Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

Intellectual Property of AI systems: a preliminary study

Relatore:
Chiar.mo Prof.
Danilo Montesi

Correlatore:
Dott.
Flavio Bertini

Presentata da:
Andrea Capriotti

Sessione II
Anno Accademico 2022/2023

Sommario

Questa tesi esplora il quadro giuridico riguardante la protezione della proprietà intellettuale nei sistemi di Intelligenza Artificiale (AI) allo stato dell'arte, sia in Italia che in altre parti del mondo. La progettazione e lo sviluppo di tali sistemi richiedono la creazione di modelli complessi e sofisticati, i quali, per essere tutelati adeguatamente, richiedono diritti legali. Tuttavia, stabilire la proprietà intellettuale nei sistemi di Intelligenza Artificiale è spesso un compito arduo, dato il carattere unico di tali modelli.

Nello specifico, in questo studio si porrà particolare attenzione alle Reti Neurali Artificiali (ANN), un modello computazionale che si basa sulle reti neurali biologiche, al fine di esplorarne il funzionamento e comprendere la natura della proposta che verrà trattata. L'obiettivo principale di questo studio preliminare, difatti, è quello di sviluppare una proposta concreta per favorire una protezione legale adeguata ed efficace per i sistemi di Intelligenza Artificiale, in particolare per le Reti Neurali Artificiali.

Attraverso un'analisi del quadro giuridico relativo alla protezione del software nell'ambito giuridico italiano, la tesi individua lacune e limitazioni nello stato attuale dei diritti di proprietà intellettuale nei sistemi di AI. Sulla base di questa analisi, viene presentata una proposta attentamente giustificata ed esposta per affrontare le sfide legali legate alla protezione della proprietà intellettuale per le ANN. Inoltre, lo studio estende le sue considerazioni ai quadri giuridici che regolamentano l'AI in diverse giurisdizioni relative ad altri paesi nel mondo, al fine di valutare la potenziale applicabilità e portata di questa proposta.

Contents

List of Figures

Chapter 1

Introduction

Artificial Intelligence, commonly known as AI, has emerged as one of the most transformative and disruptive technologies of the last few decades. From autonomous vehicles and virtual assistants to healthcare and robotics, AI has revolutionized various industries and has the potential to transform many more. However, as AI technologies continue to evolve and become more popular and invested in, the need to establish laws and rules to preserve and protect AI systems from competitors is becoming one of the most important topics in this field.

One of the key challenges in this regard is the issue of *intellectual property* (IP) rights in AI systems. The designing and the development of those systems involves the creation of complex *models*, and the creators rightfully want the legal rights to protect them. However, determining the ownership of intellectual property in AI systems can be challenging due to the unique nature of each of those models. In order to safeguard them, it is necessary to identify and protect crucial components commonly shared between those systems, which are fundamental for their purpose and functioning, but this is not always a straightforward task.

In particular, an AI model that is receiving significant attention in the last few years, and therefore will be discussed in more depth in this thesis, is the *Artificial Neural Network* (ANN). ANNs are a type of machine learning algorithm that are developed after the "hardware" of the human brain and they have been used in a wide range of cutting edge applications, including image recognition, speech recognition, and natural language processing. As ANNs become more advanced and widely used, the need for legal protection of their IP rights has become increasingly important.

The main aim of this thesis is to conduct an in-depth analysis of the legal framework governing the protection of intellectual property in artificial intelligence systems in Italy. Concurrently, European and international regulatory provisions will be taken into consideration. The ultimate goal of this research is to develop a concrete proposal for an adequate and effective legal protection system for such artificial intelligence systems.

Firstly, this paper is going to examine the current state of intellectual property laws for software, which will provide a foundation for understanding the legal issues surrounding intellectual property rights in AI systems in Italy. Software has been a subject of intellectual property protection for several decades, and the legal framework governing software IP rights has evolved over time. The exploration of intellectual property laws in the realm of software will establish the groundwork for an in-depth analysis of IP protection in AI systems. By drawing comparisons between these two domains, the aim is to discern the differences that underscore the need for more targeted regulations and laws addressing AI system protection.

Furthermore, we will provide an overview of Artificial Intelligence, Artificial Neural Networks, and Machine Learning to establish a foundation for proposing hypotheses on how to protect ANNs and their development processes. By gaining a deeper understanding of these fields, it is possible to identify the legal challenges associated with IP protection for ANNs, **in order propose an effective solution** to address them. This proposal for the protection of AI systems, will be carefully justified and explained in order to understand the reasons why it could be effective if it were possible to implement it within Italy's jurisdiction.

Finally, this thesis will examine the current state of the law on intellectual property rights in Artificial Intelligence systems by analyzing the legal framework for IP protection in various jurisdictions, in order to identify gaps and limitations within them. This analysis will provide insights into the legal implications of IP protection in AI and evaluate our proposal in the basis of those regulations and laws.

Chapter 2

Software and Intellectual Property: state of the art

Software is arguably one the most important component of modern technology, powering everything from our smartphones to almost every device we're using on a daily basis.

This technology has been rapidly evolving and developing over the past century, and as a result, the need for new laws to regulate its protection has become increasingly important.

The legal landscape surrounding software and *intellectual property* has undergone significant changes over the years, as lawmakers have worked to adapt to new technologies and emerging challenges. The aim of this chapter is to provide an overview of software intellectual property and the functioning of the Turing Machine. This understanding is essential for grasping the concept of source code and its legal protection in the state of the art, a topic that will be explored further as it is crucial to understand it for our proposal.

2.1 Software

Software is often described as a set of instructions, programs, and data that allows hardware devices like computers and smartphones to perform specific tasks and functions. These functions serve specific purposes and each of them are implemented using algorithms which are step-by-step procedures that define the logic and instructions necessary for solving problems.

Software power and capabilities are **limited** by two main factors:

1. The underlying **hardware** and the **practical constraints** of computing devices.

2. The **computational infeasibility** of certain problems.

The computational limit of mechanical computation is a fundamental concept in the theory of computation, particularly in relation to what can and cannot be computed using mechanical or algorithmic processes. It's closely tied to the idea of decidability and the *Church-Turing Thesis* which suggests that any effectively calculable function (i.e., any function that can be computed by an algorithm) can be computed by a *Turing Machine*, thus this thesis serves as a foundational principle in the this theory.

