Gymnasium Fränkische Schweiz
Ebermannstadt

Abiturjahrgang 2023

SEMINARARBEIT

im W-Seminar Künstliche Intelligenz

Thema

Artificial intelligence in computer games
- AI learns to play a 2-dimensional game through reinforcement learning

Verfasser/in:	Philipp Hofmann
Seminarleiter/in:	Herr Reuter
Abgabetermin:	08.11.2022

Ergebnis der vorgelegten Seminararbeit	Punkte
Ergebnis der Präsentation mit Prüfungsgespräch	Punkte
Gesamtergebnis in doppelter Wertung: auf ganze Punktzahl gerundetes Ergebnis von (3x schr. + 1x mdl.):2	Punkte

In die Notenliste
eingetragen am:

____

Dem Direktorat
vorgelegt am:

____

____
(Unterschrift Seminarleiter/in)

Content

1. Abstract

Although at first glance it does not appear to have any benefit in teaching an AI to play a computer game, the insights can be applied to many, significantly more important, subject areas. For example, learning to play a racing game can provide important insights into the AI's learning behavior for further development of autonomous driving, since both processes provide roughly the same data on which to build an AI.[1] This paper will first show milestones of the past. And further, discuss the structure as well as the learning behavior of an AI in a 2D game. Finally, a trained agent will take place in a game using a reinforcement learning algorithm[2] to illustrate the results. This agent will not be playing the game itself but will take on the form of an enemy in that environment.

2. Introduction

Whether in TV series, video games, or even in the everyday use of our cell phones: It's hard to imagine modern life in the 21st century without artificial intelligence. Let's take cell phones in everyday life as an example. Most people are not aware of how much influence artificial intelligence takes in simple tasks like picture quality enhancement or image stabilization[3]. A more well-known area that is using AI to increase user satisfaction is the social media industry. All commonly used algorithms to evaluate what posts to recommend to you are controlled by AI. Whether it's TikTok, Instagram, or Twitter, all these platforms (and therefore companies) use the same technology to manipulate users to stay on their platform for as long as possible, namely autonomous learning algorithms.[4] Although this all sounds bad, and not very consumer-protective, there are also many potentially positive applications for AI. The most representative project in which AI is used now is self-driving cars. Most algorithms can learn better in simulated situations (also known as data mining) than in situations that cannot be simulated.[5] These simulations can take many different forms, and algorithms can also learn from human-generated data for example in racing games.

[1] (Laura Rice, 2022)
[2] Reinforcement learning (or RL) is a type of unsupervised learning algorithm using a reward system to "teach" the AI how to behave (Błażej Osiński, 2018)
[3] Image stabilization is the reduction in image blur induced by camera and/or object movements (Image Engineering, 2019)
[4] (Kaput, 2022)
[5] (Christine I. Noshi, 2018)

Other lesser-known ways to collect data for use in artificial learning are the infamous "Completely Automated Public Turing test to tell Computers and Humans Apart" (short CAPTCHA) where a human must identify a blurred or distorted object or text in a random image.[6] This data is then sent to Google servers, for example, where it is converted to training data for use by AI. Although this all sounds very interesting and could be investigated further, discussing all possible use cases for AI and its different types is beyond the scope of this paper. Here we will just be focusing on the specific usage of artificial learning and intelligence in the context of video games with two dimensions.

3. History and Milestones

Whether it's a robot that can play chess incredibly well, an artificial intelligence that beats the world champion of Go[7], or "just" the opponents from your favorite game that have once again tactically outplayed you: Computers are getting smarter and smarter, and especially in the context of games, they seem to have long surpassed human sophistication.[8] Artificial intelligence can either learn the desired behavior extremely quickly, especially in computer games, due to the above-average amount of data, or it can further train the already existing behavior. This was not always the case in AI technology; like many things, this behavior also had to be developed first. The beginning of modern artificial intelligence research was marked by a conference held in New Hampshire in 1956. This conference also decided on the name "artificial intelligence" and the problems they want the algorithm to solve. In the following years, many algorithms were developed that no longer serve today's requirements. Among these was the idea of Turing's intelligent machine (a further development of the Turing Machine), as well as the first chess robot, which however represents more a normal algorithm than an AI nowadays.[9]

One of the first analog games in which an AI was used to compete as an opponent for a human is chess, where "Deep Blue" was able to defeat a champion, Garry Kasparov, in chess for the first time in May 1997.[10] This game lasted for several days, with six games played, three of which were drawn.

