

Driving trainer with ML-Agents

LAB University of Applied Sciences

Bachelor of Engineering, Information and Communication Technology

2023

Alejandro García Villegas

Abstract

Author(s) García Villegas, Alejandro	Publication type Thesis, UAS	Completion year 2023
	Number of pages 50	
Title of the thesis Driving trainer with ML-Agents		
Degree, Field of Study Bachelor of Engineering, Information and Communication Technology		
Organisation of the client		
Abstract <p>Artificial intelligence has witnessed remarkable progress in recent years, with machine learning being a crucial milestone in transforming the field. Reinforcement learning, a model that trains an algorithm through trial and error, has emerged as a significant breakthrough with the potential to revolutionize various industries, including the automotive sector.</p> <p>This thesis aimed to create a program capable of training a car to complete a road using the user's driving example as input. A program in Unity was developed to achieve the objective, which uses reinforcement learning to train the model.</p>		
Keywords Artificial intelligence, machine learning, reinforcement learning, agent, autonomous car		

Contents

1	Introduction.....	1
2	Autonomous cars.....	2
2.1	Definition	2
2.2	First attempts.....	3
2.3	Present situation.....	4
2.4	Expectations for the Future.....	5
3	Artificial Intelligence	8
3.1	Definition	8
3.2	History	9
4	Machine Learning	12
4.1	Introduction.....	12
4.2	Supervised Learning.....	14
4.2.1	Classification	16
4.2.2	Regression	18
4.3	Unsupervised Learning.....	18
4.4	Semi-supervised Learning	20
4.5	Reinforcement Learning.....	23
5	Game engines	28
5.1	Unity	28
5.1.1	Video Games.....	28
5.1.2	Non-gaming Industries.....	30
5.2	Unreal Engine.....	30
6	Artificial Intelligence in Game Engines	33
6.1	Introduction.....	33
6.2	AirSim.....	35
6.3	CARLA	36
6.4	ML-Agents	37
7	Practical case	40
7.1	Introduction.....	40
7.2	Unity Preparation.....	40
7.3	AI Creation	44
8	Summary.....	49
	References	51

1 Introduction

Computers and other technological tools are indispensable and influential in the modern world. They have developed rapidly over the years and have become widespread in our daily routines. With their vast array of capabilities, it is clear that computing can help us tackle many of the challenges we face today.

Of all the challenges we face in the modern era, few are as significant or worrisome as the frequency of car accidents. The fact that many of these incidents are caused by human negligence is beyond dispute. To address this issue, we must develop cars capable of driving themselves safely and competently.

Although driving is an everyday activity that many people around the globe share, it is essential to acknowledge that there can be subtle divergences in the regulations and norms of each country. Notably, using separate signals for turning left, right, or going straight was more common in Finland, whereas a single sign is often used for going forward or turning left in Spain. Additionally, the prevalence of roundabouts and traffic light-controlled intersections differed between the two nations. These are just a few examples of the variations that exist between countries.

As previously noted, computer science is a rapidly expanding and vast field encompassing many subfields, including artificial intelligence. Within AI, reinforcement learning is a particularly promising area gaining more and more significance. Reinforcement learning algorithms can train a model through trial and error without needing a pre-existing database. The growth of artificial intelligence has been remarkable, surpassing what was thought possible just a few years ago. Given this growth, why not harness the power of reinforcement learning to train cars and improve driving safety in the future?

This thesis aims to create a driving trainer that can recognize areas for improvement in a user's driving based on an example provided by the user. However, developing a machine learning program that can accurately perform this task is a challenging undertaking. It requires a significant amount of time and effort, as well as identifying the most appropriate policy for the problem at hand.

2 Autonomous cars

2.1 Definition

An autonomous car, also known as a self-driving car or driverless car, is a vehicle that can travel without human input. This vehicle can also sense and react to its environment at the same level as an experienced human driver. According to the Society of Automotive Engineers, we can find six levels of driving automation, ranging from “*fully manual*” to “*fully autonomous*.” The U.S. Department of Transportation has also adopted these levels. (Synopsis 2023)

This disruptive technology is expected to transform the transportation industry in the following years, whose development is driven by several factors, such as the desire to reduce traffic accidents, the need to improve transportation efficiency, and the potential for new business models. They also have the potential to reduce energy consumption and carbon emissions by optimizing driving patterns. (Arrow 2022)

As we can see in Image 1 and technically according to the definition of an autonomous car, a “*fully autonomous*” car would fit inside level 5 of driving automation, and it would be a car that can drive itself all the time. However, when we think of an autonomous car, we usually think of one that automatically drives you wherever you want to without any other human input. These cars would fall into level 3 or 4 of driving automation. (Sipe 2023)

SYNOPSIS

LEVELS OF DRIVING AUTOMATION

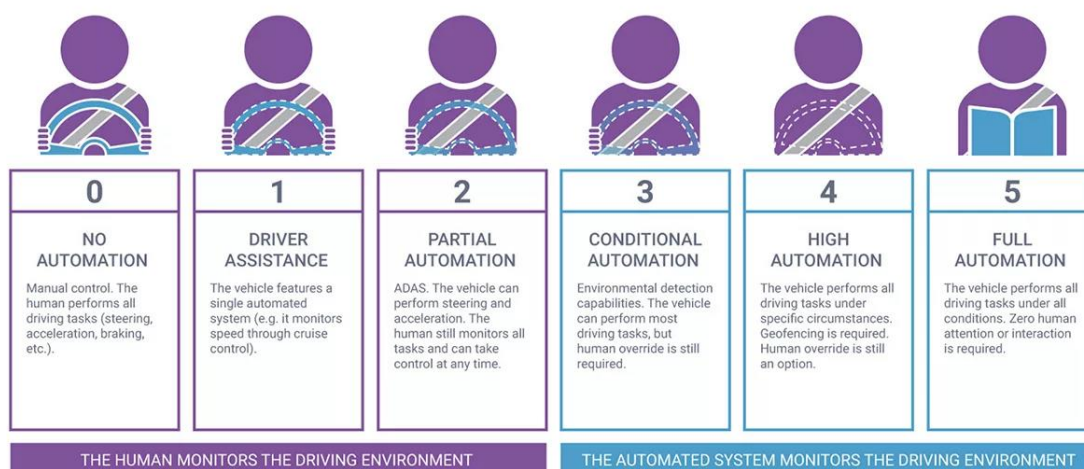


Image 1. Different levels of driving automation (Synopsis 2023)

Autonomous cars use complex sensors, algorithms, and machine learning systems to execute the needed software to drive. They create a map of their surroundings based on the information they get from the different sensors, and they monitor the position of nearby vehicles, road signs, and pedestrians. Most of these sensors use pulses of light or ultrasounds and video cameras to measure all the needed parameters. Then, the software's job starts, processing all this sensory input and sending the instructions to the actuators. (Synopsys 2023)

But despite the many benefits of this technology, several challenges need to be addressed before becoming widely available. These include technical challenges, such as ensuring the safety of the technology used, and regulatory challenges, such as establishing clear guidelines and standards for these cars and addressing legal liability in the event of an accident. (Arrow 2022)

2.2 First attempts

The concept of autonomous cars can be traced back to the early 20th century when inventors began exploring technology to automate the driving process. In the 1920s, Houdina Radio Control, a radio equipment firm, built a radio-controlled vehicle that could drive itself around a track. Still, the technology needed to be more advanced to make self-driving cars a practical reality at the time. (Arrow 2022)

Fast forward to the 1950s and 1960s, and several car manufacturers and research institutions began experimenting with autonomous vehicle technology. General Motors (GM) was one of the pioneers in this field, developing a series of experimental vehicles that used radar sensors and electronic controls to assist the driver or take over some driving functions. The 1956 GM Firebird II was a notable example, equipped with an early version of an automated driving system that could follow a wire embedded in the road. (Guthrie 2023)

However, this concept started to appear in everyone's life around the 1980s, with some films and series becoming famous then. During those years, some attempts were made in this area of work, and The Robotics Institute, with the help of Carnegie Mellon, created the Navlab5, the first "*self-driving*" car, which we can see in Image 2. It was successfully piloted from Pittsburgh to San Diego (around 3800km long route). Despite the actual driver being responsible for accelerating and braking, the car itself was able to utilize cameras and sensors to steer and navigate over 3500km without human intervention, which is considered a huge milestone even if we cannot properly call it an "*autonomous*" car. The same companies created other cars, but some were semi-autonomous or fully autonomous but in limited scenarios. (Arrow 2022)



Image 2. Navlab5 (Russell 2015)

In the following years, several research institutions and car manufacturers invested heavily in autonomous vehicle technology, aiming to make self-driving cars a reality. In 1995, Mercedes-Benz introduced the first adaptive cruise control system, which used radar sensors to adjust the car's speed to maintain a safe distance from the vehicle in front. In 1997, the Defense Advanced Research Projects Agency (DARPA) began funding research into autonomous cars, culminating in the launch of the DARPA Grand Challenge in 2004. (Guthrie 2023)

In 2005, the DARPA Grand Challenge was held again, with several teams completing the course. This marked a significant milestone in the development of autonomous vehicle technology, demonstrating that self-driving cars were no longer a far-fetched dream but a real possibility. (Guthrie 2023)

Since then, several car manufacturers, including Toyota, Google, and Tesla, have invested heavily in autonomous vehicle technology, intending to develop self-driving cars that could one day become a common sight on our roads. Significant advancements have been made in the technology, with improved sensors, more sophisticated algorithms, and greater processing power enabling autonomous cars to navigate complex environments and make split-second decisions. (Arrow 2022)

2.3 Present situation

Nearly ten years ago, it was predicted that we would be using self-driving cars by today, but it seems that the difficulties involved in this technology still need to be fully estimated. There

is still a lot of time and work required to achieve Level 5 autonomous cars since, nowadays, even though scientists and businesses are obtaining constant but slow progress, we can only find Level 2 and some Level 3 cars for sale. The technology to deliver autonomous vehicles safely is indeed around 80% developed. Still, the last 20% is increasingly tricky as challenges, such as unusual and rare street or road events, still need to be solved. (Sipe 2023)

Several companies, such as Waymo, Tesla, Cruise, and Uber, have tested their autonomous vehicles on public roads for years. These tests have helped advance the technology, identify safety issues, and refine algorithms. However, autonomous cars are still in the early stages of development. (Guthrie 2023)

Safety is one of the biggest challenges facing autonomous car development right now. Safety is a critical issue for autonomous vehicles, and there have been accidents involving autonomous cars in the past. Safety concerns arise from the complex algorithms that govern autonomous vehicles and the ability of sensors to detect and respond to road hazards accurately. However, as technology advances, safety concerns will be addressed, and the safety of autonomous vehicles will continue to improve. (Arrow 2022)

Another challenge companies are facing for autonomous cars is regulatory and legal hurdles. The regulatory framework for autonomous cars is still being developed, and it is still being determined how autonomous vehicles will be regulated and integrated into the transportation system. The legal framework must also be designed to address liability issues and insurance coverage. (Arrow 2022)

Despite the challenges facing the development of autonomous cars, there have been significant technological advances. Autonomous cars' growth is driven by increased investment in research and development by companies, research institutions, and government agencies. In addition, advancements in machine learning and AI algorithms are helping improve autonomous vehicles' performance and safety. (Sipe 2023)

2.4 Expectations for the Future

This technology will undoubtedly continue being developed, but slower, with less hype and money invested. The increasing cost of capital also makes it harder for new start-ups to get funds in this area. (Sipe 2023)

Autonomous cars are expected to change transportation and significantly impact various industries and sectors. For instance, the manufacturing and supply chain sectors will have to adjust to meet the needs of autonomous cars. Self-driving vehicles will require more

advanced parts and components, including sensors, cameras, and software, requiring increased production and supply chain management. This, in turn, will create new job opportunities and stimulate economic growth. (Sipe 2023)

However, Mercedes-Benz believes we could see level 4 cars on the road by 2030. During a media event, Markus Schäfer, chief technology officer, told journalists that advanced self-driving capabilities are soon achievable. According to the definition previously seen in Image 1, a level 4 car could self-drive in almost every situation, allowing occupants to conduct other activities or even sleep while in motion. (Guthrie 2023)

Mercedes-Benz was able to offer a system called Drive Pilot by May 2022, which is capable of managing the car's speed, braking, and steering and is approved for use at speeds up to 60km/h, making it one of the level 3 autonomous cars approved for use on public roads in Germany and Nevada. (Guthrie 2023) This fact makes it more possible to believe that they will achieve what Schäfer said.

The insurance industry will also have to adjust to the arrival of autonomous cars. With the decrease in accidents and fatalities caused by human error, insurance companies must reevaluate their business models to reflect the lower risk of accidents. The shift towards car-sharing and on-demand transportation will also require new insurance policies that cover multiple drivers and vehicles. (Arrow 2022)

The real estate industry may also experience changes due to autonomous cars. With reduced demand for parking spaces and garages, the need for parking infrastructure may change, leading to a shift in how we design and develop urban areas. (Arrow 2022)

Another expectation for autonomous cars is to reduce the stress and frustration associated with driving. With autonomous vehicles, drivers will no longer have to worry about traffic, navigation, or finding parking, freeing up time and energy for other activities. This can lead to a decrease in stress-related health problems and an increase in overall well-being. (Arrow 2022)

Autonomous cars can also improve accessibility and mobility for people with disabilities or older people with difficulty driving or accessing public transportation. With autonomous vehicles, these individuals can have more independence and freedom, allowing them to live a more fulfilling life. (Arrow 2022)

Furthermore, autonomous cars are expected to create new opportunities for businesses and entrepreneurs. For instance, autonomous delivery vehicles can revolutionize delivering goods, improving efficiency and reducing business costs. Additionally, autonomous cars

can enable new advertising, entertainment, and e-commerce forms, as passengers have more free time and opportunities to engage with digital content. (Sipe 2023)

3 Artificial Intelligence

3.1 Definition

Artificial intelligence is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable. (IBM)

Artificial Intelligence (AI) is a complex and multifaceted field that has been a subject of research and development for several decades. It enables problem-solving by combining computer science and robust datasets. It is also a branch of computer science that creates intelligent machines that simulate human cognitive functions such as learning, problem-solving, decision-making, and perception. AI uses techniques such as machine learning, natural language processing, deep learning, and neural networks, among others, to create intelligent systems that can perform a wide range of tasks, such as predictions or classifications based on input data. (Anyoha 2017)

According to Stuart Russell and Peter Norvig's textbooks, nowadays delve AI into four potential goals or definitions, which differentiate them based on thinking vs. acting. These four definitions are: systems that think like humans, systems that act like humans, systems that reason, and systems that work rationally. (IBM)

We can find three primary levels of artificial intelligence: narrow, general, and strong AI. Each one of them can offer different and essential things, especially in the field of computing and gaming. (Wirtz 2023.)

