University of Bath

Department of Mechanical Engineering

Internship Report

AI in Robotics Internship Report

Elliot Routier

239252454

July – August 2024

Maciej Bednarczyk – Chief of Architectural Software

Sherpa Mobile Robotics

Haguenau, Grand Est, France

September 2024

Acknowledgements

First and foremost, I would like to thank Maciej Bednarczyk for being such a great mentor at Sherpa Mobile Robotics, he has kindly taken me in and tracked my progress throughout the internship. His help has been invaluable and my experience at Sherpa would not have been the same were it not for Maciej.

I would also like to thank the Research and Development department at Sherpa for being so welcoming and friendly, I am truly grateful to have met these people and worked alongside them. The working environment was so positive, it made going to work in the morning so much more exciting than it already was.

Finally, I would like to thank the Arthur Clement's Fund for their financial support, which made it possible for me to complete this internship. Without this assistance, I would not have been able to pursue this valuable opportunity.

Summary

In today’s rapidly evolving industrial landscape, companies are under constant pressure to stay competitive by improving efficiency and automating workflows. In their search to automate and streamline workflows, companies look to new horizons to stand out from their competition and keep up with market standards. Sherpa Mobile Robotics focuses on facilitating such processes. Leading innovators in the sector of Autonomous Mobile Robots (AMRs), Sherpa Mobile Robotics plays a crucial role in the shift from manually focused sectors to automated ones. Automating industrial processes and improving operational efficiency, reducing the need for manual labour has become their objective. During my internship, I had the opportunity to participate and contribute in this race to innovate and constantly evolve to stand out from competition while keeping up with market trends.

Involved in research aimed at innovating current solutions within Sherpa, I delivered weekly presentations focused on the topic of AI intending to raise awareness and prospect new technologies.

During my time at Sherpa, I also had the opportunity to discover various fields within the company. By working with the production team, speaking with the sales division, collaborating with the R&D department and interacting with the software engineers I was able to gain a diverse insight into engineering as a profession.

This internship report will serve as an insight into the different opportunities I have experienced at Sherpa Mobile Robotics and will also clarify where and how the field of Artificial Intelligence is applicable to the sector of AMRs. As the need to innovate is extremely important, researching and developing new technologies becomes a fundamental factor to the success of Sherpa Mobile Robotics.

Table of Contents

Introduction to Sherpa

Sherpa Mobile Robotics (SMR) is a robotics company founded in 2020. Based in Haguenau, France, its CEO, Stephane Fauth, envisioned a solution that would complement SMR's current sister enterprise – NORCAN.

NORCAN, founded in 1987, is a leader in France and among the leaders in Europe in the field of tailor-made, mechanically assembled solutions based on aluminium profiles. When Stephane Fauth took over NORCAN back in 2014 he encountered some of the challenges that industrial manufacturers are faced with day to day. More specifically, following an increase in demand for NORCAN's maintenance chariot, the CEO imagined a motorised version of the chariot. From this idea, Sherpa Mobile Robotics was born.

To address the growing need for automation and efficiency in the setting of industrial manufacturing, this strategic move would allow industrial manufacturers to enhance their productivity, reduce manual labour and answer for a gap in the market. Beyond meeting these criteria, Stephane also envisioned NORCAN to be a primary integrator of the Sherpa range of robots.

Today, the field has become highly competitive with companies like Boston Dynamics, Geek+ and Mobile Industrial Robots at the forefront of SMR's competition. However, Sherpa still manages to stand out in a market as competitive as that of industrial AMR's. With several unique robots that answer to highly specific scenarios and the "follow-me" function that remains the intellectual property of Sherpa Mobile Robotics, the company maintains a strong foothold in the market. The upcoming section will address these stand-out features.

Sherpa-D

The Sherpa-D is SMR's latest robot release. Featuring a singular fork designed to transport dollies, wheeled pedestals, stacks of bins and rolling trays, its maximum payload amounts to a total of 250 kg with a maximum height of 1.2 metres. Weighing 137 kg, this AMR is easily integrated in solutions for moving stacks of bins from a point A to a point B.

Sherpa-B

This fully autonomous AMR, specially designed to improve order preparation processes, eliminates unnecessary manual interventions. Equipped with a LiDAR and flexible add-ons (possible through NORCAN) this 75kg robot can lift weights of up to 200 kg at speeds of 7 km/h. The Sherpa-B's main use-case is for any picking and handling actions thanks to its top conveyor that can be adjusted depending on the client's requirements.

Sherpa-P

The Sherpa-P is designed to transport pallets and crates of up to 1 200 kg. Its purpose – interconnecting production zones with warehouses. This robot is customizable to meet client requirements on demand. Its base can be equipped with multiple extensions to adapt it to a specific environment. Weighing just 250 kg, the use case of the Sherpa-P has been thwarted due to the arrival of the new Sherpa-F…

Sherpa-F

Similar to Sherpa-P, this AMR is designed to transport pallets and crates. However, it does not require specific infrastructure to function, a requirement that has led Sherpa-P to be fully replaced by the Sherpa-F. weighing in at over 200 kg this robot has a weight capacity of 1200 kg. With no environment modifications required, its reliability is also based on its ability to recognise and pick up pallets even when they are not aligned.

Follow-Me Function

The “Follow-Me” feature is Sherpa Mobile Robotics’ standout innovation and remains exclusively their intellectual property. The feature has provided Sherpa with a significant advantage in the market proving to be more than just a unique perk but also a factor influencing clients to opt for a Sherpa robot.

Fundamentally, the Follow-Me function allows a robot to automatically track and follow the nearest individual without the requirement of programming any missions. Minimising downtime and eliminating the need to program specific missions this makes the robot more intuitive to use and streamlines operations. Whether it’s quickly displacing a pallet from a point A to a point B under time constraints or helping a robot adapt to a new warehouse, its applications are vast.

This hands-free solution offers a face paced, dynamic and reliable approach. Allowing businesses to maximise efficiency and adapt to quickly changing scenarios, this simple yet versatile solution has made the “Follow-Me” feature a powerful selling point for Sherpa Mobile Robotics.

Learning Objectives

Offering insights I had not expected, this experience was truly eye-opening, allowing me to gain a real-world understanding of how a company operates and what it means to work as an engineer abroad.

Before starting, I knew my role at Sherpa Mobile Robotics would involve producing weekly reports on the current state and prospects of AI in the world of robotics. I was guided by Maciej Bednarczyk, chief of architectural software at Sherpa, who mentored me throughout the internship, keeping track of my progress and the work that I was tasked with.

The primary task – producing weekly reports on the current state and prospects of AI in the field of robotics – allowed me to delve deep into the latest technologies in the field of AI and contribute to raising awareness within the company about AI's potential in the sector. But the internship wasn't just about the research, I was also presented with the valuable opportunity to experience the balance between personal and work life. Learning firsthand about the dynamics of a company provided a holistic experience contributing to my growth from a professional, but also a personal point of view.

Discovering Sherpa

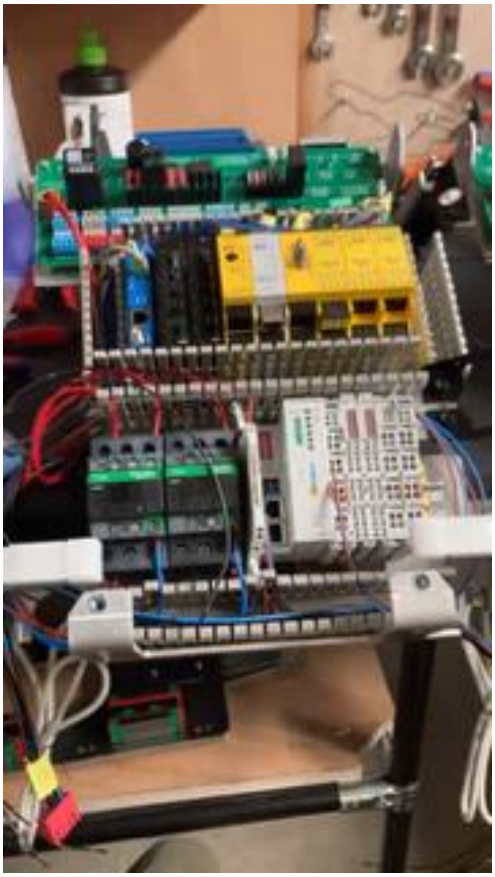
Before discussing the work that unfolded during my internship at Sherpa, I was encouraged, upon arrival, to immerse myself within the company by engaging with as many teams and members as possible. This step was crucial for me to understand the dynamics of the company and integrate as the summer intern. During my first few days, I interacted with a variety of departments including the sales team, R&D engineers, commercial division, human resources and software engineers. This exposure to the company was essential to grasp the company's atmosphere and to settle in comfortably.