[6] (Maruzani, 2021) (Google, 2020)
[7] An abstract strategy board game for two players in which the aim is to surround more territory than the opponent. (American Go Association)
[8] (Jiachao Fang, 2018)
[9] (Lenzen, 2020, p. 19 ff.)
[10] (IBM, 2007)

With this global success under its belt and armed with further knowledge in the field of artificial intelligence, the company DeepMind developed several versions of an AI for the strategy game "GO!". After several iterations, the development team has released the latest version, AlphaGo Zero, which has now far surpassed its predecessors entirely without human examples, using only the basic rules (of which there are only two in GO anyway, aside from the movement rules). After only three days of training, the AI was already able to play GO better than the version that beat the World Champion 4:1 in 2016.[11] However, this was a long road that was only completed in the 2000s - 2010s. Another breakthrough was only then: the concept of deep learning based on artificial neural networks (ANNs). Since then, the years have been characterized by more and more truly marketable products based on ANNs.[12] Extreme learning curves as seen in the AlphaGo Zero algorithm is an excellent example of what can be done with a large amount of data by not giving the neural network[13] any further information and letting it learn everything by trial and error. An example, that is a bit closer to the scope of the AI in this paper, among others, is the Pokémon trainer’s ANN in the game series of the same name, which first appeared in 1998. In these games, the AI decides on moves ("attacks" of the Pokémon) against the player himself.

[11] (Deep Mind, 2017)
[12] (Lenzen, 2020, p. 24 f.)
[13] Artificial neural networks (ANNs) are computational models inspired by networks of biological neurons (Z.R. Yang, 2014) (Science Direct, 2016)

4. Restrictions in modern games

Of course, games nowadays are fundamentally different from games in the 1990s in the sense that most games AI tries to learn today are digital. Even though computer programs such as artificial learning algorithms can much better comprehend a game that is written in machine language from the ground up there still is a significant number of limitations on how AI can be used in modern-day programs and computer games.[14] Some problems of the past still remain a barrier that we cannot seem to get around, such as the sheer absurdly high amount of possible moves after n[15] turns in a chess game. Because of the exponential growth of possibilities in that game, there (practically) is no such thing as a perfect chess algorithm since such an algorithm would take years to process.

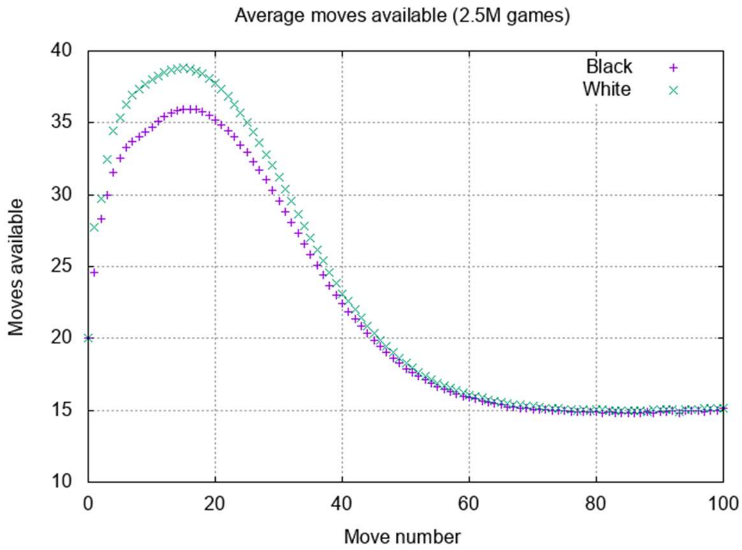

Figure 1: Number of moves available after n turns (Retrieved November 6, 2022, from https://chess.stackexchange.com/questions/23135/what-is-the-average-number-of-legal-moves-per-turn, self-approved)

5. An Example: AI in a 2D game[16]

In this paragraph, I will be talking about how the idea of the remake of an “intellectualized” Undertale came about and what kinds of issues as well as solutions became visible after starting with the project.