Narrow AI, or weak AI, refers to creating systems focused on a specific task. These artificial intelligence programs can only serve a few determined functions and must work correctly outside their essential duties. Narrow AI has been described as having a preprogrammed purpose. (Wirtz 2023.)

General AI refers to the creation of systems that have the potential to learn anything a human can. These machines can experience many of humans' cognitive functions (reasoning, logic, planning, communication, and learning). (Wirtz 2023.)

When we talk about Strong AI, we refer to a theoretical type of AI that involves creating machines with the ability to exhibit human-like intelligence or even surpass it. This requires a self-aware consciousness that could solve problems, learn, and plan. (IBM)

One of the main advantages of AI is its ability to process vast amounts of data at incredible speeds, which can lead to breakthroughs in fields such as healthcare, finance, and science. For example, AI is used to develop more accurate and efficient diagnostic tools for diseases such as cancer, optimize investment portfolios, and predict natural disasters. AI is also being used to improve customer service through chatbots and virtual assistants, which can provide 24/7 support and assistance. (Anyoha 2017)

However, the development and use of AI also pose some challenges and risks. One of the most pressing concerns is the potential impact of AI on employment, as many tasks previously performed by humans may become automated. This could lead to significant job displacement, particularly for workers in industries such as manufacturing and transportation. (Anyoha 2017)

Another significant challenge associated with AI is the potential for bias and discrimination, which can arise when AI systems are trained on limited or incomplete data sets. This can lead to unfair treatment of particular groups, such as minorities or women. Additionally, using AI in surveillance and law enforcement areas raises concerns about privacy and civil liberties. (Anyoha 2017)

3.2 History

Artificial intelligence started as a concept due to some science fiction movies in the first half of the 1900s. The creation of some characters, such as the “heartless” Tin Man from *The Wizard of Oz*, made some scientists ask whether machines could think, leading to a generation of people assimilating the concept of AI culturally. One of those mathematicians, Alan Turing, often called the father of computer science, explored the mathematical possibility of AI and suggested that if humans used available information and reason to make decisions, machines could do the same thing. (Anyoha 2017)

In the 1950s, researchers believed that it would be possible to create a machine that could perform any intellectual task that a human being could do. This led Turing to publish a paper discussing how to build intelligent machines and test their intelligence. In his work, he offered a test known as the Turing Test, where a human would try to distinguish between a computer and a human during a text conversation. If the machine could convince the human that it is another human, it was said to have passed the Turing test. (IBM) Nowadays, this test has a different purpose and meaning than it was meant to have since AI has evolved a lot, but it remains an essential part of the history of AI.

In 1956, the Dartmouth Summer Research Project on Artificial Intelligence was held. At this conference, Allen Newell, Cliff Shaw, and Herbert Simon presented the Logic Theorist program, designed to mimic a human's problem-solving skills. This is considered by many people the first artificial intelligence program. The significance of this event led to the next twenty years of research in the field. (Anyoha 2017)

In the 1960s and 1970s, AI flourished. Computers were upgraded in many ways and became more accessible, and machine learning algorithms also improved and became more common. Optimism was high, and the even higher expectations led to discordance between what was possible then and what people were aiming for. The following years were invested in improving machines and algorithms, and research was done on what was possible, which meant a significant improvement in AI. (Anyoha 2017)

Researchers began exploring new AI approaches, including machine learning and neural networks, in those years. Machine learning involves training computers to recognize patterns and make predictions based on data, while neural networks are modeled after the structure and function of the human brain. These approaches allowed AI systems to learn from experience and adapt to new situations. (Anyoha 2017)

However, progress in AI research slowed down in the 1980s and 1990s as funding for AI projects decreased, and many researchers turned their attention to other fields. This period is often referred to as the "AI winter." Despite this slowdown, there were still some critical developments in AI during this time, such as the development of expert systems and the integration of AI into everyday technology, such as voice recognition systems and video games. These expert systems were designed to mimic the decision-making processes of human experts in a particular field. These systems were also often used in medicine, where they could help diagnose diseases and suggest treatments. (IBM)

In the late 1990s and early 2000s, computing power and data storage advances led to a resurgence in AI research. This period saw the development of machine learning techniques such as support vector machines and random forests, large datasets such as ImageNet, and deep learning algorithms that could process and analyze this data. (IBM)

In 2015, DeepMind created AlphaGo, a program developed to play Go, and in 2016 it beat the world champion Go player in a five-game match. (IBM) This is considered a massive milestone for AI, and it became a huge inflection point in the perception and development of new technologies.

The future of AI will likely involve continued advances in machine learning and neural networks and the development of new approaches to AI, such as quantum computing and

neuromorphic computing. There is also likely to be an increased focus on developing transparent and explainable AI systems so that their decision-making processes can be understood and validated.

4 Machine Learning

4.1 Introduction

Machine learning is a branch of artificial intelligence (AI) and computer science that focuses on using data and algorithms to imitate how humans learn, gradually improving its accuracy. (IBM)

Machine learning has become one of the fastest-growing fields in artificial intelligence and computer science thanks to its ability to make predictions or decisions based on data analysis without being explicitly programmed. It has transformed how we solve complex problems and is used in various fields, from healthcare and finance to self-driving cars and speech recognition software. (Pratt 2020)

Technological advances have made it possible to create many products based on machine learning, such as recommendation engines on web pages and self-driving cars. Thanks to these advances, it was possible for machine learning to become an essential component of the growing field of data science. Using statistical methods, it is possible to train algorithms, typically created using frameworks that accelerate solution development, to make classifications or predictions. (IBM)

At its core, machine learning involves developing algorithms that can learn from data without being explicitly programmed. These algorithms are designed to improve their performance over time by learning from the patterns and relationships present in the data. This approach enables machines to make predictions or decisions based on what they have learned from the data. (Pratt 2020)

Deep learning and machine learning tend to be used interchangeably. Still, the most significant difference is that deep learning, apart from being a sub-field of machine learning, can use unstructured data and automatically determine the distinct categories since classical machine learning is more dependent on human intervention to learn. Machine learning needs a human expert to determine the set of features to understand the differences between data inputs. (IBM)

Machine learning is divided into four categories: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. The algorithm is trained using labeled data in supervised learning, where the desired output is known. The algorithm learns from the labeled data and can make predictions based on new, unseen data. On the other hand, unsupervised learning involves analyzing unlabeled data and identifying patterns and relationships in the data without prior knowledge of the output. In semi-supervised

learning, the models are trained with labeled and unlabeled data to extract the best of both supervised and unsupervised learning. Reinforcement learning is used to train machines to make decisions based on feedback from their environment. In Image 3, we can see a summary of the four categories. (Pratt 2020)

Supervised learning	Unsupervised learning	Semi-supervised learning	Reinforcement learning
<p>Data scientists provide input, output and feedback to build model (as the definition)</p> <p>EXAMPLE ALGORITHMS:</p> <p>Linear regressions</p> <ul style="list-style-type: none"> sales forecasting risk assessment <p>Support vector machines</p> <ul style="list-style-type: none"> image classification financial performance comparison <p>Decision tree</p> <ul style="list-style-type: none"> predictive analytics pricing 	<p>Use deep learning to arrive at conclusions and patterns through unlabeled training data.</p> <p>EXAMPLE ALGORITHMS:</p> <p>Apriori</p> <ul style="list-style-type: none"> sales functions word associations searcher <p>K-means clustering</p> <ul style="list-style-type: none"> performance monitoring searcher intent 	<p>Builds a model through a mix of labeled and unlabeled data, a set of categories, suggestions and exemplar labels.</p> <p>EXAMPLE ALGORITHMS:</p> <p>Generative adversarial networks</p> <ul style="list-style-type: none"> audio and video manipulation data creation <p>Self-trained Naive Bayes classifier</p> <ul style="list-style-type: none"> natural language processing 	<p>Self-interpreting but based on a system of rewards and punishments learned through trial and error, seeking maximum reward.</p> <p>EXAMPLE ALGORITHMS:</p> <p>Q-learning</p> <ul style="list-style-type: none"> policy creation consumption reduction <p>Model-based value estimation</p> <ul style="list-style-type: none"> linear tasks estimating parameters

Image 3. Types of machine learning models. (Pratt 2020)

The ability of machine learning algorithms to analyze large and complex data sets has made them valuable tools in fields such as healthcare, finance, and marketing. In healthcare, machine learning algorithms analyze patient data and develop personalized treatment plans based on their medical history and genetic makeup. In finance, machine learning predicts market trends and identifies fraudulent activities. Machine learning algorithms identify customer behavior patterns in marketing and develop targeted advertising campaigns. (Pratt 2020)

Advancements in computer hardware and software, along with the increasing availability of large data sets, have accelerated the growth of machine learning. Developing deep learning algorithms, which can handle even larger and more complex data sets, has further boosted the potential applications of machine learning. (Keita 2022)

The growth of machine learning has led to the development of new job roles, such as data scientists and machine learning engineers, and has created a need for specialized training programs in these areas. As the amount of data available to us continues to grow, machine

learning is expected to play an increasingly important role in enabling us to extract insights and make better decisions. (Keita 2022)

4.2 Supervised Learning

Supervised learning involves using a labeled dataset to train a model that can predict the correct output for new, unseen data. The labeled dataset consists of input variables (also known as features or predictors) and output variables (also known as labels or targets). The goal of the model is to learn the underlying relationship between the input variables and the output variables so that it can accurately predict the output for new, unseen input data. (Keita 2022)

The supervised learning model works using input data and adjusting its weights until it has been fitted appropriately. Usually, a process known as cross-validation is used to ensure that the model avoids overfitting or underfitting. Some methods used in supervised learning include neural networks, naïve Bayes, linear regression, logistic regression, random forest, and support vector machine. (IBM)

In supervised learning, the training data provided also works as the supervisor that teaches the machine, applying the same concept as a student learning under the supervision of a teacher. To achieve this, the model knows about each type of data thanks to it being labeled, and once the training is complete, the model is tested using a subset of data from the training set. It is easier to understand if we look at Image 4, which shows us the working of supervised learning. (JavaTpoint 2021)

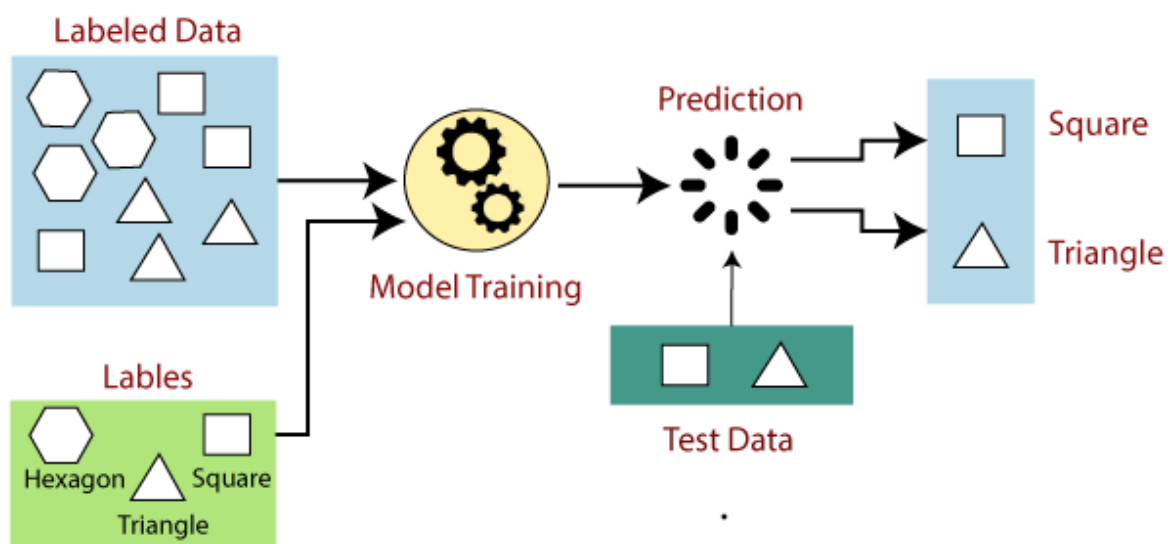


Image 4. Diagram of how supervised learning works. (JavaTpoint 2021)

To work with supervised learning, first, we have to determine the training dataset we will use and collect the training data. When we have enough data, we must split the dataset into training, testing, and validation. At this point, we have to determine the input features of the training dataset, which should be accurate enough so that the model can predict the output correctly. Then, we choose the algorithm we think is the most suitable for the model, we execute it, and, finally, we evaluate the accuracy of the final model. (JavaTpoint 2021)

For a better understanding, and based on the diagram in Image 3, let's suppose we have a dataset of different types of shapes. The first step is to train the model for each figure, which we achieve by telling the model to focus on the sides of the figures. After training, we test if our model can identify the shapes of the test set. The already trained machine will try to classify new shapes it finds based on the number of sides. (JavaTpoint 2021)

One common technique used in supervised learning is the use of decision trees. Decision trees are an algorithm that recursively splits the input data into smaller subsets based on the values of the input features. The goal is to create a tree-like structure where each leaf node represents a prediction. Decision trees are easy to interpret and can handle categorical and numerical data. (IBM)

Another popular technique used in supervised learning is the use of neural networks. Neural networks are a type of algorithm that is inspired by the structure of the human brain. They consist of layers of interconnected neurons and can learn complex relationships between input features and output. Neural networks are widely used in applications such as image recognition, natural language processing, and speech recognition. (IBM)

Support vector machines (SVMs) are another popular technique used in supervised learning. SVMs work by finding a hyperplane that separates the input data into different classes. SVMs can handle non-linear relationships between input features and output and are widely used in text classification and image recognition applications. (Keita 2022)

