.md

November 18th,2021:

- Worst case scenario, if GRF does not work at all, I will use the Unity environment:
  https://github.com/Unity-Technologies/ml-agents/blob/main/docs/ML-Agents-Overview.md
- Had a meeting today with Prof. Kit. He said: While the GRF issue is being solved, start learning about Reinforcement Learning. You can write that in your interim report as literature review. For the interim report, you should also write about the process you went through to choose the environment, as well as the list of objective for the future. He also said something about a trade off in RL (exploration vs exploitation).

November 19th, 2021:

- This Kaggle course could be useful to do:
  https://www.kaggle.com/learn/intro-to-game-ai-and-reinforcement-learning
- I am currently looking for suitable courses/resources to study the theory of Reinforcement Learning:
  1. David Silver, the creator of AlphaZero, UCL course on YouTube (10 lectures) (17 hours) (slides available) https://deepmind.com/learning-resources/-introduction-reinforcement-learning-david-silver
     https://youtube.com/playlist?list=PLqYmG7hTraZBiG_XpjnPrSNw-1XQaM_gB
  2. Playlist that summarises the Intro to RL book (8 videos, one for each chapter) (1.5 hours):
     https://youtube.com/playlist?list=PLnn6VZp3hqNvRrdnMOVtgV64F_O-61C1D
  3. Stanford CS234: (15 lectures) (19 hours) (Slides available)
     https://youtube.com/playlist?list=PLoROMvodv4rOSOPzutgyCTapiGlY2Nd8u
     http://web.stanford.edu/class/cs234/modules.html
  4. Not as relevant: Intro to Deep Learning: https://youtube.com/playlist?list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI Lecture 5 is on Deep Reinforcement Learning. Course page:
     http://introtodeeplearning.com/

5. "Silly" course on RL from Udacity: A lot of short videos that seem to simplify ideas (might be useful when I need a slower explanation of something) https://classroom.udacity.com/courses/ud600
6. MATLAB RL course taught by Brain Douglas! (5 videos) (1.25 hours): https://youtube.com/playlist?list=PLn8PRpmsu08qw_IwpgVNsKiJQpvvW0MmM
7. Very Relevant Practical 3 hour course on RL using Python: https://www.youtube.com/watch?v=Mut_u40Sqz4
8. (64 videos) (27 hours) it looks useful for trying to find specific things as there are a lot of videos, each on a topic: https://youtube.com/playlist?list=PLyqSpQzTE6M_FwzHFAyf4LSkz_IjMyjD9

- I will start with number 7, as it might actually help me with the practical side that I am currently stuck at, as well as providing fresh ideas (more suitable environment maybe?).
- I think the most robust theoretical resource is number 1, along with the textbook it uses. I can then use number 2 to supplement it.
- I might look at number 6 for some more intuitive knowledge, since it is taught by Brain Douglas.
- Starting with number 7: At (min 26), there is a link to a list of third party environments with open AI Gym. However, the page does not load! I have to find a way to access this page or ask about it.
- Never mind, I found the list by googling it! https://github.com/openai/gym/blob/master/docs/third_party_environments.md
- List of environments found in the list:
  1. **gym-derk: GPU accelerated MOBA environment**: This is a 3v3 MOBA environment where you train creatures to fight each other. It runs entirely on the GPU so you can easily have hundreds of instances running in parallel. There are around 15 items for the creatures, 60 "senses", 5 actions, and roughly 23 tweakable rewards. It's also possible to benchmark an agent against other agents online. It's available for free for training for personal use, and otherwise costs money (for research use). https://gym.derkgame.com
  2. **Procgen:** 16 simple-to-use procedurally-generated gym environments which provide a direct measure of how quickly a reinforcement learning agent learns generalizable skills. The environments run at high speed (thousands of steps per second) on a **single core.** Games are customisable so I might be able to use multi-agent RL in them..? There are games like pac-man and dodgeball etc. **https://github.com/openai/procgen**
  3. **SlimeVolleyGym: A simple environment for single and multi-agent reinforcement learning:** A simple environment for benchmarking single and multi-agent reinforcement learning algorithms on a clone of Slime Volleyball game. Only dependencies are gym and numpy. Both state and pixel observation environments are available. The motivation of this environment is to easily enable trained agents to play against each other, and also facilitate the training of agents directly in a multi-agent setting, thus adding an extra dimension for evaluating an agent's performance. A tutorial demonstrating several different training methods (e.g. single agent, self-play, evolution) that require **only a single CPU machine** in most cases.
  4. **Unity ML Agents:** (the same one I found two days ago) https://github.com/Unity-Technologies/ml-agents
  5. **gym-games:** Gym implementations of the MinAtar games, various PyGame Learning Environment games, and various custom exploration games. **https://github.com/qlan3/gym-games**

November 20th, 2021:

- In GRF, I can run the code without errors related to GPU when I change the representation from Pixels to Simple115. However, I cannot see what is happening! I don't think I can do that because I need to see what is happening initially to understand what is going on. I will look into the youtube tutorial playlist and see what he says about the representation when he wrote the code. Ok, I checked and he says it will run faster with Simple115.
- I tried changing the representation to "extracted", which should be a minimap, but I got a lot of errors.

November 21st, 2021:

- I figured out how to connect to the GPU server and run the code on it!! I do not get any errors! I did it by connecting to the server within the code itself. However, I do not see the game being rendered since it is not running on my machine anymore. I need to figure out how to let the code run on the server but see the pixels at the same time..

- Here is the code I used: (it uses the paramiko python package)

```
import paramiko
# Connect to remote host
client = paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
client.connect('athens.ee.ucl.ac.uk', username='zceesaa', password='*********')
# Setup sftp connection and transmit this script
sftp = client.open_sftp()
sftp.put(__file__, '/tmp/test.py')
sftp.close()
# Run the transmitted script remotely without args and show its output.
# SSHClient.exec_command() returns the tuple (stdin,stdout,stderr)
stdout = client.exec_command('python /tmp/test.py')[1]
print("success")
for line in stdout:
    # Process each line in the remote output
    print (line)
client.close()
sys.exit(0)
```

November 22nd, 2021:

- I am currently watching Andrew Ng's lectures on Machine Learning. This is mainly for my COMP0036 module. However, it will be a useful foundation for this project, even though it is not focused on Reinforcement Learning.
- I made a big mistake by starting the project with choosing a game/environment, when I don't understand anything about Reinforcement Learning! I cannot gauge whether a game is too simple for my project, or too complicated for a single machine (time and power-wise). Therefore, I will focus on learning the theory now and see where this takes me.
- The problem is, even if I find a simpler environment, tensorflow-gpu will always require an NVIDIA gpu, which I don't have. So, the problem is not just with GRF!
- Min(50) of https://www.youtube.com/watch?v=Mut_u40Sqz4 He confirms that CUDA is needed for GPU. However, GPU is used to boost the performance, so you don't need it to run all games! He says that it will boost performance a lot in Deep Learning, but not much in Reinforcement Learning!! I think the best thing to do is to give up on GRF and pick a much simpler game, and not even use tensorflow-gpu! Just tensorflow.
- On the same video, the testing and evaluation part, he shows how you can see how much time is spent on a single episode, and other metrics. These will be useful as a start to gauge if a game is suitable or not because I prefer simpler games where the training time is not very high (such as days/hours).
- **Model-Free vs Model-Based RL:** https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html I should read this later to learn about the different algorithms.

November 23rd, 2021:

- In the comments of the tutorial (number 7) he gave this link to someone who needed help with Multi-agent RL: https://docs.ray.io/en/latest/rllib-env.html it includes: "A Unity3D soccer game being learnt by RLlib via the ExternalEnv API." and "Another example shows, how to run a similar setup against a Unity3D external game engine.": https://github.com/ray-project/ray/blob/master/rllib/examples/serving/unity3d_server.py
- In Lecture 1 of David Silver's course, he said that agents took around 4 days to learn those Atari games, and they used GPU.. so i think i do need that.. i'm just not sure what the best thing to do. use my own computer? (slow but 100% doable), or use remote GPU servers? (works but I don't know how to see the rendering), or use UCL's computer clusters? (I have to be in UCL to do that, but it will be the most powerful).
- In Multi-agent RL, from the perspective of one agent, the rest of the agents are simply part of the environment.

November 24th, 2021:

- I started writing my Interim report. I had a look at the structure it should have an started to formulate my own document. I started by adding on the background theory part as I go with the lectures (of David Silver).

November 25th, 2021: (up to 2nd of December)

- I should look at Google Colab's GPU. I have been told that I can use their free GPU (but only up to 12 hours continuously). I can test the GRF code on it and see if in renders the game on my screen!
- In the Interim report, I should mention the difficult decision I have to take at one point, which is deciding on a game environment. It is not that easy. There is this trade-off between either choosing a game that is too complex where training will take weeks (which is not suitable since the project ends in March), or choosing a simpler game but that might be "solved" to a certain degree (meaning that most algorithms perform well on the game to the point that they are all similar).
- I have a couple of research papers to read and include in the literature review part of the interim report. I will start with the paper titled: "**Google Research Football: A Novel Reinforcement Learning Environment**".
- I should have read this paper a long time ago! It really helps in knowing what I am really researching and tells me a lot about the game as well as the research area. Below I will quote important information from the paper that is relevant for the interim report:
    1. Recent progress in the field of reinforcement learning has been accelerated by virtual learning environments such as video games, where novel algorithms and ideas can be quickly tested in a safe and reproducible manner.
    2. The goal of reinforcement learning (RL) is to train smart agents that can interact with their environment and solve complex tasks (Sutton and Barto, 2018). Real-world applications include robotics (Haarnoja et al., 2018), self-driving cars (Bansal, Krizhevsky, and Ogale, 2018), and control problems such as increasing the power efficiency of data centers (Lazic et al., 2018).
    3. While a variety of reinforcement learning environments exist, they often come with a few drawbacks for research, they may either be too easy to solve for state-of-the art algorithms or require access to large amounts of computational resources. At the same time, they may either be (near-)deterministic or there may even be a known model of the environment (e.g. Chess). Finally, learning environments may have restrictive licenses or depend on closed source binaries. (so not fully open-source).
    4. an RL environment should ideally provide the tools to a variety of current reinforcement learning research topics such as the impact of stochasticity, self-play, multi-agent setups and model-based reinforcement learning, while also requiring smart decisions, tactics, and strategies at multiple levels of abstraction.
    5. This provides a challenging RL problem as football requires a natural balance between short-term control, learned concepts such as passing, and high level strategy.
    6. Existing environments exhibit a variety of drawbacks that we address with the Google Research Football Environment:
        a) **Easy to solve.** many commonly used scenarios can now be solved to a reasonable degree in just a few hours with well-established algorithms. For instance, ~50 commonly used Atari games.
        b) **Computationally expensive.** training agents in recent video-game simulators often requires substantial computational resources that may not be available to a large fraction of researchers due to combining hard games, long episodes, and high-dimensional inputs (e.g. pixels) For example, Starcraft.
        c) **Lack of stochasticity.** The real-world is not deterministic which motivates the need to develop algorithms that can cope with and learn from stochastic environments. Robots, self-driving cars, or data-centers require robust policies that account for uncertain dynamics. It remains an open question whether modern RL approaches such as self-imitation generalize from the deterministic setting to stochastic environments.
        d) **Lack of open-source license**. open-source licenses enable researchers to inspect the underlying game code and to modify environments if required to test new research ideas.

e) **Known model of the environment**. (e.g. Chess) current state-of-the art algorithms often exploit that the rules of these games (i.e., the model of the environment) are specific, known and can be encoded into the approach. As such, this may make it hard to investigate learning algorithms that should work in environments that can only be explored through interactions.

f) **Single-player.** In many available environments such as Atari, one only controls a single agent. However, some modern real-world applications involve a number of agents. Cooperative multi-agent learning also offers many opportunities and challenges, such as communication between agents, agent behaviour specialization, or robustness to the failure of some of the agents. Multiplayer environments with collaborative or competing agents can help foster research around those challenges.

7. **Other football environments.** (e.g. DeepMind MuJoCo Multi-Agent Soccer Environment). In contrast to these environments, the Google Research Football Environment focuses on high-level actions instead of lowlevel control of a physics simulation of robots. Furthermore, it provides many useful settings for reinforcement learning, e.g. the single-agent and multi-agent settings as well as single-player and multiplayer player modes. Google Research Football also provides ways to adjust difficulty, both via a strengthadjustable opponent and via diverse and customizable scenarios in Football Academy, and provides several specific features for reinforcement learning research, e.g., OpenAI gym compatibility, different rewards, different representations, and the option to turn on and off stochasticity.
8. **Other related work.** For instance, the DeepMind Control Suite (Tassa et al., 2018) focuses on continuous control, the AI Safety Gridworlds (Leike et al., 2017) on learning safely, whereas the Hanabi Learning Environment (Bard et al., 2019) proposes a multi-agent setup. As a consequence, each of these environments are better suited for testing algorithmic ideas involving a limited but well-defined set of research areas.
9. **Football Engine:** The length of the game is measured in terms of the number of frames, and the default duration of a full game is 3000 (10 frames per second for 5 minutes). The length of the game, initial number and position of players can also be edited in customized scenarios (see Football Academy below). Players on a team have different statistics , such as speed or accuracy and get tired over time.
10. **Opponent AI Built-in Bots**. The environment controls the opponent team by means of a rule-based bot. The difficulty level θ can be smoothly parameterized between 0 and 1, by speeding up or slowing down the bot reaction time and decision making. Some suggested difficulty levels correspond to: easy (θ = 0.05), medium (θ = 0.6), and hard (θ = 0.95). For self-play, one can replace the opponent bot with any trained model. Moreover, by default, our non-active players are also controlled by another rule-based bot. In this case, the behavior is simple and corresponds to reasonable football actions and strategies, such as running towards the ball when we are not in possession, or move forward together with our active player. In particular, this type of behavior can be turned off for future research on cooperative multi-agents if desired.
11. **State & Observations.** We define as state the complete set of data that is returned by the environment after actions are performed. On the other hand, we define as observation or representation any transformation of the state that is provided as input to the control algorithms. The definition of the state contains information such as the ball position and possession, coordinates of all players, the active player, the game state (tiredness levels of players, yellow cards, score, etc) and the current pixel frame.
12. We propose three different representations. Two of them (pixels and SMM) can be stacked across multiple consecutive time-steps (for instance, to determine the ball direction), or unstacked, that is, corresponding to the current time-step only. Researchers can easily define their own representations based on the environment state by creating wrappers similar to the ones used for the observations below.
13. **Pixels.** The representation consists of a 1280 × 720 RGB image corresponding to the rendered screen. This includes both the scoreboard and a small map in the bottom middle part of the frame from which the position of all players can be inferred in principle.

14. **Super Mini Map.** The SMM representation consists of four 72 × 96 matrices encoding information about the home team, the away team, the ball, and the active player respectively. The encoding is binary, representing whether there is a player or ball in the corresponding coordinate.
15. **Floats.** The floats representation provides a compact encoding and consists of a 115-dimensional vector summarizing many aspects of the game, such as players coordinates, ball possession and direction, active player, or game mode.
16. **Actions.** The actions available to an individual agent (player) are displayed in Table 1. They include standard move actions (in 8 directions), and different ways to kick the ball (short and long passes, shooting, and high passes that can't be easily intercepted along the way). Also, players can sprint (which affects their level of tiredness), try to intercept the ball with a slide tackle or dribble if they posses the ball. We experimented with an action to switch the active player in defense (otherwise, the player with the ball must be active). However, we observed that policies tended to exploit this action to return control to built-in AI behaviors for non-active players, and we decided to remove it from the action set. We do not implement randomized sticky actions. Instead, once executed, moving and sprinting actions are sticky and continue until an explicit stop action is performed (Stop-Moving and Stop-Sprint respectively).
17. **Rewards.** The Football Engine includes two reward functions that can be used out-of-the-box: SCORING and CHECKPOINT. It also allows researchers to add custom reward functions using wrappers which can be used to investigate reward shaping approaches.
18. **SCORING** corresponds to the natural reward where each team obtains a +1 reward when scoring a goal, and a −1 reward when conceding one to the opposing team. The SCORING reward can be hard to observe during the initial stages of training, as it may require a long sequence of consecutive events: overcoming the defense of a potentially strong opponent, and scoring against a keeper.
19. **CHECKPOINT** is a (shaped) reward that specifically addresses the sparsity of SCORING by encoding the domain knowledge that scoring is aided by advancing across the pitch: It augments the SCORING reward with an additional auxiliary reward contribution for moving the ball close to the opponent's goal in a controlled fashion. More specifically, we divide the opponent's field in 10 checkpoint regions according to the Euclidean distance to the opponent goal. Then, the first time the agent's team possesses the ball in each of the checkpoint regions, the agent obtains an additional reward of +0.1. This extra reward can be up to +1, i.e., the same as scoring a single goal. Any non-collected checkpoint reward is also added when scoring in order to avoid penalizing agents that do not go through all the checkpoints before scoring (i.e., by shooting from outside a checkpoint region). Finally, checkpoint rewards are only given once per episode.
20. **Accessibility.** Researchers can directly inspect the game by playing against each other or by dueling their agents. The game can be controlled by means of both keyboards and gamepads. Moreover, replays of several rendering qualities can be automatically stored while training, so that it is easy to inspect the policies agents are learning.
21. **Stochasticity.** In order to investigate the impact of randomness, and to simplify the tasks when desired, the environment can run in either stochastic or deterministic mode. The former, which is enabled by default, introduces several types of randomness: for instance, the same shot from the top of the box may lead to a different number of outcomes. In the latter, playing a fixed policy against a fixed opponent always results in the same sequence of actions and states.
22. **API & Sample Usage.** The Football Engine is out of the box compatible with the widely used OpenAI Gym API (Brockman et al., 2016).
23. **Technical Implementation & Performance.** The Football Engine is written in highly optimized C++ code, allowing it to be run on commodity machines both with GPU and without GPU-based rendering enabled. This allows it to obtain a performance of approximately 140 million steps per day on a single hexacore machine (see Figure 3).
24. **Football Benchmarks:** The goal in the Football Benchmarks is to win a full game against the opponent bot provided by the engine. We provide three versions of the Football Benchmarks that only differ in the strength of the opponent AI as described in the last section: the **easy, medium, and hard** benchmarks. This allows researchers to test a wide range of research ideas under different computational constraints such as single machine setups or powerful distributed settings. We expect

that these benchmark tasks will be useful for investigating current scientific challenges in reinforcement learning such as **sample efficiency, sparse rewards, or model-based approaches**.