In practical terms, this means that if a problem can be mechanically computed using an algorithmic process, it can be simulated by a *Turing Machine*. Conversely, if a problem cannot be simulated by a *Turing Machine*, it is beyond the scope of mechanical computation.

2.2 The Turing Machine

In 1936 a paper entitled *On Computable Numbers, with an Application to the Entscheidungsproblem*[1] written by Alan Turing revolutionized the field of computability theory. This article's aim was to address a fundamental question known as the *Entscheidungsproblem* (the decision problem), which basically seeks to determine if it exists an effective method or algorithm that can decide, for any given mathematical statement or formula, whether it is provable within a given formal system or not.

In this paper Turing came out with the concept of an idealized computing device that could manipulate symbols on an infinite tape following a predetermined set of rules that was able to solve any given computable problem. This notion was formalized by introducing the concept of the *Turing Machine* a theoretical model that is capable of solving any problem as long as there is an algorithmic procedure to solve it by simulating the procedure itself.

This revolutionary theoretical device provided a unifying framework for understanding computation, and revealed the limits of what can be algorithmically computed.

2.2.1 Implementation

The *Turing Machine* **key components** are:

- **Tape:** The tape is an infinite length strip divided into cells. Each cell has a fixed size and can hold a symbol from a finite set of symbols. The tape serves as the memory of the *Turing Machine*.
- **Read-Write Head:** The read-write head is positioned over a specific cell on the tape and can read the symbol at that position, write a new symbol, and move left or right along the tape.
- **State Register:** The state register keeps track of the current state of the *Turing Machine*. It determines the behavior of the machine at any given moment.
- **Transition Table:** The transition table defines the rules or instructions for the *Turing Machine*. It specifies the actions to be taken based on the current state and the symbol read by the head.

The **basic operations** of this machine include:

- Reading/Writing the symbol in which the head is positioned.
- Moving the head to the left/right along the tape.
- Changing the state register.

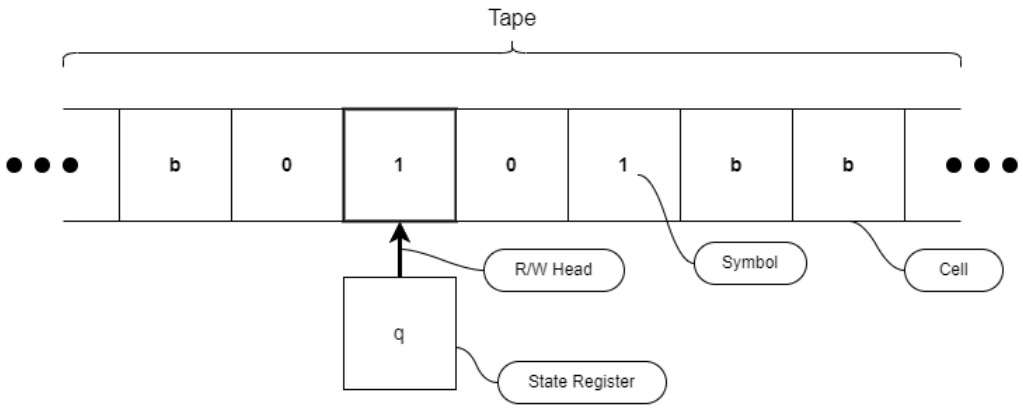

Figure 2.1: A Turing Machine example

2.2.2 Formal definition

A *Turing Machine* (one-taped and deterministic) is formally defined by a *7-tuple* (Q,Σ, Γ, δ, q_0,b,F) with the following meanings:

- Q is the **set of states**. Each state is a unique configuration of the *Turing Machine*.
- Σ is the **input alphabet**. It is a finite set of symbols that are read by the machine on the tape.
- Γ is the **tape alphabet** which includes Σ and additional symbols used for tape manipulation such as the blank character.
- δ is the **transition function** which is defined as: $\delta : Q \setminus F \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$.
  It basically defines the behaviour of the machine by specifying what action the machine should take based on the current state, the symbol read from the tape, and the current direction of the read/write head. The transition function maps (current state, tape symbol) pairs to (next state, symbol to write, direction to move) triples.
- $q_0 \in Q$ is the **initial state** of the machine.
- b $\in \Gamma$ is the **blank character**.
- F $\subseteq Q$ is the **set of the final states** (accepting states). If the machine ends in these states, the input is accepted.

2.2.3 Software and the Turing Machine

It is possible to think of software programs as being analogous to a *Turing Machine* by underscoring the core mechanism at the foundation of both. In fact the concept of software programs aligns with the idea of a theoretical construct that takes an input, processes it based on a **set of instructions** which basically corresponds to a program's **source code**, and produces an output.

This comparison underscores a shared objective: to process input data systematically to generate meaningful output through a given function that encapsulates the instruction to follow in order to fulfill the above mentioned objective, with the theoretical boundaries of the *Turing Machine* model.