The main advantages of supervised learning are that the model we produce can predict the output based on prior experiences, and we can have an exact idea about the type of objects we are working with. On the other hand, these models are only sometimes suitable for complex tasks. Also, they cannot predict the correct output if the test data differs from the training dataset. In addition to that, it also takes lots of computation time. (JavaTpoint 2021)

Supervised learning can be broadly categorized into two types: classification and regression. In classification, the output variable is categorical, which means that the model needs to predict which class the input data belongs to. For example, a classification model may indicate whether an email is spam. In regression, the output variable is continuous, which

means that the model needs to predict a numerical value. For example, a regression model may anticipate the price of a house based on its features. Image 5 shows a clear diagram to understand when to use classification or regression. (Keita 2022)

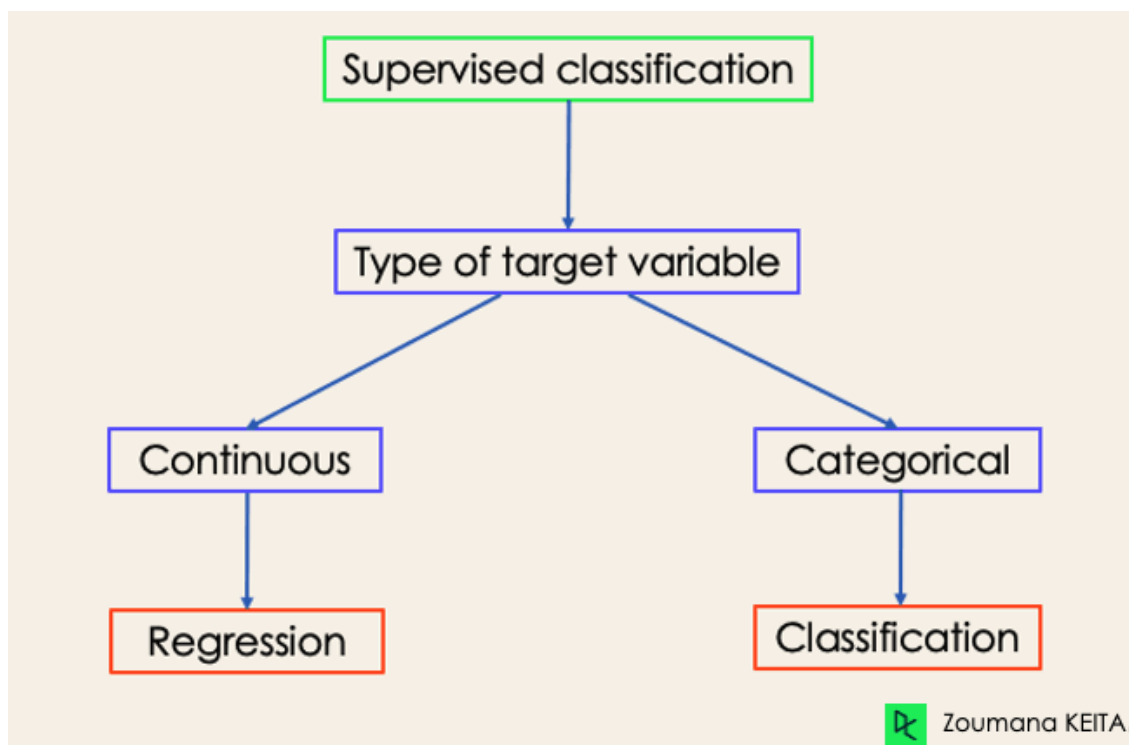


Image 5. Difference between regression and classification. (Keita 2022)

4.2.1 Classification

Classification is a supervised machine learning method where the model tries to predict the correct label of a given input data. In classification, the model is fully trained using the training data, and then it is evaluated on test data before being used to perform prediction on new unseen data. (Keita 2022)

In other words, classification is a commonly used technique in supervised learning, particularly in applications such as image recognition, text classification, and fraud detection. In classification, the algorithm is trained on a labeled dataset, where each data point is associated with a set of input features and a corresponding output label. The algorithm learns to map the input features to the correct output label to predict the designation of new, unseen data points. (JavaTpoint 2021)

In classification, we can differentiate between two types of learners: lazy and eager learners. Eager learners are the ones that build a model from the training dataset before making any prediction. They spend more time during the training to better generalize from learning the correct weights, but they require less time to make predictions. Lazy learners, on the

other hand, wait to create a model immediately from the training data. They work memorizing the training data, and when a prediction is needed, for the nearest neighbor from the whole training data, which makes them very slow during prediction. (Keita 2022)

Classification models can be used in several applications in multiple domains of our daily life, such as healthcare, education, transportation, and sustainable agriculture. Some of the examples of use include training a machine on historical patient data that can help healthcare specialists to accurately analyze their diagnoses, training a model to analyze data with the help of Natural Language technologies to classify documents per category, to predict geographical locations with rise traffic volume, or to help improve farmers' productivity without damaging the environment. (Keita 2022)

We can define four main tasks for classification in machine learning: binary, multi-class, multi-label, and imbalanced sorts. The goal of a binary classification task is to classify the data into two mutually exclusive categories so that it can be defined as a true or false task. On the other hand, the multi-class classification has two or more mutually exclusive class labels, where the goal is to predict to which class a given example belongs to. Most of the binary classification algorithms can also be used for multi-class classification. Conversely, the model will try to predict 0 or more classes for each example in multi-label classification tasks. In this case, there is no mutual exclusion between the labels. This scenario can be observed in domains such as auto-tagging in Natural Language Processing, where a text can contain multiple topics. For the imbalanced classification, the number of examples is unevenly distributed between the classes, meaning more of one category than the others in the training data. Since using conventional predictive models when dealing with an imbalanced dataset could not be so effective, the most commonly used approach is to include sampling techniques. (Keita 2022)

The most common metrics for evaluating classification performance include accuracy, precision, recall, and F1 score. However, in addition to these standard metrics, there are also more advanced techniques for evaluating the performance of classification algorithms. For example, receiver operating characteristic (ROC) analysis is a technique for evaluating the performance of binary classification algorithms based on the trade-off between true positive rate and false positive rate. Another popular technique is confusion matrix analysis, which is a method for visualizing the performance of a classification algorithm in terms of its true positives, false positives, true negatives, and false negatives. (JavaTpoint 2021)

4.2.2 Regression

Regression in machine learning consists of mathematical methods that allow data scientists to predict a continuous outcome (y) based on the value of one or more predictor variables (x). Linear regression is probably the most popular form of regression analysis because of its ease-of-use in predicting and forecasting. (Kurama 2023)

Multiple techniques can be found to work with regression. Linear regression uses a best-fit straight line to see the linear relationship between the dependent variable and one or more independent variables. Generally, linear models make predictions by simply computing a weighted sum of the input features plus a constant called bias. Bias is a deviation introduced to the equation for the predictions made. For this technique to work, the dependent variable has to be continuous, and the nature of the regression line has to be linear. We can also find simple linear regression if the problem only has one independent variable, multiple linear regression if the problem has more than one, and multivariate linear regression if the problem has various output variables. (Kurama 2023)

While a linear regression model can understand patterns by fitting in a simple linear equation, it might need to be more accurate when dealing with complex data. That's where polynomial regression comes in. In this technique, the degree of the equation is greater than one, creating curved lines to fit the data. Sometimes it is also necessary to use regularization to avoid overfitting. (Kurama 2023)

Logistic regression is a type of regression model used when the target variable is categorical, such as binary (yes or no) or multiclass (more than two categories). In logistic regression, the algorithm predicts the probability of an event occurring based on the input variables. The output of logistic regression is a probability value between 0 and 1, and a threshold is set to classify the output as belonging to a particular category. (Kurama 2023)

Several metrics can be used to evaluate the performance of regression models, including mean squared error (MSE), root mean squared error (RMSE), and R-squared. MSE measures the average squared difference between the predicted and actual values, while RMSE is the square root of the MSE. R-squared is a statistical measure representing the proportion of variance in the target variable explained by the input variables. (Kurama 2023)

4.3 Unsupervised Learning

Unsupervised learning refers to the use of artificial intelligence (AI) algorithms to identify patterns in data sets containing data points that are neither classified nor labeled. (Pratt 2020)

Unsupervised learning allows algorithms to automatically discover the underlying structure in datasets without needing labeled examples or human supervision. It involves identifying patterns and similarities in data points not tagged or categorized beforehand, allowing for discovering hidden relationships and structures within the data. (Pratt 2020)

Unsupervised learning can be divided into two broad categories: clustering and dimensionality reduction. Clustering algorithms group data points into clusters based on their similarities. In contrast, dimensionality reduction algorithms reduce the number of features in a dataset while preserving the most relevant information. (Pratt 2020)

Clustering is a widespread technique in unsupervised learning and involves grouping similar data points into clusters based on their distance or similarity in a high-dimensional space. There are different clustering algorithms, including hierarchical and k-means clustering. Hierarchical clustering creates a tree-like structure of nested clusters. In contrast, k-means clustering involves partitioning the data into k clusters by minimizing the sum of squared distances within each collection. (Pratt 2020)

On the other hand, dimensionality reduction involves reducing the number of features in a dataset while retaining the most critical information. This is done by identifying the most essential elements in the dataset and discarding the less important ones. Principal component analysis (PCA) is a popular technique for dimensionality reduction. It involves transforming the data into a new set of variables called principal components, which represent the essential features of the data. (Pratt 2020)

Unsupervised learning can be more unpredictable than a supervised learning model. It might add unforeseen and undesired categories to deal with unusual data, creating clutter instead of order. Since no labels or categories are contained within the data sets being used to train the systems, each piece of data is an unlabeled input object or sample. (Pratt 2020)

One of the main benefits of unsupervised learning is that it can be used when no prior knowledge or labeled data is available. This makes it ideal for exploring and discovering new patterns in data and for situations where data is too large or complex to be marked manually. It can also work with real-time data. (Pratt 2020)

However, one of the main challenges of unsupervised learning is that evaluating the algorithm's performance can be tricky since there is no labeled output to compare against. There are several methods for evaluating the performance of clustering algorithms, including the silhouette score, which measures the similarity of data points within a cluster and the dissimilarity between groups. (Pratt 2020)

Other disadvantages include the need for complete insight into how or why a system reaches its results. Also, unsupervised learning can be dangerous for customer segmentation since cluster analysis could overestimate the similarities in the input objects and thereby obscure individual data points that may be important. (Pratt 2020)

4.4 Semi-supervised Learning

Semi-supervised learning offers a medium solution between supervised and unsupervised learning, using more minor labeled data to guide the classification of larger, unlabeled data sets. It is often used to solve the problem of needing more labeled data or when it is too costly to label enough data. (IBM)

Unlike unsupervised learning, semi-supervised learning can be used for various problems, ranging from classification and regression to clustering and association. On the other hand, while supervised learning relies on a labeled dataset to train the model, semi-supervised learning uses labeled and unlabeled data to improve model accuracy. By leveraging the information from labeled and unlabeled data, semi-supervised learning can help improve the efficiency and accuracy of machine learning models. These two features make semi-supervised learning find many applications while maintaining the accuracy of the results. (Altexsoft 2022)

There are several approaches to semi-supervised learning. One of the most popular approaches is to use unsupervised learning techniques to preprocess the unlabeled data. For example, clustering can help identify data points that are similar to each other, while dimensionality reduction can help remove redundant features from the data. These techniques can extract useful information from the unlabeled data, which can then be used to improve model accuracy. (IBM)

Another approach to semi-supervised learning is to use generative models, such as auto-encoders or variational autoencoders, to learn a probability distribution over the data. These models can be trained on the unlabeled data to generate synthetic data similar to the actual data. The synthetic data can then be used as additional labeled data to train the model, allowing the model to learn from a more extensive and diverse dataset. This can be achieved using self-training, co-training, or graph-based label propagation. (IBM)

Self-training is the procedure in which you take any supervised method for classification or regression and modify it to work semi-supervised. In Image 6, we can see how the workflow is. First, we pick a small amount of labeled data with their respective tags, and we use this dataset to train a base model with the help of ordinary supervised methods. Then we apply the process known as pseudo-labeling, which consists of taking the partially trained model

and using it to make predictions for the rest of the database still unlabeled. The labels generated are called pseudo, as they are produced based on the initially labeled data with limitations. From this point, we take the most confident predictions made with the model. If any pseudo-labels exceed this confidence level, we add them to the labeled dataset and create a new combined input to train an improved model. This process can go through several iterations. (Altexsoft 2022)

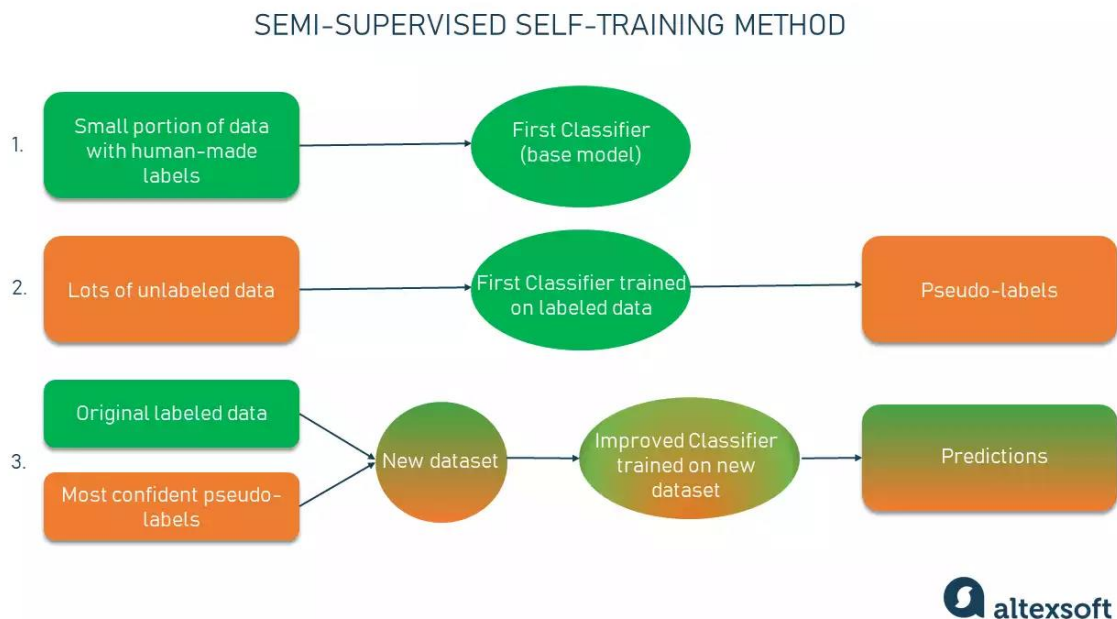


Image 6. Semi-supervised self-training method. (Altexsoft 2022)