Towards the end of this introductory period, I had the opportunity to meet the production team, responsible for assembling the robots. They were very welcoming and friendly; not only did they guide me through the assembly process, but they entrusted me with assembling a Sherpa-D myself. Thrilled at the challenge, I was given flexibility by my mentor,

Maciej Bednarczyk, allowing me to focus on this project before engaging in my weekly AI reports in the field of robotics.

Working alongside Louis, Adam, Thomas and Romain from the production team was a highly enjoyable experienced. Patient and pedagogical, they taught me the intricacies of robot assembly and shared valuable tips and tricks. I spent one week assembling the mechanical aspect of the robot and another week setting up the electrical panel. Despite my limited prior knowledge in both areas, I learned a great deal and thoroughly enjoyed the hands-on experience that assembling a robot has to offer.

Overall, my time with the production team was both enlightening and fulfilling. Creating a complex piece of engineering and engaging with a cheerful and supportive group of people made this experience a highlight of my internship at Sherpa Mobile Robotics.



The current state and prospects of AI in the field of robotics

After spending the first couple of weeks exploring different departments within the company and working with the production team to assemble a robot from scratch, I was now ready to begin the focus of my internship – weekly reports on the current state and prospects of AI in the field of robotics.

Having discussed with my mentor, Maciej Bednarczyk, it was agreed that instead of producing a traditional, weekly, written report which might not directly be useful for Sherpa, I would deliver weekly presentations to the team at SMR. These presentations would be open to anyone curious to learn more about AI, creating an open source of knowledge for anyone to access in engaging and active presentations. As such, these talks would be aimed at raising awareness within the company on the topic of AI, ensuring everyone had a solid understanding of its current state and eventually exploring how this rapidly evolving technology could be practically integrated into Sherpa's operations.

The following is a structured overview of the weekly work I completed at Sherpa, excluding the first couple of weeks and the week of August 12 – August 15:

Elliot Routier
Intern at Sherpa Mobile Robotics
Internship Period: July 1 – August 30
Location: Haguenau, France

Week 3 (July 15 – July 19)

- **Main Topic: Introduction to AI**
- **Goal: Familiarise with basic AI concepts and prepare a presentation.**
- **Work Done: Researched fundamental AI concepts such as machine learning and deep learning. Created and delivered a presentation on the introduction to AI.**

Week 4 (July 22 – July 26)

- **Main Topic: AI Learning Processes**
- **Goal: Understand different types of AI learning focusing on backpropagation.**

- **Work Done: Researched AI learning methods in particular learning about backpropagation. Prepared and delivered a presentation on AI learning processes.**

Week 5 (July 29 – August 2)

- **Main Topic: Computer Vision**
- **Goal: Explore computer vision and its applications in robotics.**
- **Work Done: Studied image recognition and machine vision systems. Created and delivered a presentation on computer vision in robotics.**

Week 6 (August 5 – August 9)

- **Main Topic: Natural Language Processing (NLP)**
- **Goal: Investigate NLP applications in robotics and address the topic of prompt engineering.**
- **Work Done: Researched NLP technologies. Delivered a presentation on NLP and prompt engineering in robotics.**

Week 8 (August 19 – August 23)

- **Main Topic: AI Solutions in Robotics**
- **Goal: Prospect various AI solutions for robotics.**
- **Work Done: Researched AI-driven solutions and case studies in industrial robotics. Presented findings on AI applications in robotics.**

Week 9 (August 26 – August 30)

- **Main Topic: Final Recap**
- **Goal: Summarise all previous topics in a final presentation.**
- **Work Done: Consolidated all topics covered in the internship. Delivered a final recap presentation to the company.**

Key Presentations:

- **Week 1 (July 15 – July 19): Introduction to AI.**
- **Week 2 (July 22 – July 26): AI Learning Processes.**
- **Week 3 (July 29 – August 2): Computer Vision.**
- **Week 4 (August 5 – August 9): Natural Language Processing (NLP).**

- **Week 5 (August 19 – August 23): AI Solutions in Robotics.**
- **Week 6 (August 26 – August 30): Final Recap Presentation.**

What follows in this technical internship report will focus on detailing the key findings during my time at Sherpa Mobile Robotics. Working my way through each week, I will provide technical details on the topics of AI, its learning processes, Computer Vision, Natural Language Processing and AI applications in robotics. Since the last week was focused on providing a summary of the previous weeks, I will not provide a breakdown for this last week.

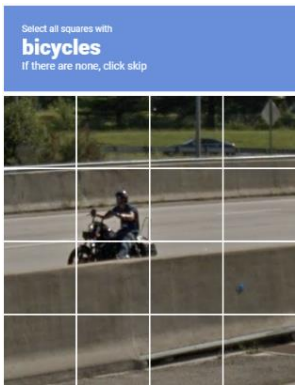
An Introduction to AI

Artificial intelligence refers to the development of computer algorithms to perform tasks that would typically require human intelligence. These tasks include reasoning, learning, problem solving, understanding natural language, recognising patterns and making decisions. AI systems can process huge amounts of data, and its use case is applicable to various sectors ranging from mobile robotics to healthcare. Today, the term artificial intelligence can be misleading and therefore it is important to understand what that means on a more basic level.

Data

Essentially, Artificial Intelligence refers to the ability of a machine to think on its own. In this introduction of AI and for the rest of the report I will be focusing mainly on a branch of AI known as Machine Learning. Machine Learning is a subset of AI that allows systems to learn automatically from data, hence the name “Machine Learning”. Acquiring this data is the challenging aspect of Machine Learning – the quality of the AI model depends on the quality of the data. Think of it like feeding a child – if the child is to grow strong and healthy, he will need the best nutrition available. Similarly, a Machine Learning model will only perform well if it is fed with reliable, well-structured data.

Another challenging aspect of machine learning – classifying and labelling the data gathered. Once properly labelled, this data becomes the foundation for training an AI model enabling it to learn improve and eventually, think. Ever had Google ask you to identify objects, like bicycles from a set of images? That's Google collecting and classifying data which is then used to train their machine learning models like Gemini for instance.

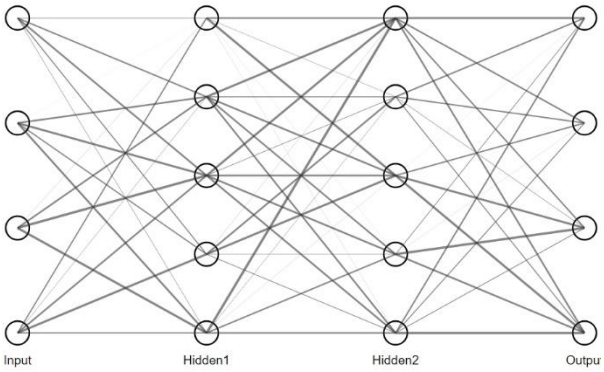


To put in perspective the shear volume of data that an AI can consume, imagine reading through a collection of 500 000 books. Assuming a reading speed of 50-60 pages per hour and that each book is 300 pages long, it would take an average human about 1 000 years to finish these 500 000 books whereas an AI could read through this information in the span of minutes.

Neural Networks

The very foundation of a machine learning system relies on a neural network. A neural network consists of two principal components:

- Neurons
- Neural Links



A neural network is designed to process input data and produce an output, which can be used for making predictions or providing answers. Here is a breakdown of how it works:

1. **Neural Links:** Information flows through the network from the input to the output layer. This data is propagated through the network by neural links also known as weights or coefficients, which determine the importance of the information that is being transferred between neurons.