5.1. Idea and inspiration

The inspiration for the topic "Undertale remake" comes from my enthusiasm for the 2015 released indie game “Undertale”, developed by Toby Fox[17]. The game was nominated for the best indie game the same year it came out and blew away its fans and community with how captivating the storyline was written.[18] The only “problem” the game had from an AI-enthusiast-perspective was that both the dialogues as well as

[14] (Bartleby research, not stated)
[15] $n \in \mathbb{N}$
[16] (Hofmann, 2022)
[17] (Fox, 2022)
[18] (GameStar, 2015) (Orland, 2015)

the encounters with enemies in-game were hard coded into the game so there wasn't any way for the algorithm to evolve or get better at the game itself. And I wanted to change that.

5.2. Problems

Having the idea to start a project is one thing but implementing it is a whole other story so here are all the (little) problems I ran into while writing the code and designing the "finished" product.

Right away the first very important decision to make as a game developer is what game engine to use. As the original Undertale game was made in GameMaker Studio I also opted for GameMaker as my primary game engine.

The second very important question was what kind of machine learning algorithm I wanted to use for the simulated enemy in that game. I decided on reinforcement learning since this could be interesting as the player only has a limited movement range of 410x410 (minus 25 on both axis for player size) and the enemy (who must react to player movement) only has the option to pick between 5 different moves. For training the AI, The Tensorflow Keras framework and a custom OpenAI Gym environment was used to simulate the game. This posed the issue of not being able to migrate the AI to GameMaker after it had finished its training.

5.3. Solutions and Workarounds

Possible solutions for solving the AI-migration-problem could be to migrate the game itself to python using a python game rendering library like pygame or further investigate in the GameMaker language (GML)[19] to read the generated weight data for the neural network and process it for the enemy object to use it. As a short-term fix, the "trained AI state" has been hardcoded to how the AI would play if the neural net was trained perfectly. The "untrained AI state" is coded in properly as it just relies on randomly picking a legal move to execute.

There also was the problem of Tensorflow not accepting the custom environment given to it. This problem was later fixed by manipulating the box-shape of the observation space in the environment.[20]

---

[19] The programing language used by GameMaker (YoYo Games, 2022)
[20] (Hofmann, 2022, pp. learner.py, line 116)

5.4. Restrictions

The restrictions of the neural network are small but present. Different values for the environment variables for the algorithm can produce widely spread results depending on exactly those values. Setting a higher sample size (for example nb_steps_warmup=200.000)[21] strongly accelerates the learning process for the first 200.000 generations but the iteration speed falls off immensely after these repetitions, so using a lower sample size of about 20.000 but a bigger step sum of 50.000 to 500.000 (depending on your system specs) could vastly accelerate the learning process of the AI. Another really big factor for performance is the number of nodes used in the net. In this particular example, the AI is built on six dense nodes (layers), with eight neurons per layer in- or decreasing these numbers could also take a large impact on system performance.

Apart from the restrictions of the ANN itself, there are limitations for the enemy and the player due to the way the game is designed. The fact that the enemy can only use a maximum of five legal attacks at any given point in the game massively reduces the load on the AI. This restriction could be revisited, as it is purely a game design choice.

6. The project

This part of my work can be found at
https://github.com/Pytonballoon810/Seminararbeit-AI-2022

7. Conclusion

All in all, it should have become clear that AI is an impressive technology that, although we have achieved grandiose breakthroughs again and again in recent years, is still a long way from reaching a "perfect" state. We can only wait and marvel at what research into artificial intelligence will bring. As far as the project is concerned, it is of course far from the best theoretical solution, but that was not the goal anyways. The game demo shows very nicely the concrete difference between an untrained and a perfectly trained AI on a graphical level, with only two possible input axes. In real life, n-dimensional diagrams, and dynamic graphs[22] can process much more data at once and with much better contextuality. Of course, the game here also serves only to demonstrate the