Co-training trains two individual classifiers based on two data views, and it is used when only a small portion of labeled data is available. Each view is a different set of features that provide additional information about each instance, meaning they are independent given the class. Also, each view is sufficient to predict the sample data class accurately. We can see how this method works in Image 7. First, we train a separate model for each view with the help of a small amount of labeled data. Then, the bigger pool of unlabeled data is added to receive pseudo-labels. Classifiers co-train one another using pseudo-labels with the highest confidence level. Suppose the first classifier confidently predicts a data sample's genuine label while the other makes a prediction error. In that case, the data with the confidence pseudo-labels assigned by the first classifier updates the second classifier and vice-versa. The final step combines the predictions from the two updated classifiers to get one classification result. This method can also go through many iterations to construct an additional labeled dataset from the vast amount of unlabeled data. (Altexsoft 2022)

SEMI-SUPERVISED CO-TRAINING METHOD

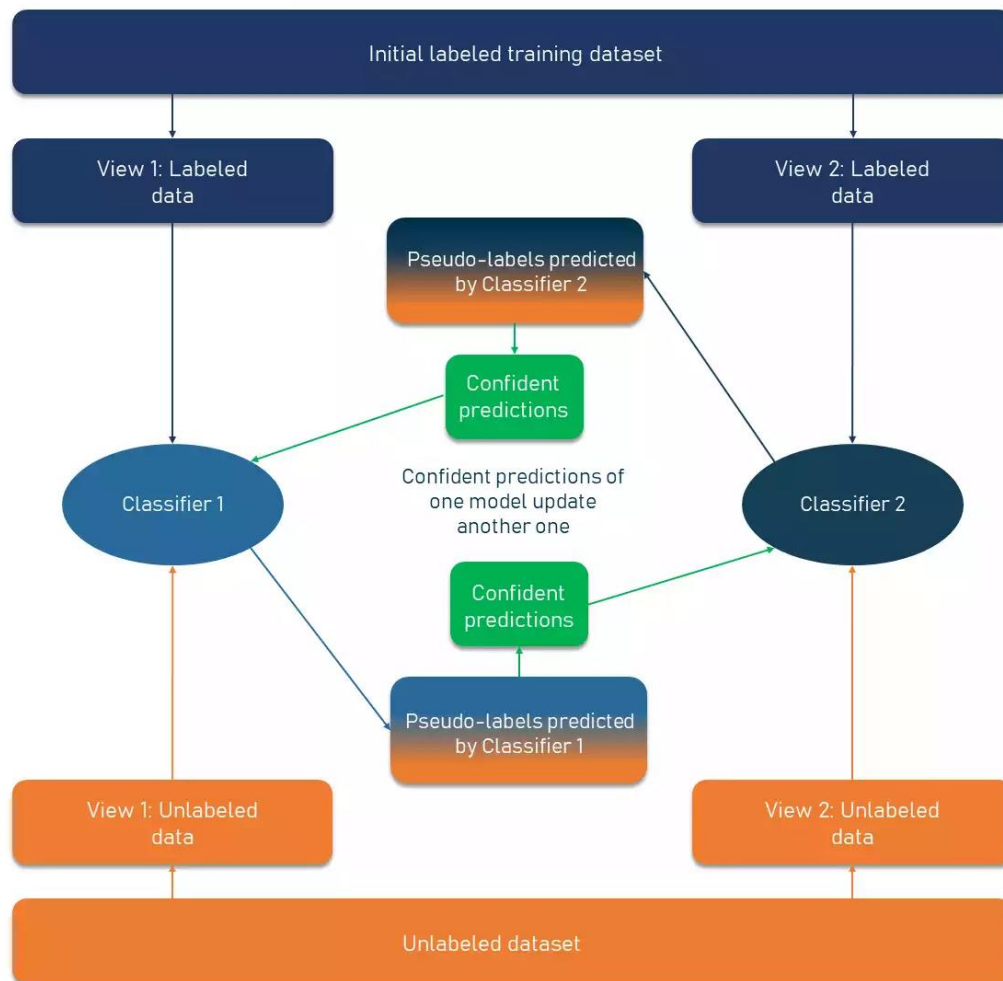


Image 7. Diagram of how the co-training method works. (Altexsoft 2022)

Another popular way to run semi-supervised learning is to represent labeled and unlabeled data in graphs and then apply a label propagation algorithm. In this method, most of the data points of a network are unlabeled. The task is to spread the labels throughout the network. One way of doing this is picking a point and counting up all the different paths that travel through the network from this point to each labeled node. If we do that, we will find that different walks lead to one label or another. From that, we assume that this point belongs to the category with more walks leading from the point to this category. And then, we repeat the process for every point of the graph. This method can be seen in personalization and recommender systems. It can predict customer interests based on information about other customers. (Altexsoft 2022)

Semi-supervised learning has been applied to many applications, including image classification, speech recognition, natural language processing, and anomaly detection. For example, semi-supervised learning has been used in image classification tasks, where only a tiny fraction of the dataset is labeled. Semi-supervised learning can help to improve the model's accuracy by leveraging the unlabeled data to learn more about the structure of the data. Similarly, in natural language processing, semi-supervised learning can help improve models' performance by using large amounts of unlabeled text data to learn better language representations. (IBM)

Despite its advantages, semi-supervised learning also has some limitations. One of the main challenges is selecting the suitable method and parameters for the problem. Semi-supervised learning also requires significant computational resources and expertise in choosing the appropriate form and parameters for the problem. Additionally, the quality of the unlabeled data can affect the model's performance, so it is essential to select it to ensure that it represents the problem carefully. (IBM)

4.5 Reinforcement Learning

Reinforcement Learning (RL) is the science of decision making. It is about learning the optimal behavior in an environment to obtain maximum reward. This optimal behavior is learned through interactions with the environment and observations of how it responds, similar to children exploring the world around them and learning the actions that help them achieve a goal. (Verma & Diamantidis 2021)

Reinforcement learning is a model where the algorithm is trained using trial and error instead of sample data. A sequence of successful outcomes will be reinforced to develop the best policy for a given problem. It has gained significant attention recently due to its potential to enable machines to make decisions in complex, dynamic environments. It can also be considered a type of unsupervised learning involving an agent interacting with an environment, taking actions, and receiving feedback as rewards or penalties. The agent learns to associate specific actions with higher rewards and others with lower rewards and adjusts its behavior accordingly. Over time, the agent becomes better at making decisions that lead to higher rewards. (IBM)

The quality of the actions is measured by not just the immediate reward they return but also the delayed reward they might fetch. Being capable of learning the activities that result in eventual success in an unseen environment without the help of a supervisor makes reinforcement learning a robust algorithm, as seen in Image 8. (Verma & Diamantidis 2021)



Image 8. Diagram of how Reinforcement Learning works. (Synopsis 2021)

Reinforcement learning has its roots in psychology and neuroscience, where researchers have long been interested in understanding how animals and humans learn from feedback. In the 1950s, researchers began developing mathematical models of learning based on reinforcement, and in the 1980s, researchers began applying these models to artificial intelligence. (Verma & Diamantidis 2021)

At the heart of reinforcement learning is the concept of the agent, the environment, the actions, the rewards, and the policy. The agent is the entity that makes decisions and takes actions in the environment. The environment is the setting in which the agent operates. The actions are the choices that the agent can make in the environment. The rewards are the feedback signals that the agent receives based on its actions. The policy is the set of rules that the agent uses to decide what actions to take in different situations. (Verma & Diamantidis 2021)

A reinforcement learning system is based on the hypothesis that all goals can be described by maximizing expected cumulative rewards. The agent must learn to sense and change the state of the environment using its actions to derive a maximal reward. A useful abstraction of the reward signal is the value function, which faithfully captures the 'goodness' of a state. While the reward signal represents the immediate benefit of being in a particular state, the value function captures the cumulative reward expected to be collected. The objective of an RL algorithm is to discover the action policy that maximizes the average value that it can extract from every state of the system. (Verma & Diamantidis 2021)

RL algorithms can be broadly categorized as model-free and model-based. Model-free algorithms do not build an explicit model of the environment. They are closer to trial-and-error algorithms that run experiments with the environment using actions and derive the optimal policy from it directly. These algorithms can either be value-based or policy-based. Value-based algorithms consider the optimal policy to be a direct result of estimating the value function of every state accurately. Using a recursive relation, the agent interacts with the environment to sample trajectories of states and rewards. Once the value function is known, discovering the optimal policy is achieved by acting greedily concerning the value function at every state of the process. Policy-based algorithms, on the other hand, directly estimate the optimal policy without modeling the value function. By parametrizing the policy directly using weights, they render the learning problem into an explicit optimization problem. In policy-based algorithms, the agent also samples trajectories of states and rewards; however, this information is used to explicitly improve the policy by maximizing the average value function across all states. Policy-based approaches suffer from a high variance that manifests as instabilities during training. Value-based strategies, though more stable, are not suitable for modeling continuous action spaces. (Verma & Diamantidis 2021)

On the other hand, model-based RL algorithms build a model of the environment by sampling the states, taking actions, and observing the rewards. The model predicts the expected reward and future states for every state and possible action. While the model-free algorithm deals with regression problems, the model-based one deals with density estimation problems. Given a model of the environment, the RL agent can plan its actions without directly interacting with the environment. This is like a thought experiment a human might run when solving a problem. By mimicking this process, reinforcement learning allows machines to learn more naturally and intuitively. (Verma & Diamantidis 2021)

Several algorithms and techniques are used in reinforcement learning, including Q-learning, policy gradients, and actor-critic methods. These techniques allow the agent to learn from experience and improve decision-making. For example, Q-learning is a model-free algorithm that enables the agent to discover the optimal policy by iteratively updating the values of each state-action pair based on the rewards received. Policy gradients, on the other hand, use a gradient-based optimization approach to optimize the policy directly. (Verma & Diamantidis 2021)

The main benefits of reinforcement learning include focusing on the problem as a whole, not needing a separate data collection step, and working in dynamic and uncertain environments. Reinforcement learning does not need to divide the problem into subproblems since it directly works to maximize the long-term reward. Also, the training data is obtained via the

direct interaction of the agent with the environment, as the data is the agent's experience, not a separate collection of samples that has to be fed to the algorithm. These algorithms are also inherently adaptive and built to respond to environmental changes. (Verma & Diamantidis 2021)

On the other hand, reinforcement learning has to deal with some challenges, such as needing extensive experience, dealing with delayed rewards, and needing more interpretability on the models. These methods autonomously generate training data by interacting with the environment, so the environment's dynamics limit the data collection rate. Complex environments with high-dimensional state spaces can greatly slow the learning curve by needing extensive exploration. Also, the agent can often trade off short-term rewards for long-term gains. This makes it difficult for the agent to discover the optimal policy. This is especially true when the outcome is only known once a large number of sequential actions are taken. Regarding interpretability, the reasons for the agent's actions might not be evident to an external observer since it has learned the optimal policy and it is deployed in the environment. (Verma & Diamantidis 2021)

In recent years, there has been significant progress in reinforcement learning, with researchers developing new algorithms and techniques that have improved the ability of machines to learn from feedback. For example, deep reinforcement learning has emerged as a powerful technique that combines reinforcement learning with deep neural networks, allowing devices to learn from raw sensory input such as images or sound. (IBM)

Reinforcement learning has many applications in robotics, game AI, and autonomous vehicles. In robotics, reinforcement learning can teach robots how to perform tasks such as grasping objects or navigating through complex environments. In game AI, reinforcement learning can teach agents to play games such as chess, Go, or video games. In autonomous vehicles, reinforcement learning can teach cars to make decisions in complex driving scenarios. (IBM)

It also has the potential to transform many industries in the coming years. For example, in healthcare, reinforcement learning can be used to develop personalized treatment plans for patients, optimizing the treatment regimen based on the patient's individual needs and responses to treatment. In finance, reinforcement learning can be used to develop trading strategies that adapt to changing market conditions and maximize returns. (IBM)

However, ethical concerns are also associated with using reinforcement learning. For example, suppose reinforcement learning is used to make decisions that affect human lives, such as autonomous vehicles or medical treatment plans. In that case, there is a risk that the decisions may be biased or unfair, and they can be not that intuitive for the observers.

It is essential to carefully consider the potential consequences of using reinforcement learning in these contexts and to ensure that these systems are transparent and accountable.
(IBM)

5 Game engines

5.1 Unity

Unity is an incredibly versatile game engine that has transformed the world of video game development. Since its initial release in 2005, Unity has become one of the leading game engines in the industry, providing a robust and accessible platform for game development that anyone can use. It is considered easy for beginner developers and famous for indie game development.

One of the key features of Unity is its flexibility, which allows developers to create games for a wide range of platforms. The engine supports various programming languages, including C#, JavaScript, and Boo, and offers a range of tools and features, including physics, animation, and lighting tools. Additionally, Unity has a vast library of pre-made assets that developers can use to speed up development. These assets can be accessed through Unity's asset store, a marketplace that allows developers to buy and sell assets.

Another key strength of Unity is its strong community of developers. The Unity community is active and vibrant, with developers sharing their knowledge and resources through forums, tutorials, and other resources. Unity's comprehensive documentation also ensures that developers can access the information they need to create engaging games and interactive content.

Beyond its use in the gaming industry, Unity has also found applications in other fields. For example, it is increasingly being used in architecture and engineering to create virtual simulations of buildings and infrastructure. Unity is also used in education to create interactive educational content, making it an essential tool for teachers and educators.

Unity's popularity and versatility have also led to the creation of a vast ecosystem of third-party tools and plugins, further expanding the engine's capabilities. These tools and plugins cover a range of functionalities, from artificial intelligence to virtual reality, and are readily available through the asset store.