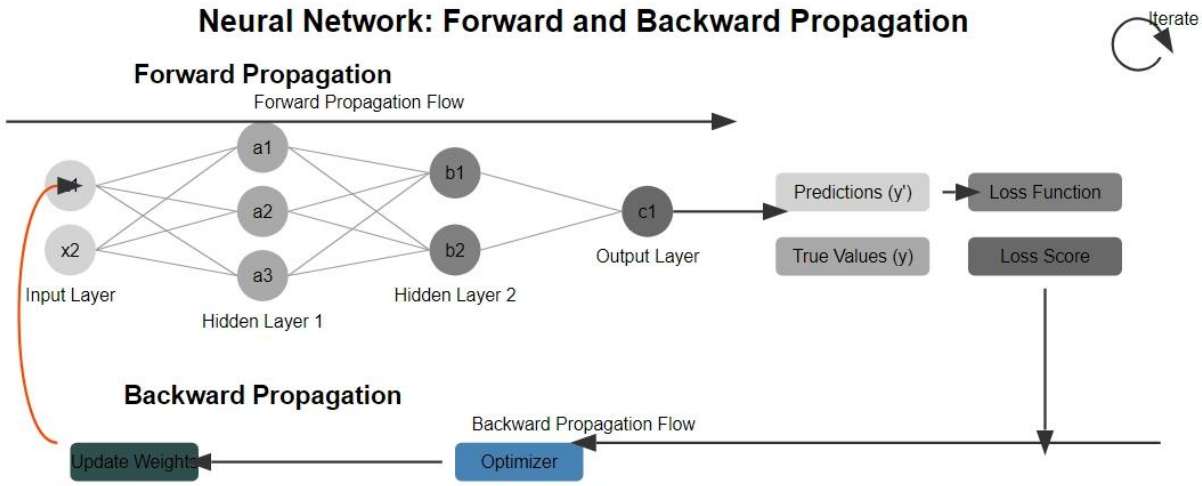
2. **Neurons:** each neuron receives information from the neural links, processes it using an activation function, and passes it to the next layer of neurons for this information to be processed again.
3. **Network Structure:**
    - **Input Layer:** the raw data enters the network
    - **Hidden Layers:** layers between the input and output layer responsible for recognising patterns, hierarchal learning, extracting features and introducing non-linearity to a model.
    - **Output Layer:** final prediction based on the processed information

A Neural network processes data through its interconnected layers of neurons where each layer contributes to pattern recognition and feature extraction. The propagation of this information ultimately leads to the generation of an output. The structure of a neural network is what allows a machine learning to process data as fast as it does.

The Learning Process

The learning process is arguably one of the most technical aspects of machine learning. While it involves many complex details, especially in the underlying mathematics, I will focus on the key steps to give a clear overview without diving into every detail.

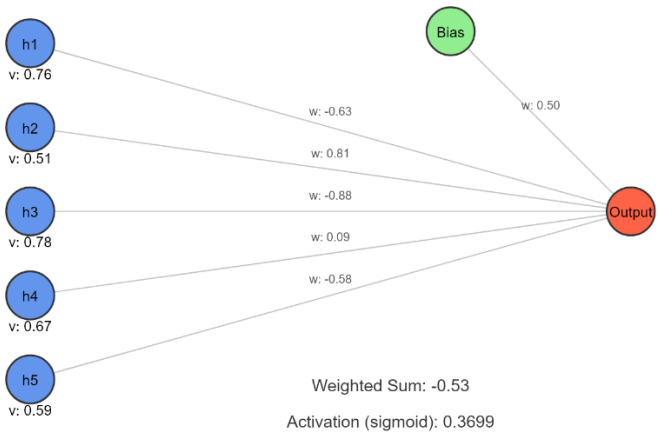
This phase introduces several concepts that have not yet been mentioned. They are not necessary to understand the functioning of neural networks, but they are crucial in understanding how they learn.



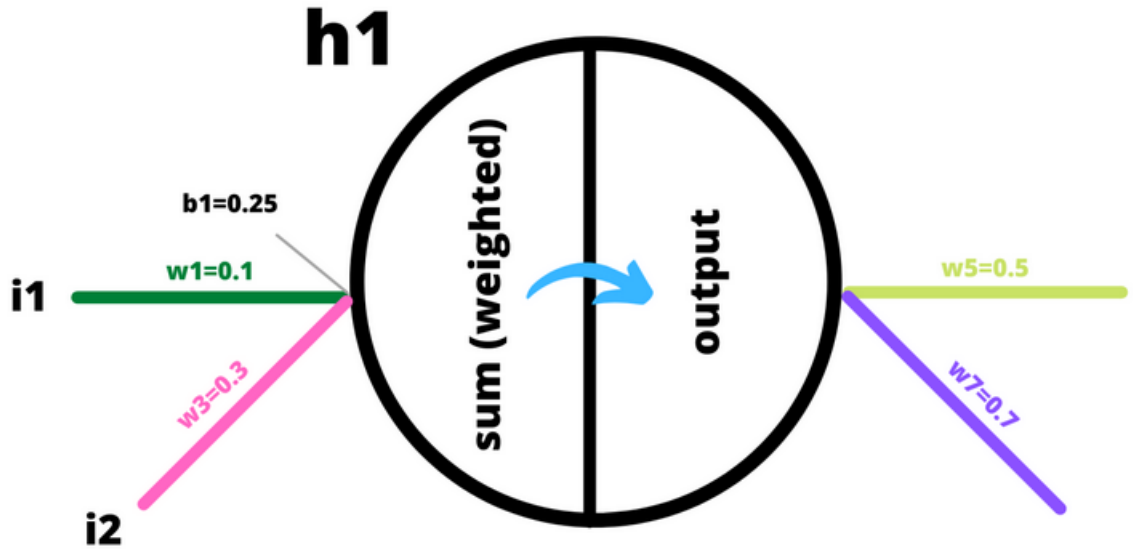
This diagram outlines the different steps involved in this process. My aim is to guide you along this diagram, step by step, to provide a more holistic understanding. During my presentation to the team this diagram was also used as visual support to guide my audience through the process.

Activation Functions

Having touched very briefly on activation functions when introducing the concept of neural networks, I will now expand on this topic. Activation functions process the data passed through the neural network to generate an output from this information. Essentially the function applies a weighted sum to the input data and its respective weights (or coefficients as I like to call them because they determine the importance of the propagated information).



From the diagram above, the blue circles labelled h1, h2, h3, h4 and h5 represent neurons from a hidden layer. Each neuron contains a value (denoted by the letter “v”) ranging from 1 to 0 which is transmitted along a neural link. Each neural link holds a weight (denoted by the letter “w”). A weighted sum of the neurons, and their respective weight, produces an output – in this case -0.53. This activation function is applied to every single neuron in a neural network and allows for the processing and propagation of information within this system.



Peeking inside a single neuron, we see how the input data is averaged using a weighted sum of values and weights to produce an output that is transmitted to the next layer of neurons.

The final step involved in activation functions is the activation itself. Many activations may be used when creating a neural network, to list a few: ReLU, Sigmoid, Leaky ReLU, Tanh (hyperbolic tangent) and Softamx. Each have their use case depending on the outcome required. For the sake of simplicity, I will only explain the activation process of ReLU. The activation aspect of activation functions allows the weighted sum to be propagated through to the next layer (or not). The following steps describe how ReLU translates the weighted sum as an output:

- If the weighted sum is $x > 0$, then the output is x
- If the weighted sum is $x = 0$, then the output is 0
- If the weighted sum is $x < 0$, then the output is 0

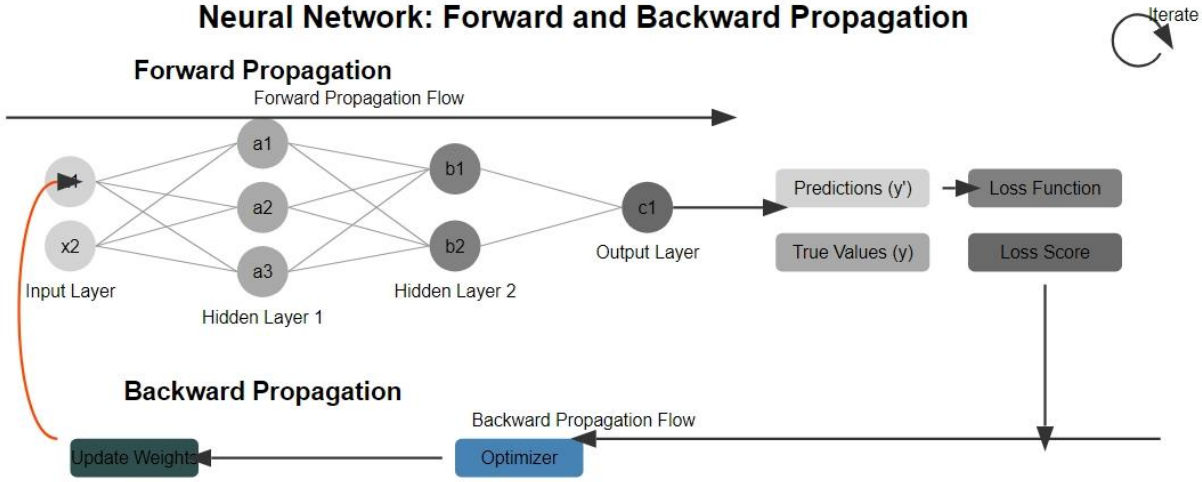
The ReLU function allows the elimination of all negative, non-zero entries into the neural network such that only relevant information is processed by the network. The ReLU function contains a flaw which can be resolved by using different activation functions, such as the ones mentioned earlier.