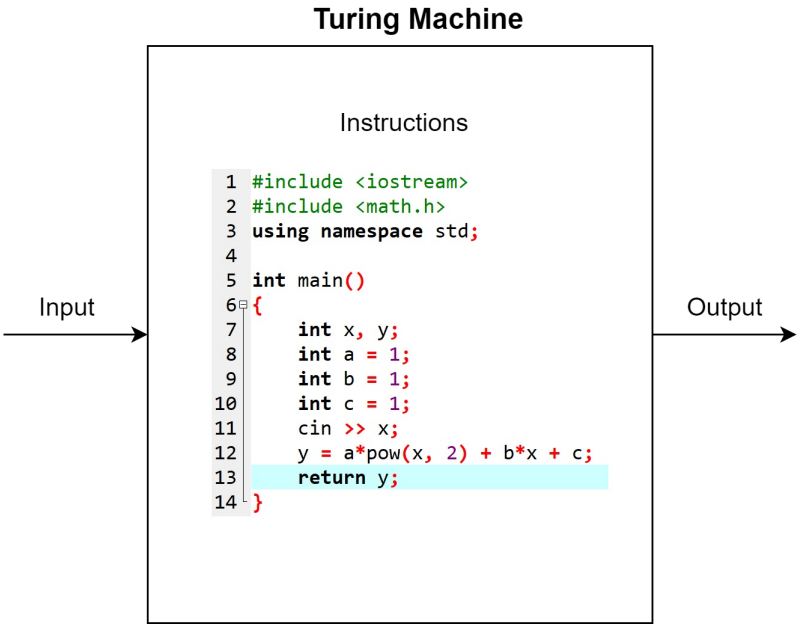

Figure 2.2: Turing Machine analogy with software programs

2.3 Software IP in Italy: State of the art

In the 1970s and 1980s, there were extensive discussions on whether the patent system, the copyright system, or a sui generis system, should provide protection for computer software.

In 1984 WIPO (*World Intellectual Property Organization*) officially adopted a definition of software which was quite wide and approximate in order to fit the steady development of Computer Science, and keep it valid over the course of the years. The definition was the following:

> *Software is an organized and structured set of instructions or symbols, contained in any form or medium (tape, disk, film, circuit), capable of directly or indirectly causing a particular function, task or result to be performed or obtained by an electronic information processing system.*

In Italy there isn't a specific law or organization like WIPO that actually addresses in a generic what a software program really is, however they belong to the category of intangible legal goods, as they are not consumable (and therefore not subject to wear and tear) and can be used simultaneously by an indefinite number of subjects, without their usefulness being diminished. From a legal point of view, therefore, software falls within the intellectual creations, and as such it is protected either as:

- Work of authorship (pursuant to Article 2575 of the Italian Civil Code), and therefore subject to the copyright law.
- Industrial invention (pursuant to Article 2585 of the Italian Civil Code), and therefore subject to patent.

2.3.1 Copyright Law

According to the Article 2575 of the Italian Civil Code:

> *Formano oggetto del diritto di autore le opere dell'ingegno di carattere creativo che appartengono alle scienze, alla letteratura, alla musica, alle arti figurative, all'architettura, al teatro e alla cinematografia, qualunque ne sia il modo o la forma di espressione [2424, n. 4, 2579, 2580].*

This means that any creative work of authorship which belongs to the field of science, literature, music, the the visual arts, architecture, theater, and cinematography, in any form or expression, falls under the protection by the copyright law which is known as *Legge sul diritto d'autore (L. n. 633/1941).*

In fact in the Article 2, Paragraph 8 of the above mentioned law, it is set out clearly that:

In particolare sono comprese nella protezione: [...] 8) I programmi per elaboratore, in qualsiasi forma espressi purché originali quale risultato di creazione intellettuale dell'autore. Restano esclusi dalla tutela accordata dalla presente legge le idee e i principi che stanno alla base di qualsiasi elemento di un programma, compresi quelli alla base delle sue interfacce. Il termine programma comprende anche il materiale preparatorio per la progettazione del programma stesso

This means that computer programs, in any form expressed, provided that they are original as a result of the intellectual creation of the author are protected by this law. Furthermore it states that the ideas and principles underlying any element of a program, including those underlying its interfaces, are excluded from the protection granted by this law. Finally it underlines that the term *program* also includes the preparatory material for the design of the program itself.

It is worth to note that the protection provided by the Law n. 633/1941 implies that just the expressive form of the software itself is protected. This means that the ideas, the functions and algorithms on which the program is based are not safeguarded by this specific law, therefore what it's really being protected is the *source code* of a program as it is considered to be like a literary work, just like books, as it is stated in article 1 of this very same law:

[..] Sono altresì protetti i programmi per elaboratore come opere letterarie

Let's keep in mind that is how a traditional source code is conceived and protected in Italy, in order to understand some considerations on AI later in this thesis.

2.3.2 Limits of the copyright

It was previously stated that, through the *Legge sul Diritto D'autore*, it is only possible to protect the source code of a program but not the function that it computes. As we mentioned earlier we can consider the code of a program as a set of instruction that a *Turing Machine* follows in order to compute a specific function on a given input, but what if we use a different code to give the very same instructions to the machine in order to carry out the same output? The result given by our software product would be exactly the same using a different code, therefore avoiding the copyright law infringement. It would be just like rewriting a book using different words (code), keeping its original plot (the computed function). Let's see an example:

Consider the program P, written through the code C designed to compute the function f representing the quadratic polynomial:

$$f(x) = x^2 + 2x + 1$$

For the sake of simplicity, we're going to take this exact same polynomial where all coefficients are set to 1.

Let's now consider a second program P'. This program is written through a different code C', however it computes a function f' which is the same as f, so $f(x) = f'(x) = x^2 + 2x + 1$, hence we basically have two programs that do the same thing but written differently as they carry out the same function. Let's see these two different programs written in C programming language:

Program P:

```
#include <stdio.h>

int calculate_polynomial(int input) {
    int result = input * input + input + 1;
    return result;
}

int main() {
    int x_value = 5;
    int function_result = calculate_polynomial(x_value);
    printf("Foo result for x = %d: %d\n", x_value, function_result);
    return 0;
}
```

Program P':

```
#include <stdio.h>

int main() {
    int num = 5;
    int result = num * num + num + 1;
    printf("Result for num = %d: %d\n", num, result);
    return 0;
}
```

This obviously applies both to codes written in the same language and codes written in different languages for example Java and Python:

Program P written in Python:

```
def compute_function(x):
    return x**2 + x + 1

x_value = 5
result = compute_function(x_value)
print(f"f({x_value}) = {result}")
```

Program P' written in Java:

```
public class FunctionCalculator {
    public static void main(String[] args) {
        int xValue = 5;
        int result = computeFunction(xValue);
        System.out.println("f(" + xValue + ") = " + result);
    }

    public static int computeFunction(int x) {
        return x*x + x + 1;
    }
}
```

Our observation leads us to the conclusion that the scope of copyright law is bounded by certain considerations.