5.1.1 Video Games

Unity has significantly impacted the gaming industry. Its versatility allows developers to create games for various platforms, such as desktops, mobile devices, consoles, VR, and AR devices. The engine's user-friendly interface is suitable for developers of different experience levels. Additionally, Unity has a vast library of pre-made assets accessible through the Unity Asset Store, including 3D models, sound effects, and plugins. In Image 9, we can see

an example of one of the possible assets in the Asset Store. This feature saves developers' time, enabling them to create high-quality games by incorporating professional-grade assets. (Unity 2023)

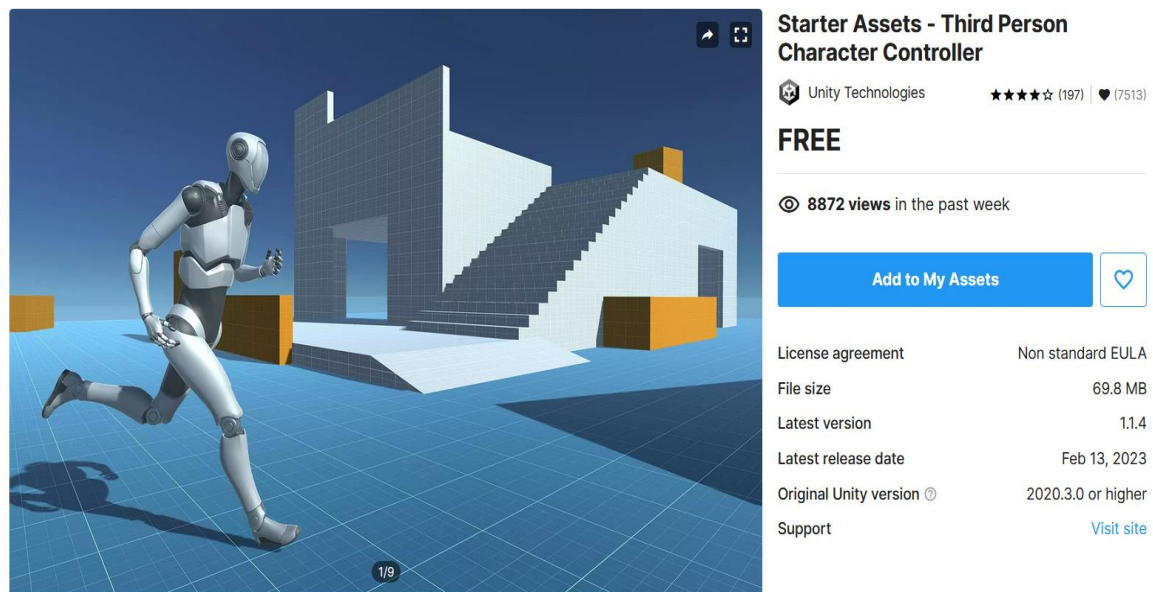


Image 9. Screenshot of an asset for a third-person character controller in the Unity Asset Store.

The ecosystem of tools and plugins makes it easier for developers to create high-quality games and interactive content with Unity, providing a range of resources that can be leveraged to create engaging and immersive experiences for users. (Unity 2023)

One of the most significant advantages of Unity is its scripting language, C#. C# is a modern, object-oriented programming language that is easy to learn and use. Its syntax is intuitive, making writing clean and efficient code easier. Additionally, C# allows developers to access platform-specific APIs, which is essential for creating games on different platforms. Unity's implementation of C# is optimized for game development, making it faster and more efficient than other programming languages. (Unity 2023)

Another essential aspect of Unity is its physics engine. The engine's physics system provides accurate and realistic simulations of physical phenomena such as collisions, gravity, and object movement. This feature is essential for creating realistic and engaging games, particularly those involving physics-based puzzles or simulations. Unity's physics engine is powerful and flexible, allowing developers to adjust and fine-tune parameters to achieve the desired effect. (Unity 2023)

5.1.2 Non-gaming Industries

Aside from its impact on the gaming industry, Unity has also made significant strides in non-gaming sectors such as architecture, engineering and construction (AEC), and healthcare. The engine's real-time 3D capabilities have proven helpful in these industries, allowing developers to create immersive experiences that facilitate better decision-making and improved outcomes. (Unity Technologies 2023)

In the AEC industry, Unity is used to create virtual simulations of buildings, infrastructure, and urban environments. These simulations allow architects, engineers, and construction professionals to visualize designs, test different scenarios, and identify potential issues before construction begins. This feature saves time and resources and improves outcomes by reducing errors and minimizing waste. Unity's real-time 3D rendering capabilities are also beneficial in marketing and sales, where architects and developers can use the engine to create interactive presentations and virtual tours of properties. (Unity Technologies 2023)

In healthcare, Unity is used to create simulations and training programs for medical professionals. For instance, Unity-based simulations can train surgeons in minimally invasive procedures or simulate medical emergencies for doctors and nurses. Unity's real-time 3D capabilities also help create educational content and patient engagement tools, such as virtual reality therapy for patients with phobias or mental health disorders. (Unity Technologies 2023)

Furthermore, Unity's AR and VR capabilities are also relevant in non-gaming industries. In retail, Unity creates augmented reality experiences that enhance customer engagement and improve sales. For instance, Unity-powered AR applications can allow customers to preview products before purchasing, such as trying on clothes or testing furniture in their homes. In education, Unity is used to create virtual and augmented reality experiences that enhance learning and provide a more immersive educational experience. (Unity Technologies 2023)

5.2 Unreal Engine

Unreal Engine is a game engine developed by Epic Games that is designed to provide a complete suite of development tools for creating high-quality video games, simulations, and visualizations. It was first introduced in 1998 as a first-person shooter game, and it has since evolved into a multi-platform engine that supports a wide range of applications. It is written in C++ and includes high portability, and it has been used in various games and

other industries, especially film and television. The latest generation was launched in April 2022. (Unreal Engine 2023)

It is similar to Unity, but one of the critical strengths of Unreal Engine is its advanced graphics capabilities. It includes a highly optimized rendering engine with photorealistic lighting, dynamic global illumination, and advanced post-processing effects. This allows developers to create immersive and engaging experiences that rival those in major blockbuster movies. (Unreal Engine 2023)

The engine is built on a modular architecture, which makes it easy for developers to customize and extend its functionality. The engine's architecture comprises several core modules, including the rendering system, physics simulation, animation, audio, input, networking, and more. This modular design allows developers to create custom tools and workflows that fit their needs. (Unreal Engine 2023)

Another significant advantage of Unreal Engine is its physics engine. It includes a powerful physics simulation system that allows developers to create highly realistic and interactive environments, as seen in Image 10. The engine can simulate rigid body dynamics, soft body dynamics, and cloth simulation. This enables developers to create games with highly dynamic and interactive worlds. (Unreal Engine 2023)

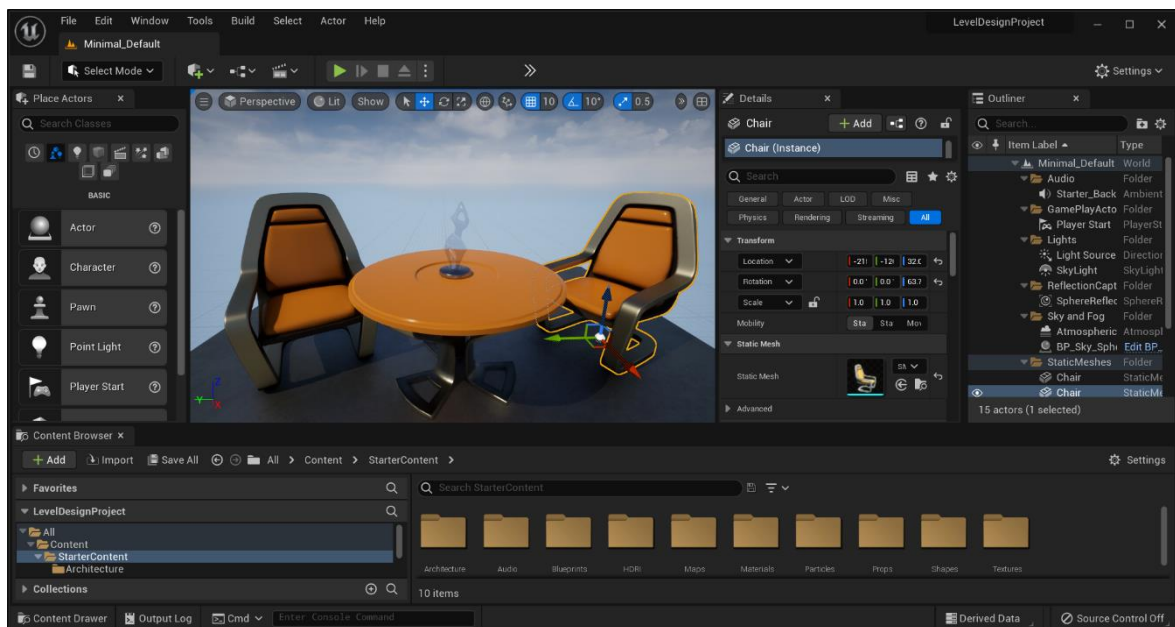


Image 10. Example of the interface for the Unreal Engine level editor. (Epic Games 2023)

Unreal Engine is also highly customizable, with various programming languages supported, including C++, Blueprint visual scripting, and Python. This allows developers to create custom game logic and workflows and extend the engine with their tools and features. (Unreal Engine 2023)

Another critical strength of Unreal Engine is its thriving community of developers. Epic Games has worked hard to build a supportive, inclusive community that shares knowledge and expertise. This community includes thousands of developers worldwide who share their experiences, tips, and resources through online forums, social media, and other channels. (Unreal Engine 2023)

In addition to its primary application in game development, Unreal Engine has also been used for various other applications. For example, it has been used to create architectural visualizations, product demos, and industrial simulations. It has even been used in film and television production, with high-profile movies such as *The Mandalorian* and *The Lion King* using Unreal Engine to create stunning special effects and environments. (Unreal Engine 2023)

One of the most significant advantages of Unreal Engine is its scalability. It can be used by small indie developers working on their first game up to large AAA game studios working on high-budget blockbuster games. This scalability is partly due to the engine's royalty-based revenue model, which allows small developers to use the engine for free and pay a percentage of their revenue when their game is released. (Unreal Engine 2023)

6 Artificial Intelligence in Game Engines

6.1 Introduction

Artificial intelligence has helped game developers to create modern video games as we know them. Without artificial intelligence, most video games would not be functional since the game characters would not react to their environment or the player's actions in the game world. (Wirtz 2023.)

It has become an essential tool for game developers to create more complex and realistic gameplay experiences. AI techniques are used to improve the gameplay mechanics, create intelligent and responsive NPCs, and generate dynamic and varied game worlds. In addition, AI enhances the game's visual and audio effects and analyzes player behavior to improve game design. (Wirtz 2023.)

Creating responsive and intelligent virtual players and non-playable game characters is hard. Especially when the game is complex. To create intelligent behaviors, developers have had to resort to writing tons of code or using highly specialized tools. (Unity Technologies 2023.)

One of the most significant benefits of using AI in game development is procedural generation. This technique uses algorithms to create game content, such as terrain, buildings, and NPCs. Procedural generation can help developers save time and resources by automating the creation of repetitive or time-consuming tasks and creating more dynamic and unique game experiences. AI can generate random worlds, creating infinite possibilities for exploration and discovery. (Unity Technologies 2023.)

Reactive AI is another essential technique used in game engines. It involves using simple decision-making algorithms to enable NPCs to react to environmental changes. Examples of reactive AI include obstacle avoidance and attack behaviors. Deliberative AI, conversely, involves more complex algorithms that allow NPCs to make informed decisions based on their current state and objectives. Deliberative AI techniques include pathfinding, behavior trees, and decision-making algorithms. (Wirtz 2023.)

Pathfinding is critical to game development, especially for games with large open worlds. Pathfinding algorithms calculate the best path for NPCs to take through a game environment, considering obstacles and other factors. The most commonly used pathfinding algorithms are A* and Dijkstra's, although many algorithms can achieve similar results. (Wirtz 2023.)

Behavior trees are another critical AI technique used in game engines. They consist of nodes representing different actions or decisions, such as moving to a particular location or attacking the player. Behavior trees allow developers to create more intelligent and believable NPCs by defining their decision-making processes in a structured and modular way. This technique enables NPCs to respond dynamically to different situations, making the gameplay experience more engaging. (Wirtz 2023.)

In Image 11, we can see an example of a behavior tree. Following this example, it is possible to imagine an AI that tries to walk through a door. Since the nodes of a behavior tree are resolved from left to right and from bottom to top, in this example, we could have an AI walking to a door and trying to open it. If it succeeds, the selector node (a particular node for behavior trees) will be achieved, and the AI will continue with the “Walk through Door” and “Close Door” nodes. However, if the nodes for the selector start failing for some reason (the door is locked, for example), it will start trying to activate the following nodes (in this case, unlock and open the door). This will continue until some node from the selector one works or until it reaches out of options, leading to the sequence failing and the AI abandoning this action. (Simpson 2014)

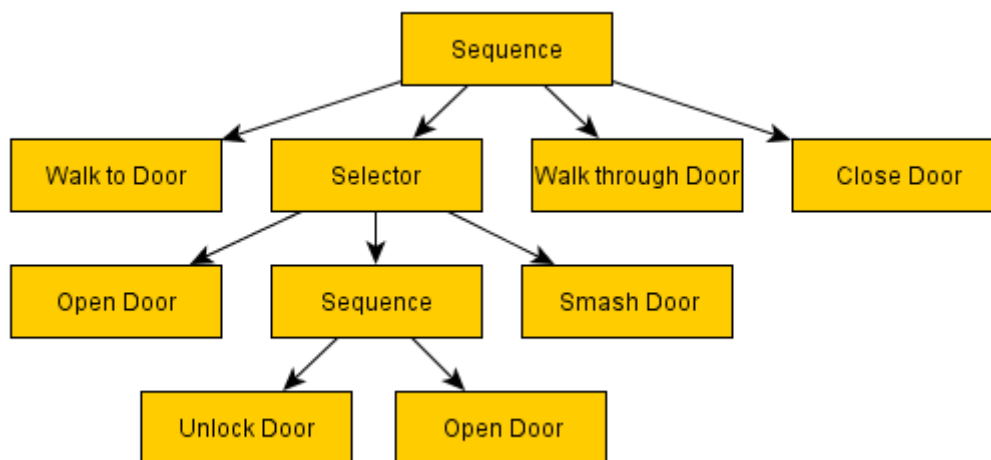


Image 11. Example of a behavior tree. (Simpson 2014)

Machine learning is an increasingly popular technique in game development. Machine learning algorithms can train NPCs to perform specific tasks, such as recognizing patterns in player behavior or learning to play a particular game. Machine learning can create more adaptive and responsive NPCs that can understand and improve over time. This technique can create challenging enemies that adapt to the player's strategies, providing a more rewarding gameplay experience. (Wirtz 2023.)