25. **Experimental Setup:** As a reference, we provide benchmark results for three stateof-the-art reinforcement learning algorithms: **PPO** (Schulman et al., 2017) and **IMPALA** (Espeholt et al., 2018) which are popular policy gradient methods, and Ape-X **DQN** (Horgan et al., 2018), which is a modern DQN implementation. We run PPO in **multiple processes on a single machine**, while IMPALA and DQN are run on a **distributed cluster** with 500 and 150 actors respectively. In all benchmark experiments, **we use the stacked Super Mini Map representation** and the same network architecture. We c**onsider both the SCORING and CHECKPOINT rewards**. The tuning of hyper-parameters is done using easy scenario, and we follow the same protocol for all algorithms to ensure fairness of comparison.
26. **Results:** The experimental results for the Football Benchmarks are shown in Figure 4. It can be seen that **the environment difficulty significantly affects the training complexity** and the average goal difference. We observe that the **CHECKPOINT reward function appears to be very helpful for speeding up the training for policy gradient methods but does not seem to benefit as much the Ape-X DQN** as the performance is similar with both the CHECKPOINT and SCORING rewards. We conclude that the **Football Benchmarks provide interesting reference problems for research** and that there remains a large headroom for progress, in particular **in terms of performance and sample efficiency on the harder benchmarks.**
27. **Football Academy:** Training agents for the Football Benchmarks can be challenging. To allow researchers to quickly iterate on new research ideas, we also provide the Football Academy: a diverse set of scenarios of varying difficulty. These 11 scenarios (see Figure 5 for a selection) include several variations where a single player has to score against an empty goal etc. Using a simple API, **researchers can also easily define their own scenarios and train agents to solve them.**
28. **Experimental Results:** The experimental results indicate that the Football Academy provides a set of diverse scenarios of different difficulties suitable for different computational constraints. The scenarios where agents have to score against the empty goal (Empty Goal Close, Empty Goal, Run to Score) appear to be **very easy and can be solved by both PPO and IMPALA** with both reward functions **using only 1M steps**. As such, these scenarios can be considered “unit tests” for reinforcement learning algorithms where **one can obtain reasonable results within minutes or hours instead of days or even weeks. harder tasks may be used to quickly iterate on new research ideas on single machines before applying them to the Football Benchmarks** (as **experiments should finish within hours or days**).
29. **Promising Research Directions:** In this section we briefly discuss a few initial experiments related to three research topics which have recently become quite active in the reinforcement learning community: s**elf-play training, multi-agent learning, and representation learning** for downstream tasks.
30. **Multiplayer Experiments:** The **Football Environment provides a way to train against different opponents, such as built-in AI or other trained agents**. Note this allows, for instance, for self-play schemes. When a policy is trained against a fixed opponent, it may exploit its particular weaknesses and, thus, it may not generalize well to other adversaries. We conducted an experiment to showcase this in which a first model A was trained against a built-in AI agent on the standard 11 vs 11 medium scenario. Then, another agent B was trained against a frozen version of agent A on the same scenario. While B managed to beat A consistently, its performance against built-in AI was poor. The numerical results showing this lack of transitivity across the agents are presented in Table 2.
31. Multi-Agent Experiments: The **environment also allows for controlling several players from one team simultaneously, as in multi-agent reinforcement learning**. We conducted experiments in this setup with the **3 versus 1 with Keeper scenario from Football Academy**. We **varied the number of players that the policy controls from 1 to 3**, and trained with Impala. As expected, **training is initially slower when we control more players, but the policies seem to eventually learn more complex behaviors and achieve higher scores**. Numerical results are presented in Table 3.
32. **Representation Experiments:** T**raining the agent directly from raw observations, such as pixels, is an exciting research direction.** While it was successfully done for Atari, i**t is still an open challenge**

for most of the more complex and realistic environments. In this experiment, we compare several representations available in the Football Engine. **Pixels gray** denotes the raw pixels from the game, which are resized to 72 × 96 resolution and converted to grayscale. **While pixel representation takes significantly longer time to train**, as shown in Table 4, **learning eventually takes place (and it actually outperforms handpicked extensive representations like 'Floats').**

33. **Conclusions.** It is challenging and accessible, easy to customize, and it has specific functionality geared towards research in reinforcement learning. We expect that t**hese components will be useful for investigating current scientific challenges** like s**elf-play, sample-efficient RL, sparse rewards, and model-based RL.**

December 2nd, 2021:

- After reading the paper, the best way for me to start training agents is to reproduce the easy experiments done using the three algorithms (PPO, IMPALA, DQN) on the empty goal scenario. This is because it could be trained within hours since it is an easy scenario. Also, by reproducing these results, I will learn how to use the environment properly and it will prepare me to start carrying out my own experiments. An interesting place to start is the 3 vs 1 scenario, where I can use multi-agent RL.
- As shown in bullet point 32 above, training using the pixel representation is very difficult! I made a mistake by reading this paper too late because I have been trying to run it using pixels ever since I started!
- I will try to use Google Colab's GPU servers now.
- If this doesn't work, I will have to use UCL's servers and try to export a video of the rendering. If that doesn't work, I will save the agent after it trains, then I can check how it performs on my own machine since that doesn't need Tensorflow.
- I will follow this tutorial to use Google Colab: https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d
- I learnt that Bash commands can be run by prefixing the command with '!'.
- I am having issues running my code on Google Colab as it doesn't allow installing gfootball, which is the library for GRF.
- I will try to upload the gfootball folder I already installed on my laptop. Ok, this method works, but I have to manually upload all the libraries! For some reason, cloning from github doesn't work with me in Google Colab.
- Even after manually uploading the libraries it doesn't work. I also tried putting them in the site packages folder, still doesn't work. I will have to use the UCL GPU server from now on and see how it goes.

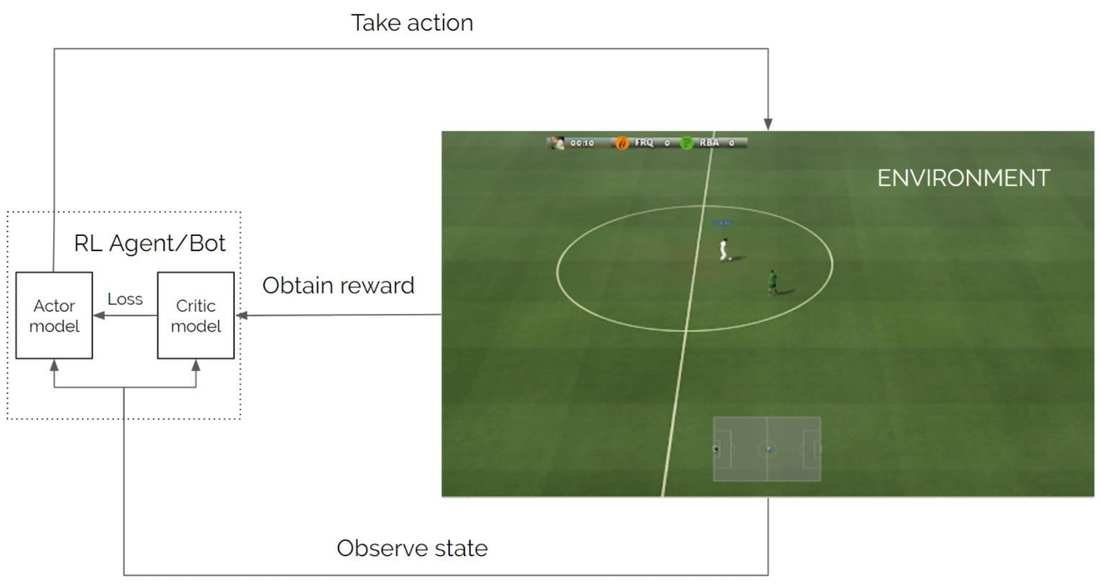
December 5th, 2021:

- I figured out how to run the test code (from the online tutorial I tried before) locally, by using a different representation than pixels (Extracted). However, in the online tutorial, he still gets to view the game in pixels while it trains agents in the Extracted representation. I am not sure how he does it, as this could be all I need to know in order to move forward. Actually, the representation he uses is simple115, rather than Extracted. Right now, the code runs! I still need to know how to save the agent then see how it performs by watching it play. I also need to go back and understand every part of the code so that I can change it and fix the bug that I get at the end.
- important tutorial on reproducing GRF RL results: https://towardsdatascience.com/reproducing-google-research-football-rl-results-ac75cf17190e
  the blog contains instructions on how to generate and watch videos of the game! This could solve my issue if it works with football academy as I can train the agent using the simple155 representation, then watch the video to see what happens.
- Go back to the 3 hour RL tutorial as I think he shows how to save trained agents.

December 6th, 2021:

- In the GRF paper, they only use two types of rewards, scoring and checkpoints, where checkpoints is a measure of how close you are to the opponent. I think a possible way to improve the results of all algorithms is to implement new reward functions! For example, passing through all defenders and being head-to-head with the keeper should have a high reward (as opposed to just calculating how close to the goal the player is).

- In the interim report: why did I not choose a game other than GRF since it caused many issues? Answer: because other decent options are from the OpenAI gym, which is the same as GRF since it uses the OpenAI Gym, meaning that the problems I face will be the same on other games.
- I will go through the tutorial slowly and understand the code. I will make a python file for each step in the process so that I can show my supervisor the progress and also explain the progress in the interim report.
- The tutorial makes an agent using the Proximal Policy Optimisation model (PPO).
- I already created a virtual environment and installed the system dependencies and python packages required for this project. I also already tested the game running on my laptop and it worked on Linux (on Windows it wasn't good).
- Now, in the python code, we start by importing the gfootball library:
```
import gfootball.env as football_env
```
- Then, we create an environment object (env):
```
env = football_env.create_environment(env_name="academy_empty_goal",
representation="pixels", render=True)
```
  env_name chooses the scenario that will be used. academy_empty_goal scenario is where our player spawns at half-line and has to score in an empty goal on the right side.
  representation='pixels' means that the state that our agent will observe is in the form of an RGB image of the frame rendered on the screen. (This representation does not work on my laptop when using tensorflow and this is where i spent a lot of time trying to solve it!
- After that, we can check the state dimensions and the number of possible actions the agent can take at that state:
```
state_dims = env.observation_space.shape
print(state_dims)
n_actions = env.action_space.n
print(n_actions)
```
  This prints:
```
(72, 96, 3)
19
```
  where the frame dimensions are 72x96, and 3 since it is in RGB. 19 is the number of actions the agent can take. (This is the end of Part 1).
- In part 2, we incorporate the Actor-Critic models. The figure below (taken from the tutorial) shows the structure of the PPO algorithm. The actor model takes as input the current observed state and decides what action is best to take. This action is fed into the environment, which produces a new state and a reward is obtained. This reward could be positive (if a goal is scored), or negative (if an own goal is scored). This reward is the input to the critic model, which evaluates if the action taken by the Actor led our environment to be in a better state or not and gives its feedback to the Actor, hence its name. It outputs a real number indicating a rating (Q-value) of the action taken in the previous state. By comparing this rating obtained from the Critic, the Actor can compare its current policy with a new policy and decide how it wants to improve itself to take better actions.



- Before we code the two models, we start by letting the agent pick random actions in the environment for 128 steps:

```
ppo_steps = 128
states = []
actions = []
values = []
masks = [] # checks if the match is in completed/over state, in which case we
want to restart the game
rewards = []
actions_probs = []
actions_onehot = []
for itr in range(ppo_steps): # collect 128 interactions with the game
   observation, reward, done, info = env.step(env.action_space.sample()) # this
takes a random action in the game and gives as output the observation, reward,
and other info, as well as if the game is done.
   if done:
       env.reset() # reset if the game is done
env.close()
```

- Starting with the actor model, it is coded as follows: (I still do not understand all of it):

```
def get_model_actor_image(input_dims):
   state_input = Input(shape=input_dims)

   # Use MobileNet feature extractor to process input image
   feature_extractor = MobileNetV2(weights='imagenet', include_top=False)
   for layer in feature_extractor.layers:
       layer.trainable = False

   # Classification block
   x = Flatten(name='flatten')(feature_extractor(state_input))
   x = Dense(1024, activation='relu', name='fc1')(x)
   out_actions = Dense(n_actions, activation='softmax', name='predictions')(x)

   # Define model
   model = Model(inputs=[state_input], outputs=[out_actions]) # the model takes
as input the current state and outputs a list of actions
   model.compile(optimizer=Adam(lr=1e-4), loss='mse')

   return model
```

- Then, the function is called to construct the actor model:

```
model_actor = get_model_actor_image(input_dims=state_dims)
```

- After that, the PPO loop is edited to take into account the actor model, rather than picking random actions in each state:

```
for itr in range(ppo_steps): # collect 128 interactions with the game
    state_input = K.expand_dims(state, 0)
    action_dist = model_actor.predict([state_input], steps=1) # uses the actor
model to predict the best actions
    action = np.random.choice(n_actions, p=action_dist[0, :]) # picks an action
based on the action distribution from the actor model
    action_onehot = np.zeros(n_actions)
    action_onehot[action] = 1
    observation, reward, done, info = env.step(action)
    if done:
        env.reset() # reset if the game is done
env.close()
```

- Now we incorporate the critic model. It is similar in code to the actor model, the difference is that there is only one output node from the neural network, which is the q_value, and it is a real number.

```
def get_model_critic_image(input_dims):
    state_input = Input(shape=input_dims)

    # Use MobileNet feature extractor to process input image
    feature_extractor = MobileNetV2(include_top=False, weights='imagenet')
    for layer in feature_extractor.layers:
        layer.trainable = False

    # Classification block
    x = Flatten(name='flatten')(feature_extractor(state_input))
    x = Dense(1024, activation='relu', name='fc1')(x)
    out_actions = Dense(1, activation='tanh')(x) # output node from the neural net
is 1 since it is only the q_value


    # Define model
    model = Model(inputs=[state_input], outputs=[out_actions])
    model.compile(optimizer=Adam(lr=1e-4), loss='mse')

    return model
```

- Then, the function is called to construct the critic model:

```
model_critic = get_model_critic_image(input_dims=state_dims)
```

- After that, the PPO loop is edited to take into account the critic model, and new data is appended to existing lists:

```
for itr in range(ppo_steps): # collect 128 interactions with the game
    state_input = K.expand_dims(state, 0)
    action_dist = model_actor.predict([state_input], steps=1) # uses the actor
model to predict the best actions
    q_value = model_actor.predict([state_input], steps=1)  # uses the critic model
to predict the q value.
    action = np.random.choice(n_actions, p=action_dist[0, :]) # picks an action
based on the action distribution from the actor model
    action_onehot = np.zeros(n_actions)
    action_onehot[action] = 1
    observation, reward, done, info = env.step(action)
    mask = not done

    states.append(state)
    actions.append(action)
    actions_onehot.append(action_onehot)
    values.append(q_value)
    masks.append(mask)
    rewards.append(reward)
    actions_probs.append(action_dist)
```

```
   state = observation # changing the state variable into the new observation or
the next state, otherwise we use the same initial state as input to out models.

   if done:
       env.reset() # reset if the game is done
env.close()
```

- In part 3, we implement an algorithm called: Generalized Advantage Estimation (GAE). i will come back to it since I do not understand it.
- In part 4, we use a custom PPO loss function instead of MSE.

December 7th, 2021:

- Continuing with the tutorial, in part 4, a custom PPO loss function is used instead of MSE. I am getting errors when running the code, even though it is the same code as the tutorial. I will try to fix it.
- The error was due to a silly mistake where I should use `q_value = model_critic.predict` instead of `q_value = model_actor.predict`
- I found another error in the tutorial's code, in the actor loss assignment where the model_actor.fit is used, there was an extra bracket. However, I still get an error due to: Data cardinality is ambiguous, Make sure all arrays contain the same number of samples. I think the issue is with reshaping the arrays when fitting the model. I will ignore this now and continue with part 5.
- In part 5, finishing touches are added, which include… I need to solve the Data cardinality error otherwise the code doesn't carry on to the next step which is saving the model! Why do I get this error and in the tutorial he doesn't? when the code is the same?

December 8th, 2021:

- I will not spend more time trying to fix the data cardinality bug. Instead, I will make a simpler version of the tutorial's code to avoid this issue.
- My method worked!! I got rid of the extra parts of the code where they take into consideration the custom PPO loss function, which causes data cardinality errors when fitting the model. Now, i will move on into trying to improve the current model and learn how to save it and test it!
- The code finally works and the agent is training. The first thing I noticed, the player got stuck at one point in training where he decides not to move (he only performs action 13). This is probably due to the simple model used. However, the objective now is to get to the end of the code where the model is supposed to save, then test loading the model!
- Ok, the code finished running completely with no errors. It took around 30 minutes to do 50 iterations of 128 steps each. I think it took a long time because the representation is in pixels. Now, I need to learn how to save the model properly because when I run the test code it doesn't load the file (since it didn't save properly).
- (check whether I have a gpu or cpu tensorflow. If I have the gpu one, then how did it suddenly work on my PC? Answer: I am using tensorflow-gpu!! What! Maybe it just ignores the gpu part and uses the cpu?
- I managed to save the model and then load it and test it! it works!! all I had to do was remove this if statement from my code (because all iterations were getting 0 rewards so it never needs to save a model):
  ```
  if avg_reward > best_reward:
  ```

  Test code:
  ```
  env = football_env.create_environment(env_name='academy_empty_goal',
  representation='pixels', render=True)

  n_actions = env.action_space.n
  dummy_n = np.zeros((1, 1, n_actions))
  dummy_1 = np.zeros((1, 1, 1))

  model_actor = load_model('my_first_model_actor.hdf5')
  model_critic = load_model('my_first_model_critic.hdf5')


  state = env.reset()
  ```

```
    done = False

    while True:
        state_input = K.expand_dims(state, 0)
        action_probs = model_actor.predict(state_input, steps=1)
        action = np.argmax(action_probs)
        next_state, _, done, _ = env.step(action)
        state = next_state
        if done:
            state = env.reset()
```

- I will run the same code now on UCL's GPU server and see how long it takes.
- Well, the code runs instantly regardless of the number of iterations, which means something is wrong and it's not working properly. I will come back to this later. For now, I will focus on the actual model running locally and improve it.
- I made a GitHub repo for my project, to act as a backup location as well.

December 9th, 2021:

- Next steps:
    1. Improve the current PPO model.
    2. Let the model solve the simple empty goal scenario and check how many steps are necessary to achieve that.
    3. Use other models (e.g IMPALA and DQN) and solve the simple empty goal scenario. compare with the paper.
- Supervisor said to focus on Multi-agent!!
- I should try to start on the multi-agent part as soon as possible. I will use the christmas break to work on the project. I will watch the lectures to learn the theory of RL, improve the single-agent model to solve the empty goal scenario, then jump into multi-agent algorithms. The supervisor emphasised the importance of having a balanced game (3 vs 3) and so on. I will try to explain that there are already existing scenarios and I might not have enough time to design my own scenario. 5 vs 5 is available but it might be too much given the available resources.

December 10th, 2021:

- The RL Kaggle course could be useful in terms of the code they provide.
- I should try training using the simple115 representation, then testing it on the pixel representation to see if the agent can perform regardless of the representation or if they only know how to play in the representation they trained in. (I think it must be the latter but I hope it is the first).
- I tried to use Kaggle's gpu to run my code but I have the same issue as the Google Colab one, which is that I cannot import gfootball and hence cannot create the environment.
- I am going to focus on writing the interim report as it is due wednesday.

December 12th, 2021:

- This github repo contains relevant code for training RL agents in GRF: https://github.com/Ujwal2910/Deep-RL-on-Gfootabll-Google-football-OpenAI-style-environment
- I tried to run the game on Kaggle's servers again but with no luck.