While it does offer protection to the *source code*, there are instances where this provision is not enough to safeguard the entirety of a software product. Consequently, while copyright offers valuable protection to the expression of ideas in code, it may necessitate complementary legal measures or strategies to comprehensively secure the holistic value and innovation embedded within a software creation.

2.3.3 Patent protection for software inventions

In Italy, there exist two distinct legal frameworks that address the patent safeguarding of industrial innovations:

- Italian Civil Code, Article 2585.
- Industrial Property Code (D.Lgs 30/2005).

The Italian Civil Code Article 2585 states:

> *Possono costituire oggetto di brevetto le nuove invenzioni [2569, 2593] atte ad avere un'applicazione industriale, quali un metodo o un processo di lavorazione industriale, una macchina, uno strumento, un utensile o un dispositivo meccanico, un prodotto o un risultato industriale e l'applicazione tecnica di un principio scientifico, purché essa dia immediati risultati industriali [2586]. In quest'ultimo caso il brevetto è limitato ai soli risultati indicati dall'inventore.*

This means that new inventions capable of industrial application, such as a method or process of industrial production, a machine, a tool, a utensil or a mechanical device, a product or an industrial result, and the technical application of a scientific principle, provided it yields immediate industrial results, may be the subject of a patent. In the latter case, the patent is limited to the results specified by the inventor.

This a general provision which means that is much more vague and approximate in this matter, in fact it's giving a very broad a definition and leaves a lot of room for interpretation for what can or cannot be subject to patent.

A specific legal framework is defined by the Industrial Property Code (Legislative Decree, 10 February 2005, known as CPI), which states that a software, to be eligible for a patent, must be part of an invention that satisfies the following criteria under Article 45, paragraph 1:

a) Novelty: The invention must involve something that is entirely new and not already part of existing knowledge or prior art.

b) Non-obviousness: The invention should demonstrate an inventive step, meaning it should not be easily derived from what is already publicly available.

c) Industrial Applicability: The invention should serve an industrial purpose, as opposed to being purely commercial in nature.

Furthermore, in accordance with Article 45 paragraph 2 of CPI, are not considered inventions:

a) Discoveries, scientific theories, and mathematical methods.

b) The plans, principles, and methods for intellectual activities, for leisure or for commercial activities, and **computer programs**.

This means that the first point excludes the patentability of algorithms as they are included in this category. Instead, the prohibition of patentability of computer programs "as such" cited in the second point, does not effectively exclude the protection of software through a patent, if integrated within the scope of an invention.

As a matter of fact this same exclusion is applied to the European Patent Convention (EPC), to which the Italian Industrial Property Code is compliant. However, in addition of what the CPI says, the EPC states that the subject of an invention is not excluded if it possesses **technical character**(T258/03). According to the Article 52(2) of EPC:

> *A computer program generates a technical character if, when it is run on a computer, it produces a further technical effect which goes beyond the "normal" physical interactions between program (software) and computer (hardware).*

Therefore, a software, in order to be patentable, has to propose to solve a technical problem and offer a solution that allow to obtain a technical effect.

Chapter 3

Artificial Intelligence

Artificial Intelligence, popularly known as AI, is a term created by John McCarthy, an American mathematician and computer scientist which defined it as:

> "*[...] 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.*"

Artificial Intelligence finds application in diverse areas, such as healthcare for diagnosing diseases and personalizing treatments, finance for algorithmic trading and fraud detection, autonomous vehicles for self-driving cars and drones, NLP for chatbots and language translation, gaming for adaptive game AI and procedural content generation, retail for recommendation systems and inventory management, robotics for industrial automation and healthcare assistance and much more.

In particular, this chapter will begin with a concise exploration of the meaning, scope, and history of Artificial Intelligence. Subsequently, it will primarily delve into a specific branch of AI known as Machine Learning, concluding with an elucidation of Artificial Neural Networks in order to better understand the context in which this thesis is going to conceive its hypotheses.

3.1 A brief history of Artificial Intelligence

In 1950, Alan Turing published a paper titled "Computing Machinery and Intelligence" [2] in which he posed the question: *Can machines think?* and suggested that the answer might be found by observing whether a machine's responses in natural language conversations could be indistinguishable from those of a human. In fact the paper discusses

what we now refer to as the *imitation game*. This game forms the basis of the *Turing Test*, where a human evaluator interacts with both a machine and another human through written communication. If the evaluator cannot reliably distinguish the machine's responses from the human's, then the machine is considered to have passed the test.

1956 is considered to be the official start of the AI field as the first AI conference took place at Dartmouth College and important people like John McCarthy, Marvin Minsky, and Claude Shannon participated, making AI a big topic of interest. During this conference, participants engaged in discussions, presentations, and brainstorming sessions about various aspects of AI. John McCarthy, who coined the term "Artificial Intelligence," played a pivotal role in organizing the event. The conference was notable for laying the foundation for the formal study of AI as a distinct field. It introduced key concepts, debated the potential of machine learning, and discussed the challenges that needed to be overcome to achieve artificial intelligence. While the conference itself did not result in immediate breakthroughs, it played a crucial role in shaping the direction of AI research and sparking interest in the development of intelligent machines.

The initial enthusiasm about the potential of this new field pushed British and U.S. governments to heavily funding research about AI in the middle of the 60s, however, due to the underestimation of the problem, the initial excitement quickly fell off and by the start of the 70s the research was almost completely cut off: this period is known as the "AI winter".

During the 80s AI gradually restored thanks to a number of factors, including the development of new algorithms, the availability of more data and the advent of the first microprocessors. During this period, AI research experienced a revival driven by the commercial achievements of expert systems, which were AI programs designed to emulate the knowledge and analytical abilities of human experts. By 1985, the market for AI had reached over a billion dollars. In the 90s the initial expectations projected significant advancement, but the enthusiasm subsided once more by the late 1980s and early 1990s. The process of programming this knowledge demanded substantial effort, often involving 200 to 300 rules. This led to a phenomenon known as the "black box" effect, wherein the machine's reasoning process remained unclear. Consequently, the tasks of development and maintenance became exceedingly challenging, particularly given the availability of quicker and more cost-effective alternatives, especially for less intricate and less costly endeavors.