Another important use of AI in game development is to enhance the visual and audio effects of the game. AI algorithms can generate realistic lighting effects, create practical facial expressions and body movements for NPCs, and even synthesize sound effects. These techniques help to create a more immersive game environment that feels more like a living world. (Wirtz 2023.)

AI can be used to analyze player behavior to improve game design. By analyzing data on player actions, developers can gain insights into how players interact with the game, which game mechanics are most engaging, and which aspects of the game need improvement. This information can be used to make data-driven decisions about game design, leading to a better gameplay experience for players. (Unity Technologies 2023.)

However, the main problem with the advancement of artificial intelligence research is creating new environments for training models that can solve complex problems. This is the most significant and first problem to solve since creating these new environments is time-consuming and requires specialized domain knowledge. (Unity Technologies 2023.)

6.2 AirSim

AirSim is an open-source, cross-platform simulation platform developed by Microsoft that has gained significant attention and popularity among researchers and developers working in autonomous vehicles and drones. The platform provides a realistic and high-fidelity simulation environment for testing and experimenting with various artificial intelligence (AI) algorithms, enabling users to simulate real-world scenarios without the risks and costs associated with physical testing. (Microsoft Research 2021)

AirSim was initially developed to support the research and development of autonomous aerial vehicles (UAVs). However, the platform has evolved to support other applications, including robotics, computer vision, and machine learning. The platform's flexibility and versatility make it an ideal tool for researchers and developers on various AI applications. (Microsoft Research 2021)

One of the critical features of AirSim is its realistic physics engine, which accurately simulates the behavior of UAVs and other vehicles in different environments and weather conditions. The platform uses Unreal Engine 4 and builds on top of it to provide a simulation environment as close to reality as possible. The platform also supports a variety of sensors, including cameras, lidars, and GPS, which can be used to generate realistic sensor data for testing and validation purposes. (Microsoft Research 2021)

AirSim includes a set of pre-built scenarios and challenges that can be used to evaluate the performance of different algorithms in various systems. These scenarios include tasks such as object detection, tracking, and obstacle avoidance, which are essential for autonomous vehicle development. Users can also create custom scenarios and challenges using the platform's built-in editor, which enables them to design and customize the environment to suit their needs. (Microsoft Research 2021)

One of the benefits of using AirSim is that it enables developers to experiment with various AI algorithms in a controlled and safe environment. This is particularly important for autonomous vehicles, where safety is critical. Using AirSim, developers can test and refine their algorithms before deploying them in the real world, reducing the risk of accidents and improving the reliability and performance of autonomous vehicles. (Microsoft Research 2021)

6.3 CARLA

CARLA (Car Learning to Act) is an open-source simulation platform developed by the Computer Vision Center at the Universitat Autònoma de Barcelona. It provides a realistic and high-fidelity environment for testing and developing various AI algorithms related to autonomous driving. CARLA has gained popularity among researchers and developers due to its flexibility, modularity, and extensibility. (CARLA)

Despite CARLA being grounded on Unreal Engine to run the simulations, one of the main goals of this technology is to serve as a tool that users can easily access and customize. To achieve this, the simulator meets the requirements of many use cases within the general driving problem, such as learning driving policies and training perception algorithms. (CARLA)

CARLA supports a wide range of sensors, including cameras, lidars, and GPS, which can be used to generate realistic sensor data for testing and validation purposes. The platform also provides a range of pre-built scenarios and challenges that can be used to evaluate the performance of different algorithms in various systems. These scenarios include object detection, lane following, and obstacle avoidance, which are essential for autonomous driving development. (CARLA)

CARLA is designed to be modular and extensible, allowing users to easily customize and extend the simulation to meet their specific needs. The platform provides a user-friendly interface that enables users to configure and customize the simulation environment easily. Users can also create custom scenarios and challenges using the platform's built-in editor, which allows them to design and customize the environment to suit their needs. (CARLA)

One of the benefits of using CARLA is that it enables developers to experiment with various AI algorithms in a controlled and safe environment. This is particularly important for autonomous driving, where safety is critical. By using CARLA, developers can test and refine their algorithms before deploying them in the real world, reducing the risk of accidents and improving the reliability and performance of autonomous vehicles. (CARLA)

6.4 ML-Agents

ML-Agents is an open-source toolkit that provides a platform for researchers and developers to create and train intelligent agents using machine learning techniques. Unity Technologies developed the toolkit specifically designed to integrate with the Unity game engine. ML-Agents enables developers to build agents to interact with their environment and learn from their experiences, making it a powerful tool for game development and other applications. (Unity Technologies 2022)

It enables deep reinforcement and imitation learning to train intelligent agents in environments created through games and simulations. With this toolkit, it is possible to train intelligent agents for 2D, 3D, and VR/AR games and simulations. These agents can control NPC behavior, automate testing, evaluate different game design decisions, etc. (Unity Technologies 2022) This toolkit is friendly to start working with AI and ML in games since you can find robust documentation and example projects. Also, it is possible to quickly create intelligent models without much coding and with several starter environments. (Unity Technologies 2023)

The environments for working with ML-Agents in Unity often look similar to those shown in Image 12. In this case, it is possible to observe one of the examples provided by the ML-Agents assets, consisting of an agent that tries to push a block to the green goal in its location. The environments are created with many agents working simultaneously because, if the computing power is high enough to run it, having more agents working simultaneously means a faster learning curve since the model receives more examples of the desired actions simultaneously. (Zhang 2021)

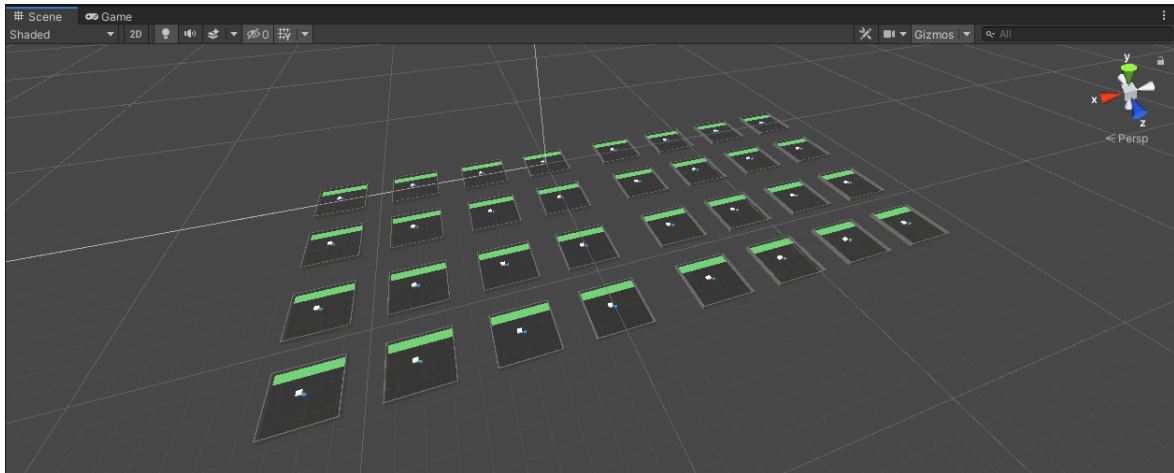


Image 12. Example of one of the environments created for starting in ML-Agents. (Zhang 2021)

One of the critical features of ML-Agents is its support for reinforcement learning. Reinforcement learning is particularly well-suited for game development, where agents can learn to optimize their behavior to achieve specific goals or objectives. Using ML-Agents, developers can create agents to learn from their interactions with the game world and improve their performance over time. (Unity Technologies 2022)

ML-Agents supports many sensors, including cameras, lidars, and other sensors that can generate accurate data for training and testing purposes. This makes ML-Agents a powerful tool for creating realistic simulations that can be used to train agents in various scenarios. (Unity Technologies 2022)

Another advantage of ML-Agents is its support for distributed training, which enables developers to distribute the training workload across multiple machines and GPUs. This can significantly speed up the training process and allow developers to experiment with larger, more complex scenarios. ML-Agents also provides a Python API that enables developers to integrate ML-Agents with various programming languages and AI frameworks, such as TensorFlow and PyTorch. (Unity Technologies 2022)

ML-Agents has been used in various research and development projects related to game development and robotics. For example, ML-Agents has been used to develop intelligent agents for multiple games, such as Capture the Flag and Hide-and-Seek. ML-Agents has also been used to formulate agents for robotic applications, such as autonomous navigation and object recognition. (Unity Technologies 2023)

In addition to its support for reinforcement learning, ML-Agents supports other machine learning techniques, such as supervised and unsupervised learning. This enables developers to experiment with various AI algorithms and procedures to find the best approach for their specific application. (Unity Technologies 2023)

ML-Agents has a growing community of users and contributors actively developing new features and capabilities for the platform. The platform constantly evolves, with new updates and releases regularly. ML-Agents' open-source nature also enables users to contribute to the project, providing feedback, bug reports, and new features to improve the platform. (Unity Technologies 2023)

In addition to its applications in game development and robotics, ML-Agents has the potential to be used in a variety of other applications, such as autonomous vehicles, natural language processing, and computer vision. ML-Agents enables researchers and developers to experiment with new and innovative AI applications by providing a platform for developing and training intelligent agents. (Unity Technologies 2023)

7 Practical case

7.1 Introduction

Having discussed the fundamental concepts of artificial intelligence, particularly reinforcement learning, it is now time to explore the practical applications of this technology. This case study aims to showcase the practical implementation of these concepts by utilizing reinforcement learning to assist individuals in solving real-life problems. Specifically, the objective is to develop a reinforcement learning agent capable of recognizing the layout of a track and driving accurately along its path. Once trained, this agent could be employed in the driving process to help and enhance the overall task.

Reinforcement learning is a suitable approach for tackling this problem, considering the absence of a pre-existing database. Since the agent needs to learn through trial and error, reinforcement learning is an effective methodology. Creating an agent that can accurately recognize and navigate any possible track is a complex task, making it impractical to solve using traditional programming techniques alone. Additionally, artificial intelligence methods such as machine learning would also face challenges due to the difficulty of feature extraction and the need for a readily available database. Given these considerations, using reinforcement learning is justified as the most appropriate approach to address this problem.

To achieve this objective, a program in Unity has been developed featuring a car and a track, which will serve as the training agent and environment. The environment and the model will be appropriately configured and prepared. Subsequently, the agent will undergo training using reinforcement learning algorithms, enabling it to learn through trial and error. As mentioned earlier, this scenario constitutes an initial reinforcement learning problem. Therefore, before commencing the training process, the environment will be set up with multiple checkpoints along the track. Furthermore, the agent will be programmed to ensure the resulting model can be trained effectively and successfully navigate the path.

7.2 Unity Preparation

Some issues were encountered during the installation process for the necessary programs, Python, PyTorch, and ML-Agents, for training in this practical case. Initially, an incompatible software version was installed, resulting in compatibility issues between different programs. To address this, thorough research was conducted on the internet to identify the appropriate software versions compatible with each other. As a result of these efforts, Image 13 displays the final program, ML-Agents, operating successfully without issues.



```

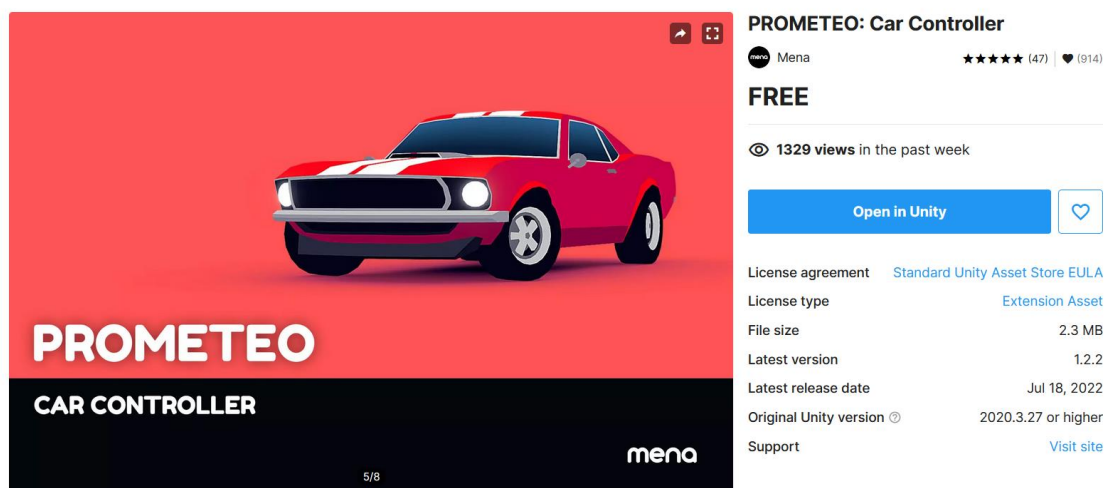
Version information:
ml-agents: 0.29.0,
ml-agents-envs: 0.29.0,
Communicator API: 1.5.0,
PyTorch: 1.7.0+cu110
[INFO] Listening on port 5004. Start training by pressing the Play button in the Unity Editor.

```

Image 13. ML-Agents program working correctly and waiting for input from the Unity Editor.

As mentioned earlier, the environment is crucial in creating and training a reinforcement learning agent. While reinforcement learning algorithms can learn the solution for a given problem, achieving acceptable performance often requires significant prior work.

Furthermore, certain assets were utilized to streamline the development process to save time and effort. Specifically, the "PROMETEO: Car Controller" asset, depicted in Image 14, was employed to provide a car model, while the "Modular Lowpoly Track Roads FREE" asset, shown in Image 15, was utilized for generating the track. The Unity program uses a sample scene from the track asset with an already-made road.