December 14th, 2021:

- This lecture on Deep Learning 2: Introduction to TensorFlow could be useful programming-wise: https://www.youtube.com/watch?v=JO0LwmIlWw0
- I am still writing the interim report, since the 12th.

Everything above was considered in the interim report, below will include logs after the interim report.

December 27th, 2021:

- Still struggling to solve the data cardinality issue

- I think i found a way around the data cardinality issue, by reshaping all inputs and outputs. However, I now get this error message:

  ```
  TypeError: You are passing KerasTensor(type_spec=TensorSpec(shape=(), dtype=tf.float32, name=None), name='Placeholder:0', description="created by layer 'tf.cast_4'"), an intermediate Keras symbolic input/output, to a TF API that does not allow registering custom dispatchers, such as `tf.cond`, `tf.function`, gradient tapes, or `tf.map_fn`. Keras Functional model construction only supports TF API calls that *do* support dispatching, such as `tf.math.add` or `tf.reshape`. Other APIs cannot be called directly on symbolic Kerasinputs/outputs. You can work around this limitation by putting the operation in a custom Keras layer `call` and calling that layer on this symbolic input/output.
  ```
- I found a way around that issue, which is to disable something from tensorflow by writing:
  ```
  from tensorflow.python.framework.ops import disable_eager_execution
  disable_eager_execution()
  ```
- It works but the code is now extremely slow!
- I will take a step back, go back to the simpler model, and try to implement the checkpoint reward function.
- Then, I will go back to the train.py code and use the simple155 representation as it will run quicker.
- I managed to add the checkpoint reward function by looking at the source code and figuring out how to add it.
- it seems that the checkpoint reward function is not contributing anything to the reward. maybe it only works in the full match? let's try it. Ok, tried it, still nothing!!
- I tried to use the simple155 representation on train.py, while increasing the ppo_steps to 1280, but no progress. This means one of two things, training needs way more steps than this, or that the algorithm is simply bad and I need to construct a new one. I will do the latter
- I realised why my avg_reward is always zero. It is because of the test_reward() function! it does not do enough steps to test the model so the player has no time to score. I will abandon the test_reward() function for now and just set avg_reward to the sum of rewards in that iteration.
- After several tries, a goal was scored using train.py using 128 steps and 30 iterations (on the 4th iteration tho). The model was saved and now I will load it and see if it works on pixels (since it was trained using simple155).
- As expected, using a different representation does not work because the state dimensions inputted are different. Error message:
  ```
  ValueError: Input 0 of layer "model" is incompatible with the layer: expected shape=(None, 115), found shape=(1, 72, 96, 3)
  ```
- I Figured out something extremely important! I can still render the game while training/testing using simple115 representation! it does not have to be pixels!
- It also seems like way more steps are needed for the agent to learn. my machine is definitely not suitable even for the simplest of tasks so I need to use a remote server.
- it seems like when a model is saved and loaded, it literally repeats the exact sequence of actions it did in training.
- Next steps: load an already available model in train.py so that training builds up. Also, are actions taken randomly at start? it seems so. This is why high number of steps is needed at the start.
- I also noticed something, when the number of steps is high, the agent sometimes get stuck doing a specific action (e.g. not moving) and then the rest of the iteration is useless! I don't know why this happens and what to do about it. I have an idea, I could force reset the environment after like 300 steps since that's too long anyway for one try.

December 28th, 2021:

- I ran the code overnight to do 100k iterations. However, only around 24k steps were done during that time. It is important to note that the processing speed decreased with time! for example, 1000 steps at the start are done in less than 3mins, compared to an hour or more later!
- I will learn how to run the code on UCL's GPU server now. As it is necessary.
- Ok, it seems that ToServer.py code is not working as expected. It connects to the remote server but does not execute train.py. However, I managed to connect to the remote server and run train.py using the following cmd command on PyCharm's terminal:
  ```
  cat train.py | ssh zceesaa@athens.ee.ucl.ac.uk python
  ```
- The error message I got was:
  ```
  File "<stdin>", line 1, in <module>
  ```

```
ModuleNotFoundError: No module named 'gfootball'
```

- The issue now is similar to that with Kaggle and Google Colab servers. I cannot use the modules imported!
- I asked someone and figured out how to do it. I need to put the names of all packages in a requirements.txt file, which is within the virtual environment. Then, pip this into the remote server and then run the code. Not sure how to do it exactly but at least I know what I should be doing.

January 2nd, 2022:

- I was offered some help on the remote server issue. A lot of progress was made. MobaXterm is really useful as it provides a user friendly interface to interact with the remote server, especially in terms of viewing my own files! Also, I just realised the server is basically my UCL account desktop.
- Time was spent trying to fix the GitHub issues I had with pulling the repository into my local project on Windows. This is not a priority so I will continue with it later.
- pip was not available on the server so a lot of time was spent trying to install pip there. It turns out it is there but the command is pip3 instead of pip.
- requirements.txt file was added to the server. All there is left to do is to pip install the requirements.txt file and then we can run the code.
- The project was added to the server by making a zip file of the virtual environment. Another way to do it is by cloning the github repo but then the libraries are not included.

January 5th, 2022:

- I trained the ppo model on my laptop for 10k steps, twice, and the outcome is two models where the player does not move, the action is always 16 and 18 for the other model. This raises the question, what is the reason for only picking one action? Is it the low number of iterations? or is it a problem in my code? I'm not sure, but the model has scored goals when training, so surely it will follow those actions to score a goal again. OH! I think I know what is happening! The agent, for some reason, only cares about the very last action taken before scoring a goal, since it comes with a +1 reward, and it thinks it is the best action and does it all the time. This is why all my previous models also spammed a single action!
- This is weird because the function get_advantages is used for this exact purpose. It should take into account previous actions that led to the positive reward. I think my next step is to check this function and understand how it exactly works and why it is not working.

January 6th, 2022:

- actions 16 and 18 are not even attacking actions, so that's even weirder.
- Ok, the reason the agent does the same action every time is this line in the test code:
  ```
  action = np.argmax(action_probs)
  ```
- the maximum probability (which should mean the best action to take, is for some reason always action 16 which is not the best one at all. replacing that line with the following lets the agent pick other actions:
  ```
  action = np.random.choice(n_actions, p=action_probs[0, :])
  ```
- But I'm still not sure if this is the right thing to do because in the tutorial they use argmax and it works.. I doubt that the reason for sticking with one action is the low number of steps because in the tutorial they only use 128 steps and it worked!
- by printing action_probs I checked that the probabilities slightly change from state to state, here is an example of one:
  ```
  [[0.05263972 0.06114597 0.02849054 0.05847295 0.02916158 0.06725094
    0.03290011 0.06154972 0.03884064 0.04107085 0.04371261 0.05049962
    0.06471127 0.07276074 0.04498716 0.02579279 0.08692873 0.05852281
    0.08056127]]
  ```
- The probabilities add up to 1 which is a good sign. The highest action is the 17th and the 19th. I just realised that since it is a python array it starts from 0 so array index 16 is actually the 17th action in the game. This means that the highest actions preferred by the model are 17 which is: start dribbling (effective when having a ball), sticky action. As for action 19, it lets the game's built-in AI generate an action.
- wait, there are 20 actions in total when considering the 19th AI action, because actions also start from action 0. Now, in the game, only 19 actions are given as a choice, so are they the actions excluding the first one or

the last one? I think the last one since it is an add-on to the default action set. But then the chosen and performed actions don't make any sense! I can easily test that by manually inserting actions.

- By manually trying actions, It turns out that the 19 actions are labeled from 0 to 18, meaning that the extra action (V2 action set) that lets the built-in AI generate an action is not available.
- This means that the action the agent is stuck at is action 16, which is: perform a slide (effective when not having a ball).
- The issue might actually be the low number of steps. Actually, to be more specific, an issue that occurs while training is that the agent gets stuck doing one action (staying still, for example) and the defenders do not do anything, which just wastes a lot of steps and nothing happens. A possible solution to avoid this is to include a check function in the code that resets the environment when a certain action is done let's say 30 times in a row.
- Maybe the issue is with the get_advantages() function? because it does all 10k rewards at once. I tried to understand if this is the issue but I am not sure.
- I brought the steps back at 128 for 10 iterations, it seems like all models that scored a goal have the same issue. The action with the maximum probability is the same for all states when testing!
- Ok I think I found out something! So out of the 10 iterations, 3 to 9 were saved, as they scored one goal. the first few iterations were stuck at action 14, but then the following models were stuck at action 3, which is moving upwards. The final model was not stuck at action 3! it changed to action 1 when the player moved up, as at that state the action probabilities changed and the maximum was action 1, which is moving to the left. Actually, a goal was only scored in iteration 3.. I forgot to reset the total rewards counter before each new iteration..
- ok i have more understanding of what is going on now! a new iteration does not start all over again, but it builds on previous iterations! How does it do that? Well, after the first iteration (128 steps) the model is trained using model.fit with the data from the 128 steps. That is one iteration, and if a goal is not scored then it is useless for us because only the scoring reward function is used (the checkpoint one does not work for some reason, maybe only in full matches?). After model.fit, the probability distribution for the list of actions to choose from changes, and the probability of good actions start to dominate after every iteration where the player scores a goal.

January 8th 2022:

- I ran train.py with 128 steps and 150 iterations. After a couple of hours, the code finished running.
- The final model saved managed to solve the scenario and score the goal every time!!!!! Finally!

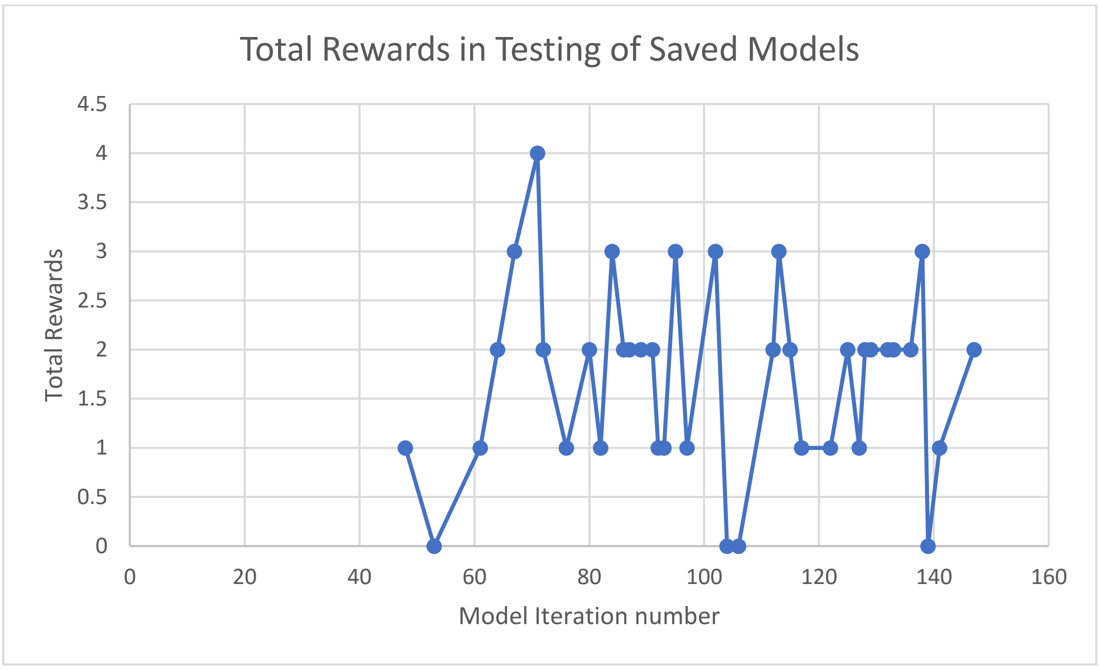
January 11th 2022:

- I will analyse the results of the previous experiments. What I have in mind is to let each saved model play the scenario for 10 times, and count how many goals it scores. Then I could plot the model, or number of iterations (or total training steps) against goals scored out of 10 tries. I could also take the probability distribution of actions for each saved model to see how it develops.
- For the following experiments, I should try to get a video of the whole training process rendered.. using the write dump thing.
- In the first successful/saved model,which is model_actor_48_1.0, the player just runs top right, obviously because np.argmax is used. I am thinking of letting the agent pick a random action based on the probability distribution, instead of the maximum probability action every time. This is because model_actor_53_1.0 just stands still, same issue as before, which will affect its results.
- I decided to do both, one with argmax and one with random choice.
- Ok, model_actor_61_2.0 scores a goal every time since the max probability action is to shoot the ball. I think using argmax is not good for the test because the plot will be two horizontal straight lines connected by a vertical line.
- Ok it seems like using random.choice is much better! The issue is, the scenario is way too easy and once the agent scored four goals it practically solved it! The plot will not look smooth but it is good to do and compare with another harder scenario!
- I should have saved all iterations from the training, that way the plot will look smoother, but even with that, it will most likely not be smooth as well! As I said, the agent improves only when a goal is scored, due to only using the scoring reward function!
- I edited test.py such that it loads each saved model, and tests it by counting how many goals are scored in 5 tries, then writes that data to a text file with an example action probability distribution.
- I am watching the lectures for RL and writing the theory I learn in the Final Report document.

- I'll try the full game scenario to see if the checkpoint reward works, if not then I have to ask someone.

January 12th, 2022:

- Figure of test results:



- To explain the previous figure, training was done at 128 steps for 150 iterations. After each iteration, if the agent scored a goal, it saves the model. Training ended with around 50 saved models, the first of which was the 48th iteration. Testing was done on all saved models by giving them 5 tries to score a goal. Total rewards shown in the y-axis indicate how many goals were scored from those 5 tries (if an own goal was scored it subtracts 1 from the total rewards). The plot is not very useful for two reasons. Firstly, after the 50th iteration, the model already becomes good enough to score goals, and we can't show the contrast by comparing with early iterations since they were not saved. Even if they were saved, it is very time consuming to test all iterations as it will take more time than training itself! The second reason is only giving them 5 tries instead of 10 or 50 or 100. This is also because it is very time consuming to do so on a single machine, as the server issue has not been solved yet!
- Another useful output from the test was the action probability distribution for each saved model (iteration). It is too long to paste all of it here but I will include some for illustration. It is an array of 19 elements, each one is the probability of picking a certain action at a specific state. The array does not change much with other states as the scenario is pretty simple.

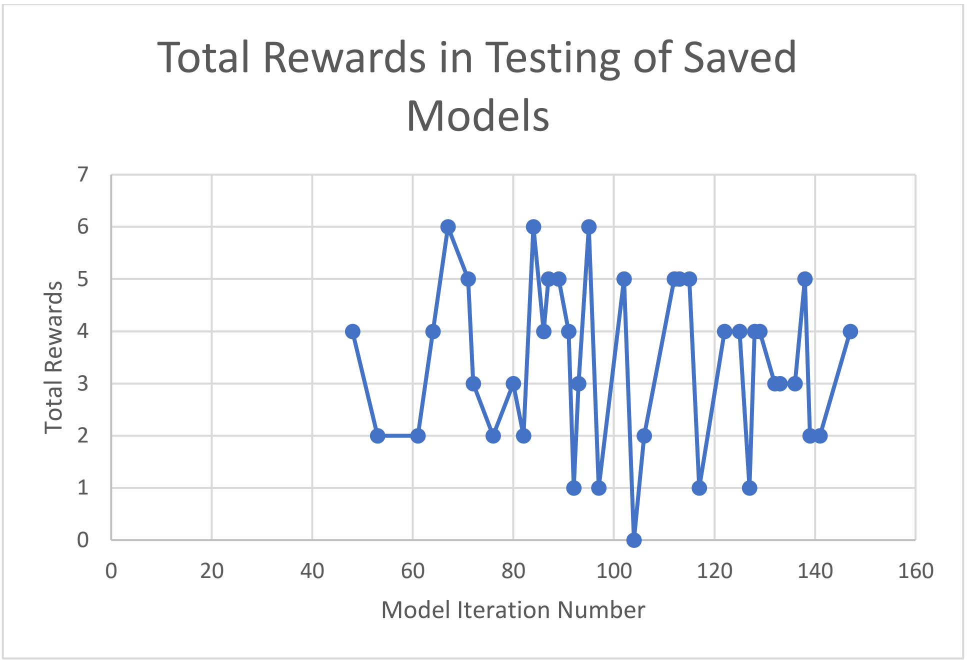
  Model 48: [[0.00463779 0.03474213 0.00189585 0.00095825 0.22312225 0.00178112
   0.01238063 0.0833588 0.02486224 0.02135637 0.12644684 0.0072421
   0.18433538 0.00153058 0.00122976 0.0023917 0.00328999 0.03352042
   0.23091777]]

  Model 92: [[1.15470518e-03 3.59623926e-03 3.64654057e-04 1.13296104e-04
   2.42232442e-01 4.70989646e-04 2.10855366e-03 1.57290593e-01
   6.96919672e-03 1.16787581e-02 1.42140806e-01 1.43009645e-03
   2.42935851e-01 2.99931795e-04 2.47082236e-04 6.42781612e-04
   6.65327825e-04 1.17751947e-02 1.73883453e-01]]

  Model 147: [[5.89877600e-04 9.83541016e-04 1.10406196e-04 4.41773045e-05
   1.43370420e-01 1.76776215e-04 9.38430778e-04 2.04716504e-01
   4.07689391e-03 9.44945775e-03 1.71779886e-01 6.68688968e-04
   3.27994645e-01 8.44620590e-05 5.86770439e-05 3.18585255e-04
   2.28398421e-04 8.39966908e-03 1.26010552e-01]]
- As shown above, as training continues, the probability distribution becomes more inclined towards a specific set of actions (which lead to scoring a goal). In this case, we can see that array element 4, 7, 10, 12 and 18 are the ones with the most probability at the end of training. These correspond to running top right, running to the bottom, performing a high pass, shooting, stop dribbling. Shooting has the highest probability at the end of training with a probability of around 33%, followed by running to the bottom with 20%.
- In Lecture 3: any value function can be used to compute a better value function.

January 14th, 2022:

- I did another test for 5 tries, to add the rewards obtained to the same models. The result of added rewards (now out of 10 tries) is shown in the figure below, it is still useless..



January 15th, 2022:

- I am now reading the docs of gfootball for multi-agent implementation.
- I learnt that an episode is defined as the time from kick-off to a goal leaving the pitch, or a goal is scored, basically when the game stops. Here are examples of customising when an episode ends when building a scenario:

  builder.config().end_episode_on_score = True

  builder.config().end_episode_on_out_of_play = True

  builder.config().end_episode_on_possession_change = True

- I changed the scenario into 5_vs_5 match. I tested my agent that solved the empty goal scenario and the action probability distribution at the start is different in this scenario, which makes sense since the state is completely different now with 5 players one each side.
- in my multi-agent experiments, should one model/policy control more than one player in a single team? Or should each player have his own model/policy? As for the other team, should their model train at the same time as the other team? Or should each training be against a rule-based bot? I think training should definitely be against a rule-based bot, if both teams were training then it might be chaotic and both teams might not learn how to play the game properly, compared to if trained against an already established opponent.
- I think I found the answer to my question above. Of course it should be one model for the team! and each player is an agent in the model! The goal is to find a "joint policy" between the agents that maximises the rewards.
- I will now implement the same PPO model but make it control more than one player!
- found an interesting paper on multi-agent GRF! https://arxiv.org/pdf/2110.04507.pdf
- This is their github repo: https://github.com/TARTRL/TiKick
- This repository implements MAPPO, a multi-agent variant of PPO: https://github.com/marlbenchmark/on-policy
- sparse rewards seems like one of the biggest problems in RL.