In the 2000s some solutions developed by AI researchers were being widely used however Artificial Intelligence had its boom during the 2010s until today. According to the Council of Europe this was due to two main reasons:

- First of all, access to massive volumes of data. To be able to use algorithms for image classification and cat recognition, for example, it was previously necessary to carry out sampling yourself. Today, a simple search on Google can find millions. [3]
- Then the discovery of the very high efficiency of computer graphics card processors to accelerate the calculation of learning algorithms. The process being very iterative, it could take weeks before 2010 to process the entire sample. The computing power of these cards (capable of more than a thousand billion transactions per second) has enabled considerable progress at a limited financial cost (less than 1000 euros per card). [3]

Another factor stems from the rise of Deep Learning, which turned out to be a huge success, driving a major improvement in interest and funding within the field of Artificial Intelligence. Interestingly, UNESCO notes that the number of published Machine Learning research pieces jumped by a whopping 50% from 2015 to 2019[4] and WIPO also pointed out that AI took the lead as the most active emerging tech, scoring the highest count in both patent applications and granted patents in a report dated 2 April 2022. [5]

3.2 Intelligence and Artificial Intelligence concepts

Despite AI being a over 60 years old field, experts are still debating to provide an accurate and general definition for Artificial Intelligence which consequently creates struggle in providing laws and regulations on the matter. This task is quite intricate due to the abstract nature of intelligence itself, which can be understood in various ways across multiple disciplines. Intelligence is often demonstrated through a range of functions such as adapting to new situations, learning from experience, abstract thinking, and efficient resource use. These functions, despite their differences, collectively contribute to better performance and goal achievement by acquiring and processing information. [6]

Although AI draws inspiration from natural intelligence and seeks suitable solutions for information processing, human and artificial intelligence are distinct, notably in terms of hardware complexity. While human brains are far more intricate than current AI systems, capable of embodying a general intelligence in diverse ways, AI excels in specific tasks like numerical calculations and rule-based processing, outpacing human capabilities.

Whilst AI and human intelligence differ substantially in hardware and complexity, AI's capacity to excel in certain tasks highlights its potential. The ongoing pursuit of

refining AI techniques and systems holds promise for bridging the gap between artificial and natural intelligence in the future. The distinction between systems specialized in performing specific tasks and systems capable of emulating human intelligence has led to the division of AI into two types: weak AI and strong AI.

3.3 AI types

As it was just stated, artificial intelligence can be broadly categorized into two main types:

- Weak AI
- Strong AI

They are also respectively known as *Narrow AI* and *General AI*.

Weak AI it's a type of AI designed and trained to perform specific tasks or functions, and drives most of the AI that surrounds us today for example Alexa, Siri and various autonomous vehicles. These AI systems excel in performing well-defined tasks within a limited domain but they are not capable of understanding contexts.

On the other hand, strong AI, represents a more advanced form of Artificial Intelligence. Strong AI systems possess human-like cognitive abilities, allowing them to understand, learn, and apply knowledge across a wide range of tasks and domains. Unlike weak AI, strong AI has the potential to reason, comprehend context, and demonstrate a level of consciousness comparable to human intelligence. However, achieving strong AI remains a complex and ambitious goal, as it requires replicating human cognitive capabilities on a deep and comprehensive level.

It is important to note that existing laws and regulations regarding intellectual property exclusively concern to weak AI, as strong AI remains a theoretical concept without established legal frameworks. Therefore the proposal that is going to made will be valid only within the realm of weak AI creations.

3.4 Artificial Intelligence approaches

AI includes a diverse range of approaches that aim to replicate or simulate human-like intelligence in machines. These approaches can be broadly categorized into two main categories: rule-based AI and data-based AI. Each category represents a unique approach to solving problems and achieving intelligent behavior in machines:

- Rule-Based AI: also known as symbolic AI, it basically uses a set of pre-defined rules to make decisions or take actions. These explicit rules are created by a human expertise in the specific domain in which it operates and guide the system's decision-making process. Rule-based AI is effective for tasks that can be precisely defined through rules, such as expert systems or knowledge-based systems, for example an AI system capable of playing chess. Within this category fall generic software programs. As a matter of fact, the above mentioned rules can be codified into a set of instructions within a source code, which, according to the mentioned Italian laws, can be protected through copyright.

- Data-Based AI: also known as data-driven AI, this type of systems are trained through large amounts of data, and they learn to make decisions by identifying patterns and correlations in the data. This type of AI is often used in applications where it is not possible or practical to write a set of rules that can accurately capture the decision-making process, for example image classifiers. For this reason the source code of these creations can't capture their functioning through a protected set of instructions like in traditional software, which is why this thesis' aim is to try propose an effective way to protect specifically those types of systems.

Numerous AI applications often take advantage of a combination of these two primary approaches with a more *hybrid approach*. This *hybrid approach* blends certain characteristics and aspects from both rule-based and data-based AI categories, resulting in more advanced and complex AI systems.

3.5 Machine Learning

Machine Learning (ML) is an evolving branch of Artificial Intelligence, primarily considered a data-based approach as it centers on employing data and algorithms to mimic human learning, by teaching them to classify, predict, and reveal significant insights in data using statistics techniques to guide decisions in diverse fields ranging from pattern recognition, computer vision, spacecraft engineering, finance, entertainment, and computational biology to biomedical and medical applications.

Understanding the functioning and learning techniques of machine learning methods in general is crucial because artificial neural networks are indeed a subcategory of machine learning. ANNs are a specific type of model used within the broader field of machine learning, and they share many of the same fundamental learning techniques and challenges as other machine learning approaches. This is important to underline because this study will mainly regard supervised learning techniques applied to ANNs.