PROMETEO: Car Controller

Mena ★★★★★ (47) | ❤️ (914)

FREE

👁️ 1329 views in the past week

[Open in Unity](#) [❤️](#)

License agreement	Standard Unity Asset Store EULA
License type	Extension Asset
File size	2.3 MB
Latest version	1.2.2
Latest release date	Jul 18, 2022
Original Unity version	2020.3.27 or higher
Support	Visit site

5/8 **mena**

Image 14. Screenshot from the Unity Asset Store of the asset used for the car.

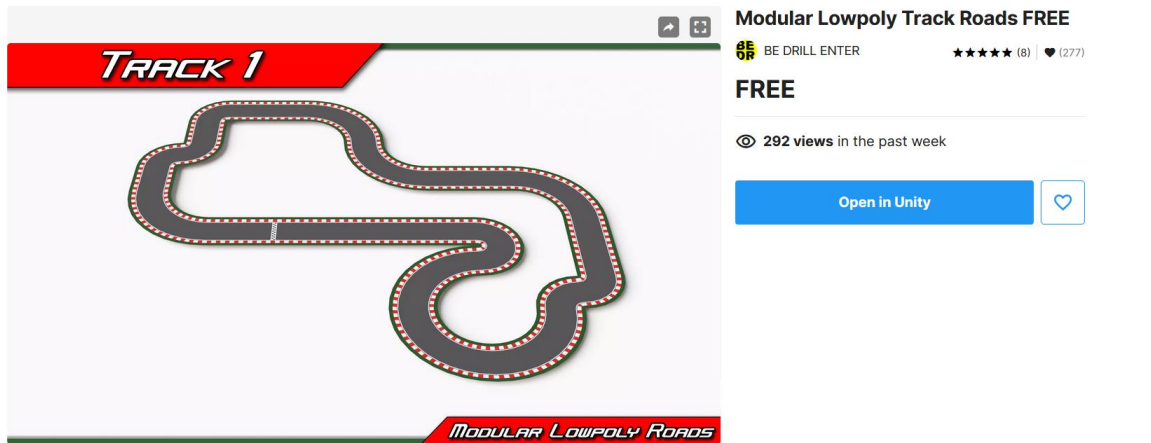


Image 15. Screenshot from the Unity Asset Store of the asset used for the track.

However, some modifications were made to both the track and car prefabs to meet the project's requirements. Precisely, the car's variables were adjusted in a specific manner to optimize the driving experience, as illustrated in Image 16. Additionally, I added some objects from the track asset to create the boundaries of the road, as shown in Image 17.

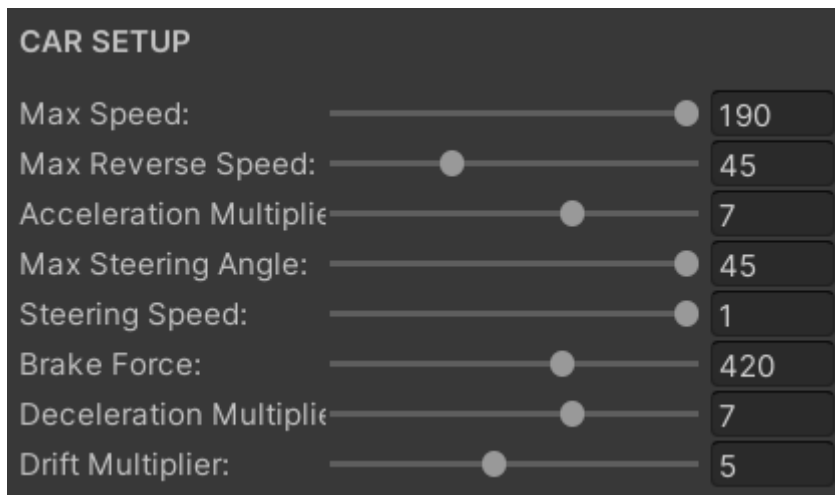


Image 16. Screenshot showing the values of the variables of the car used in the program.

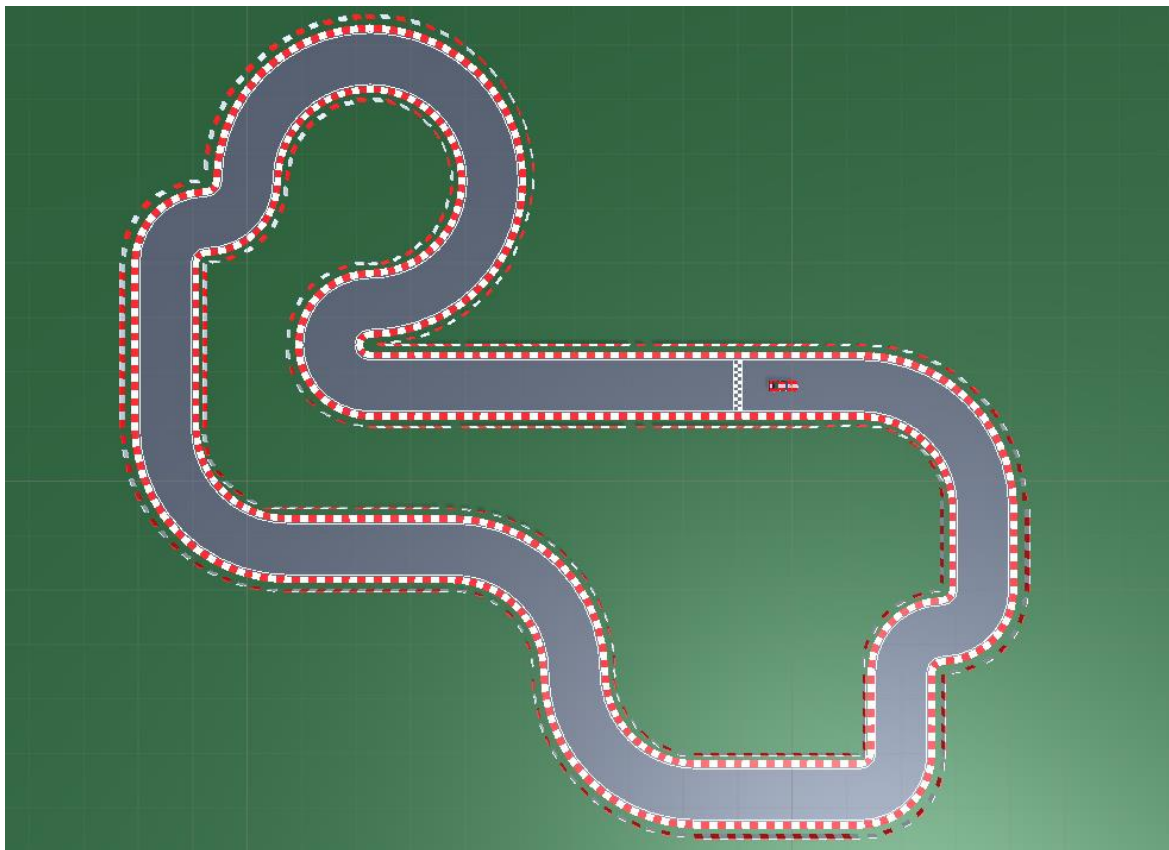


Image 17. Screenshot from the Unity program showing the track.

It is essential to ensure the proper preparation of the environment for reinforcement learning. In the case of problems like this, the environment typically requires the inclusion of invisible walls and checkpoints along the track. Additionally, it can be beneficial to have an invisible wall blocking the finish line, preventing the car, which starts just after the finish line, from choosing to go backward as a solution to reaching it. These measures help create a controlled and structured environment that guides the learning process of the reinforcement learning agent. In Image 18, we can see the track with the invisible walls and the checkpoints ready.

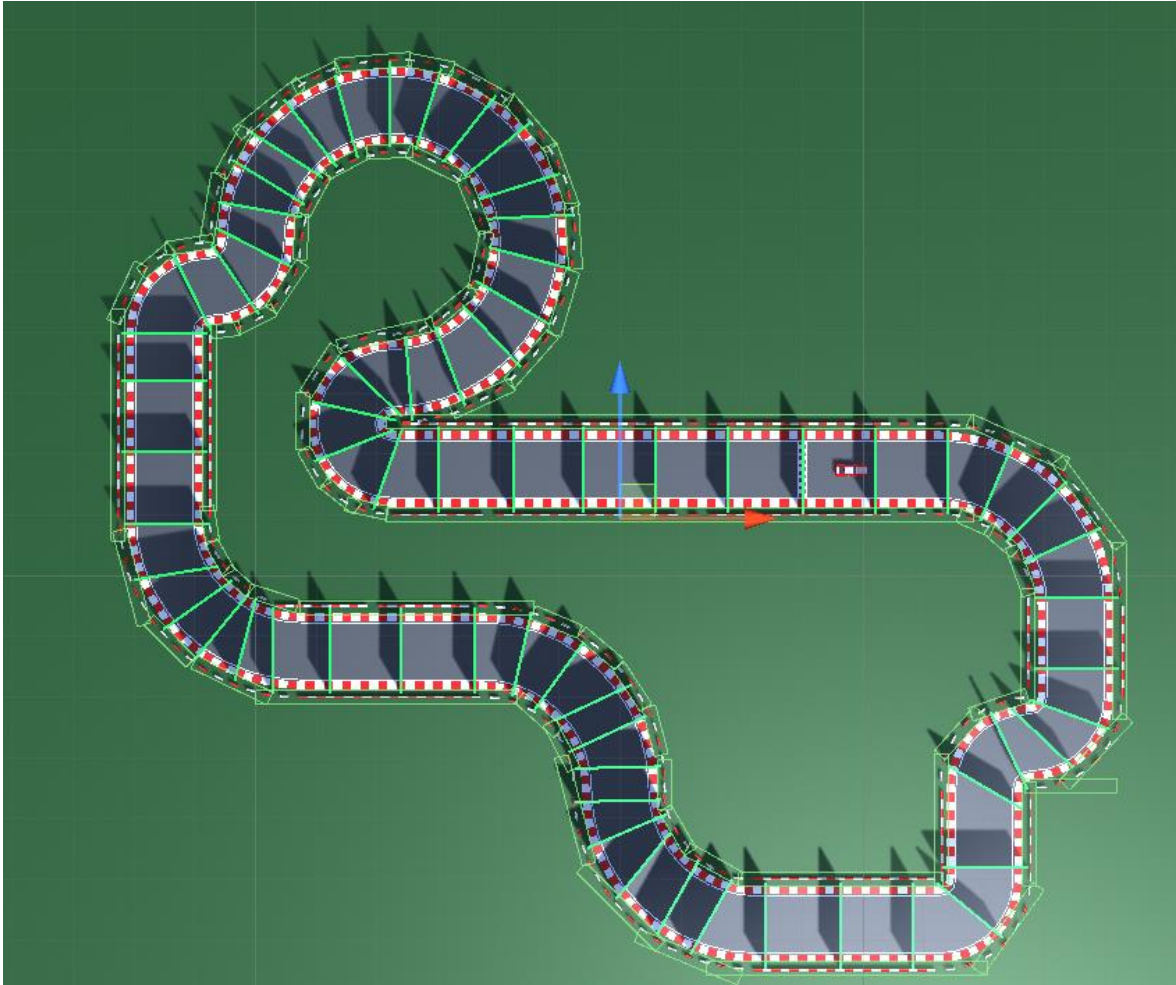


Image 18. Screenshot of the track with the invisible walls and the checkpoints.

7.3 AI Creation

Once the Unity project was prepared, the next step was to create the AI model that would be used to train the agent. To begin, a .yaml file containing the model's configuration was created. A webpage called YAML validator, commonly used for making such files, facilitated this process. Additionally, Windows Notepad was employed to edit and save the file correctly. The parameters necessary for training the model were loaded and utilized within the file.

```
behaviors:
  PrometeoCarController:
    trainer_type: ppo
    hyperparameters:
      batch_size: 256
      buffer_size: 10240
      learning_rate: 1.0e-4
      beta: 5.0e-4
      epsilon: 0.2
      lambda: 0.99
      num_epoch: 3
      learning_rate_schedule: linear
    network_settings:
      normalize: true
      hidden_units: 128
      num_layers: 2
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
    max_steps: 5000000
    time_horizon: 64
    summary_freq: 20000
```

Image 19. Screenshot of the parameters for the training of the model.

In Image 19, the configuration of the agent's model can be observed. The reinforcement learning algorithm chosen for training is Proximal Policy Optimization (PPO). Additionally, it was decided to include three cars in the project. This decision was made to expedite the training process, as the agent learns more quickly when provided with multiple examples rather than just one, despite the potential slight decrease in CPU performance. The output from the model is directly displayed within the Unity project, allowing us to witness the automatic movement of the cars. This training process generates the model that the actual agent will utilize once it has been deemed to be functioning correctly. The model can be further trained to optimize its performance or applied to an agent as necessary.

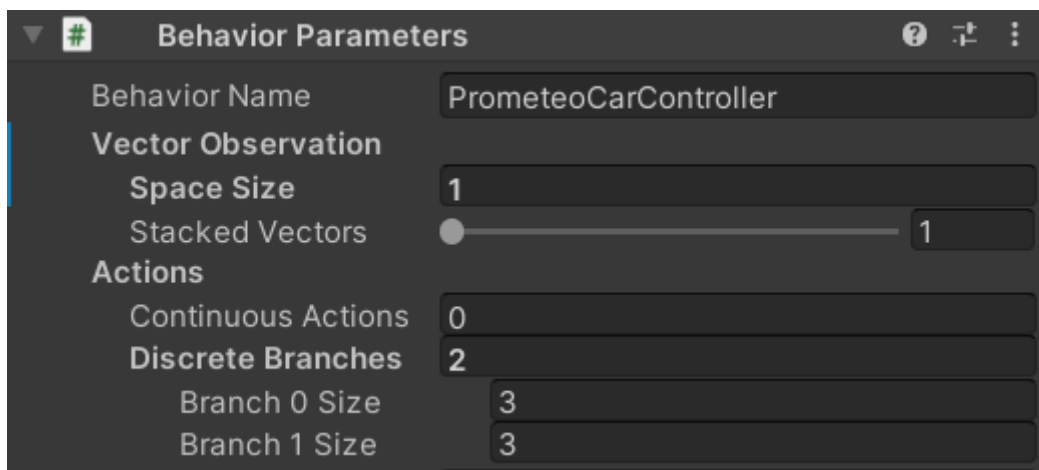


Image 20. Screenshot of the parameters for the behavior of the agent

In Image 20, the parameters for the agent's configuration can be observed. It is possible to observe that the space size of the vector for the observation is one, which means that the agent is receiving one observation at a time, which will later process and work with. Also, it can be seen that two branches for discrete actions are being used, each one with a size of three, which means that there are two groups of three possible actions to commit. In this case, they are applied to moving forward, backward or not moving, and turning to the left, to the right or not turning. This allows the model for the car to know what different actions it can take, which will later lead to it receiving positive or negative rewards depending on the output. For example, moving forward is always rewarded positively, and moving backward is always rewarded negatively to prioritize completing the track in the proper order. However, it is also true that if an action leads to the car going through the correct checkpoint, it will be rewarded positively regardless of the action taken. The same goes for the car stopping or hitting a wall; it will be awarded negatively irrespective of what action led to it.