- From the paper: “there remain many problems in building agents for the GRF: (1) Multiple Players: In the GRF, there are both cooperative and competitive players. For cooperative players, the joint action space is very huge, thus it is hard to build a single agent to control all the players. Moreover, competitive players mean that the opponents are not fixed, thus the agents should be adapted to various opponents. (2) Sparse Rewards: The goal of the football game is to maximize the goal score, which can only be obtained after a long time of the perfect decision process. And it is almost impossible to receive a positive reward when starting from random agents. (3) Stochastic Environments: The GRF introduces stochasticity into the environment, which means the outcome of taking specific actions is not deterministic. This can improve the robustness of the trained agents but also increases the training difficulties.”
- I should contact the game creators to ask why the checkpoint reward is not working, and maybe I can make my own reward!
- Actually, defining my own reward function is kind of cheating the whole idea. or is it? because if the rewards function gives a higher reward the closer you are to the goal, isn’t this the same as hard-coding it such that the player goes closer to the goal? I think that’s not necessarily true.
- I found this in wrappers.py:

```
class CheckpointRewardWrapper(gym.RewardWrapper):

  """A wrapper that adds a dense checkpoint reward."""

          def __init__(self, env):

                   gym.RewardWrapper.__init__(self, env)

                    self._collected_checkpoints = {}

                  self._num_checkpoints = 10

                  self._checkpoint_reward = 0.1
```

- I guess the checkpoint reward is 0.1 but when does it apply?!
- after line 322 in wrapper.py they compute the reward based on distance but it is very complicated. I guess there is no need to write my own reward function.
- What is a wrapper? e.g. a simple question; we got two types of array sorting techniques, lets create a wrapper for it.
  The instance you describe, where you have two classes which can do array sorting using different algorithms sounds like the 'Strategy' pattern, where you provide a way to perform an operation on some object, but the algorithm used for that operation may be different depending upon the structure of that object.
  For example, one of your sort algorithms may be great for arrays of length less than 100, but then performance might be an issue for it. The other algorithm might be good for bigger arrays. You can create a 'wrapper' for the two algorithms which supports the sort operation but decides which algorithm to use based on the array length.
  The vast majority of wrappers exist to hide some sort of complexity.

- IT WORKS!! The checkpoint reward funciton finally works! how did it not work before? It also works in the empty goal scenario! I think I know why it did not work before. There should be no space between scoring and checkpoint: `rewards='scoring,checkpoints'`
- “CHECKPOINT is a **reward by encoding the domain knowledge that scoring is aided by advancing across the pitch**. More specifically, we divide the opponent's field in 10 checkpoint regions according to the Euclidean distance to the opponent goal. NOTE: CHECKPOINT rewards are only given once per episode.”
- once per episode means I should try to make the episode as short as possible. I will check those episode ending options to see what can also be changed.

- Also, the way I tested the model was very time-consuming and such a manual process. If I want to plot graphs I should use matplotlib just like the minor module this term. I just need to automatically save the figure and as a back-up I should also write the test data to a text file.
- ok I give up, I don't know where to find them. in academy_empty_goal.py, one of the lines is:

  ```
  builder.config().end_episode_on_possession_change = True
  ```

- I checked config.py and ending episodes is not from there, i don't know where to find it. That's ok, ending the episode on possession change is really good! I just need to add that to 5_vs_5 scenario or whatever I will be using!
- From the paper:

  "As for the observations, we construct a 268-dimensional vector (as described in Section 3.1) from the raw observation for each player. As for the actions, we extend the original 19 discrete actions to 20 discrete actions by adding an extra build-in action. All the non-designated players will be assigned the build-in action. When the player takes the build-in action, the player will behave like a build-in agent. Currently, such build-in agents are obtained from a rule-based tactic, and we will try to convert it to a learning-based controller in future work. To be clear, the build-in action is only used for the full game and dropped for all the academic scenarios."

- To train a policy controlling multiple players, one has to do the following:
- pass number_of_players_agent_controls to the 'create_environment' defining how many players you want to control
- instead of calling '.step' function with a single action, call it with an array of actions, one action per player
- It is up to the caller to unpack/postprocess it in a desirable way. A simple example of training multi-agent can be found in examples/run_multiagent_rllib.py

January 16th, 2022:

- Worked on the remote server issue. I realised that the virtual environment I was using in the server did not have a bin folder so I could not activate it.
- I moved my linux virtual environment that works to my windows OS then to the server (by zipping and unzipping). Then I tried to activate it but got an error message saying: Missing }.
- I then ignored that and typed: pip freeze > requirements.txt, which writes the libraries in that text file for them to be installed. The issue is that gfootball library was not in the text file!
- I typed: pip install -r requirements.txt. It worked but all the libraries were already there! it said: requirement already satisfied.

January 17th, 2022:

- I figured out how to train a policy with more than one player. It is by passing `number_of_left_players_agent_controls=2` to `football_env.create_environment()`
- I then need to pass an array of actions when calling `env.step()` instead of one action.
- it seems like adding the extra player will produce some errors within the code, starting with:

  n_actions = env.action_space.n

  AttributeError: 'MultiDiscrete' object has no attribute 'n'

- I will find the MultiDiscrete class in the source code and see what is the case with more than one player.
- I did a nice trick: Changed it back to one player and changed the code and got this error:

  n_actions = env.action_space.m

  AttributeError: 'Discrete' object has no attribute 'm'

- The goal was to know what class it is for a single player and it is Discrete!

- After googling, it seems that these are classes from the gym environment. I found the docs: https://github.com/openai/gym/blob/master/gym/spaces/discrete.py

  https://github.com/openai/gym/blob/master/gym/spaces/multi_discrete.py

- This line of code in discrete.py shows n being an attribute: `def __init__(self, n, seed=None, start=0):`
- In multi_discrete.py, it is nvec: `def __init__(self, nvec, dtype=np.int64, seed=None):`
- While editing the code and removing errors, the most difficult one so far is the actual neural network, or building the model. I don't understand how to use functions like Dense() etc.
- By looking at the model summary, the output is already a 2 by 19 array, which is good! the error message i get is: ValueError: Error when checking input: expected input_2 to have shape (2, 19) but got array with shape (1, 19).
- Input_2 turns out to be the `oldpolicy_probs` array.
- I found out that when two players are trained by the policy, the state dimensions for simple115 become (2,115), meaning that there is a state representation per player! isn't this redundant?

'simple115'/'simple115v2': the observation is a vector of size 115.

It holds:

- the ball_position and the ball_direction as (x,y,z)

- one hot encoding of who controls the ball.

  [1, 0, 0]: nobody, [0, 1, 0]: left team, [0, 0, 1]: right team.

- one hot encoding of size 11 to indicate who is the active player

  in the left team.

- 11 (x,y) positions for each player of the left team.

- 11 (x,y) motion vectors for each player of the left team.

- 11 (x,y) positions for each player of the right team.

- 11 (x,y) motion vectors for each player of the right team.

- one hot encoding of the game mode. Vector of size 7 with the

  following meaning:

  {NormalMode, KickOffMode, GoalKickMode, FreeKickMode,

  CornerMode, ThrowInMode, PenaltyMode}.

Can only be used when the scenario is a flavor of normal game

(i.e. 11 versus 11 players).

- From the explanation of simple115 above, it seems that only one state is enough! no need to add one for each player as they will be identical anyways.. well, I don't think I can control that and it also does not matter much. I can print both states during training to see if they are identical or not :)
- Spending a lot of time re-formatting the structure of the data being used to train the model. This is necessary in order to apply training to more than one player.
- The whole code finally runs! I noticed something when rendering, the game suddenly resets mid-attack, which wastes a whole iteration sometimes. I should make it only reset when the ball is out. The issue is that I might not have control on resetting, as this depends on the boolean value of the variable (done), which comes from env.step(). I think the only thing I can do is to increase the steps in one iteration.

January 18th, 2022:

- Had a meeting with Prof Kit. Suggested plan on multi-agent training: train one team against bots, starting from the easiest difficulty then increasing the difficulty and developing the policy. the easiest opponent would be the lazy bot that does not move! That could be a good start! Also, 2 vs 2 game + keeper will be the scenario used. If decreasing the size of the stadium was possible, I will also do it.

January 19th to 21st, 2022:

- worked on ucl's remote server issue. Almost figured out how to import the gfootball library on the server by following the instructions on this page: https://intranet.ee.ucl.ac.uk/it/faq/deep-machine-learning
- I got permission denied errors when trying pip3 commands.
- I got in touch with the IT services in our department and got offer some help. They are still working on it and it turns out that they need to install the SDL (Simple DirectMedia Layer) dependency on the server as it won't work without it.

January 22nd, 2022:

- I created a new scenario called 3_vs_3.py to carry out my experiments in. Each team has two players and a goalkeeper.
- I should not include this line of code in 3_vs_3.py because the game resets every time my team loses the ball: `builder.config().end_episode_on_possession_change = True`

Up to February 7th, 2022:

- IT services in our department working on making the game run on their servers. They managed to make it work only on one server called medusa, they will ask for permission from the owner of the server if I can use it.
- While they solve the issue, I researched some alternative algorithms other than the PPO used. I also tried to reduce the size of the stadium in the game to run it on my PC but I could not. I should try again later.
- The solution on the medusa server requires the use of docker. Docker is completely new to me so I started learning how to use it and trying to figure out how to copy my files to the container and run my code (the IT services said I have to do this myself).

February 8th, 2022:

- Still trying to solve the issue of using Docker.
- This video is useful! Docker Tutorial for Beginners [2021]
- docker ps shows you the list of RUNNING containers. I don't have any containers running.
- docker ps -a shows all the containers, not just the ones running! Things are starting to make some sense now.
- docker run -it gfootball (this runs a container in interactive mode and loads the gfootball image)
- root@9ecd69888fb5:/gfootball#
- root user has the highest privileges, 9ecd69888fb5 is the name of the machine, or the container ID, which is automatically generated by docker. :/ says where we are in the file system. / represents the root directory, the highest directory in the file system. # means highest privileges. if it was a normal user, it will be a $.
- history is a command that shows a list of recent commands. then you can repeat them by typing ! followed by the command index.
- pwd = print working directory
- ls = list
- touch command creates a new file
- mkdir creates a new folder, mv renames the folder
- I Finally understand how to use docker!! I don't know how the IT services did it last time with me somehow, but I can do it with 2 open server sessions. one of them runs the gfootball docker, and the other copies the file into it! maybe I have to copy my files every time I use it!! we can solve this later.
- For now, it says my code cannot run because keras is not installed!
- After installing keras, i get this error:

```
  File "simple_test.py", line 2, in <module>
```

```
    from keras.models import load_model
  File "/usr/local/lib/python3.6/dist-packages/keras/__init__.py", line 25, in <module>
    from keras import models
  File "/usr/local/lib/python3.6/dist-packages/keras/models.py", line 19, in <module>
    from keras import backend
  File "/usr/local/lib/python3.6/dist-packages/keras/backend.py", line 39, in <module>
    from tensorflow.python.eager.context import get_config
ImportError: cannot import name 'get_config'
```

- I think it is because the python version is 3.6! while i use python 3.8 on my linux!
- Acually, it is much simpler than that! I had the same issue on my linux a while back and a simple fix is to import from tensorflow.keras instead of keras!
- New error:

```
error: XDG_RUNTIME_DIR not set in the environment.
error: XDG_RUNTIME_DIR not set in the environment.
Failed on SDL error: Video subsystem has not been initialized
You can solve this problem by:
1) If you are running inside Docker make sure container has access to XServer by running (outside of the container):
  > xhost +"local:docker@"
2) Switch to off-screen rendering by unsetting DISPLAY environment variable
3) Disable 3D rendering (which makes environment run much faster).
```

- I will try without rendering now.
- I got rid of the error! This might be the error that the IT services were getting! there might be no need for using docker and medusa because rendering is not necessary! I will text him in the morning.
- New error due to loading a model that does not exist, but that's fine, because my models are not in the server yet. I will run the training code instead now.
- It works!!! I can finally run my code with no issues on UCL's medusa server!!

February 8th, 2022:

- It turns out that the error the IT services were getting was a different one. This means that I have to use docker on medusa, but that's ok!

February 16th, 2022:

- I can say that I tried to define my own reward function but it was too complicated and the source code was not commented enough to be understood

- I am submitting the progress report.

February 17th, 2022:

- I'm trying to save a video of an episode by writing dumps in the environment.
- I could screen record the rendering on linux using ctrl+alt+shift+R.

- in the scenario file, when calling .Addplayer(), adding the argument lazy=True makes the player lazy, meaning: `lazy: Computer doesn't perform any automatic actions for lazy player.`
- by testing, the argument works on players that are not controlled by an agent.
- I decided not to make the opponent lazy as a first step in training. This is because when training starts and the lazy opponent has the ball, a lot of training time will be wasted since our players do not know how to tackle and get the ball.
- A good idea to speed up training might be to let the opponent play without a goalkeeper! This is a very good idea because shooting in this game is only a single button (it is not as complex as other football games), meaning that there aren't different strength levels to a shot, and shooting is always towards the goal by default. Therefore, letting our players receive a reward of +1 by doing that, rather than a goalkeeper saving it and our model not knowing it was a good thing to do, is indeed something worth doing.
- ACTUALLY, since I can train on the server simultaneously, I might do both at once!!! That could be the main comparison in my research!
- I should consider the stacked representation where the environment does 4 runs simultaneously to reduce training time.
- in python, import * means import all.
- I tried to track down where the source code of `builder.config().right_team_difficulty = 0` and similar lines are. I found that it is called from gfootball/env/scenario_builder.py, which in turn imports from gfootball_engine. Now, I'm trying to find that within gfootball_engine.
- while looking at gfootball_engine, I found .cpp and .hpp files. cpp files have distinct purposes, and hpp files contain declarations, while .cpp files contain implementation of what is declared in the .hpp file.
- I found something regarding team difficulty in match.cpp as: `GetScenarioConfig().left_team_difficulty);`
- it could probably be traced through the imports, but it doesn't matter for now.
- I will test training on different ppo steps to see what number of steps is suitable for a 3vs3 game. The best would be the average steps for one attack towards the goal.
- I have a very bad issue. I just tested training on the medusa server, and the speed is almost exactly equal to that of my local machine! In fact, it also experiences the same problem of becoming extremely slow after just a couple of iterations!
- I will go back to my machine and try to pinpoint where exactly in my code does the program start to become slow!
- ok, after testing different positions in the code (using the time library in python), it turns out that model.predict() takes more time as the code runs, which means as more data gets collected, predicting takes a longer time. For example, after 3 iterations of 256 steps each, the delay is around 0.14 seconds for predicting the actor and critic.
- ok, so I found the model.predict() function in the libraries and it is in line 1603 of tensorflow/python/keras/engine/training.py
- OMG! after doing some research, it turned out that calling model_actor() directly rather than model_actor.predict() is so much faster! it completes 256 steps in one second!
- this however provides the prediction in the form of a tensor, which is converted to a numpy array using .numpy(). This conversion is only possible by enabling eager execution! Therefore, I now have to deal with the error in model.fit() which I solved before by disabling eager execution.

February 20th, 2022:

- I'm trying to resolve this error now: TypeError: You are passing KerasTensor(type_spec=TensorSpec(shape=(), dtype=tf.float32, name=None), name='Placeholder:0', description="created by layer 'tf.cast_4'"), an intermediate Keras symbolic input/output, to a TF API that does not allow registering custom dispatchers, such as `tf.cond`, `tf.function`, gradient tapes, or `tf.map_fn`. Keras Functional model construction only supports TF API calls that *do* support dispatching, such as `tf.math.add` or `tf.reshape`. Other APIs cannot be called directly on symbolic Kerasinputs/outputs. You can work around this limitation by putting the operation in a custom Keras layer `call` and calling that layer on this symbolic input/output.
- Most people online suggest that it is an issue with tensorflow and numpy versions' compatibility.
- This is the same error I got in december 27th, which I solved by disabling eager execution.
- I tried with numpy 1.20 and 1.19 but I got the same error.

- I can't find a solution to this error. I will disable eager execution and try to convert the tensors into numpy arrays. even if I had to access the values in the tensors and assign them to a new numpy array variable.
- Ok, there are two possible ways to convert from a tensor to a numpy array. First one is to use .eval() and make some sort of graph, which is complicated, the other is to access each element in the tensor then use .item() on it to access only the value without the tensor, then assign that to my numpy array.
- I get an error: Tensor has no attribute item(). this is because the tutorial i saw uses item() on Torch.
- np.array(tensor) also only works when eager execution is enabled.
- ok it seems like I have to use .eval()
- I tried many methods, such as this: https://stackoverflow.com/questions/61006702/cannot-use-the-given-session-to-evaluate-tensor-the-tensors-graph-is-different and this: .eval(session=tf.compat.v1.Session())
- They all do not work, they only work when i use random variables such as y = tf.constant(2) but not when I call model(x)

February 21st, 2022:

- I uninstalled tensorflow-gpu 2.7 and installed tensorflow-gpu 2.4 but now I get an error that tensorflow cannot be imported. I then installed tensorflow-gpu, which is 2.8, and it worked, but I still get that error when using model.fit()
- I also installed the latest version of numpy (1.22.2) but I still get the same error.
- This page suggests an alternative solution: https://stackoverflow.com/questions/69033906/tensorflow-model-fit-format-data-correctly-typeerror-cannot-convert-a-symbol
- The solution is: Disabling eager execution is not a satisfying solution.

  I suggest you try converting training_data as a Dataset with tf.data.Dataset class or as a tensor with the tf.Tensor class prior to model.fit.

- Before trying out this, I will try running it on the server.
- server gave me the error that the 3vs3 scenario does not exist, which is true because I did not add it yet. So I changed it to 5vs5.
- Then, I get an error: AttributeError: 'Tensor' object has no attribute 'numpy'
- I think it's because the tensorflow version is 1.15.2. Yep, this is why: Note that eager execution is enabled by default in TensorFlow 2.0. So the answer above applies only to TensorFlow 1.x
- I should type: tf.enable_eager_execution()
- I get a new error!in line 293, which is still under model.fit() but I think it successfully passes the error I used to get about the tensors! the new error is : tensorflow.python.framework.errors_impl.NotFoundError: Resource localhost/logdir:./logs/N10tensorflow22Summar yWriterInterfaceE does not exist. [Op:WriteGraphSummary]
- numpy version on server is 1.18.1
- I removed callbacks=[tensor_board] from model_actor.fit(), to see what error message I will get, and now it is: "tensors, but found {}".format(keras_symbolic_tensors)) tensorflow.python.eager.core._SymbolicException: Inputs to eager execution function cannot be Keras symbolic tensors, but found [<tf.Tensor 'input_2:0' shape=(?, 2, 19) dtype=float32>, <tf.Tensor 'input_3:0' shape=(?, 2, 1) dtype=float32>, <tf.Tensor 'input_4:0' shape=(?, 2, 1) dtype=float32>, <tf.Tensor 'input_5:0' shape=(?, 2, 1) dtype=float32>]
- seems like it is the same error after all!