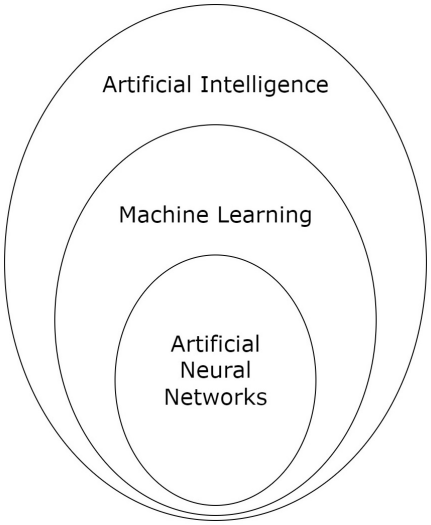

Figure 3.1: An overview of AI, Artificial Neural Networks, and Machine Learning

This field focuses specifically on the learning aspect of AI by developing algorithms that best represent a set of data. ML uses subsets of data to generate an algorithm that may use novel or different combinations of features and weights than can be derived from this principle: [7]

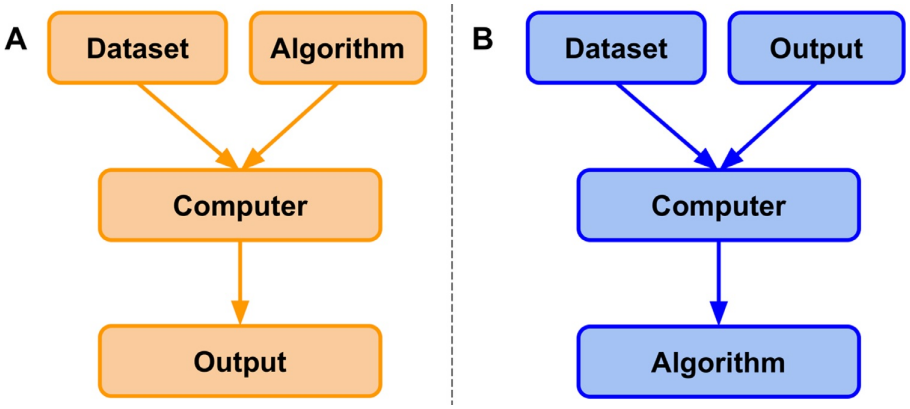

Figure 3.2: Classical programming vs Machine Learning programming

In scenario B, a computer receives a dataset along with linked outputs. The computer learns and develops an algorithm that explains the connection between the two. This algorithm is applicable for making predictions on upcoming datasets.

3.5.1 Learning methods

Within the realm of Machine Learning, there are two fundamental domains, each offering distinct methodologies and approaches according to the specific problem that we seek to resolve. Those two main types of learning methods are:

- Supervised Learning
- Unsupervised Learning

Supervised learning entails learning a mapping between a set of input variables X and an output variable Y and applying this mapping to predict the outputs for unseen data. [8] In order to pursue this objective, in a supervising learning problem the computer is supplied with labeled training data, consisting of observations paired with their corresponding known output labels. The objective is to learn general patterns, often referred to as a *model*, that establish a connection between inputs and outputs. This enables the system to make predictions for new, unseen data, where inputs are observed but their corresponding outputs are not yet known.

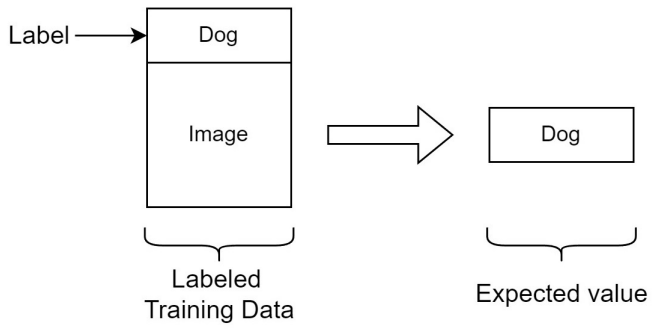

Figure 3.3: The computer during the training is fed with labeled training data (input X) and it has to grasp general patterns to match with the expected value (Output Y)

Unlike supervised learning, which involves training with labeled data, unsupervised learning seeks to find patterns within a dataset and categorize instances accordingly. The goal of this training is to "explore", without any hint of what the model is really looking for, as matter of fact these algorithms are considered "unsupervised" because they don't rely on a predefined target, instead they determine patterns through the algorithm. Some of the most common methods in unsupervised learning include clustering, association, and anomaly detection.

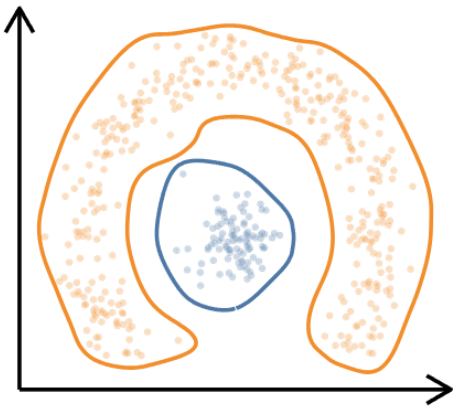

Figure 3.4: An example of the result of density-based clustering, a machine learning technique that autonomously identify clusters in datasets based on their density or proximity to each other

Other Machine Learning methods include:

- Semi-supervised Learning
- Reinforcement Learning
- Self-supervised Learning

Semi-supervised learning consists of an hybrid version of ML which combines supervised and unsupervised methods. During its training the model is fed with a given combination of labeled and unlabeled datasets; the labeled data is used to guide its learning of the unlabeled data. Semi-supervised learning offers a solution when there is an insufficient amount of labeled data for a supervised learning algorithm, especially in cases where labeling a significant amount of data is expensive.[9]

Reinforcement Learning is a paradigm of machine learning that deals with sequential decision problems. In reinforcement learning, the model learns from its experiences by taking actions, observing the outcomes, and assigning a score or reward to the action to evaluate its correctness or quality. This ensures that learning is not solely focused on the current state, but also on the long-term consequences.

Finally, Self-supervised Learning is a form of machine learning in which the system learns without relying on labeled data. Instead, it learns to identify patterns within the data by generating its own labels.