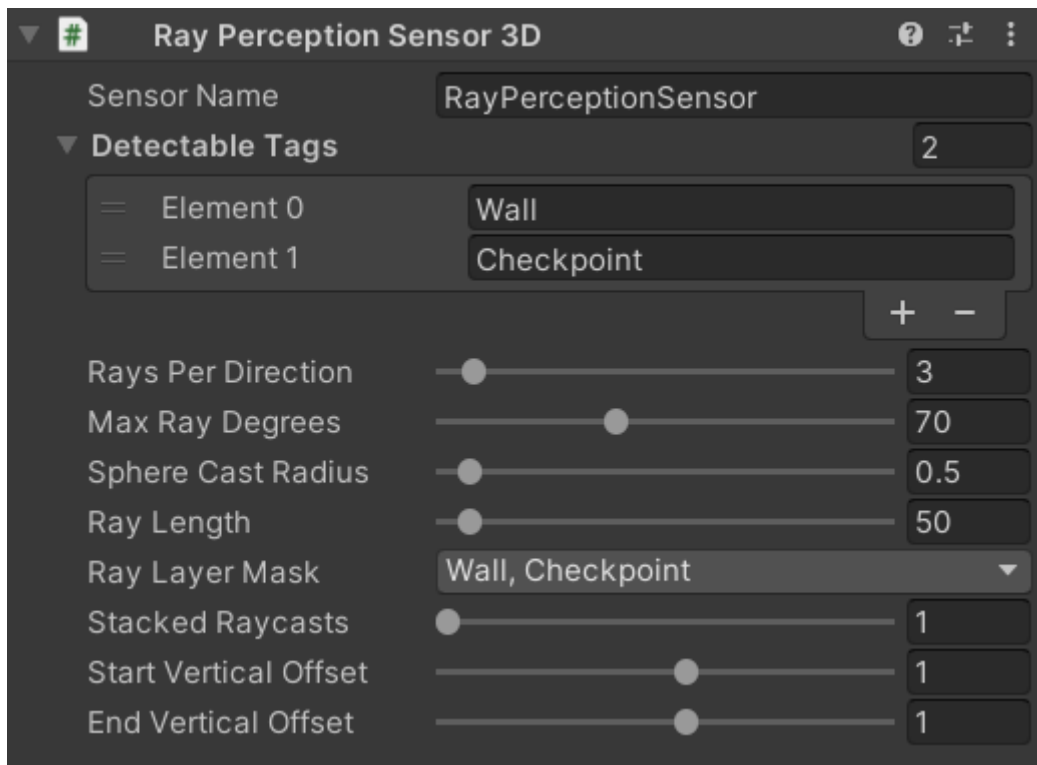


Image 21. Screenshot of the configuration for the sensors of the car.

In Image 21, it is possible to see the configuration for the car's sensors to detect the track limits and checkpoints. It is observable that two tags are being applied as detectable objects: walls and checkpoints. This is important to ensure that the observer of the car is not overwhelmed with information and is only provided with necessary information. Three rays per direction are being used, which means that a total of seven rays are being used to detect these previous objects (three for the left-front, three for the right-front, and one for the front) as it is set to a max of 70 degrees. Also, 50 units are selected as a maximum ray length to ensure they reach the objects and do not finish without detecting anything. It is also possible to observe how a vertical offset of one is set to the start and the end of the ray, making it possible for the beams to detect the objects that are in the actual car's point of view and not the one that could be under or over it.

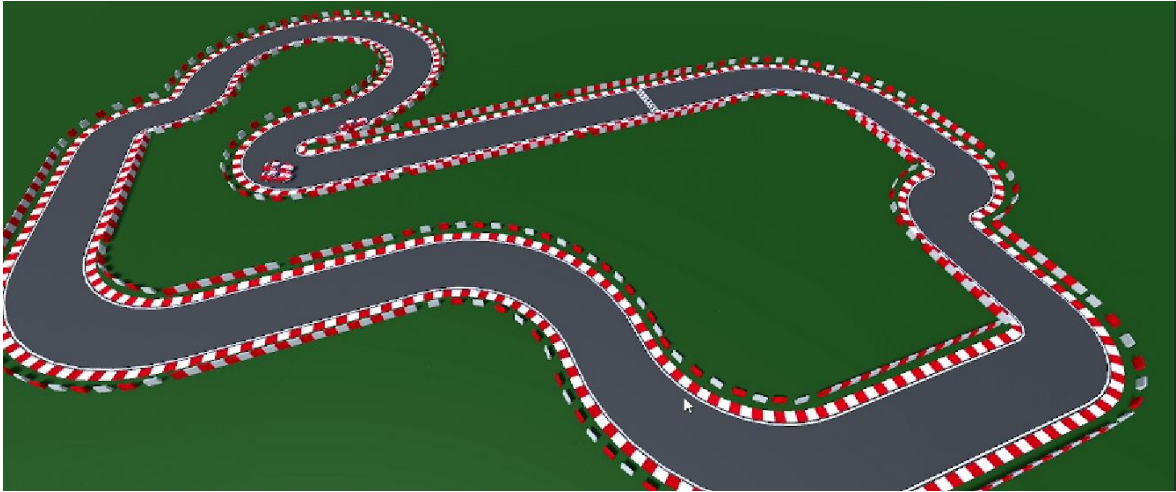


Image 22. Screenshot of the project with the agent being trained.

In Image 22, the training of the model is depicted. It can be observed that the cars have made progress along the track, although one of them is moving in the opposite direction. This real-time training setup enables users to monitor the training process and intervene if necessary. Once the model is created and trained successfully, generating new models based on the existing trained model becomes possible. This facilitates faster training of new models or allows the application of the same trained model to new cars or tracks, enabling learning in different environments.

8 Summary

Throughout this thesis, various concepts related to artificial intelligence have been explored. It commenced with an introduction to the fundamental notions of artificial intelligence. It gradually progressed towards a more focused discussion on reinforcement learning, encompassing explanations of machine learning and related concepts within this domain. Also, models were explored, projecting light on their functionality, and different machine learning algorithms were shown.

Following the theoretical foundation, the thesis reached the practical case study, preceded by an overview of essential software tools relevant to artificial intelligence and their application in Unity. The results obtained were explained within the practical case, and the methodology employed to achieve those outcomes was detailed.

In the practical case of this thesis, a program was developed in Unity to train a car to autonomously navigate a track, regardless of the specific type of track employed for the training process, as long as the required resources are available. A reinforcement learning model was created, enabling the car to learn and improve its driving skills through several training iterations.

During the development of the practical case, difficulties with the model's performance were encountered. Despite investing time in training the agent, the complexity of the problem and limited time and CPU power resulted in initially poor performance. Adjustments were made to the model's parameters to address this issue, and user inputs were provided to the cars. Manual driving demonstrated the desired behavior, giving the agent an example. These interventions led to a notable improvement in performance, although it still needed to meet the desired level of proficiency.

In terms of the final results, it can be stated that the primary objective of creating an agent capable of autonomous driving still needs to be achieved. However, the outcomes did give surprising findings. It was observed that providing user inputs before training the model significantly improved the agent's performance. Furthermore, allowing the agent more time to train or utilizing a computer with higher CPU power would likely enhance its performance. These observations highlight the potential for further improvement and indicate the significance of pre-training inputs and computational resources in achieving better results.

While the original objective of creating a real driving trainer that surpasses user performance and instructs them on improvement has yet to be achieved, given the current complexity, it remains a possibility for the future. However, realizing this ambitious goal would require substantial time and effort. Finding the optimal tools and algorithms to face such a

problem can be time-consuming, particularly for individuals or students new to the field and exploring it experimentally. However, the task may be more viable for enterprises with access to experienced programmers and powerful computing resources.

In the future, with more time and resources, it would be desirable to enhance the user experience by improving the application's interface. This could involve implementing more straightforward and intuitive controls for the car, enabling users to interact more effectively with the agent's model. This would improve usability and lead to more refined and precise examples provided to the agent during training, potentially accelerating the learning process.

In conclusion, this thesis provided a valuable learning opportunity, as it delved into artificial intelligence, a completely new field for the author. Additionally, the project facilitated a more profound familiarity with Unity, a software that had yet to be extensively utilized by the author. The exploration of artificial intelligence concepts and hands-on experience in Unity contributed to the author's knowledge and skills in these areas.

References

- Altexsoft 2022. Semi-Supervised Learning, Explained with Examples. Altexsoft. Retrieved on 2 May 2023. Available at <https://www.altexsoft.com/blog/semi-supervised-learning/>
- Anyoha, R. 2017. The History of Artificial Intelligence. Harvard University. Retrieved on 11 April 2023. Available at <https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/>
- Arrow Electronics 2022. The history of self driving cars. Arrow. Retrieved on 29 March 2023. Available at <https://www.arrow.com/en/research-and-events/articles/the-history-of-self-driving-cars>
- CARLA Team 2023. CARLA. CARLA Simulator. Retrieved on 17 March 2023. Available at <https://carla.org>
- Epic Games 2023. Level Editor. Unreal Engine 5. Retrieved on 27 May 2023. Available at <https://docs.unrealengine.com/5.0/en-US/level-editor-in-unreal-engine/>
- Guthrie, S. 2023. Level Four self-driving 'doable' this decade, says Mercedes-Benz. Drive. Retrieved on 29 March 2023. Available at <https://www.drive.com.au/news/level-4-self-driving-technology-mercedes-benz/>
- International Business Machine Corporation. What is Artificial Intelligence? IBM. Retrieved on 22 February 2023. Available at <https://www.ibm.com/topics/artificial-intelligence>
- International Business Machine Corporation. What is machine learning? IBM. Retrieved on 11 April 2023. Available at <https://www.ibm.com/topics/machine-learning>
- JavaTpoint. Supervised Machine Learning. JavaTpoint. Retrieved on 18 April 2023. Available at <https://www.javatpoint.com/supervised-machine-learning>
- Keita, Z. 2022. Classification in Machine Learning: An Introduction. Datacamp. Retrieved on 18 April 2023. Available at <https://www.datacamp.com/blog/classification-machine-learning>
- Kurama, V. 2023. Regression in Machine Learning: What It Is and Examples of Different Models. BuiltIn. Retrieved on 18 April 2023. Available at <https://builtin.com/data-science/regression-machine-learning>
- Microsoft Research 2021. AirSim announcement. GitHub. Retrieved on 2 May 2023. Available at <https://microsoft.github.io/AirSim/>

Pratt, M. 2020. What is Unsupervised Learning? TechTarget. Retrieved on 2 May 2023. Available at <https://www.techtarget.com/searchenterpriseai/definition/unsupervised-learning>

Read the Docs. Introduction – CARLA Simulator. Retrieved on 17 March 2023. Available at https://carla.readthedocs.io/en/latest/start_introduction/

Russell, P. 2015. How Autonomous Vehicles Will Profoundly Change The World. ResearchGate. Retrieved on 29 March 2023. Available at https://www.researchgate.net/figure/Navlab-5-vehicle-by-Carnegie-Mellon_fig4_338412717

Sandle, T. 2017. Microsoft AI simulator includes autonomous car research. Digital Journal. Retrieved on 15 March 2023. Available at <https://www.digitaljournal.com/tech-science/microsoft-ai-simulator-includes-autonomous-car-research/article/508552>

Simpson, C. 2014. Behavior trees for AI: How they work. Game Developer. Retrieved on 27 May 2023. Available at <https://www.gamedeveloper.com/programming/behavior-trees-for-ai-how-they-work>

Sipe, N. 2023. We were told we'd be riding in self-driving cars by now. What happened to the promised revolution? The Conversation. Retrieved on 29 March 2023. Available at <https://theconversation.com/we-were-told-wed-be-riding-in-self-driving-cars-by-now-what-happened-to-the-promised-revolution-201088>

Synopsys, Inc 2023. What is an Autonomous Car? Synopsys. Retrieved on 24 March 2023. Available at <https://www.synopsys.com/automotive/what-is-autonomous-car.html>

Unity Technologies 2022. Unity ML-Agents Toolkit. GitHub. Retrieved on 8 February 2023. Available at <https://github.com/Unity-Technologies/ml-agents>

Unity Technologies 2023. Unity Machine Learning Agents. Unity Technologies. Retrieved on 8 February 2023. Available at <https://unity.com/products/machine-learning-agents>

Unity Technologies 2023. Unity. Unity Technologies. Retrieved on 2 May 2023. Available at <https://unity.com/>

Unreal Engine 2023. The world's most open and advanced real-time 3D creation tool. Unreal Engine. Retrieved on 2 May 2023. Available at <https://www.unrealengine.com/en-US/>

Verma, P. & Diamantidis, S. 2021. What is Reinforcement Learning? Synopsys. Retrieved on 3 May 2023. Available at <https://www.synopsys.com/ai/what-is-reinforcement-learning.html>

Wirtz, B. 2023. AI in Unity: AI Levels, Setting the Stage for Unity AI, Unity AI & Its Role in Automation. GameDesigning. Retrieved on 22 February 2023. Available at <https://www.gamedesigning.org/learn/unity-ai/>

Zhang, J. 2021. An introduction to machine learning with Unity ML-Agents. Go Coder One. Retrieved on 27 May 2023. Available at <https://www.gocoder.one/blog/introduction-to-unity-ml-agents/>