February 22nd, 2022:

- I will install the same versions of tensorflow and numpy on my machine and see if it works.
- I cannot install tensorflow 1.15.2 on my machine, probably because of python's new version.
- I tried converting the inputs to tf.data.dataset and now I get this error: ValueError: Failed to find data adapter that can handle input: (<class 'list'> containing values of types {"<class 'tensorflow.python.data.ops.dataset_ops.TensorSliceDataset'>"}), (<class 'list'> containing values of types {"<class 'tensorflow.python.data.ops.dataset_ops.TensorSliceDataset'>"})
- I tried tf.TensorArray and I still get the same error. It seems like I need to take a step back and learn about Keras, especially what kind of data types are valid inputs.

- I think a good idea is also to look for example codes that make a keras model without using .predict() and see how they use model.fit().
- from https://www.tensorflow.org/api_docs/python/tf/keras/Model : Input data. It could be:

1. A Numpy array (or array-like), or a list of arrays (in case the model has multiple inputs).
2. A TensorFlow tensor, or a list of tensors (in case the model has multiple inputs).
3. A dict mapping input names to the corresponding array/tensors, if the model has named inputs.
4. A tf.data dataset. Should return a tuple of either (inputs, targets) or (inputs, targets, sample_weights).
5. A generator or keras.utils.Sequence returning (inputs, targets) or (inputs, targets, sample_weights).

- I have a genius idea to workaround this issue! I can disable eager execution in between predicting and fitting the model!! Will it work? Let's try.

- Ok, I got this error: ValueError: Calling `Model.fit` in graph mode is not supported when the `Model` instance was constructed with eager mode enabled. Please construct your `Model` instance in graph mode or call `Model.fit` with eager mode enabled.

- so this means the model must be created and fit within the same method (either enabled or disabled). That's fine! I can just enable eager execution when I want to use .numpy()

- it seems like this is not possible: ValueError: tf.enable_eager_execution must be called at program startup.

- Ok, nothing is working, I am looking for alternatives to Keras, such as PyTorch.

- my python version is 3.8.10, that's why i can't install tensorflow 2.4 since it is only..? I forgot to continue typing.

February 23rd, 2022:

- I have two options now, either try to covert tensors to numpy arrays when eager execution is disabled (not using .numpy(), or try different versions of tensorflow and numpy.
- I got this error [Cannot convert a symbolic Keras input/output to a numpy array]. using tensorflow 2.4.1 and numpy 1.19.5.
- This is one of the solutions if you use tf2.x and you dont want to close tf eager_execution. Convert your loss function to a loss layer, and make the parameters advantage and old_prediction as Input layer. for example,

  ```
  class PPO_loss_layer(tensorflow.keras.layers.Layer):

  def call(self,y_true, y_pred,advantage, old_prediction):

  ....

  y_true = Input(...)

  advantage= Input(...)

  old_prediction= Input(...)

  loss_layer = PPO_loss_layer()(y_true, y_pred,advantage, old_prediction)

  model = Model(inputs=[y_true,advantage,old_prediction],outputs=loss_layer )
  ```

  from: https://lifesaver.codes/answer/typeerror-cannot-convert-a-symbolic-keras-input-output-to-a-numpy-array-47311

  - I think the best thing to do is understand this possible solution and implement it. it might be the same suggested solution in the error I get: [You can work around this limitation by putting the operation in a custom Keras layer `call` and calling that layer on this symbolic input/output.]

- I can confirm that I confidently found the source of the problem. In my code, model_critic.fit() works perfectly, but model_actor.fit() gives the error. Why? Because when compiling the models (model.compile()), the critic uses a simple mse loss, whereas the actor uses the ppo loss function. This confirms that the above suggested solution is accurate and I should follow it.
- Dense() is used to make hidden layers and the output layer in the model. From this tutorial: https://www.youtube.com/watch?v=wQ8BIBpya2k
- I understood something else (by going back to the ppo GRF tutorial).. The error does not say that the inputs in the main code are the tensors! It refers to the tensors defined in the function `get_model_actor_simple(input_dims, output_dims):`
- From this tutorial: https://www.tutorialspoint.com/keras/keras_customized_layer.htm I learnt how to make the custom loss layer. This is the code:

```
from keras import backend as K from keras.layers import Layer

class MyCustomLayer(Layer):

   def __init__(self, output_dim, **kwargs):

      self.output_dim = output_dim

      super(MyCustomLayer, self).__init__(**kwargs)

   def build(self, input_shape): self.kernel =

      self.add_weight(name = 'kernel',

      shape = (input_shape[1], self.output_dim),

      initializer = 'normal', trainable = True)

      super(MyCustomLayer, self).build(input_shape) #

      Be sure to call this at the end

   def call(self, input_data): return K.dot(input_data, self.kernel)

   def compute_output_shape(self, input_shape): return (input_shape[0],
self.output_dim)
```

- the __init__ function shows that the output dimension should be passed when making an instance of the class.
- This Tensorflow playlist tutorial is good, I watched some of it and learnt some useful stuff to use, such as: Sequential is used for models with one input and one output. Also, how to edit the font on PyCharm. Also, how to ignore outputted tensorflow messages. https://www.youtube.com/watch?v=5Ym-dOS9ssA&list=PLhhyoLH6IjfxVOdVC1P1L5z5azs0XjMsb
- While I learn how to use Keras, I'm training the agent on the empty goal scenario using the simple representation, making use of the fast training now by using Model() instead of Model.predict(). I'm training it for 100k steps, split into 781 iterations of 128 steps each. Next, I will try to train using Pixels and find the difference in time. Maybe also when rendering is on!
- training on simple representation took 743s or 12.4 minutes! I will try using pixel representation, after testing the 100k step agent
- I got an error importing load_model, I think because of the tensorflow version, I'll reinstall the new one.
- I was right, I re-intalled tensorflow 2.7 and it works.
- The 100k step MSE model using simple representation is really bad, as expected, due to the fact that rewards are only considered for immediate actions, not actions that come before it, so the advantages function is not being used, unlike when PPO is used.
- I even tried a saved model that scored a goal in training, still same result.
- What if I try using model.predict() in testing? I tried and the model is still bad. However, time taken for each prediction using model.predict() = 0.04s, compared to 0.002s.
- I should also load the model and continue training it, to see if that increases predicting time when using model(x) and model.predict(). Ok, using model(x), it takes around 0.002 seconds and one whole iteration was 1.8 seconds. As for model.predict(), it takes around 0.03 seconds to predict and

one whole iteration was 10.2 seconds. This was important to see whether Model(x) scales or not. As written online, they say it does not scale well, but given that 100k steps is enough, it does. Unless they mean something else.

February 24th, 2022

- I am currently watching the tutorial series for keras.
- In Tutorial 3, min7, Dense() is used to create the layers with the activation function relu, but the output layer does not have an activation function, even though it should have softmax, why? because it will be done inside the loss function (EXACTLY what I need to do in my case!).
- Actually, I'm not sure, it depends if his loss function operates differently that the custom PPO layer we want to create.
- Go back to the PPO GRF tutorial and check where he got the keras code from, I remember he referenced someone.
- printing model.summary() in between layer creations helps in debugging
- I finally understand it! my keras model uses Functional API which is more flexible than Sequential API. I must include this in the final report.
- I can name the layers using name=.
- using a for loop to print the features of the model (for each layer) is useful for debugging.
- I realised my neural network is simply a default one! I need to look up a more suitable one for Reinforcement Learning. Here is an example code for Actor Critic model in keras's website, and it shows that my model should be fairly ok: https://keras.io/examples/rl/actor_critic_cartpole/ here is a PPO example too: https://keras.io/examples/rl/ppo_cartpole/
- Convolutional Neural Networks have applications in image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series.
- **Regularization** is a set of techniques that can prevent overfitting in neural networks and thus improve the accuracy of a Deep Learning model
- Dilution (also called Dropout or DropConnect) is **a regularization technique for reducing overfitting in artificial neural networks by preventing complex co-adaptations on training data**.
- all above are from Tutorials 4 and 5, which is not too relevant to my case. Tutorial 6 talks about RNNs (Recurrent Neural Networks), GRUs (Gated recurrent units), LSTMs (long short-term memory" units), and Bidirectionality, which is: Bidirectional long-short term memory(Bidirectional LSTM) is the process of making any neural network o have the sequence information in both directions backwards (future to past) or forward(past to future).

- I still don't understand how I get two outputs for my actor and critic (one for each player) even though I do not explicitly define it to! The only explanation is that the input is now two states instead of one. Well, I pass the output dimensions to get_model_actor_simple(), but I only use it to define oldpolicy_probs, and not the output! one way to perform a sanity check would be to print the result of predictions and make sure it is different for each player (it should be different otherwise they take the same actions everytime).

March 3rd, 2022:

- A solution proposed to my issue is: Also, in model.compile(...) use experimental_run_tf_function=False. I will try it now.
- it seems like the argument has been removed from compile() and it was available in older tf versions maybe, it can still be typed without an error but seems like they set it to always True (line 2772 of keras/engine/training.py)
- based on https://github.com/tensorflow/tensorflow/issues/35138 , it can be used in TF2.1. I will try it on the server since on my laptop I need an older version of python and TF in the server is 1.15.2.
- Good news! IF experimental_run_tf_function=True, I get this error message:

```
"tensors, but found {}".format(keras_symbolic_tensors))
```

```
tensorflow.python.eager.core._SymbolicException: Inputs to eager execution function cannot be Keras symbolic tensors, but found [<tf.Tensor 'input_2:0' shape=(?, 2, 19) dtype=float32>, <tf.Tensor 'input_3:0' shape=(?, 2, 1) dtype=float32>, <tf.Tensor 'input_4:0' shape=(?, 2, 1) dtype=float32>, <tf.Tensor 'input_5:0' shape=(?, 2, 1) dtype=float32>]
```

- IF experimental_run_tf_function=False, I can get past model.fit() without an error!! I get another error after the second iteration:

```
 _six.raise_from(_core._status_to_exception(e.code, message), None)

  File "<string>", line 3, in raise_from

tensorflow.python.framework.errors_impl.NotFoundError: Resource localhost/logdir:./logs/N10tensorflow22SummaryWriterInterfaceE does not exist. [Op:WriteGraphSummary]
```

- It's clear that this error is not related to the main issue! even though the error is in the critic model.fit(). A simple fix would be to try TF2.1 either on the server or my machine.
- I tried TF2.1 on the server and it works! I ran the whole code with no errors!!! Next step: install whatever python version is compatible with TF2.1 and use it to train the agent for now, then come back to the server and fix issues such as having to copy files to docker every time, making a models folder, and save models properly, and move them out of the server to my linux machine (either through github or through my laptop). I should try using github on the server!

March 5th, 2022:

- I managed to connect to medusa's docker using the command prompt on UCL's computers. the python version in docker is 3.6.9. I will install the same version on my linux to install TF2.1.
- Current python version on my linux is 3.8.10.
- I installed python 3.6.9 on linux, but I don't know how to use it to install TF2.1. even after creating a new venv for my project, it still uses python 3.8 by default, but uses 3.6 if I use the terminal and type python3.6 file.py
- I will just use the server as its errors seems easily fixabled. and use my linux with python 3.8 to test the models! I only need to save the last model.
- Successfully cloned my repo into the server, and created a folder to collect saved models.
- It works! The mode is currently training for the empty goal scenario for 100k steps. It is extremely fast as well, a single iteration (128 steps) takes less than a second!
- "Containers that need to store data **permanently** must use **Docker** volumes"
- if the screen shuts down, the connection to the server stops and training halts.
- Training is done! time taken to finish whole training: 1092.166493654251s
- 100k steps were done in 18.2 minutes!
- Note: most of the time is in between iterations (when the epochs are printing) I need to do something about that maybe.
- After testing model_actor_780 (781st iterations with 100k steps), it only performs action 6 and has this probability distribution:

```
[[7.3878120e-08 4.1386321e-20 4.4386667e-19 1.2971148e-16 1.5951541e-07

  5.6914121e-18 9.9999583e-01 3.8132897e-24 8.0598659e-18 1.0729903e-15

  3.9525826e-06 5.7630979e-24 1.0158150e-19 1.5642817e-20 7.4843652e-24
```

```
 6.4169954e-25 3.3468648e-20 1.4235481e-16 6.2303438e-18]]
```

- This is another example of the action probability distribution:

```
probs =[[7.3073004e-08 4.3480500e-20 4.6354395e-19 1.3761179e-16 1.5946249e-07
  5.9495731e-18 9.9999583e-01 4.1462403e-24 8.2272707e-18 1.1014231e-15
  3.9285987e-06 6.1242641e-24 1.0489524e-19 1.6058188e-20 8.1346532e-24
  6.8590825e-25 3.5611699e-20 1.5002777e-16 6.4428953e-18]]
```

- It shows that the model reached a point where it thinks action 6 is always the best action to use, with a probability of 99.999%! I think this is caused by the addition of the checkpoint reward function, which is completely unnecessary in the empty goal scenario. It also shows that 100k steps should be enough to solve the problem! Next step: do the same thing but remove the checkpoint reward function.

March 6th, 2022:

- I run the same scenario to train for 1M steps overnight. Unfortunately, the wifi disconnected after training was done so models were not saved. However, I have the terminal output saved as a word doc.
- time taken to finish whole training: 10516.5s = 175 mins = 2 hours and 55 mins. This is good! it means that time increases lineary (comparing with 18mins for 100k steps), as opposed to using model.predict(), which was increasingly time consuming as training went on.
- unfortunately, even the terminal output saved starting from iteration 1496. it seems like all the rewards are +1, which means that the agent solved the scenario before the 1500th iteration! This means that it was solved within 191k steps.
- This tensorflow message gets printed after every iteration:

```
tensorflow/core/profiler/internal/gpu/cupti_tracer.cc:1307] function cupti_interface_->Subbe( &subscriber_, (CUpti_CallbackFunc)ApiCallback, this)failed with error CUPTI could not be loaded or symbol could not ound.

2022-03-06 04:36:18.748665: E tensorflow/core/profiler/internal/gpu/cupti_tracer.cc:1346] function cupti_interface_->ActyRegisterCallbacks( AllocCuptiActivityBuffer, FreeCuptiActivityBuffer)failed with error CUPTI could not be loaded or symcould not be found.
```

- From online: I had a similar error when trying to get tensorboard graph, I think it only affects you if you plan to use tensorboard.
- I think I can ignore the message.
- I will train again but only for 200k steps since more than that in unnecessary.
- For some reason, one of the images opened yesterday is still running! I found a way to access it which is by: https://stackoverflow.com/questions/20813486/exploring-docker-containers-file-system

```
# find ID of your running container: docker ps

# create image (snapshot) from container filesystem

 docker commit 12345678904b5 mysnapshot

# explore this filesystem using bash (for example)

 docker run -t -i mysnapshot /bin/bash
```

- IT WORKS! I can also check files in docker containers that are not running anymore! I opened the container I trained 1M steps on yesterday and I found my git repo and all the saved models!
- from the saved models, the rewards were consistently 0 up until the 490th iteration, after that it is consistently a reward of 1 (iteration 500 had a reward of 0). This means it took around 500*128 = 64k steps for the PPO model to solve the empty goal scenario! This is my first result and this is a very good one!

- from the paper: "The scenarios where agents have to score against the empty goal (Empty Goal Close, Empty Goal, Run to Score) appear to be very easy and can be solved both PPO and IMPALA with both reward functions using only 1M steps. " The key to solve it was only using the scoring reward function! Confusing the agent with rewards that are more detailed than needed is a disadvantage here. To illustrate this, after 100k steps of training using the checkpoint reward, the player was moving towards the corner, which is close to the goal so he receives a high reward (close to 1). This makes the scoring reward of 1 become not so important. If the scoring reward could be scaled to like +10 then using both would result in a better combination!
- Ok, I solved the issue of copying files to docker every time! I'll just this container ID every time: dbc38f8261ec
- If I ssh into the server using UCL's computer and lock the PC then it does not get disconnected, but if I use my laptop then it disconnects after the screen goes black. It might be a good idea to use the PC in UCL but find a way to let it not sign out automatically. I think I can do that in the Robert's building PCs (like the mech eng ones).
- I tried testing a model using model() and model.predict(), same choice of actions but .predict() is slower when both are rendering.
- This is the action probability distribution when using model(): (100k steps both reward functions)

```
[[3.27980582e-04 8.21312527e-16 7.34094021e-16 1.91411814e-13
  4.22871020e-03 9.38587976e-15 9.76639569e-01 7.63175008e-21
  2.12753399e-14 2.03689669e-12 1.88037772e-02 2.02128013e-20
  4.77440020e-16 5.22155313e-17 1.89211426e-20 7.35782729e-21
  1.01655944e-16 6.67580791e-13 7.81090755e-15]]
```

- This is the action probability distribution when using model.predict():

```
[[3.27980582e-04 8.21312527e-16 7.34094021e-16 1.91411814e-13
  4.22871020e-03 9.38587976e-15 9.76639569e-01 7.63175008e-21
  2.12753399e-14 2.03689669e-12 1.88037772e-02 2.02128013e-20
  4.77440020e-16 5.22155313e-17 1.89211426e-20 7.35782729e-21
  1.01655944e-16 6.67580791e-13 7.81090755e-15]]
```

- They are exactly the same, which means there is no difference between them as far as we can tell! I would say it is the same case during training as well.
- I will now test the models from the 1M steps of only scoring reward function.
- This is the action probability distribution of the final saved model (1M steps):

```
[[4.6844081e-29 1.0242690e-30 1.7417108e-28 2.9709256e-25 2.1288327e-29
  1.0000000e+00 2.8984759e-17 3.0024972e-17 9.9093939e-31 8.1520618e-29
  2.5548047e-17 1.9075100e-28 2.3034560e-32 1.8866539e-28 1.0399123e-27
  1.9283614e-28 3.7554880e-17 1.6558234e-24 1.0745699e-26]]
```

- It solved the scenario by favouring action 5, which is running to the right, as shown by the array above.
- As for iteration 500, this is the test of its model: it also runs towards the goal but sometimes it moves to the bottom and misses:

```
[[3.9161279e-13 9.3694963e-14 3.6734047e-14 3.3657216e-10 5.6593949e-13
  4.5169646e-01 1.7383181e-02 9.3733910e-03 1.5260873e-13 4.1935639e-12
  2.7926471e-03 5.0182298e-14 2.9080870e-14 7.5707059e-14 1.2520643e-12
  4.4623466e-13 5.1875436e-01 4.4730092e-12 1.1195610e-13]]
```

- As for iteration 400, it shoots towards its own goal and this is the distribution:

```
[[2.3658203e-10 1.6268178e-11 5.8721309e-12 1.3059875e-07 2.3169434e-10
  1.8349461e-01 2.4905980e-01 1.9582178e-01 6.0546887e-11 2.0686410e-09
  1.5216345e-01 9.0456644e-12 8.9032497e-12 8.8252071e-12 4.3498907e-10
  6.7601647e-11 2.1946022e-01 1.1950859e-09 4.4441676e-11]]
```

- I think using model() instead of model.predict() increased exploitation in training and decreases exploration, how? I am not sure, but it should be the case because all models before 490th iteration had 0 reward! so no goals scored randomly! It might be due to the experimental_run_tf_function but i am not sure.
- This is the code used to train the model for 1M steps: (actually no need to paste it here because it is very long. If I include it in the report I can check the github commit around today's date.
- Now i am preparing the code to train 3 vs 3. I am trying to understand the point of this part of the scenario file:

```
if builder.EpisodeNumber() % 2 == 0:
 first_team = Team.e_Left
 second_team = Team.e_Right
else:
 first_team = Team.e_Right
 second_team = Team.e_Left
```

- I added print statements and what happened was that the Else part gets executed 4 times when the environment is being created, then the IF part gets executed once when the game starts, so that should mean the first team is on the left. This is also confirmed by adding more players to the second team to check.
- The lazy parameter only works on players not controlled by your agent. The opponent players must not be lazy! they had a corner and a lazy player went to play it, and they never do, so the game will finish in that corner and the whole game is wasted. Same thing with the goalkeeper, if they are lazy they never play the goalkick!
- The best thing to do now is to train 2 models, one with a normal 3 vs 3 scenario, but the other 3 vs 2 such that the opponent has no goal keeper! This will speed up training as any shot on goal will be a reward of +1.
- For some reason, if I remove the goalkeeper from the scenario, the game does not start!
- Ok all scenarios must have a goalkeeper, they get rid of him by letting him respawn far away from his goal. That might work but it does not help much because he starts running towards his goal when the game starts.
- something worth adding to the report: Trade-off: how easy should the game be? if we remove the goalkeeper or the offside rule, our agent will score more and learn faster. However, they will get used to this level of difficulty and so might not be able to improve when it is harder? as in they might overfit to the easy mode.
- Since epochs seem to consume most of the training time now, I will reduce it to 1 instead of 8. It seems like it is more useful in supervised machine learning and so on.
- An important trade off as well is the number of steps in each iteration. The best answer is the approximate length of one attack in steps. I will set it to 256 as it seems good after testing it. it resets the environment after each iteration though, and sometimes it resets when a goal is almost scored. Well, that's because the environment is reset by me after saving the models. I can delete that line of code and then it won't reset the game every end of iteration! it think this is the best idea, I can let it reset every 10 iterations in case they get stuck and do not move around.