3.5.2 The overfitting problem

Supervised Learning aim is to learn a specific concept or function in order to grasp the connection between the labeled data and the expected value. The learned model should be able to accurately represent the data observed during the training as well as showing the capability to carry out thorough predictions on unseen data.

In order to do so it is mandatory to avoid the overfitting problem in which the model memorize the pattern that links the labeled data to the output, instead of understanding the connection between these two which is an issue that frequently happens when the model becomes overly complex or when there is an excessive number of features relative to a limited set of training examples.

This challenge is often referred to as the *bias/variance trade-off* [10], in which:

- Bias represents the model's average error across various training sets, indicating the difference between the average predicted values and the true underlying mean
- Variance signifies the model's responsiveness to the training set, illustrating the spread of predicted values around their mean for a specific point.

Balancing this prediction error involves in fact a trade off in which it is necessary either to reduce bias or variance; Lowering bias cause the variance to rise variance and consequently the complexity of the models. In order to deal with this issue there are regularization techniques that introduce penalties that alter the calculations of parameter values, helping to manage the balance between model complexity and accuracy.

The settings of this bias/variance trade-off are also bound on the data set it's being used and the artificial neural network architecture, therefore it is necessary to consider these decision along with other components.

3.5.3 Biases in data generation

In machine learning a bias in the context of data generation refers to systematic errors introduced during the process of collecting or generating data. These type of errors can be generated from a variety of factors and can significantly impact both the quality and the representativeness of the data as well as influencing the training of the model.

AI biases can also involve treating certain groups or people in a discriminative way. This typically affects groups that have been historically discriminated against and marginalised due to factors like gender, social class, sexual orientation, or race, but not in all cases. This can happen if the model is built with biased and prejudiced assumptions during the developing of the model but it is not always the case, in fact it can also

happen if the model uses training data that is not accurate or representative, therefore the bias could be unintentional [11].

These are some of the biases that need to be dealt with when developing a machine learning model:

- Historical Bias: A type of bias intrinsic in the society which is reflected on the collection of data [11].
- Specification Bias: This term denotes a bias in the choices and specifications of what constitutes the input and output in a learning task. For example the choices of target variables (like "approve" in a bank loan) can lead to biases. These specifications are made by the designer of the system which therefore needs a strong understanding of the problem as well as the skill to transform the problem into suitable elements [12].
- Measurement Bias: It refers to the distortion or inaccuracy in data collection methods, leading to misleading or unreliable measurements. It can be caused by a variety of factors such as faulty instruments, biased sampling techniques, or human error during data collection.
- Sampling Bias: also called population bias, occurs when the selection of samples from a population is not done randomly and in a representative manner. This can lead to distorted results because the trends and characteristics of the population might not be accurately reflected in the selected samples so the samples may not be representative of the entire population.
- Annotator Bias: When the data labelling process is done manually, the annotator is responsible for reviewing each given output and deciding whether to approve or not approve it based on the designated criteria or guidelines. When doing so, the human operator can transfer biases to the data used to train the model, therefore biasing the trained model.

3.5.4 Transparent and opaque systems

Machine learning includes a very broad variety of methods, both for supervised and unsupervised learning. These methods not only vary in their predictive performance but also in their capacity to offer explanations, often leading to a trade-off between the two objectives. Highly accurate systems tend to be more opaque, meaning they have a reduced ability to justify their decisions.

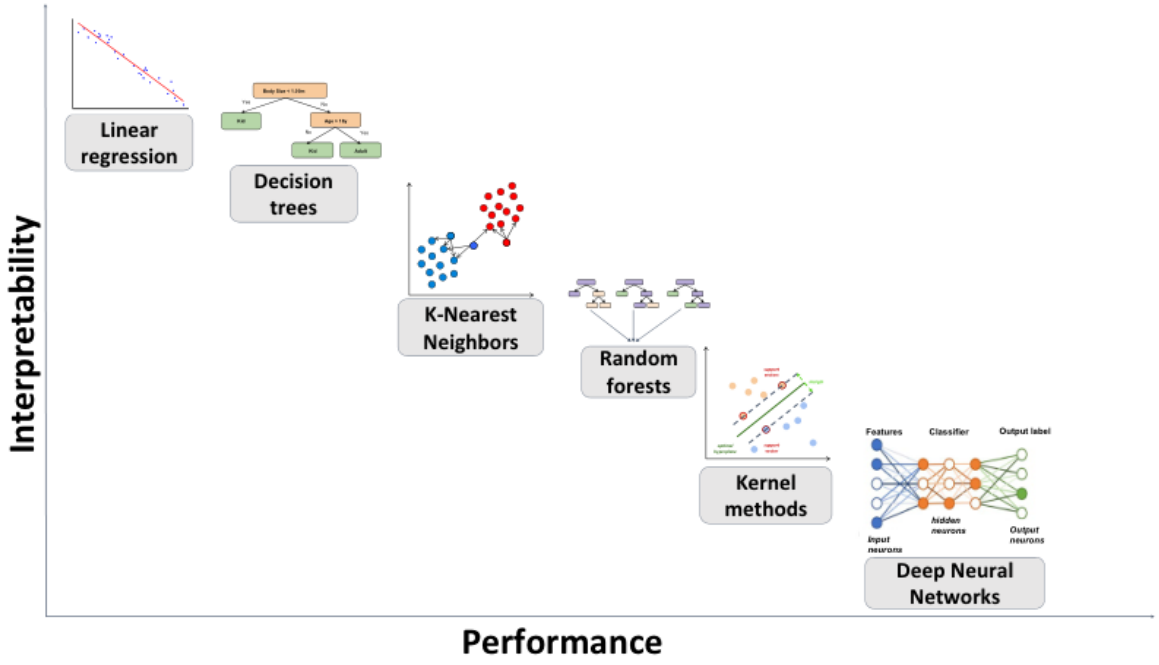

Figure 3.5: ML methods spanning from simpler and more interpretable approaches to more advanced algorithms [10]

3.5.5 Blackboxes

In Machine Learning, *black boxes* refer to models or algorithms whose internal workings are complex, intricate, not easily interpretable by humans or not interpretable at all. These models can produce accurate predictions or classifications, but their decision-making process is not transparent or easily understandable. Paradoxically, as we've seen in the Figure 3.5, the better is the predictive accuracy of a machine learning model, the more opaque it is. A missing step in the construction of an ML model is precisely the explanation of its logic, expressed in a comprehensible, human-readable format, that highlights the biases learned by the model, allowing to understand and validate its decision rationale. This limitation impacts not only information ethics, but also accountability, safety and industrial liability [13]. As a matter of fact it is considered to exist three different forms of opacity [14]:

1. Opacity as intentional corporate or institutional concealment of their algorithms in general, including those of ML.

2. Opacity as technological illiteracy that prevents society from understanding a field as specialized as that of computer programming.