Bias

You may have noticed in the diagrams above, additional neurons labelled “bias”. The bias helps adjust the weighted sum to influence the input for the next neuron layer. Unlike other neurons, the bias isn’t linked to individual neurons but applies to the entire layer. This added parameter increases the complexity of neural networks, helping fine-tune the outputs.

Loss Functions

Referring to the learning process diagram from mentioned earlier,



after a prediction or an output has been generated, a loss function compares the values generated by the neural network with the expected or true value. This comparison results in a loss score, which indicate how the AI model’s predictions align with expected outcomes. The loss score is later used in the learning process to guide the fine-tuning of parameters to improve its performance.

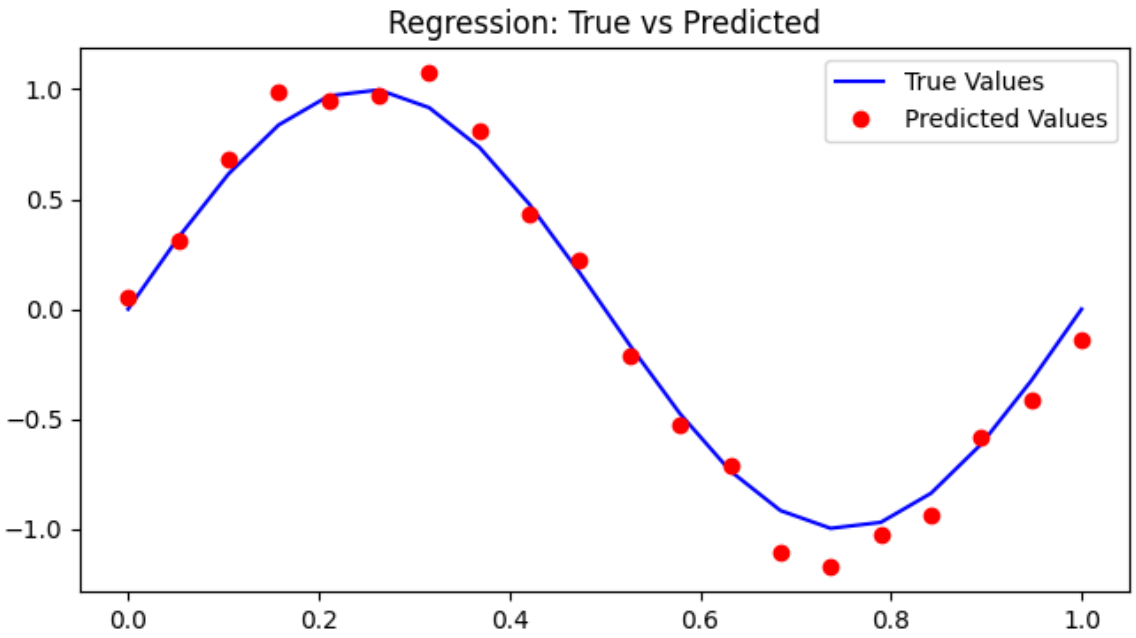
Similar to activation functions, there are different types of loss functions, and each have their ups and downs. For demonstration purposes we will look at two common loss functions:

- Mean Squared Error (MSE)
- Categorical Cross Entropy (CCE)

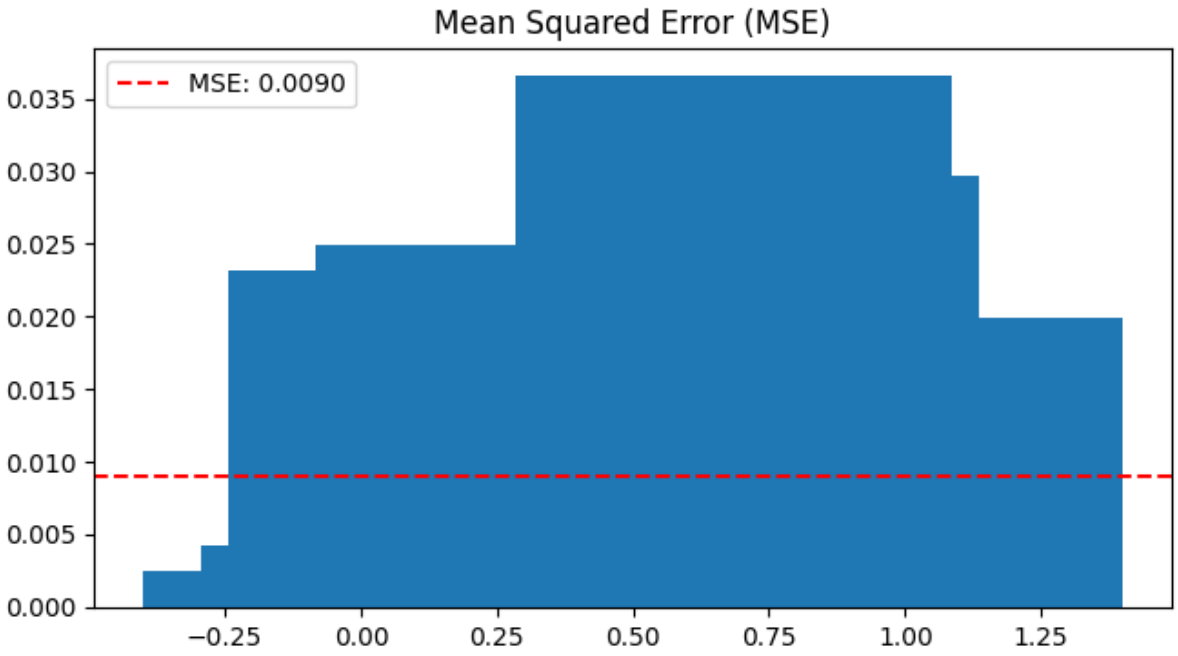
The Mean Square Error is typically used in regression problems. It calculates the average squared difference between predicted and true values. The smaller the score, the better the model is performing.

The Categorical Cross Entropy is used for classification tasks. It measures the difference between the predicted probability distribution and actual class distribution. This score is calculating using probabilities and logarithmic functions, making it more complex than the MSE method.

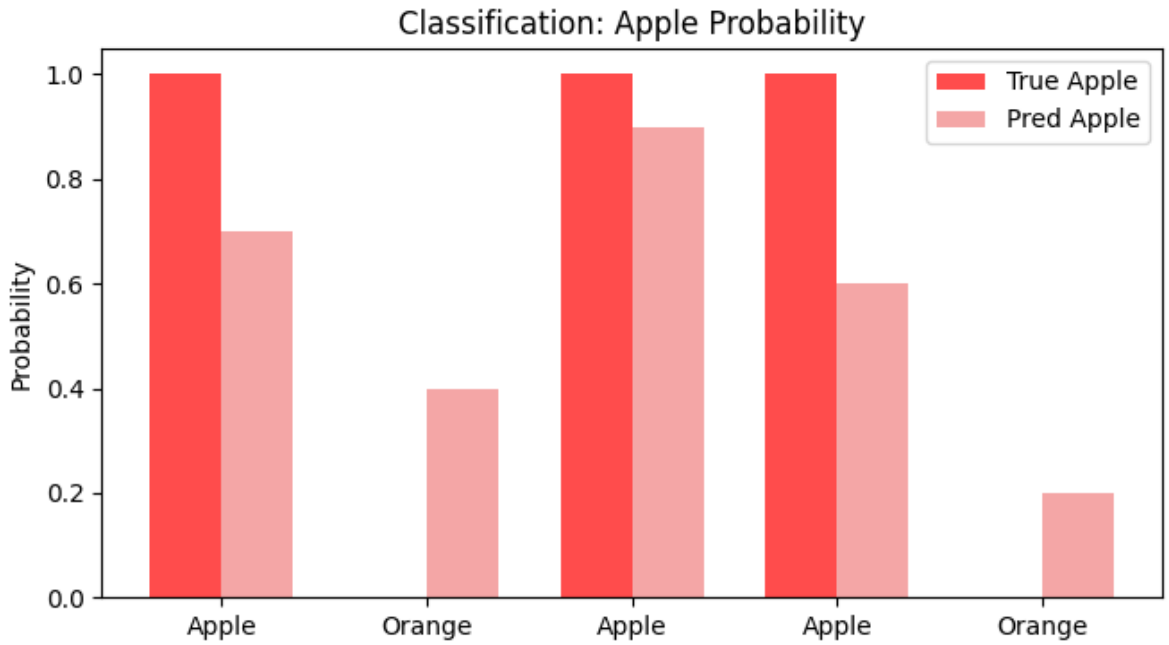
Let's have a look at some examples.