- I tried to open the container I use now using the same method as before, but it seems like it does not save any new changes to the container, as it is no longer running! I found out by typing git status, and it was not up to date as results from latest experiment were not committed.
- Therefore, I will use the random container that is somehow still running, which is: 1a13524680b5
- Actually, that will make no difference I think. This is because I am copying a container not accessing it. I need to know how to access it.
- Seems like the way to do this is by using the exec command:

```
39 medusa.ee.ucl.ac.uk % docker exec -it  dbc38f8261ec /bin/bash

Error response from daemon: Container
dbc38f8261ecc6bbcee157cc7020749e0bdf1081418d6539400d647d0c2070dd is not running
```

- As expected, it should only work on running containers, so I will use the 1a13.. one.
- I cannot run the code because it does not recognise my custom scenario! I added it to docker using:

```
docker cp /home/zceesaa/3_vs_3.py 1a13524680b5:/gfootball/gfootball/scenarios/3_vs_3.py
```

- I checked and it was indeed added, but still gives an error after running the code.

March 8th, 2022:

- Looking back at the paper, the only multi-agent experiment they did was using the 3 vs 1 scenario using IMPALA. I think it is a great idea if I just do the same thing but using PPO!
- I am thinking of changing epochs and shuffle in model.fit() but i'm trying to understand their effect. do they mean the training data as in only the 128 steps? Or all the data saved in the model? I will leave it as before for now.
- The paper includes results using PPO in 3 vs 1 but it is not multi-agent.
- when two players are controlled by the agent, their names are in blue and the built-in bot is in green.
- when only 1 player is controlled by the agent, they are all in blue. I think I can ignore that as I don't have control over it anyways. I will just trust that they wont let one of the controlled players be the goalkeeper since they did not include controllable=False! Also, there are no offsides in the scenario.
- Training finished for 3 vs 1 for two players. time taken to finish whole training: 14003.78s = 3 hours and 53 mins.

March 10th, 2022:

- I think the last training took an extra hour compared to 1M steps on empty goal scenario because there are two players controlled by the agent which doubles the amount of data being trained each step.
- I will test how fast it is if epoch=1 compared to epoch=8, as well as stacked=True.
- I tested the 1M step 3vs1_two model. The final model just moves back and stays still. Model 400 (400th iteration) walks top left until it leaves the field. The player is improving but very slowly since the reward is only for scoring and not a single goal was scored.
- i'm testing the effect of epoch using simpleModel.py, 30 iterations were done in 28 seconds using epoch=8 for both critic and actor models.
- when epoch=1 for both critic and actor models, time taken to complete 30 iterations became 21.7 seconds!
- stacking is only possible on 3 representations: pixels, pixel grey, and extracted. I will leave it for now.
- I will now train 3 vs 1, controlling 1, 2, and 3 players, using scoring only and both rewards. So this is 6 agents being trained for 5M steps. I will try to let them train simultaneously.
- Training was suddenly stopped by a Windows update.. It seems like training 6 scenarios at the same time using the same docker container is ok! From the number of iterations completed in that time period (1-2 hours roughly), it seems like training speed is not affected by all scenarios running at the same time! unless the speed due to using epoch=1. Also, training using one player is faster than two/three players. For

example, one player finished 2200 iterations, two players finished 1600 iterations, and three players finished 1400 iterations.

- The most recent model for one player runs towards the goal, which is a good sign! still needs a lot more training though.
- the most recent model for 2 players is very similar to the one player model!
- I realised something, I did not save my model critic, which is needed when loading a saved model!

March 12th, 2022:

- I finished training 6 scenarios for 5M steps.
- This gets printed in the terminal each iteration:

```
Train on 256 samples

2022-03-11 21:04:19.399094: I tensorflow/core/profiler/lib/profiler_session.cc:225] Profiler session started.

2022-03-11 21:04:19.399304: E tensorflow/core/profiler/internal/gpu/cupti_tracer.cc:1307] function cupti_interface_->Subscribe( &subscriber_, (CUpti_CallbackFunc)ApiCallback, this)failed with error CUPTI could not be loaded or symbol could not be found.

2022-03-11 21:04:19.399399: E tensorflow/core/profiler/internal/gpu/cupti_tracer.cc:1346] function cupti_interface_->ActivityRegisterCallbacks( AllocCuptiActivityBuffer, FreeCuptiActivityBuffer)failed with error CUPTI could not be loaded or symbol could not be found.

 32/256 [==>...........................] - ETA: 0s - loss: 0.08692022-03-11 21:04:19.406829: E tensorflow/core/profiler/internal/gpu/cupti_tracer.cc:1329] function cupti_interface_->EnableCallback( 0 , subscriber_, CUPTI_CB_DOMAIN_DRIVER_API, cbid)failed with error CUPTI could not be loaded or symbol could not be found.

2022-03-11 21:04:19.406939: I tensorflow/core/profiler/internal/gpu/device_tracer.cc:88]  GpuTracer has collected 0 callback api events and 0 activity events.

WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch update (0.530852). Check your callbacks.

256/256 [==============================] - 1s 4ms/sample - loss: 0.0081batch update (0.530852). Check your callbacks.

Train on 256 samples

2022-03-11 21:04:20.561134: I tensorflow/core/profiler/lib/profiler_session.cc:225] Profiler session started.

2022-03-11 21:04:20.561244: E tensorflow/core/profiler/internal/gpu/cupti_tracer.cc:1307] function cupti_interface_->Subscribe( &subscriber_, (CUpti_CallbackFunc)ApiCallback, this)failed with error CUPTI could not be loaded or symbol could not be found.

2022-03-11 21:04:20.561269: E tensorflow/core/profiler/internal/gpu/cupti_tracer.cc:1346] function cupti_interface_->ActivityRegisterCallbacks( AllocCuptiActivityBuffer, FreeCuptiActivityBuffer)failed with error CUPTI could not be loaded or symbol could not be found.

 32/256 [==>...........................] - ETA: 0s - loss: 9.6187e-052022-03-11 21:04:20.566042: E tensorflow/core/profiler/internal/gpu/cupti_tracer.cc:1329] function cupti_interface_->EnableCallback( 0 , subscriber_, CUPTI_CB_DOMAIN_DRIVER_API, cbid)failed with error CUPTI could not be loaded or symbol could not be found.

2022-03-11 21:04:20.566105: I tensorflow/core/profiler/internal/gpu/device_tracer.cc:88]  GpuTracer has collected 0 callback api events and 0 activity events.
```

```
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch update (0.610001).
Check your callbacks.

256/256 [==============================] - 1s 5ms/sample - loss: 0.0019batch update (0.610001).
Check your callbacks.

total test reward of iteration 19530 = 2.1000001430511475
```

- I might need to solve those warning issues for next trainings.
- time taken to finish the whole training (1 player only scoring): 70950.6s = 19 hours and 42 mins
- for 1 player, with checkpoint, I think it was around 64k seconds = 17 hours and 46 mins
- time taken to finish whole training(3 players with checkpoint): 95749.5s = 26 hours and 36 mins
- time taken to finish whole training(2 players with checkpoint): 99372s = 27 hours and 36 mins
- time taken to finish whole training(3 players only scoring): 98863s = 27 hours and 28 mins
- time taken to finish whole training(2 players only scoring): 91430.3s = 25 hours and 24 mins
- The server session automatically disconnects when the code is finished. But thankfully everything is saved in the docker container. For 3vs1_three_check_5M, there is a weird thing where the same iteration model actor is saved twice, each with a different reward!
- after testing the models, I realised they all overfit, and very early on in training! For example, in 3vs1_three_check_5M, the action probability distribution has one action as a probability of 1, from iteration 600 or 800. even with the other scenarios such as controlling one player.

March 13th, 2022:

- The source of the overfitting issue is not known, it might be the epochs=1, or the new method of predicting model(), or the experimental_run_tf_funtion=false, or something else. I will quickly test training using epoch=8 and see if that makes a difference.
- Unfortunately, i cannot connect to the server due to ssh connection time out, for some reason the ucl vpn is connected but not working properly. I re-installed cisco ucl vpn but it did not work. I also let someone who is connected to eduroam directly try to ssh into medusa, but it also gave the same error. This means that the issue is not with the vpn but it's with ucl's servers!
- useful blog on PPO https://flyyufelix.github.io/2020/12/02/google-football-rl.html
- a locked blog on medium that would be useful: https://towardsdatascience.com/reproducing-google-research-football-rl-results-ac75cf17190e
- I realised they do not use tensorflow and keras for the ppo algorithm, but they use openAI baseline. Anyways, I tried to run the ppo example on GRF's Read me page but it did not work.
- I will analyse my empty goal model that I solved and see how it developed. So, for the first 400 iteration, the player keeps passing it back to the keeper. After that, he learns to run towards the goal but still passes the ball back a few times. Then, he runs all the way to goal but misses. Finally, he scores around the 500th iteration.
- An idea for my overfitting issue, if i can, I should set a maximum possible probability (say 0.8), and a minimum for each action (say 1%) so that the agent does not always choose the same action and get stuck. I can do that! I can manually change the action distribution within the for loop and set the maximum and minimums.

March 21st, 2022:

- I tested training for 800 iterations using epoch=8 for 3 players with checkpoint: time taken to finish whole training = 4249.6s = 70.8 mins
- The source of the overfitting issue is not epochs, as this also overfits from the 600th iteration!
- I will write the code now such that it sets max probability thresholds.

- I will use two different approaches to solve the issue, the first is setting the max threshold (will be tested on 3 players), and the other is an epsilon greedy approach where the agent takes random actions sometimes (will be tested on 2 players).
- Here is the code to set a max threshold: (used on 3 players for both reward scenarios)

```
if max(action_dist) > 0.5:
   diff = max(action_dist) - 0.5
   share = diff / (len(action_dist) - 1) # distribute the difference to other actions
   index = np.argmax(action_dist)  # index of max
   for i in range(len(action_dist)):
       if i == index:
           action_dist[i] = 0.5
       else:
           action_dist[i] = action_dist[i] + share
```

- Here is the code for the epsilon greedy approach to pick a random action: (used on 2 players for both reward scenarios)

```
if np.random.uniform() < epsilon: # epsilon greedy approach
   action_player1 = np.random.choice(action_dims[0]) # take a completely
random action
   action_player2 = np.random.choice(action_dims[0])
else:
   action_player1 = np.random.choice(action_dims[0], p=action_dist[0, 0, :])
   action_player2 = np.random.choice(action_dims[0], p=action_dist[0, 1, :])
```

- I got an error because I did not set the max threshold for each player, so i included the same code for each player and now it is training.
- training done on 2 players using checkpoint: time taken to finish whole training: 3491.5s = 58.2 mins
- training done on 2 players without checkpoint: time taken to finish whole training: 3683.9s = 61.4 mins
- training done on 3 players using checkpoint: time taken to finish whole training: 3953.9s = 65.9 mins
- training done on 3 players without checkpoint: time taken to finish whole training: 3848.0s = 64.1 mins
- I will analyse results tomorrow.

March 22nd, 2022:

- Testing 3 players with checkpoint (max threshold approach): Great success! The method works! Here is the action probability distribution:

```
probs1 =[0.00889153 0.00792799 0.00818823 0.00812976 0.00778564 0.4463397
 0.00821998 0.00791954 0.00715958 0.00835556 0.00721414 0.00794365
 0.41900745 0.00892021 0.00733383 0.00795603 0.00747091 0.0071755
 0.00806078]
```

- As shown, there are two actions with a probability around 40%, which is even better than expected! I expected the printed distribution to still be overfitting, but no, it actually affects how the model is trained.
- Testing 3 players without checkpoint (max threshold approach): Similar results as before.
- Testing 2 players with checkpoint (epsilon greedy approach): Here is the distribution:

```
probs1 =[1.9096397e-13 6.5620485e-13 1.9473485e-13 1.2890891e-13 2.8530336e-13
```

```
 8.9464087e-14 1.5707487e-13 1.7312524e-06 3.4196330e-13 3.1122278e-13
 2.4094013e-13 4.7483057e-13 3.2413072e-13 7.9738740e-14 1.2888781e-07
 2.6285132e-13 9.9999809e-01 1.2915527e-13 2.8739242e-13]
```

- There is overfitting, so it's worse than the max threshold method. The result for without checkpoint is the same.
- I will use the max threshold approach to all scenarios and train for 5M steps and see what I get! Should I improve it even more? The issue with it now is that there will be around 2 actions with a combined probability of > 80%, so it's kind of between two actions. Maybe add some completely random options such as the epsilon approach? I think I will also add the epsilon too with like epsilon = 0.2.
- I included epsilon = 0.2 with all scenarios too.

March 23rd, 2022:

- Something interrupted 4 of the 6 trainings: Network error: Software caused connection abort.
- Disconnection might be due to wifi or closing the lid of the PC, but it's weird that not all trainings were disconnected! I will avoid closing the lid just in case.
- One of the 4 thankfully finished training before it was disconnected, which is the 1 player without checkpoint.
- 1 player with no checkpoint's last iteration scored a goal!!! This is a great result! I will check when it started scoring and whether there is some randomness or as usual the rewards are 000011111.. etc.
- time taken to finish whole training: 89035.1s = 24 hours and 44 mins.
- The terminal output only saved starting from iteration 7987 unfortunately.
- A reward of +1 was given 446 times, out of 11613 iterations. The first being 8003, and it is quite frequent throughout all iterations, so it's not like the last 500 iterations were successful.
- A reward of +2 was given 10 times, out of 11613 iterations. The first being 8959.
- A reward of -1 was given 121 times, out of 11613 iterations.
- 2 player without checkpoint finished training: time taken to finish whole training: 109186.4s = 30 hours and 20 mins
- 3 player with checkpoint finished training: time taken to finish whole training: 109962.1s = 30 hours and 33 mins
- Training took more time compared to the last time I trained for 5M steps, which is I think because of increasing the iterations to 19600 such that it is divisible by 200 and the last model is saved.
- Important question: Should I save the model with the most rewards? or the last one? what is the norm? in the video tutorial I found he saves the iteration only if it perfroms better. I think the best thing to do is to add an additional check if the reward is high, then save that model regardless if it was divisible by 200 or not. For loading the model for extra training, choose the model that performs better when tested? or the last one?
- For testing, I might also add some randomness exactly like in training, since that makes more sense! I can do two separate testing methods and compare them!
- Testing 1 player without checkpoint iteration 19600: action distribution:

```
probs =[[0.02535012 0.02678265 0.02607941 0.02811905 0.02662448 0.02872757
  0.02489821 0.0244915  0.02257134 0.02617476 0.02514939 0.02635515
  0.03061564 0.02278338 0.02793105 0.0260101  0.02665315 0.02891016
  0.5257729 ]]
```

- Question: how do I know that the agent controls the player that starts with the ball?? I will check by manually inserting actions.
- the action with the highest probability is action 18 (stop dribbling) which is a bit random!
- I can confirm that the controlled player is the one with the ball.
- Below is the distribution for iteration 600, where an own goal was scored:

```
[[2.4623528e-06 2.2826813e-17 1.0308233e-07 9.6213199e-14 7.6961895e-16
```

```
 1.3135481e-15 1.7258224e-12 3.9898839e-12 3.9026039e-13 6.5555090e-05
 3.0127275e-01 2.1622218e-02 6.7927066e-12 7.3521094e-12 3.0960648e-13
 3.1929630e-01 8.0413744e-03 4.3032205e-12 3.4969920e-01]]
```

- running to the left has the least probability, which makes sense because it leads to scoring an own goal!
- I just realised why setting the max threshold works and makes a difference in the model, this is because each action_dist is appended to action_probs, which is then passed to model.fit!!
- This means I need to be more careful with how i set the threshold and how to distribute the shares. this way, the first action that reaches 50% probability will always be there, and the rest will be at around 2%.
- distribution at iteration 8800:

```
probs =[[0.01940684 0.01970894 0.0200589  0.02065573 0.02017623 0.02020881
  0.02002803 0.01977205 0.01919294 0.02123697 0.01982268 0.02037171
  0.01971531 0.02091411 0.0196225  0.13960116 0.01982417 0.01987288
  0.51981014]]
```

- as shown here, it is very similar to iteration 19600! so nothing much changes! let’s check if that’s the case with 3 players.
- for 3 players with checkpoint, actions seem to all have equal probabilities for iteration 19600:

```
probs1 =[0.04565408 0.054561   0.05349384 0.05437586 0.05175764 0.05280083
 0.05296942 0.05364088 0.05395454 0.05066006 0.05337331 0.05302405
 0.05409335 0.05252874 0.05194803 0.0513941  0.0545909  0.05153652
 0.05364287]
, probs2 =[0.2564789  0.0418518  0.04303576 0.041929   0.04211283 0.04038844
 0.04114074 0.0409983  0.04115589 0.04108817 0.04138384 0.04131985
 0.04097198 0.04096339 0.04126185 0.04021065 0.04224723 0.04001291
 0.04144855]
, probs3 =[0.16773634 0.04685935 0.04726491 0.04806994 0.04771585 0.04530344
 0.04568586 0.04538523 0.04590182 0.04537053 0.04696951 0.04652589
 0.04646816 0.04570547 0.04628614 0.04535444 0.0471799  0.04478428
 0.04543302]
```