[21] (Hofmann, 2022, pp. learner.py, line 236)
[22] Graphs that can predict the embedding of new nodes to the system. In most cases usen in neighborhood aggregation and Graph Neural Networks (GNN) (Menzli, 2022)

process. The further development of the game (especially with the implementation of a chatbot algorithm) would certainly be an interesting method to connect different neural networks even further. There are also many other widely spread subject areas which I did not touch on since talking about all these different areas of application would be far beyond the scope of this paper but after all, I still want to name a few of the more fascinating ones like machine learning in Augmented Reality (AR)[23] or full-on Virtual Reality (VR)[24]. Being able to use AI as a visual aid, possibly showing additional information for most everyday actions is especially helpful for data scientists and analysts. AI recognizes the environment and figures out what information to gather for you and where to display them in your field of view.[25] Another great application of AI is in the context of medical use as in detecting cancer in an early stage through pattern matching of biological data[26] and AI helping doctors to diagnose patients[27] (especially in the current times with the covid 19 infection this was a skill that came in handy for the doctors[28]). In conclusion, there is little left to conclude on the subject of AI. We can only look forward to the future and gaze at what research into the imitation of human intelligence will bring.

[23] (Anderson, 2021)
[24] (Gasparini, 2021)
[25] (Bezmalinovic, 2022)
[26] (Philipp Tschandl, 2020)
[27] (Divya S, 2018)
[28] (Xueyan Mei, 2020)

8. List of literature

American Go Association. (n.d.). A Brief History of Go. Retrieved November 6, 2022, from www.usgo.org/brief-history-go

Anderson, R. (2021, February 13). AR Apps With Artificial Intelligence in 2021. Retrieved November 5, 2022, from arpost.co/2021/02/13/ar-apps-with-artificial-intelligence-in-2021/#:~:text=Augmented%20reality%20and%20artificial%20intelligence,Snapchat%20lenses%20and%20Pokemon%20Go.

Bartleby research. (not stated). Advantages And Disadvantages Of AI In The Gaming System. Retrieved November 6, 2022, from www.bartleby.com/essay/Advantages-And-Disadvantages-Of-AI-In-The-PCFCZ33WCU

Bezmalinovic, T. (2022, September 11). NeuralPassthrough: Meta shows AI-based AR for VR headsets. Retrieved November 4, 2022, from mixed-news.com/en/neuralpassthrough-meta-shows-ai-based-ar-for-vr-headsets/

Błażej Osiński, K. B. (2018, July 5). What is reinforcement learning? Retrieved November 6, 2022, from deepsense.ai/what-is-reinforcement-learning-the-complete-guide/

Christine I. Noshi, A. I. (2018, December). The Role of Big Data Analytics in Exploration and Production. Retrieved November 5, 2022, from onepetro.org/SPEHOCE/proceedings-abstract/18HOCE/1-18HOCE/D012S021R001/214816

Deep Mind. (2017, October 18). AlphaGo Zero: Starting from scratch. Retrieved November 6, 2022, from www.deepmind.com/blog/alphago-zero-starting-from-scratch

Divya S, I. V. (2018). A Self-Diagnosis Medical Chatbot Using Artificial Intelligence. Retrieved November 7, 2022, from core.ac.uk/download/pdf/230494941.pdf

Fox, T. (2022). Undertale. Retrieved November 6, 2022, from undertale.com/about/

GameStar. (2015). Undertale. Retrieved November 6, 2022, from www.gamestar.de/spiele/undertale,1437.html

Gasparini, A. (2021, November 12). Stanford researchers are using artificial intelligence to create better virtual reality experiences. Retrieved November 6, 2022, from news.stanford.edu/2021/11/12/using-ai-create-better-virtual-reality-experiences/

Google. (2020). What is recaptcha? Retrieved November 5, 2022, from www.google.com/recaptcha/about/

Hofmann, P. (2022). Seminararbeit-AI-2022. Retrieved November 6, 2022, from github.com/Pytonballoon810/Seminararbeit-AI-2022

IBM. (2007, April 07). Deep Blue. Retrieved November 6, 2022, from www.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/

Image Engineering. (2019). Image Stabilization. Retrieved November 5, 2022, from image-engineering.de/library/image-quality/factors/1076-image-stabilization

Jiachao Fang, H. S. (2018, June 3). Will Artificial Intelligence Surpass Human Intelligence? Retrieved November 6, 2022, from papers.ssrn.com/sol3/papers.cfm?abstract_id=3173876

Kaput, M. (2022, April 18). What Is Artificial Intelligence for Social Media? Retrieved November 5, 2022, from deepsense.ai/what-is-reinforcement-learning-the-complete-guide/

Laura Rice, S. B. (2022). How an AI system beating a video game could lead to better autonomous vehicles. Retrieved November 4, 2022, from texasstandard.org/stories/how-an-ai-system-beating-a-video-game-could-lead-to-better-autonomous-vehicles/

Lenzen, M. (2020). *Künstliche Intelligenz; Fakten, Chancen, Risiken.* Munich: Beck.