In this example, a sine wave is generated. The sine wave is represented by an oscillating blue line that corresponds to the values we expected our model to predict. The predicted values of the model are shown as red dots following the trend of the sine wave. While the predicted values are close to the expected values, they do not match exactly. On the following page, the Mean Squared Error is calculated using the difference of two squares.

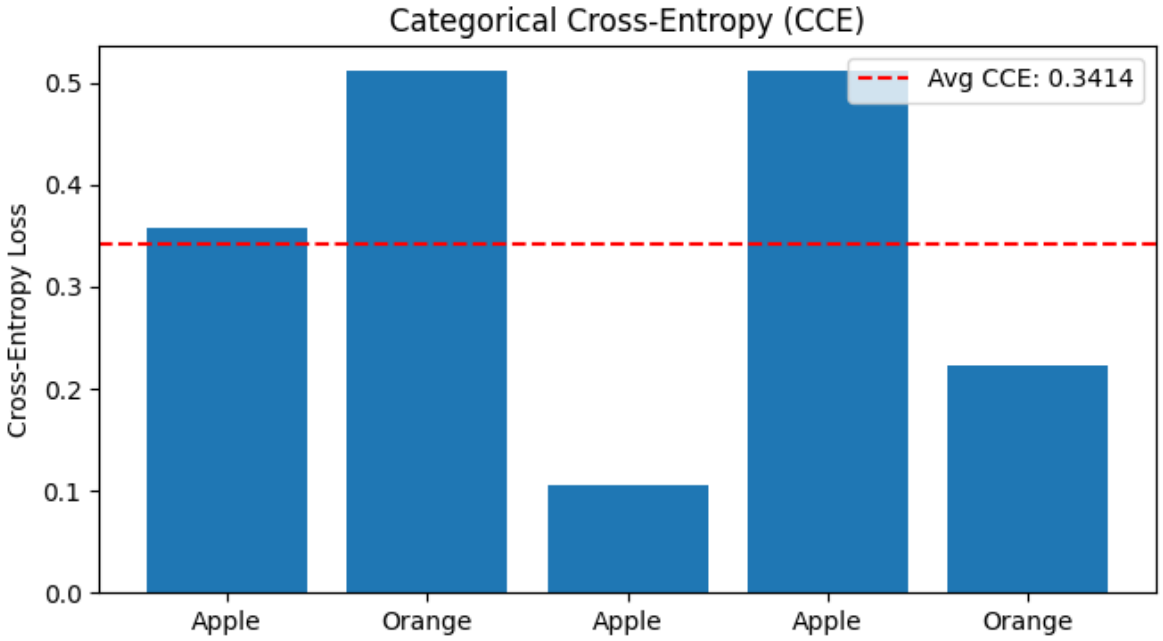


By calculating the difference of two squares, an average loss score is determined. In this case, the Mean Squared Error is 0.0090. The next graph uses Categorical Cross Entropy for classification purposes.



Imagine an AI model that predicts whether an image is that of an apple or not. The dark orange bars from represent whether the image of an apple was provided to the machine or not and the light orange bars show the machine's predicted probability of the image being an apple. The following graph shows the loss score calculated using Categorical

Cross Entropy for the apple image recognition.



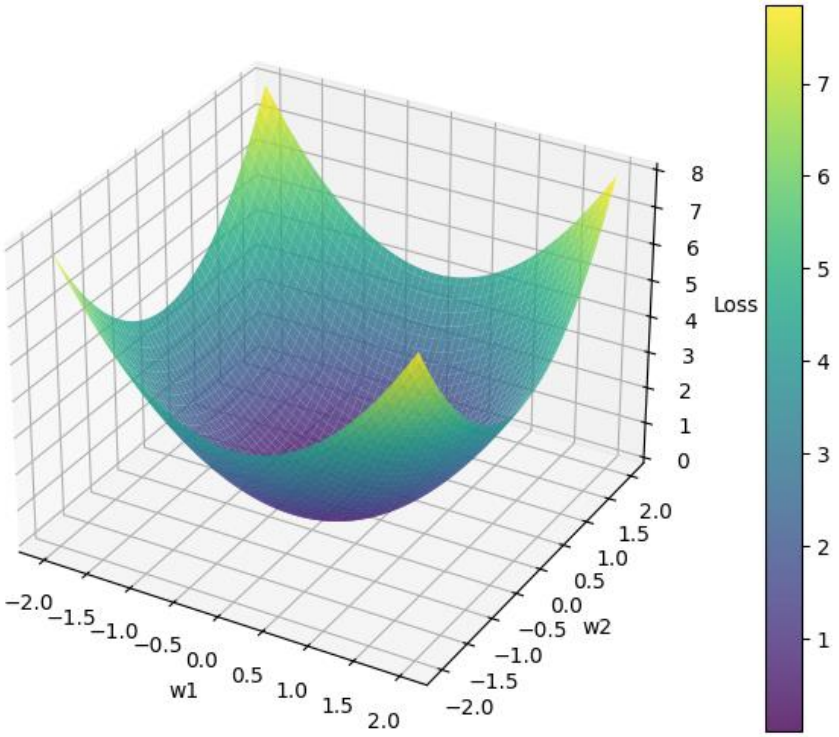
Essentially, the CCE score is calculated using probabilities and logarithmic functions to assess the accuracy of the model's predictions.

Gradient and Backpropagation

Until now we have investigated neural networks use activation and loss functions. However, we haven't seen anything on the learning aspect of machine learning. This is where gradients and backpropagations come into play. These processes work hand in hand to help the model adjust its parameters (weights and biases) to improve performance.

After the loss score has been calculated, the model's output is backpropagated through the network, from the output to the input. Backpropagation determines how each neural link and bias contribute to the loss score calculated.

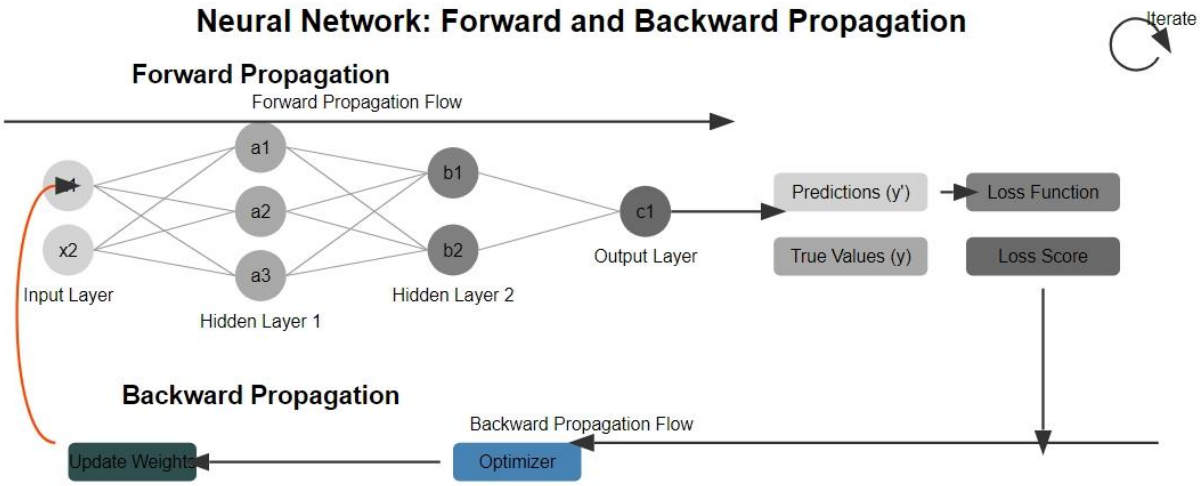
Once the contribution to the loss score for each parameter has been determined, a gradient must be calculated. A gradient indicates the direction and magnitude of changes required to reduce the loss score. To illustrate, imagine a hiker, climbing down a mountain when it is dark. To reach the bottom of the mountain, the hiker carefully places one foot in front of the other making sure that each step taken leads downhill. Similarly, in machine learning, gradients guide the parameters (weights and biases) to reduce the loss score.