- At iteration 600, there were some favourite actions, such as 0 (idle) and 9 (long pass).
- for iteration 19400, action 0 dominates, here is the distribution:

```
probs1 =[0.54967254 0.0250326  0.0255512  0.02489806 0.02537221 0.02452808
 0.02482629 0.02520601 0.02449239 0.02564426 0.02553753 0.02541355
 0.02466392 0.02439878 0.02559722 0.02367325 0.02479858 0.02520055
 0.02549301]
, probs2 =[0.5012785  0.02772195 0.02758957 0.02750321 0.02793111 0.02758056
 0.0279256  0.02742281 0.02770039 0.02745397 0.02800716 0.02746557
 0.02771207 0.02772943 0.0274932  0.02811336 0.02755909 0.02743299
 0.02837941]
, probs3 =[0.5025702  0.02769751 0.02776317 0.02764976 0.02794683 0.02744725
```

```
 0.02751768 0.02747201 0.02758073 0.02737481 0.02810205 0.02735708
 0.02756728 0.02742738 0.0276972  0.02786592 0.02771037 0.02727112
 0.02798172]
```

- it's very weird that in the last 200 iterations, action 0 moves from 50% to just 4% like the rest of actions.
- ok, I've reached the conclusion that my model has a flaw and must be fixed or I should just use the available PPO model from baseline. I tested random iterations and they all have action 0 at 50% and the rest equally distributed.
- As for 3 player without checkpoint, iteration 5400 has a good distribution:

```
probs1 =[0.02171903 0.02080814 0.02190363 0.02041025 0.02087492 0.02016648
 0.02114031 0.02248695 0.0212309  0.02071365 0.02066312 0.01902001
 0.62107366 0.02248636 0.02375922 0.02148392 0.01760456 0.02208121
 0.02037366]
, probs2 =[0.03889439 0.04144327 0.03968564 0.03813916 0.0382894  0.03933428
 0.03871341 0.0403493  0.03798693 0.0373839  0.03746706 0.03737114
 0.30390245 0.03886209 0.03905787 0.03953297 0.03506194 0.03941438
 0.03911045]
, probs3 =[0.03922672 0.03992376 0.03964855 0.04053654 0.04185037 0.03766833
 0.04041653 0.04109326 0.03794551 0.03689079 0.03660766 0.03798451
 0.29411295 0.03881558 0.0395126  0.04123149 0.0412596  0.03754266
 0.03773262]
```

- the highest probability is shooting which is good!!
- the problem is, the best action keeps changing, for example, this is for iteration 5600:

```
probs1 =[0.0230937  0.01511178 0.01651718 0.01710458 0.01747519 0.02142485
 0.01679663 0.63056374 0.02249987 0.02031682 0.01676147 0.02159914
 0.03595214 0.02308566 0.02286035 0.02039168 0.01497619 0.02390019
 0.01956885]
, probs2 =[0.03253325 0.02952621 0.02860561 0.03059888 0.03176052 0.0364581
 0.02880137 0.38649368 0.03763187 0.03331347 0.0332197  0.03411981
 0.05343588 0.03535152 0.03370246 0.03296703 0.03189994 0.03498401
 0.03459673]
, probs3 =[0.02807111 0.02739385 0.0264053  0.02527414 0.03026813 0.03233512
 0.02645955 0.4482655  0.03079797 0.0301236  0.0272898  0.03457157
 0.04518569 0.03018234 0.0322157  0.03129882 0.03135788 0.03051488
 0.03198904]
```

- action 7 (run down) is now preferred!
- for 2 players with checkpoint, distribution prefers shooting for the latest saved iteration! which is iteration 12200:

```
probs1 =[0.03128848 0.03283998 0.0310029  0.03119287 0.03157064 0.03101454
```

```
 0.03235494 0.0307682  0.03110094 0.03262362 0.03062861 0.03038162
 0.43075857 0.03219762 0.03194307 0.03109296 0.03184854 0.03304206
 0.03234984]
, probs2 =[0.03209767 0.03341002 0.03204785 0.03191517 0.03173627 0.03122408
 0.03308813 0.0318166  0.03065936 0.03293422 0.03195789 0.03154274
 0.41940683 0.03260246 0.03245196 0.03144411 0.03247859 0.03422984
 0.0329562 ]
```

- even with iteration 12000 (and 9000, 7000, 3000, 1800 , 800), shooting is preferred which is good! it seems like the model learns that shooting is good from model 400 onwards! I think I will continue training for 50M steps! maybe with time a second action such as passing will rise as the second preferred action?
- I have a good idea! for sharing the probabilities, I should not share them equally, I should give actions with higher probability more share, so that the model does not choose useless actions. For example, running to the left was very bad in one of the models, but due to equally sharing, it gets back to be chosen since it gets around 2% or so. I must try this.
- for 2 player without checkpoint, iteration 18200 was really good! they do 1-2s and pass through the defender! they just need to shoot to score! Here are the distributions:

```
probs1 =[3.0014666e-02 3.1546336e-02 2.9242557e-02 2.9292477e-02 3.0222636e-02
 5.4324251e-01 3.0010859e-02 3.5824202e-02 9.3789391e-17 2.8822776e-02
 3.0270692e-02 2.8441703e-02 3.0353472e-02 3.1139584e-02 8.9082370e-16
 8.2510035e-17 3.0703302e-02 3.0484129e-02 3.0388096e-02]
, probs2 =[2.7064685e-02 2.9075516e-02 2.8184259e-02 2.6876610e-02 2.4248783e-02
 5.9476304e-01 2.7338233e-02 2.4923455e-02 3.4026724e-20 2.6189895e-02
 2.7810462e-02 2.6757075e-02 2.7365126e-02 2.8338103e-02 9.1064107e-19
 2.3422451e-20 2.6901022e-02 2.7195370e-02 2.6968401e-02]
```

- There is something interesting here, action 8 has an extremely low probability! (running bottom left) I don't know why this especially and how it did not increase from the share. Maybe it just became low then back to normal in the next iteration?
- Also, favoured action is 5 (running to right).
- for iteration 18400 (next saved one):

```
probs1 =[4.5392118e-02 4.5135252e-02 5.1591203e-02 5.0698441e-02 1.3774242e-01
 1.7040879e-01 4.6133794e-02 5.1417444e-02 3.0905996e-12 4.8636395e-02
 5.1664613e-02 5.2909732e-02 5.1069643e-02 4.9496178e-02 7.7752223e-12
 3.7393404e-12 4.6337862e-02 5.0510202e-02 5.0855942e-02]
, probs2 =[6.2800154e-02 5.7063174e-02 5.9090868e-02 5.8157548e-02 7.8977622e-02
 7.8180172e-02 6.0647320e-02 4.8766498e-02 1.1436805e-07 6.0726859e-02
 6.8946749e-02 6.6374220e-02 6.5129504e-02 5.5955581e-02 9.7228828e-08
 1.2892575e-07 5.7482380e-02 6.2606066e-02 5.9094995e-02]
```

- seems like action 8 is slowly becoming more favourable, but not much, which is good!
- As for 1 player with checkpoint, interesting result for last saved iteration 15600 (and 15400):

```
probs =[[9.4786314e-12 1.3217130e-11 1.3255150e-11 4.9722356e-01 1.6725650e-11
  2.5293951e-11 1.6384198e-11 4.6775476e-11 1.2494885e-11 1.4813166e-10
  1.0434094e-11 1.2181446e-10 1.1858791e-11 1.3273720e-11 3.7584366e-11
  4.9337387e-01 9.4026057e-03 2.1792609e-11 6.6650662e-13]]
```

- Two actions have around 50%! However, not very good ones, (run to top and stop sprinting!)
- overfitting started as soon as the 1000th iteration! (or even earlier). the rest of training for this scenario was useless due to this.
- it seems like not using the checkpoint reward function is better in many cases. For example, the 1 player with checkpoint model is happy with running to the top all the time, because it gets some reward for checkpoint and due to resetting when leaving the field, it gets the reward a lot within one iteration, so it thinks it is good.

March 24th, 2022:

- Andrey told me about stable baselines and how easy it is to use their models, so I will give it another go.
- Started by installing in on the server: `pip3 install git+https://github.com/openai/baselines.git@master`.
- then running the example: `python3 -m gfootball.examples.run_ppo2 --level=academy_empty_goal_close`
- I get an error saying no module named sonnet. After installing it using pip3 install sonnet, I get an error saying no model called graphs, I do pip3 install graph, but then it says it cannot use sonnet.
- Online they say use pip3 install dm-sonnet instead, so I use that and it works.
- Now I get this error:
```
Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/baselines/common/vec_env/subproc_vec_env.py", line 121, in __del__
  File "/usr/local/lib/python3.6/dist-packages/baselines/common/vec_env/vec_env.py", line 98, in close
  File "/usr/local/lib/python3.6/dist-packages/baselines/common/vec_env/subproc_vec_env.py", line 104, in close_extras
  File "/usr/lib/python3.6/multiprocessing/connection.py", line 206, in send
  File "/usr/lib/python3.6/multiprocessing/connection.py", line 404, in _send_bytes
  File "/usr/lib/python3.6/multiprocessing/connection.py", line 368, in _send
BrokenPipeError: [Errno 32] Broken pipe
```
- After googling, it seems to be a server-client error.
- I will try on Linux.

- This is the error I get in Linux, which i got before: SystemError: Sonnet requires tensorflow_probability (minimum version 0.4.0) to be installed. If using pip, run `pip install tensorflow-probability` or `pip install tensorflow-probability-gpu`
- it must be a case of different versions. I have an idea! use a different docker container where tf is v1 not 2.
- It works on tensorflow 1.15!! I can run `python3 -m gfootball.examples.run_ppo2 --level=academy_empty_goal_close` successfully.
- Next step: copy GRF's run_ppo2.py code and adjust it such that it becomes multi-agent and run ppo2 on it while saving the model using that tutorial from youtube!
- I get this error running the multi-agent example code: Error while finding module specification for 'gfootball.examples.run_multiagent_rllib.py' (AttributeError: module 'gfootball.examples.run_multiagent_rllib' has no attribute '__path__')
- Flags are used for parameters when running python3 in the cmd.
- I changed the number of steps using the parameter and made it very low to see what happens at the end, and I got many tensorflow warnings and errors:
- failed call to cuInit: UNKNOWN ERROR (-1)
- 2022-03-24 19:22:56.474597: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:163] no NVIDIA GPU device is present: /dev/nvidia0 does not exist.
- So it cannot find the gpu.. These are the versions I should have right now:(TF, python, compiler, build tools, cuDNN, cuda) https://www.tensorflow.org/install/source#gpu

tensorflow_gpu-1.5.0	2.7, 3.3-3.6	GCC 4.8	Bazel 0.8.0	7	9

- it turns out that there is an NVIDIA GPU on medusa, but not within the docker container!!! what can I do here??

- I think I'll just test how many goals my 5M step models score and compare that to the paper and that's it. or do the improvement then train again.

March 26th, 2022:

- I tried installing CPU version of tensorflow, because NVIDIA GPU is not available in the docker container.

- Something weird happened, so when tensorflow-gpu is installed, I get this error: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:163] no NVIDIA GPU device is present: /dev/nvidia0 does not exist

- When only the CPU version of tensorflow is installed and I run python3 -m gfootball.examples.run_ppo2 --num_timesteps=10, I get an error saying: no module named tensorflow!

- Ok, After uninstalling and re-installing CPU version of tensorflow, now it has become the "default" I think. Now when running: python3 -m gfootball.examples.run_ppo2 --num_timesteps=10, I don't get any errors. However, I get warnings such as: WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/baselines/common/misc_util.py:58: The name tf.set_random_seed is deprecated. Please use tf.compat.v1.set_random_seed instead.

- I think the warning is due to the fact that TF.2 exists now and if you want to use version 1 functionality then use .compat.v1. Is that necessary in my case? Should I edit it? I don't know.

- However, in run_ppo2.py, `import tensorflow.compat.v1 as tf is used!` Doesn't that mean I should not be getting warnings?

- In the world of software development, "deprecated" refers to **functions or elements that are in the process of being replaced by newer ones**. (I was right)

- Seems like I can ignore the problem. The goal now is to use stable baselines! I wanna learn how to save the model and quickly train my models asap.

- I copied run_ppo2.py into my repository to see if it works, and it does! now I can customise the code as I like.
- since i can't use my own pycharm since baseline only works in the server (due to python and TF versions), I can use google Colab to play around with the code and see how I can save models.
- app.run(train) simply means the function that will run is train(), so I should add stuff inside the train() function.
- this is the function called in run_ppo2.py as ppo2.learn(): https://github.com/openai/baselines/blob/master/baselines/ppo2/ppo2.py

-
  ```
  if load_path is not None:

        model.load(load_path)
  ```

- this is within .learn(), so it builds upon stable_baselines (the tutorial im watching). so i should find out how they save the model in our case.
- Here is where the saving is done in the code:

```
if save_interval and (update % save_interval == 0 or update == 1) and logger.get_dir() and is_mpi_root:

    checkdir = osp.join(logger.get_dir(), 'checkpoints')

    os.makedirs(checkdir, exist_ok=True)
```

```
        savepath = osp.join(checkdir, '%.5i'%update)

        print('Saving to', savepath)

        model.save(savepath)
```

- let’s test by letting it save a model more frequently.
- when the code is run, this is the terminal output:

```
Stepping environment...

I0326 17:50:33.509761 140591822858048 football_env_core.py:267] Episode reward: 1.00 score: [1, 0], steps: 11, FPS: 34.7, gameFPS: 252.1

I0326 17:50:34.690863 140591822858048 football_env_core.py:267] Episode reward: 0.00 score: [0, 0], steps: 82, FPS: 54.8, gameFPS: 241.9

I0326 17:50:34.863031 140591822858048 football_env_core.py:267] Episode reward: 0.00 score: [0, 0], steps: 90, FPS: 53.9, gameFPS: 224.7

I0326 17:50:34.905639 140591822858048 football_env_core.py:267] Episode reward: 0.00 score: [0, 0], steps: 91, FPS: 53.2, gameFPS: 264.3
```

- I think steps are basically the number of steps/actions for each episode. Episodes are not necessarily 128 steps because a goal might be scored etc (i think), so the max steps per episode would be 128! let me test this now..
- Also, I get this output:

```
| misc/nupdates          | 1        |

| misc/serial_timesteps   | 128       |

| misc/time_elapsed      | 5.54      |

| misc/total_timesteps    | 1.02e+03
```

- it shows the total timesteps taken (1000) and how much time passed, even though i don’t see the steps for each episode adding up to total_timesteps!
- After that , this is outputted: Saving to /tmp/openai-2022-03-26-17-50-33-228744/checkpoints/00001
- I think this is where the model is saved.. but how do i access it! is it on my machine/server?
- Found it!! it’s within the tmp folder (one of main folders in the container), so before cd gfootball. Is there a way I can make it save within my repo for easier pushing?
- i tested with nsteps=5 and I think I understand it now. nsteps is like the batch size, so before when I was using model.fit(), it would be equivalent to ppo_steps. However, the steps printed here: “I0326 18:02:51.352060 140192221501248 football_env_core.py:267] Episode reward: 1.00 score: [1, 0], steps: 17, FPS: 6.5, gameFPS: 247.8” They mean the steps for each episode (which ends when a reward is given, regardless of nsteps), so when a goal is scored and so on depending on the scenario.
- im trying to load a saved model now, i think it has to be on the same folder (venv) then it will work. I will move the file there and try.
- Ok it works, i moved the model/file the loaded it using --load_model, but I got another error:

```
 File "/usr/local/lib/python3.6/dist-packages/joblib/numpy_pickle.py", line 579, in load
```

```
    with open(filename, 'rb') as f:

IsADirectoryError: [Errno 21] Is a directory: 'openai-2022-03-26-18-13-12-293863'
```

- I think it's because this is a directory and not a file? let me check. Yep, it is a directory and it includes 3 files: 6.monitor.csv log.txt progress.csv
- how do I know which one to load? let me check for examples.
- I FOUND IT!!!! https://github.com/openai/baselines/blob/master/README.md#training-models from the README file here, it shows how to properly save and load models! there is a flag/parameter called --save_path too!
- Unfortunately, the --save_path parameter is not defined in the ppo2.learn().. but i figured out how they automatically save models! so when the code runs, it makes an opanai folder with those three random files, but as soon as a model is saved, a checkpoint folder is created inside the openai folder for that run, and inside the checkpoint each model is saved by its number! so loading is as simple as --load_path=00001 for this model: /tmp/openai-2022-03-26-17-50-33-228744/checkpoints/00001, if it was moved to venv. it worked!
- after the model was loaded and training started again, a new openai folder was created, with the current time and date, and new models are saved starting from checkpoint 00001.
- ok, now i need to figure out the multi-agent part, then im ready to train!!!
- the thing is, how can I test the model on my linux if i can't use baseline.. that's an issue. I will try to solve it, otherwise no need for rendering I will just calculate average rewards etc and compare it to the paper.
- let's see what happens if i disconnect during training.. ok i tested it by closing the tab and I realised that a single run will have more than one openai folder.. it is probably a folder for each checkpoint?? also i found that not all of them had checkpoints saved.. so let me try again and see which one gets saved.
- ok i did it again and turned off the wifi mid-training, training stopped but did not halt the whole tab! when i turned wifi back on it continued training! wow! However, when the laptop sleeps it stopped training and the tab shuts down: Network error: Connection reset by peer
- it kept saving to the same openai folder, all checkpoints. but it still created 9 openai folders for this single run..
- oh i see why.. for some reason, when the training starts, it creates a couple of openai folders (probably contains logs or something) but then an extra one is created where the checkpoints are saved!
- so there are 8 extra folders, each with a monitor file inside it, numbered from 0 to 7. (i don't know what the point of it is).
- I deleted all folders inside /tmp by using rm -r *
- it's time to figure out multi-agent training, which should not be too different.
- by default, the number of agents is 3: `parser.add_argument('--num-agents', type=int, default=3)`
- the scenario they use in the example is test_multiagent, I should change that into 3vs1 (in env_name=..)
- I get this error when trying to run the multi-agent example:

```
  from pkg_resources import Requirement

/usr/bin/python3: Error while finding module specification for 'gfootball.examples.run_multiagent_rllib.py' (AttributeError: module 'gfootball.examples.run_multiagent_rllib' has no attribute '__path__')
```

- ok it was a silly issue. you should either do python3 file.py, or python3 -m gfootball/file (I was doing -m and file.py)
- it works but I get a lot of errors. I will copy the file into my repo and let rendering be false and see if that gets rid of them.
- I still get this error:

```
  File "gfootball_multiagent.py", line 121, in <module>

    lambda agent_id: policy_ids[int(agent_id[6:])]),
  File "/usr/local/lib/python3.6/dist-packages/ray/tune/tune.py", line 633, in run

    raise TuneError("Trials did not complete", incomplete_trials)

ray.tune.error.TuneError: ('Trials did not complete', [PPO_gfootball_6f007_00000])
```

- I googled ray and i got this https://docs.ray.io/en/latest/rllib/index.html it is a reinforcement learning library with support for Multi-agent RL. I have two options now, try to understand it and fix the error, or use the run_ppo2 and change the number of players to 3 to see what happens.
- suggested solution: I had to install the tensorflow_probability and then use the tensorflow >=1.15.0 and then downgrade my numpy to 1.19. Doing this resolved resolved the issue (so in my case just install numpy 1.19 since i have 1.18.1)
- it says numpy 1.19 is not compatible with TF 1.15.5, but it still installed it! and the code runs! but i get the same error as before! this is a mess.
- i think it's because my ray version is not the one i need? let me try. (No success)
- The issue could be caused by: Unknown config parameter `sample_batch_size`
- I found this in the docs for tune.run(), but no more details are given:
- config (*dict*) – Algorithm-specific configuration for Tune variant generation (e.g. env, hyperparams). Defaults to empty dict. Custom search algorithms may ignore this.
- i tried commenting out the sample_batch_size line and now I get so many other errors. the terminal does not output anything now but the code did not stop running..