Maruzani, R. (2021, January 26). Are You Unwittingly Helping to Train Google's AI Models? Retrieved November 6, 2022, from towardsdatascience.com/are-you-unwittingly-helping-to-train-googles-ai-models-f318dea53aee

Menzli, A. (2022, July 21). Graph Neural Network and Some of GNN Applications. Retrieved November 4, 2022, from https://neptune.ai/blog/graph-neural-network-and-some-of-gnn-applications

Orland, K. (2015, December 27). The best video games of 2015, as picked by the Ars editors. Retrieved November 5, 2022, from arstechnica.com/gaming/2015/12/the-best-video-games-of-2015-as-picked-by-the-ars-editors/4/

Philipp Tschandl, C. R. (2020, June 22). Human–computer collaboration for skin cancer recognition. Retrieved November 6, 2022, from www.nature.com/articles/s41591-020-0942-0

Science Direct. (2016). Artificial Neural Network. Retrieved November 6, 2022, from www.sciencedirect.com/topics/neuroscience/artificial-neural-network

Xueyan Mei, H.-C. L.-y. (2020, May 19). Artificial intelligence–enabled rapid diagnosis of patients with COVID-19. Retrieved November 7, 2022, from www.nature.com/articles/s41591-020-0931-3

YoYo Games. (2022). GML Code Overview. Retrieved November 1, 2022, from manual.yoyogames.com/GameMaker_Language/GML_Overview/GML_Overview.htm

Z.R. Yang, Z. Y. (2014). *Comprehensive Biomedical Physics.* Elsevier. Retrieved November 6, 2022, from www.sciencedirect.com/referencework/9780444536334/comprehensive-biomedical-physics#book-info