We can think of this gradient descent as 3-dimensional space where its y-axis represents the loss score, and the x and z-axes represent the parameters of the neural network which are weight and bias. A neural network could consist of millions of these parameters, so it is impossible to visualise or even understand how this gradient descent occurs.

The gradient is calculated using a vector of partial differential equations with respect to each parameter (weights and biases). These gradients are applied during backpropagation where the information is reversed through the neural network.

Following the gradient calculation from the backpropagation of information, weights and biases are updated to improve the model. This is a cyclical process and will iterate until the loss score has reached a minimum or until the model has reached a suitable level of performance. To summarise the learning process of machine learning, we can refer to our original diagram.



The steps involved include:

- Forward propagation – the input data passes through the network, producing an output
- Loss function – the output is compared to the expected value and a loss score is calculated to measure the model's performance
- Backpropagation – the output is sent back through the network to measure each parameters contribution to the loss score
- Gradient – they are calculated to minimise the model's loss score by optimising weights and biases

Computer Vision

Having covered some of the theory behind Artificial Intelligence, and more specifically behind machine learning, we can start looking at some of the key AI solutions in the world of robotics. In this section of the report, I will focus on a technology that is extremely popular and utilised in the world of AI, computer vision. Computer vision allows machines to see. But what does this mean?

Today, we could program a computer to recognise a dog from a cat by teaching it key feline features such as whiskers, a long tail, pointy ears… and so on. However, identifying the difference between a cat and a jaguar is far more difficult, which is why computer vision is highly valuable for recognition and classification tasks.

In this section we will look at all the processes involved in computer vision:

- Data
- Convolution
- Pooling
- Stacking
- Flattening

Data

As mentioned at the start of this report, data remains a long and tedious process of machine learning. Acquiring clean and structured data is fundamental to obtain the best quality model. Not only selecting the data but also labelling it plays a key role in the process. The same applies for computer vision, the quality of the data is directly proportional to the quality of the model.

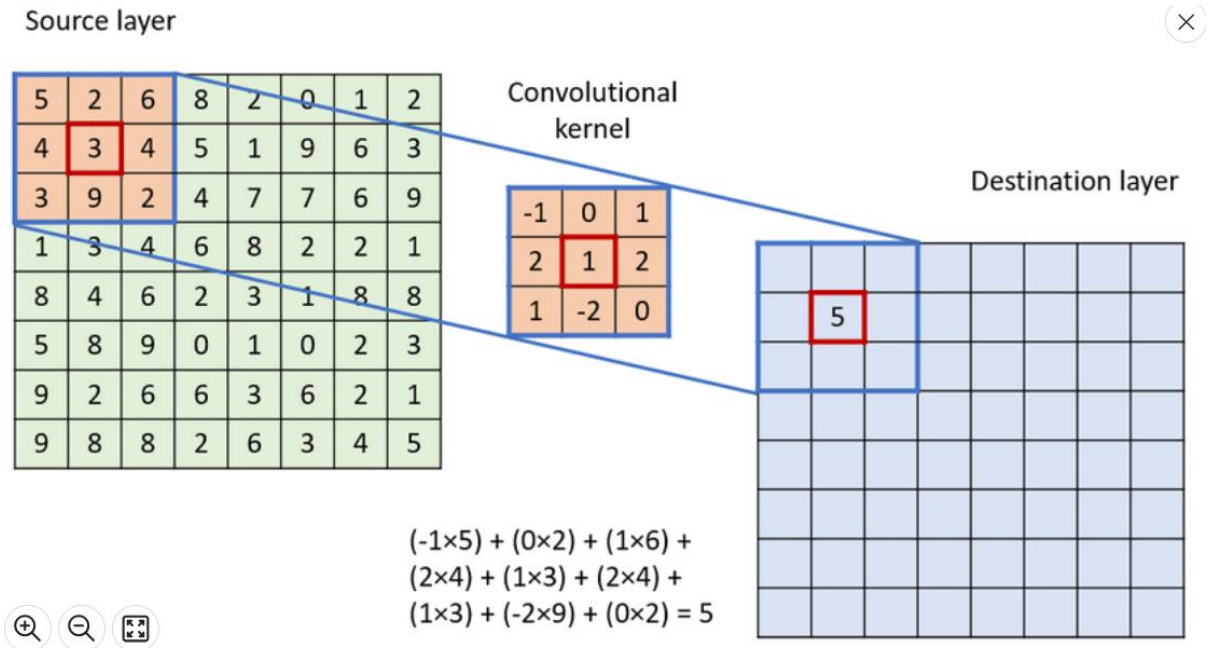
Beyond the acquisition of quality data and thorough labelling, an image must be converted from a coloured one to a grayscale one – in other words, a grayscale conversion.

This is the image of my cat undergoing a grayscale conversion. A grayscale conversion allows for the reduction of information – it makes it easier to process the information of an RGB image (coloured). An RGB image contains pigments of Green, Red and Blue in every pixel of an image. A grayscale image contains only an intensity of light for each pixel. As such, this conversion allows for the initial input data to be reduced by a threefold, making it a lot easier and faster for the network to process.

Convolution

After converting an image to grayscale, applying a convolution layer can further reduce the input data we provide to a neural network and help extract key image features. A convolution layer is a filter, applied to a grayscale image, allowing the neural network to extract key features from the image.

This filter is in the format of a 3x3 matrix of pixels, known as a kernel. The kernel scans the image, performing a weighted sum between the kernel values and the pixel values. To illustrate this process, the following image shows a kernel applying a weighted sum to corresponding pixels of an image:



From this 8x8 pixel image, the top left portion consisting of 3x3 pixels has been translated to a single pixel by use of this weighted sum. The kernel then moves across the image, applying the filter until the entire image is processed. The filtered result is known as a feature map.

Different kernels produce different feature maps. The kernel values depend on the desired extracted features of an image such as edges, contours or brightness.

Pooling

Pooling further reduces the amount of data processed by the neural network while preserving essential features from the feature maps. This step makes the model more efficient by reducing unnecessary information while maintaining key characteristics of the image.

There are two types of pooling:

- Max Pooling
- Average Pooling

Like convolution filters, pooling involves scanning an RxC matrix to an image and either perform an average of the pixels within this matrix (average pooling), or by selecting the most significant pixel within this region (max pooling).