- I tried to exit using ctrl+c and got this: WARNING tune.py:596 -- SIGINT received (e.g. via Ctrl+C), ending Ray Tune run. This will try to checkpoint the experiment state one last time. Press CTRL+C one more time (or send SIGINT/SIGKILL/SIGTERM) to skip.
- so it seems like the thing was running?..
- I'll test it with a little number of iterations to see what happens.
- This is the last error that gets printed before it freezes: AttributeError: 'RllibGFootball' object has no attribute '_agent_ids'. that's a variable in the gftooball example code so i think their code has many errors.
- i tried adding number of players controlled = 2 in gfootball_ppo2.py but got this error: ValueError: Shape must be rank 4 but is rank 5 for 'ppo2_model/pi/c1/Conv2D' (op: 'Conv2D') with input shapes: [8,2,72,96,16], [8,8,96,32]. it definitely doesn;t work with more than 1 player being controlled.
- I guess I will train one player on 3vs1 and empty goal, while learning how to use ray!

- started training one player using only scoring reward for 5M steps in 3vs1. Here is the file where models are saved: Saving to /tmp/openai-2022-03-26-22-36-42-155491/checkpoints/00001
- started training one player using both reward functions for 5M steps in 3vs1. saved here: Saving to /tmp/openai-2022-03-26-22-42-35-838678/checkpoints/00001
- started training one player empty goal scenario for 1M steps only scoring reward: Saving to /tmp/openai-2022-03-26-22-47-11-179202/checkpoints/00001
- started training one player empty goal scenario for 1M steps both rewards: Saving to /tmp/openai-2022-03-26-22-50-54-886228/checkpoints/00001

March 27th, 2022:

- I should start writing the final report today
- training has finished. one player using only scoring reward for 5M steps in 3vs1 finished in 44100s = 12 hours and 15 mins
- one player using both reward functions for 5M steps in 3vs1 finished in 45200s = 12 hours and 33 mins
- one player empty goal scenario for 1M steps only scoring reward finished in 12400s = 3 hours and 26 mins
- one player empty goal scenario for 1M steps both rewards finished in 12000s = 3 hours and 20 mins
- now it's time to test the results!! I hope I can easily do it on Linux..
- first i'll move the saved models into my repo using mv. like this:
- mv /tmp/openai-2022-03-26-22-50-54-886228 /gfootball/GRF/venv/models
- I saved the terminal output for one player using only scoring reward for 5M steps in 3vs1 in test data folder for GRF in windows

- I will try to use baseline in Linux one more time.. it has to work otherwise i cannot render the game and see how my models perform! Python version on server: 3.6.9. CPU tensorflow version = 1.15.5. numpy=1.18.1 baselines = 0.1.6

- i followed this tutorial and installed python 3.6.9 on linux https://www.youtube.com/watch?v=LIQKC2t0mjU
- however, it's not the default version now, but at least I can use if if i type python3.6 instead of python3
- Something weird happens, when i try to install tensorflow using `python3 -m pip install tensorflow==1.15.*` it says that only versions above 2.2 are available because of python 3.8
- when i try with python 3.6: `python3.6 -m pip install tensorflow==1.15.*,` it says that only versions before TF1.4 are available!! how did it install in the server then?!
- ok in their google colab they use python 3.7 but Tensorflow is 2.8!
- I finally installed TF.1.15 by installing python 3.7!! now i,ll try setting up baselines and let the pycharm interpreter be 3.7 instead of 3.8
- when i use python3.7 to run ppo2 it says no module named gfootball.. so i'm gonna make a new project and install gfootball again.
- by watching the tutorial, he can view plots by typing tensorboard --logdir=logs
- minute 17 in the tutorial part 2 shows how to test a model with rendering. I think it should work with my environment too.
- the default python version has become 3.7 I think because when editing the bashrc file i typed python instead of python3:

  ```
  (venv) abdullah@abdullah-Swift-SF514-51:~/PycharmProjects/GRF3.7$ python --version

  Python 3.7.13

  (venv) abdullah@abdullah-Swift-SF514-51:~/PycharmProjects/GRF3.7$ python3 --version

  Python 3.8.10
  ```

- I finally learnt how to create the python interpreter in pycharm and made one for python 3.7 and now I get this output:

  ```
  (venv) abdullah@abdullah-Swift-SF514-51:~/PycharmProjects/GRF_py3.7$ python --version

  Python 3.7.13

  (venv) abdullah@abdullah-Swift-SF514-51:~/PycharmProjects/GRF_py3.7$ python3 --version

  Python 3.7.13
  ```

- YES!! everything works now! I managed to install baselines and it works on my linux! let me try rendering now!
- it turns out that testing the model requires stable_baselines, not the baseline model I trained with. I hope I don't need to train again!
- I'm having issues testing the model, but I think I understand it now! It's like this: To load and visualize the model, we'll do the following - load the model, train it for 0 steps, and then visualize:
  ```
  python3 -m baselines.run --alg=ppo2 --env=PongNoFrameskip-v4 --num_timesteps=0 --load_path=~/models/pong_20M_ppo2 --play
  ```
- I finally getting there. I'm using parts of the baseline/run.py file to test and visualise the model, while using parts of my training code to produce the my custom environment.
- I've been refining the code and fixing bugs as I go. I removed parameters when running the code as they are not necessary and add extra complexity in the code.
- The code finally works!! the final fix was adding `allow_early_resets=True` to the Monitor() function. this is so that the environment can be reset at the start.
- wow there are two renderings! one of them can be made into a full screen! i guess this is what Monitor() does.
- IT WORKS! the 3vs1 one player model with only scoring reward(as well as with checkpoint) learnt that passing the ball is the most optimal action, and they manage to score goals most of the time! I think i'll let the scenario run a set number of times to get an average for the goals and the variance, to compare it to the paper.

- I also tested the empty goal scenarios, and both (with checkpoint and without) solved the scenario by shooting twice towards the goal!!
- Instead of analysing more results, I will try my best to let the multi-agent code work! Then i can train it for 5M steps!
- ok somehow the multi-agent code works now, but it's weird, it's not giving the sample_batch or the agent_id error, but it's printing some scheduling thing..
- I changed the scenario into 3vs1 and it still works, good.
- this github repo has a good pdf (their paper) and they did some multiagent 3vs1 experiments: https://github.com/aparikh98/Multi-Agent-Coordination-Google-Football
- i'm trying to understand the weird output from rllib or ray.. It's kind of scheduling the code which is not what I want from it. i changed num_gpus into num_cpus. As for parameters like num_agents, changing the default actually makes a difference (it works), but setting the default of num_iters=1 didn't do anything, which means training did not start yet.
- ok so i figured out it's stuck as pending, so it's an issue with the CPUs and GPUs it needs to use.
- I think the issue is my code is not requesting any resources?: "Resources requested: 0/4 CPUs, 0/0 GPUs, 0.0/1.66 GiB heap, 0.0/0.83 GiB objects" I tried changing num_cpus=4 but nothing changed.
- I ran the new repo's code and I got the sample_batch error, so i think the pending thing is because i edited the original file, let me try to find the difference by comparing my file to the original example. rendering is the first difference, and it wasn't the reason. ok to save time i just ran the original file on my machine and it was pending.. so it's because of a difference it the machine (in the server i think I will get that error), so i will try to make it work in the server now).
- on the server i get this error when commenting out sample_size: AttributeError: 'RllibGFootball' object has no attribute '_agent_ids'
- let me dig deep into the source code to find what parameters are "known". ok can't find it.
- i tried commenting the last part of config ('multiagent') and I still get the same error! it seems like there is something within the code called _agent_ids, which is not in my code. (the variable in my code is agent_id)
- i tried changing it to agent_ids but nothing happened. ok I clearly need to define the attribute then! I will do it within the RllibGFootball class.
- I think i figured out how the new repo guy does multi-agent! he uses run_ppo2.py and copies the training code 3 times!
- ok i see how he does it, he changes the 3vs1 scenario such that each player becomes the first one to be defined, then i guess it is by default the one that the agent controls. I don't wanna do the same as i think i'll get that error in the sever because i can't define scenarios..
- Plan for tomorrow: just write the report. at the end of the day have another go at rllib.

March 28th, 2022:

- Andrey told me that openAI hide and seek use rllib for multi-agent, I just gotta find it
- I have a very good idea!!! look at when the gfootball example file was uploaded, and go find the version of ray that was the latest at that time!! it might be just a version issue.
- I found in the docs some info on multi-agent configuration: https://docs.ray.io/en/latest/rllib/rllib-env.html such as the 'multi-agent' and policy mapping fn:

```
config={

  "multiagent": {

    "policies": {

      # the first tuple value is None -> uses default policy

      "car1": (None, car_obs_space, car_act_space, {"gamma": 0.85}),

      "car2": (None, car_obs_space, car_act_space, {"gamma": 0.99}),

      "traffic_light": (None, tl_obs_space, tl_act_space, {}),

    },
        "policy_mapping_fn":
            lambda agent_id:
                "traffic_light"  # Traffic lights are always controlled by this policy
                if agent_id.startswith("traffic_light_")
                else random.choice(["car1", "car2"])  # Randomly choose from car policies
    },
})
```

- it might be worth a try to add that extra curly bracket between policy mapping and policies.
- oh never mind it should not make a difference because they are separated by commas, the bracket is for the lists.
- turns out the original example was edited 6 months ago, but the commit says it's only for windows support. the last fix to the code was around oct/nov 2019. The file was added around july 2019. Ray 0.7.5 was released 25th september 2019 so i will try that.
- SUCCESS! Version 0.7.5 works!!! Should I start getting the multi-agent models running or should I write the report? I think I will start writing the report first. then i'll figure out how to save the model and test it, as well as get those nice tensorboard plots for the average reward.
- ok i'm currently dumping bullet points from the lab book to the report. I'm planning to finish it within an hour.
- finished dumping everything to the report! word counts 13k.
- this is where the agent_ids comes from (thankfully i need not to worry about it anymore):

```
class MultiAgentEnv(gym.Env):
    def __init__(self):
        if not hasattr(self, "observation_space"):
            self.observation_space = None
        if not hasattr(self, "action_space"):
            self.action_space = None
        if not hasattr(self, "_agent_ids"):
            self._agent_ids = set()
```

March 29th, 2022:

- when using stacked in multi-agent:

```
policy_reward_max:
  policy_0: 1.0
  policy_1: 1.0
  policy_2: 1.0
policy_reward_mean:
  policy_0: 0.05555555555555555
  policy_1: 0.05555555555555555
  policy_2: 0.05555555555555555
policy_reward_min:
  policy_0: 0.0
  policy_1: 0.0
  policy_2: 0.0
sampler_perf:
  mean_env_wait_ms: 12.73130849951074
  mean_inference_ms: 18.928301275073004
  mean_processing_ms: 2.1594778617618493
time_since_restore: 31.79174828529358
time_this_iter_s: 31.79174828529358
time_total_s: 31.79174828529358
timestamp: 1648550757
timesteps_since_restore: 2000
timesteps_this_iter: 2000
timesteps_total: 2000
training_iteration: 1
```

- when stacked is False:

```
    policy_0: 1.0
    policy_1: 1.0
    policy_2: 1.0
  policy_reward_mean:
    policy_0: 0.0
    policy_1: 0.0
    policy_2: 0.0
  policy_reward_min:
    policy_0: -1.0
    policy_1: -1.0
    policy_2: -1.0
  sampler_perf:
    mean_env_wait_ms: 12.041867668948957
    mean_inference_ms: 7.710112404923672
    mean_processing_ms: 1.8852939561916138
  time_since_restore: 17.954442501068115
  time_this_iter_s: 17.954442501068115
  time_total_s: 17.954442501068115
  timestamp: 1648551082
  timesteps_since_restore: 2000
  timesteps_this_iter: 2000
  timesteps_total: 2000
  training_iteration: 1
```

- this is weird, it took more time when stacked=True. Anyways, it seems like it won't take a lot of time to train for 5M steps (should be a couple of hours only) other iterations took around 5 seconds only.
- this time i'm gonna write dump traces and use the example code they provide to convert it to video!
- writing dumps gives errors so i'm just gonna remove it.
- ok now i cant even run the normal code i think because there is no space: No usable temporary directory found in ['/tmp', '/var/tmp', '/usr/tmp', '/gfootball/GRF/venv']
- running df -h shows memory and some are 100% used.. i need to delete stuff:

```
Filesystem      Size  Used Avail Use% Mounted on
overlay         511G  511G     0 100% /
tmpfs            64M     0   64M   0% /dev
tmpfs            63G     0   63G   0% /sys/fs/cgroup
shm              64M     0   64M   0% /dev/shm
/dev/sda2        29G   17G   11G  61% /tmp/.X11-unix
/dev/sdb1       511G  511G     0 100% /etc/hosts
```

```
tmpfs          63G     0   63G   0% /proc/acpi

tmpfs          63G     0   63G   0% /proc/scsi

tmpfs          63G     0   63G   0% /sys/firmware
```

- i tried the space on the other container and they share the same memory.
- i found 300G of stuff lying around in my venv folder haha.. they were all named core..*, i think it was when it was writing dump files. hopefully it's fixed now after removing them.
- Fixed now!:

```
Filesystem      Size  Used Avail Use% Mounted on

overlay        511G  211G  274G  44% /

tmpfs          64M     0   64M   0% /dev

tmpfs          63G     0   63G   0% /sys/fs/cgroup

shm            64M     0   64M   0% /dev/shm

/dev/sda2       29G   17G   11G  61% /tmp/.X11-unix

/dev/sdb1      511G  211G  274G  44% /etc/hosts

tmpfs          63G     0   63G   0% /proc/acpi

tmpfs          63G     0   63G   0% /proc/scsi

tmpfs          63G     0   63G   0% /sys/firmware
```

- i tried running another session simultaneously, but it says:

```
Traceback (most recent call last):

  File "gfootball_multiagent.py", line 77, in <module>

    ray.init(num_gpus=1)
```

- definitely because it's used by the other training model. even the first one stopped working.. i'll try removing num_gpus
- ok i definitely have an issue due to only 1 gpu being available: 2022-03-29 13:22:56,966 WARNING worker.py:1779 -- The actor or task with ID ffffffffffff41d260bc01000000 is pending and cannot currently be scheduled. It requires {CPU: 1.000000}, {GPU: 1.000000} for execution and {CPU: 1.000000}, {GPU: 1.000000} for placement, but this node only has remaining {CPU: 55.000000}, {memory: 93.554688 GiB}, {object_store_memory: 12.841797 GiB}. In total there are 0 pending tasks and 1 pending actors on this node. This is likely due to all cluster resources being claimed by actors. To resolve the issue, consider creating fewer actors or increase the resources available to this Ray cluster.
- from: https://docs.ray.io/en/latest/rllib/rllib-algorithms.html

```
# 1. Policy evaluation in parallel across `num_workers` actors produces
```

#   batches of size `rollout_fragment_length * num_envs_per_worker`.

- i think these are “actors” so i need to use 1 num_workers?
- ok it works! but it does 2000 steps in around 25 seconds, which means 5M will take 17 hours! i don’t have that time. I will try letting it use only the CPU since there are 56 cores..
- ok i guess i can’t:

  ray.tune.error.TuneError: Insufficient cluster resources to launch trial: trial requested 11 CPUs, 0 GPUs but the cluster has only 1 CPUs, 2 GPUs, 93.85 GiB heap, 12.84 GiB objects. Pass `queue_trials=True` in ray.tune.run() or on the command line to queue trials until the cluster scales up.

  You can adjust the resource requests of RLlib agents by setting `num_workers`, `num_gpus`, and other configs. See the DEFAULT_CONFIG defined by each agent for more info.

- oh well:

  Exception: Unknown config parameter `num_cpus`

- ok now im using num_gpus in config but nothing in ray.init() and it works! also, when num_workers= 1, each iteration takes 25s, but when num_workers=10, each iterations takes 6s max!! this is good! i will try more workers!
- i tried 50 workers but the code is frozen haha
- i tried with num_sgd_iter=1 (not equal to workers) but still code frozen
- i tried 5 workers but i got errors. i guess 10 workers will do!
- i think for each agent, a core file is created.
- this is annoying, when i change the number of iterations i get an error about agent_id!
- ok i trained tll checkpoint 50 was saved (50th iteration), found it under ray_results/. now, how do i test it??
- this is the exact issue im having: https://github.com/ray-project/ray/issues/4569 so i should use the restore parameter with the path to the checkpoint. but then, how to i just render and not train? let’s try.
- ok im almost there but im having issues with my ray version!!: pandas.util.version.InvalidVersion: Invalid version: '0.14.0.RAY'
- it’s probably the pandas version. let me see what version it is in the server..
- ok it turns out i need to use the terminal and type python to use 3.7 (F5 does not work for some reason). I installed gfootball again and tensorflow 1.15 and ray 0.7.5 and pandas 1.1.5 and wrote a line to import pandas as pd. NOW IT WORKS!! i can test my model! there is a weird thing where it creates 3 different renderings tho..
- i passed an incorrect path to the checkpoint to check what happens, the code still runs and renders! but i get this message: FileNotFoundError: [Errno 2] No such file or directory: '/home/abdullah/PycharmProjects/GRF_py3.7/venv/models/PPO_gfootball_0_2022-03-29_14-45-46hcqi4q54/checkpoint50/checkpoint-50.tune_metadata'
- i dont get the error when passing a correct path! YESS
- ok now im trying to train for 5M steps but i get errors trying to connect to the socket of ray! i think it’s because my: (/dev/sda2   29G   17G   11G 61% /tmp/.X11-unix) is almost full? that’s what they say online, only 30GB of ram available.
- ok idk how to clear the used 17G, so i will use another location to store the socket and logs etc. using this: https://docs.ray.io/en/stable/tempfile.html
- i managed to move the logs, but not everything. also, weird thing, the memory was always this low! why did it not work now!