9. List of figures

10. Attachments

```python
"""create_environments module of "AI training python"

Used to generate the training_data.csv file which is used for
simulating different environments (ENVs)

Returns:
    None
"""
# My standard linting settings
# pylint: disable=trailing-whitespace
# pylint: disable=logging-fstring-interpolation
# pylint: disable=line-too-long

import random
import csv
from math import sqrt

# possible_moves are "shockwave", "bl", "tl", "br", "tr"
x, y = (410, 410) # sprite_width and sprite_height of play_frame in GM
middle_x, middle_y = (x/2, y/2)
middle_point = (middle_x, middle_y)
RADIUS = 150 # threshold value of the radius(distance) to choose the "shockwave" option, RADIUS is a constant
FILENAME = "training_data.csv" # the name of the file to save the training data to, FILENAME is a constant
NUM_OF_START_ENVS = 2000

def get_points(max_x:int, max_y:int, number_of_points:int) -> list[tuple]:
    """Generate a list of random points on the plane
        0 < x <= max_x
        0 < y <= max_y

    Args:
        max_x (int): the maximum x value for the randomly generated points
        max_y (int): the maximum y value for the randomly generated points
        number_of_points (int): the number of random points to generate

    Returns:
        list[tuple] or list[x, y]: list of points with x and y coordinates
    """
    points:list[tuple] = []
    for _i in range(number_of_points):
        ran_x = random.randint(0, max_x)
        ran_y = random.randint(0, max_y)
        point = (ran_x, ran_y)
        points.append(point)
    return points

def calculate_best_move(point:tuple) -> str:
    """Calculate the best move for a given point.

    Args:
        point (tuple): point with x and y coordinate in range of max_x and max_y

    Returns:
        str: the calculated best move for the given point
    """
    vector_x = point[0] - middle_point[0]
    vector_y = point[1] - middle_point[1]
    distance = sqrt((vector_x**2 + vector_y**2))
    if distance <= RADIUS: # if the point is in the circle
        best_move = "shockwave"
    elif point[0] < middle_x: # if point on the left side of coordinate system
        if point[1] < middle_y:
            best_move = "bottom_left"
        else:
            best_move = "top_left"
    elif point[0] >= middle_x:
        if point[1] < middle_y:
            best_move = "bottom_right"
        else:
            best_move = "top_right"
```

```python

    return best_move

def compile_data(point:tuple, calculated_move:str) -> tuple:
    """Compiles the data to a single argument(tuple). Would also be possible to do inline.
        Better to do it in a separate function to have the possibility to add/modify the given arguments.

    Args:
        point (tuple): point with x and y coordinate in range of max_x and max_y
        calculated_move (str): the calculated move by calculate_best_move()

    Returns:
        tuple: tuple of ((point), calculated_move)
    """
    return (point, calculated_move)

def create_csv(filename:str) -> None:
    """Sets up a csv file for the given path or creates one of no file with the given path exists.

    Args:
        filename (str): the path of the csv file. Also works with relative paths
    """
    with open(filename, "w", encoding="UTF-8", newline="") as training_file:
        csvwriter = csv.writer(training_file)
        csvwriter.writerow(["point", "move"])

def append_data(filename:str, data:list[tuple, str]) -> None:
    """Append the given data to the csv file.

    Args:
        filename (str): the path of the csv file. Also works with relative paths
        data (list[tuple, str]): the data to append to the given file.
            Takes a list of a tuple and string as the point(x, y) and the calculated best move
    """
    with open(filename, "a+", encoding="UTF-8", newline="") as training_file:
        csvwriter = csv.writer(training_file)
        csvwriter.writerows(data)

if __name__ == "__main__": # Standard python syntax to test for __name__ == "__main__" to run code in a program only from the program itself
    point_data:list[tuple, str] = []
    create_csv(FILENAME)
    for random_point in get_points(x, y, NUM_OF_START_ENVS):
        BEST_MOVE = calculate_best_move(random_point) # BEST_MOVE is a constant (for each point)
        point_data.append(compile_data(random_point, BEST_MOVE))
    append_data(FILENAME, point_data)