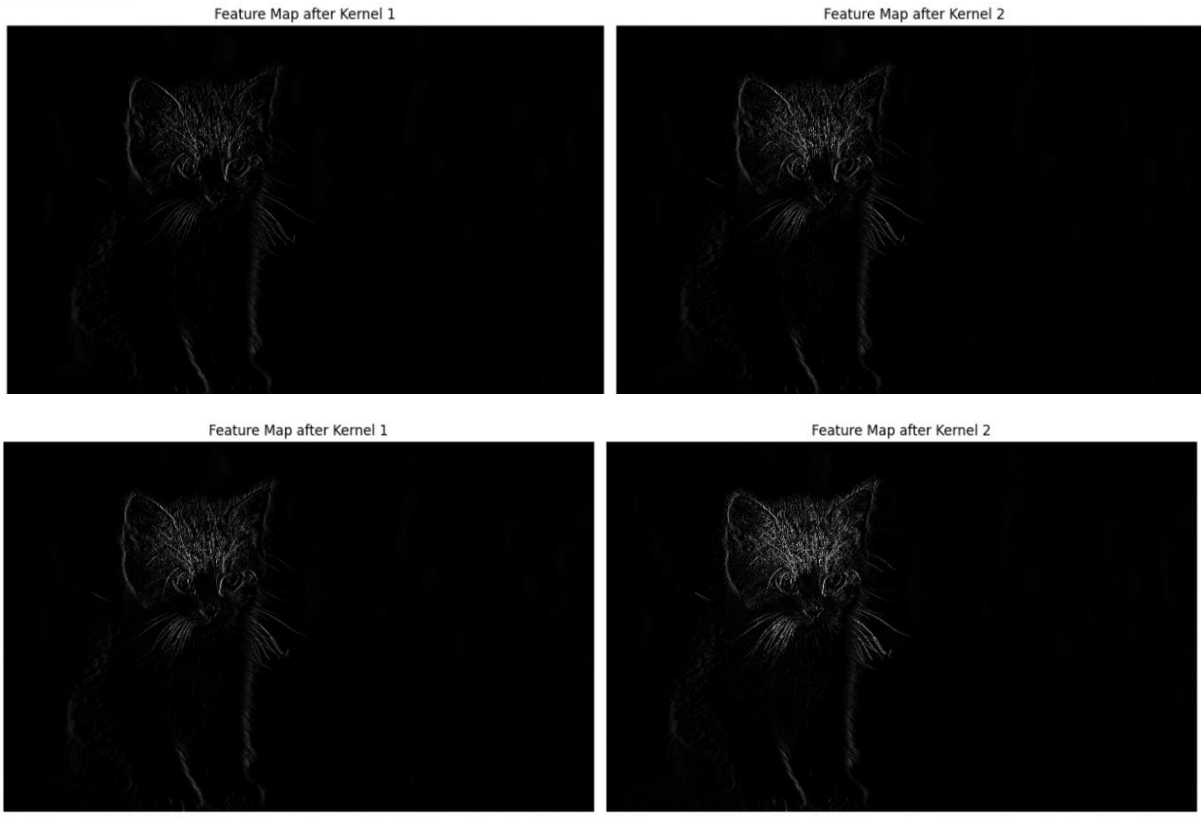
Max Pooling: imagine doing a helicopter tour over New York. The buildings that stand out are the tallest and most prominent ones. Like the skyline of a mega city, max pooling selects the most dominant pixels within a select region.

Average Pooling: an averaging operation ensures that each pixel is accounted for in the process.

Stacking

Applying convolution layers to the initial image is sometimes not sufficient to capture the important features of an image. To handle more complex feature extraction, multiple convolution layers can be stacked on top of each other. In doing so, specific characteristics may be extracted from images that have been already filtered. Here is an example of a grayscale image of a cat whose contour is being extracted:

The feature map is now filtered to extract more complex features of this image:



Through the process of stacking, we can obtain very useful insights into the image with very small amounts of data.

Flattening

Assuming we have applied multiple convolution layers to the image of our cat, such that we obtain the following feature maps:

The top left feature map allows to accentuate 3D aspects of the image, the top right image is the contour filter, the bottom left image allows to blur intensity contrasts, while the last picture aims to highlight bright features.

The flattening process converts the data format such that it can be managed by the neural network.

Currently the data exists as a 3D vector known as a tensor. These 3 vectors include height (vertical pixels of a feature map), width (horizontal pixels of a feature map) and the number of feature maps (4 in this case). Flattening will ensure that this data is translated to a 1D vector and not a 3D one.

The flattening process is very simple and consists of multiplying the vertical pixels with the horizontal ones and multiplying them by the number of feature maps.

The image of our cat is a 2,880 x 1,800 picture (excluding the pooling, grayscaling and filtering that reduce the number of pixels an image contains) and we have obtained 4 of these feature maps. Multiplying these out we acquire a total of 20,736,000 individual pixels that will be used as input data to the neural network.

Once the data is in the correct 1D vector format, that the data size has been reduced and that we have obtained our feature maps, the information can now be processed by the neural network.

Natural Language Processing – NLP

Natural Language Processing, or NLP, is a branch of Artificial Intelligence that allows machines to interpret, process and respond to human language. In the world of robotics, NLPs hold true potential to understand and execute tasks based on natural language commands, creating a more intuitive and user-friendly interface for customers using AMRs. This next section explores how NLPs function, what are its applications in the world of robotics and some of the challenges it faces in the industry.

The following is a comprehensive list of steps involved in NLP:

- Speech Recognition
- Language Parsing
- Semantic Understanding
- Command Execution

Speech Recognition

Although not a necessary step as some natural language can be written and not spoken, most NLP equipped robots use speech recognition technologies like Google Speech-To-Text or Microsoft Azure Speech Services. These technologies use a deep learning models known as Recurrent Neural Networks (RNNs) or Transformers to process audio data in real time by analysing frequencies and amplitudes of sound waves. These deep learning models can account for accents, variations in tone, background noise and even speaker emotions.

Language Parsing

Once the speech has been converted to text, language parsing breaks the text down into a structured layout. There are two types of dependency parsing that order the text into a structured layout. These are:

- Syntactic Parsing
- Dependency Parsing

Syntactic parsing analyses grammatical structures whereas dependency parsing analyses relations between words in a sentence.

Syntactic parsing identifies the role each word plays in a sentence. For example, in “Pick up the box”, the word “Pick” is identified as the verb, and the word “box” is identified as the object of the verb. As such, syntactic parsing is responsible for determining the action and target of a mission. Dependency parsing creates a dependency tree where each word is linked to others based on how they depend on each other for meaning. For example, in “Put the red book on that table”, the dependency tree would relate “red” to “book” and map “on that table” to “Put” to indicate where the action should take place.

Semantic Understanding

Once the structure of grammar and dependency is understood, the robot needs to comprehend the meaning behind the words. Semantic understanding enables robots to understand what is being asked of them beyond the literal meaning of words. This step involves Named Entity Recognition (NER), Semantic Role Labelling (SRL) and Word Sense Disambiguation (WSD).

- NER – allows AMRs to identify specific names, locations or objects. For instance, in “Bring me the book from the shelf”, the robot associates the “book” being on the “shelf”
- SRL – assigns roles to different parts of a sentence to understand relationships between entities and actions. For example, in “John gave Mary the book”, Semantic Role Labelling identifies “John” as the giver and “Mary” as the receiver
- WSD – use of contextual cues to eliminate ambiguity in human language. For example, a “bank” can mean the side of a river or a financial institution.

Command Execution

The command execution phase of NLPs involves translating the meaning of the command into a physical action using its actuators. The process is highly dependant on motion planning, object manipulation and multi-step command algorithms.

- Motion Planning – for mobile robots, motion planning involves determining the most efficient path from one point to another while avoiding obstacles.

- Object Manipulation – for robots equipped with arms for example, they must understand how to handle objects.
- Multi-Step Commands – robots need to perform multiple tasks sometimes such as “got to aisle A42”, then “pick up pallet 445” and “drop it off in zone 5”. These instructions involve task decomposition before being able to execute.

Overall, NLP allows for the translation of natural language from an unstructured to a structured environment. In a more realistic example, if we ask a robot “Add eggs and milk to my shopping list”, NLP algorithms will deconstruct this phrase and rearrange it into the following structured architecture “<Shopping list><item>eggs</><item>milk</></>” by using methods such as language parsing and semantic understanding.

NLP in robotics is a sophisticated and multi-layered process that combines speech recognition, language parsing, semantic understanding, and command execution, all enhanced by continuous learning from interactions. This integration allows robots to engage in more natural, intuitive, and human-like interactions, improving their ability to understand and act on verbal commands across various industries and applications.

Generative AI

During my internship, the research I led on artificial intelligence was strongly supported by generative AI. Most of the work was carried out using these tools, with occasional adjustments made by me. After discussing with my mentor, we decided that I would evaluate various GenAI platforms. This approach allowed me to give detailed feedback on each tool's performance and provide insights on which one was most effective. I evaluated them based on key criteria: natural language processing (NLP), programming capabilities, operability, user experience, and creativity.

To get a thorough understanding of each tool, I spent roughly a week working exclusively with each AI platform, examining its strengths and limitations. In total, I assessed four different GenAIs:

- ChatGPT
- Microsoft Copilot

- Gemini
- Claude 3.5

ChatGPT

OpenAI’s ChatGPT is currently the leader in the generative AI space, and for good reason. It was by far my favourite tool to work with as it excelled in all the areas I evaluated:

- NLP: ChatGPT demonstrates an exceptional ability to understand and generate natural language, though it sometimes leans toward overly formal and verbose responses.
- Programming: It’s highly versatile, often assisting with code generation and debugging. I used it extensively in my first-year projects, although it required multiple iterations to reach the desired result.
- Operability: It’s incredibly reliable—I've never faced server downtime. While I only used the free version, which met all my needs, the paid version at $20 per month is available for enhanced features.
- User Experience: Very straightforward interface with a useful chat history and memory features that store information across conversations for future use.
- Creativity: ChatGPT is remarkably creative, consistently offering original ideas and solutions.

Microsoft Copilot

- NLP: Copilot’s natural language understanding is somewhat basic and occasionally struggles with nuanced input.
- Programming: Its programming capabilities are underwhelming, and I wouldn’t rely on it for serious coding tasks.
- Operability: This is where Copilot shines. Its seamless integration with the Microsoft environment makes it extremely accessible. Similar to Gemini’s connection with Google, this is a standout feature.

- User Experience: Simple and intuitive, with easy access to other Microsoft apps. However, the lack of conversation history is a drawback compared to ChatGPT.
- Creativity: Copilot's creative abilities are bolstered by its access to DALL·E 3, providing up to 15 free image generations per day.
- Additional Features: The real game-changer is Copilot's ability to fetch live data from the web, eliminating the need for external searches. This feature is absent in ChatGPT, Claude 3.5, and Gemini, making Copilot unique in this aspect.

Gemini

- NLP: Better than Copilot in language processing, but still not on par with ChatGPT.
- Programming: Its programming abilities are solid and surprisingly complementary with ChatGPT. Together, they often solve problems the other struggles with.
- Operability: Accessible and integrated into the Google ecosystem, though not as tightly as Copilot with Microsoft.
- User Experience: Simple to use, though it doesn't offer much beyond the basics.
- Creativity: It gets the job done, being reasonably creative in comparison to other GenAIs.

In summary, while Gemini is a high-performing AI, it lacks a standout feature. It feels like a less robust version of ChatGPT, offering nothing particularly unique. Unlike Copilot, which excels with live web access, Gemini doesn't provide anything groundbreaking.

Claude 3.5

- NLP: Claude 3.5 surpasses ChatGPT in natural language generation, producing answers that feel more conversational and human-like.
- Programming: Claude excels in coding, outperforming both ChatGPT and Gemini. I especially enjoyed using it to generate games and animations, which were valuable during my weekly presentations.

- Operability: Unfortunately, this is its weakest point. The servers are often overloaded, making it difficult to access. Initially, when it launched in mid-2024, I had frequent access, but the current free version is limited to only five messages per day. I didn't get the chance to try the paid version, but I suspect it might offer better server access.
- User Experience: The layout and interface are user-friendly, though the limited usage is frustrating.
- Creativity: Claude 3.5 is one of the most creative AI tools I've used, regularly offering insightful and imaginative ideas.

Over the course of my internship, I became deeply familiar with these generative AI tools and continue to use them today. Each has its strengths, and I now employ them for different tasks: Copilot for web-based research, Claude when it's available, ChatGPT as a fallback when Claude is inaccessible, and Gemini, though rarely, as I wait to see if Google enhances its capabilities with live web access. I strongly recommend leveraging this comparison to make the most out of these tools in your own projects.

Fleet Management Platforms

The final project that I led at Sherpa was a benchmarking between 2 platforms that focused on robot fleet management. At Sherpa, the current fleet management we use is called InUse. InUse is an IoT that allows for the design of a web app where data can be retrieved to visualise information from your fleet of robots.

During some of my research in the beginning of my internship, I had come across another fleet management platform named WAKU that implemented AI solutions. When presenting about WAKU to some team members from SMR, they pointed out to me that this was very similar to InUse and that it could be a great idea for me to carry out an evaluation to compare these 2 platforms.

And so, I met the CEO of WAKU and got in touch with people from Sherpa that take care of the InUse contract to get a better idea of both platforms. This following section will aim to highlight the pros and cons of each platform, why we should use one over the other and more importantly what are their differences.

InUse

InUse is a French company based in Paris that specialises in the Internet of Things (IoT). IoT refers to a network of physical devices embedded with sensors and software, enabling them to collect and exchange data over the internet. InUse itself isn't specifically a fleet management platform but rather provides the tools necessary to build one. This gives Sherpa, a high degree of flexibility in customising the platform according to our needs.

The web app that Sherpa developed with InUse allows us to access Key Performance Indicators (KPIs) that are crucial for meeting Service Level Agreements (SLA) and identifying potential robot errors. It also provides real-time insights into robot operations, such as mission durations and daily task details.

While InUse is primarily used for data visualisation in our current setup, it's capable of much more. After speaking with Laurent Couillard, the CEO of InUse, I learned that the platform could integrate third-party services, manage robot maintenance, handle ticketing systems, oversee inventory and spare parts, plan tasks, and even include AI-driven chatbots. This flexibility highlights the potential of InUse beyond just visualising KPIs.

WAKU

WAKU is a German company founded in 2019, located near Sherpa in the Alsace region. The founders, Leo KaBner, Victor Splittgerber, Alexander Bresk, Florian Purchess, and Sander Nijssen created WAKU Care, a software solution designed for managing the maintenance and after-sales support of Autonomous Mobile Robots (AMRs). The main goal of WAKU Care is to optimize robot operations and help clients achieve a high return on investment (ROI) by improving how mobile robots are integrated into logistics processes.

Unlike InUse, WAKU provides a more focused, pre-built platform specifically tailored for AMR maintenance and data retrieval. While the app is very efficient and user-friendly, it lacks the customisability of InUse. Every client gets the same interface with only minor differences based on contract details. This uniformity simplifies things but may not suit companies with highly specific or evolving needs.

Comparison

The web app Sherpa developed with InUse is functional but still in a rough state, requiring further time and investment to reach its full potential. In contrast, WAKU's app is simpler, more intuitive, and highly polished because it's designed specifically for the AMR industry. However, WAKU sacrifices customisability, something InUse offers in abundance.

When comparing the AI capabilities of these platforms, they are both on similar paths. Both InUse and WAKU are in the beta phase of integrating AI-driven chatbots designed to handle ticketing issues. These chatbots will draw from historical data and internal documentation to resolve common problems without needing technician intervention, saving time for more complex tasks.

In terms of pricing, InUse is the more expensive option, but this reflects the broader range of services it offers. Over a 10-year period and for the connection of 1,500 robots, InUse costs around €300,000, while WAKU's subscription is priced at €156,000. This pricing difference highlights the trade-off between flexibility and cost.

Conclusion

In summary, comparing InUse and WAKU is challenging because they cater to different aspects of fleet management. InUse offers a highly customizable platform focused on IoT and data visualization, allowing companies like Sherpa to build a system tailored to their specific needs. On the other hand, WAKU delivers a straightforward, ready-to-use platform designed for AMR maintenance. It's cheaper, more user-friendly, and highly specialised but lacks the flexibility of InUse.

Switching from InUse to WAKU would involve a period of adaptation, as Sherpa's team would need to get familiar with a new system. However, WAKU is specifically designed for AMR manufacturers, which could offer advantages in streamlining operations. While InUse excels in data visualisation and customisation, WAKU offers a more specialised and affordable solution for Sherpa's AMR fleet management and maintenance needs.

Final Thoughts

As an engineering student, my internship at Sherpa Mobile Robotics has been an eye-opening experience, bridging the gap between theoretical knowledge and practical application in the field of Autonomous Mobile Robots (AMRs).

The hands-on experience of assembling a Sherpa-D robot provided invaluable insights into mechanical and electrical assembling of AMRs. This practical exposure, combined with my research into AI applications for robotics, has deepened my understanding of the technical challenges and innovations driving the industry forward.

My weekly presentations on topics like neural networks, computer vision, and natural language processing not only expanded my own knowledge but also contributed to the company's awareness of AI's potential in robotics. The exploration of various Generative AI tools during my research highlighted the rapid advancements in AI technology and their practical applications in engineering.

The benchmarking project comparing InUse and WAKU fleet management platforms was a motivating project, offering a real-world perspective on how engineering decisions intersect with business considerations.

Overall, this internship has been crucial in my development as an aspiring engineer. It has equipped me with practical skills and industry insights that will be invaluable in my future academic and professional work. I'm grateful for the mentorship and hands-on experience provided by the team at Sherpa Mobile Robotics, which has significantly enhanced my understanding of the exciting and rapidly evolving field of AMRs and AI in robotics.