```

```python
"""learner module of "AI training python"

The simulation engine to run the loaded environment.

Returns:
    None
Exports:
    "dqn_weights.h5f.index" exports an index file for the calculated weights
"""
# My standard linting settings
# pylint: disable=trailing-whitespace
# pylint: disable=logging-fstring-interpolation
# pylint: disable=line-too-long

import random
import os
# Import the Sequential model from keras
# as well as the flatten node to flatten out the 2-dimensional inputs and
# dense nodes as the default tensorflow deep leaning node
from keras.models import Sequential
from keras.layers import Dense, Flatten, Input # Flatten import unused for personal environment
from keras.optimizers import Adam

import pandas # for reading csv files

from gym import Env # for creating our own environment with gym
from gym.spaces import Discrete, Box

from rl.agents import DQNAgent # Multiple agents possible (see https://keras-rl.readthedocs.io/en/latest/) [DQN, NAF, DDPG, SARSA, CEM]
from rl.policy import BoltzmannGumbelQPolicy # Value or policy based reinforcement learning -> here: policy (BoltzmannGumbelQPolicy)
from rl.memory import SequentialMemory # To maintain memory

import numpy as np

#Custom imports:
import create_environments

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2' # fixing the "Could not load dynamic library 'cudnn64_8.dll'; dlerror: cudnn64_8.dll not found" error, but also disables the option to run from the GPU

# Possible to use one or multiple lines from the "training_data.csv"
TRAINING_DATA_FILE = create_environments.FILENAME
LINE_FROM_ENVIRONMENT_FILE = 1

# User specific configuration
TESTING = True # should the neural network be tested after it is finished with training?
SHOW_NET_STRUCT = True # prints out the net structure on initialization of the learning process
SAVE_WEIGHTS = True # should the weights be saved after the training is finished?
SAVE_WEIGHTS_AS = "dqn_weights.h5f" # if the weights are saved, this is the filename

def point_in_rectangle(point:tuple[int, int], rect:tuple[int, int, int, int]) -> bool:
    """Global function/
    Calculate if point is in the area of a rectangle

    Args:
        point (tuple[int, int]): x, y of the point that is given
        rect (tuple[int, int, int, int]): x1, y1, x2, y2 of the rectangle to test for

    Returns:
        bool: true if point is inside, false if it is not inside
    """
    x, y = point
    x_1, y_1, x_2, y_2 = rect
    if (x_1 < x and x < x_2):
        if (y_1 < y and y < y_2):
            return True
```

```
    return False

def point_in_circle(point:tuple[int, int], circle:tuple[int, int, int]) -> bool:
    """Global function/
    Calculate if point is in the area of a circle

    Args:
        point (tuple[int, int]): x, y of the point that is given
        circle (tuple[int, int, int]): x1, y1, radius of the circle to test for

    Returns:
        bool: true if point is inside, false if it is not inside
    """
    x, y = point
    x_1, y_1, rad = circle
    if (x-x_1)**2 + (y-y_1)**2 < rad**2:
        return True
    return False

def calculate_aoe_hit(action_id:int, target:tuple[int, int]) -> bool:
    """Global function/
    Calculate if the action belonging to the action id would hit the target

    Args:
        action_id (int): action_id converted to an action with Move.moves[action_id]
        target (tuple[int, int]): the point trying to be hit

    Returns:
        bool: true if point is hit, false if it is not hit
    """
    action = Move.moves[action_id]
    square_moves = [Move.b_l, Move.b_r, Move.t_l, Move.t_r]
    if action in square_moves:
        return point_in_rectangle(target, Move.aoe[action])
    else:
        return point_in_circle(target, Move.aoe[Move.shockwave])

class Environment(Env):
    """Gym Env Inheritance of the object of the OpenAI gym environment

    Args:
        Env (gym.Env): inheriting the gym.Env properties
    """
    def __init__(self, player_pos:tuple[int, int]) -> None:
        """The constructor function for an custom Environment

        Args:
            player_pos (tuple[int, int]): the player position for which the environment is set up
        """
        # Inherit from Env
        super().__init__()
        # self.observation_space = Box(low=np.array(playfield_zeros, dtype=np.float32), high=np.array(playfield_max, dtype=np.float32), dtype=np.float32) # action array
        self.observation_space = Box(low=0, high=1, shape=(410, 410), dtype=np.uint8)
        self.action_space = Discrete(5, start=0) # actions we can take (Move.moves)
        # LEGACY: using indexed state variable
        # self.state = random.choice([Move.shockwave, Move.b_l, Move.b_r, Move.t_l, Move.t_r]) # set start action
        self.state = random.randint(0, 4)
        self.player_position = player_pos
        self.player_position_move = 10

    def step(self, action:int) -> tuple[int, int, bool, dict]:
        """overwriting the step function from the gym.Env class
        # would be possible to user super().__init__() for perfect implementation, but is simply not required here

        Args:
            action (int): action index for the self.state variable

```

```python
        Returns:
            tuple[int, int, bool, dict]: current state, reward (either -1 or 1), done (true when player movement was simulated 10 times)
        """
        # LEGACY: using indexed state variable
        # self.state = Move.moves[action]
        self.state = [[0 for x in range(410)] for y in range(410)]
        self.state[self.player_position[0]][self.player_position[1]] = 1
        self.player_position_move -= 1
        # print(self.state, self.player_position) # DEBUGGING
        # Add player move noise
        self.player_position = self.player_position[0]+random.randint(-self.player_position_move, self.player_position_move), self.player_position[1]+random.randint(-self.player_position_move, self.player_position_move)
        if calculate_aoe_hit(action, self.player_position):
            self.state = [[0 for x in range(410)] for y in range(410)]
            self.state[self.player_position[0]][self.player_position[1]] = 1
            reward = 1
        else:
            reward = -1
        if self.player_position_move <= 0:
            done = True
        else:
            done = False

        info = {} # placeholder (required by OpenAI)

        return self.state, reward, done, info

    def render(self):
        """placeholder for a possible implementation with pygame or tkinter. Not done for time reasons and the non necessity
        """

    def reset(self) -> int:
        # Number of parameters was 3 in 'Env.reset' and is now 1 in overridden 'Environment.reset' method
        """called periodically after each iteration of the specified step amount

        Returns:
            int: current state to pass onto the next interval
        """
        # LEGACY: using indexed state variable
        # self.state = random.choice([Move.shockwave, Move.b_l, Move.b_r, Move.t_l, Move.t_r]) # set start action (same as above)
        self.state = random.randint(0, 4)
        self.player_position_move = 10 # Reset player move noise
        return self.state

#Dataclass
class Move():
    """dataclass for specifying the possible moves and packing them into lists and dictionaries
    @dataclass decorator not used since it would be overkill
    """
    shockwave = "shockwave"
    b_l = "bottom_left"
    b_r = "bottom_right"
    t_l = "top_left"
    t_r = "top_right"

    moves = [shockwave, b_l, b_r, t_l, t_r]

    aoe = {"bottom_left":(0, 0, 205, 205),        # define the area of effect to calculate if a non-perfect move would still succeed
           "bottom_right":(205, 0, 410, 205),     # point defined like (x_1, y_1, x_2, y_2)
           "top_left":(0, 205, 205, 410),
           "top_right":(205, 205, 410, 410),
           "shockwave":(205, 205, create_environments.RADIUS)}    # circle defined like (middle_x, middle_y, radius)

```

```python
class Agent():
    """agent class used to build a custom agent object
    """
    def __init__(self) -> None:
        """self.env creates our custom environment for the specified x, y start position of the player.
        Also possible to iterate over them using a for loop, for utilizing the generated training_data in the training_data.csv
        """
        self.training_data = pandas.read_csv(TRAINING_DATA_FILE)

        self.env = Environment((254, 82))
        # print(self.env.step(4)) # DEBUGGING


    def build_model(self) -> Sequential: # actions:int
        """function for building a keras model

        Args:
            actions (int): amount of actions to build the model for (in this case 5)

        Returns:
            Sequential: returns the Sequential class of the model created by the keras module
        """
        states = self.env.observation_space.shape
        actions = self.env.action_space.n
        # print(states, actions) # DEBUGGING
        model = Sequential()
        # LEGACY: no need to flatten out the custom env with an input_shape before usage # TODO
        # model.add(Flatten(input_shape=states))
        model.add(Input(shape=(410, 410, 5)))
        model.add(Dense(256, activation="relu")) # Dense node layer as standard keras neuron to generate deep reinforcement learning algorithms
        model.add(Dense(128, activation="relu"))
        model.add(Dense(64, activation="relu"))
        model.add(Dense(32, activation="relu"))
        model.add(Dense(8, activation="relu"))
        model.add(Flatten())
        model.add(Dense(actions, activation="softmax"))
        model.summary()
        print(model.input_shape)
        return model

    def build_agent(self, model:Sequential) -> DQNAgent:
        """function for building a keras model

        Args:
            model (Sequential): _description_

        Returns:
            DQNAgent: personal DQNAgent (see line 20 for more info "from rl.agents import DQNAgent")
        """
        actions = self.env.action_space.n # = 5
        policy = BoltzmannGumbelQPolicy()
        memory = SequentialMemory(limit=50_000, window_length=1)
        print(model.output_shape)
        dqn = DQNAgent(model=model, memory=memory, policy=policy, nb_actions=actions, nb_steps_warmup=20_000, target_model_update=1e-2)
        return dqn


if __name__ == "__main__":
    # All personal objects are labeled with the prefix "my_"
    my_agent = Agent()
    # LEGACY: env is created in the __init__ of Agent
    # env = agent.create_environment("(254, 82)","shockwave")
    my_model = my_agent.build_model()
    # my_model.summary()
    print("here")
```

```
    my_dqn = my_agent.build_agent(my_model)

    my_dqn.compile(Adam(learning_rate=0.01), metrics=["mae"]) # mae = mean absolute error
    my_dqn.fit(my_agent.env, nb_steps=50_000)
    if TESTING:
        scores = my_dqn.test(my_agent.env, nb_episodes=100, visualize=False) #nb_episodes = amount of testing episodes to run
        print(np.mean(scores.history["episode_reward"]))
    if SAVE_WEIGHTS:
        my_dqn.save_weights(SAVE_WEIGHTS_AS, overwrite=True)
```

11. Closing statement

I have prepared this seminar paper without outside help and have used only the sources and aids listed in the bibliography.

Dietzhof,	19.02.2023	
Place	Date	Signature