3. Opacity in the sense presented so far.

This underlines the fact that the unexplainability of black boxes can therefore purposely used for unethical and unsafe scopes.

However, it's important to note that the black box nature of AI can present both challenges and opportunities regarding IP protection. While it may make it harder to understand and protect certain aspects of AI models, it can also offer advantages in terms of safeguarding proprietary information and maintaining a competitive edge as Trade secret law can be exploited effectively when AI models are black boxes.

From now on we will be focusing exclusively to the machine learning model called Artificial Neural Network (ANN) in order to pursue the objective of this study.

3.6 Artificial Neural Networks

3.6.1 Introduction

An Artificial Neural Network (ANN) is a mathematical model that draw inspiration from the structure and functioning of the human brain, in fact researches on ANN were based on the idea that by replicating biological neural networks it was possible for machines to acquire intelligence.

In order to do so the ANN tries to emulate the functionality of biological neurons by replacing it with an artificial neuron, a simple mathematical model which constitutes the basic building block for every artificial neural network. These units are nodes interconnected by links, each with an associated weight and follow three primary sets of regulations: multiplication, accumulation, and activation.

As data enters the artificial neuron each input value is multiplied by an individual weight. Moving into the middle portion of the artificial neuron, a summation function calculates the combined value of all weighted inputs and incorporates a bias term.

Exiting the artificial neuron, the total of these weighted inputs along with the bias proceeds through an activation function, often referred to as a transfer function.

3.6.2 Artificial neuron and biological neuron

Artificial neurons aim is to mimic some of the design and functionality of the biological neuron in order to get a similar result. Let's observe some similarities:

- The cell body, called *soma*, corresponds to nodes.
- *Dendrites* are the input links.
- *Axons* are the output links.
- *Synapses* are like the weights.

Electrical impulses conveying information from neighboring neurons are received by the dendrites at synapses, which are connection points. This information is then passed from the dendrites to the soma, which processes it. Once this process is completed the output signal, in the form of impulses, travels along the axon to reach other neurons [15].

Similarly, in ANN input links convey signals received from other neurons to the node. According to the weight associated to the link, the transported signal are reduced or amplified. When it reaches the node, the information is processed through a summer and then it's passed to an activation function which embeds a threshold unit that determines

whether the signal can go further; If that's the case, the signal is propagated to the output links to reach other neurons.

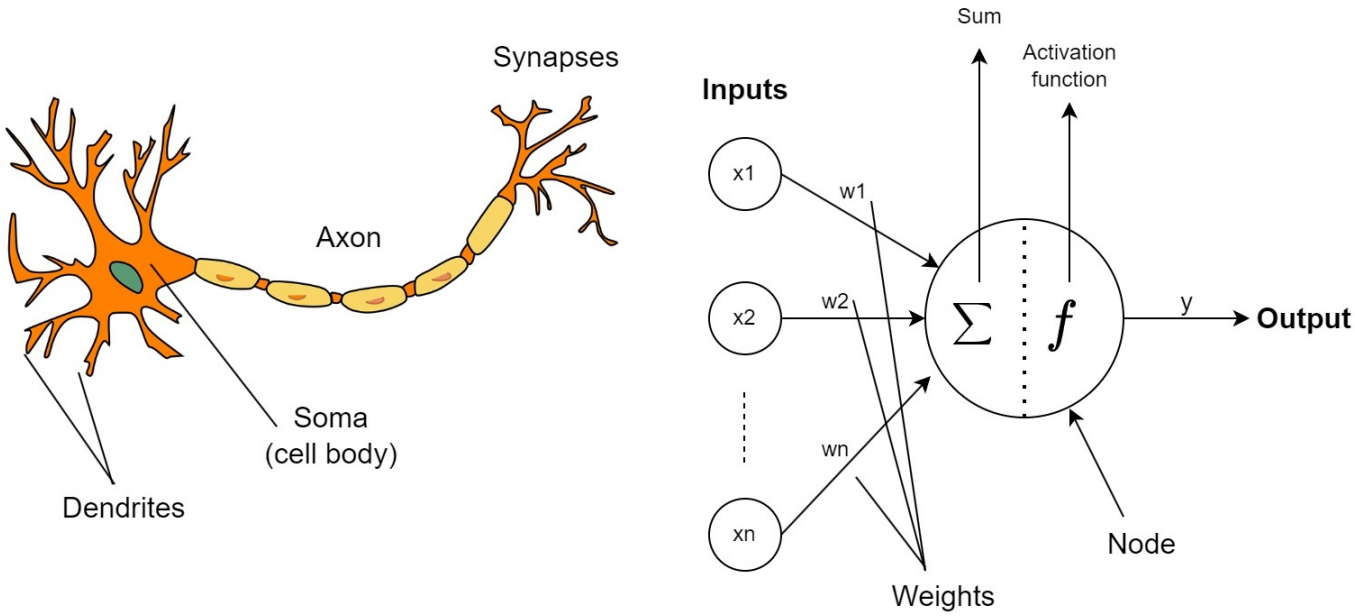

Figure 3.6: Biological neuron vs artificial neuron

The formula to calculate the output, $y(k)$, of an artificial neuron with an activation function f can be represented as follows:

$$y(k) = f\left(\sum_{i=1}^{m} w_i(k) * x_i(k) + b\right)$$

Where:

- $y(k)$ is the output of the neuron.
- k represents the discrete time, so it is an index of a specific time step within a sequence of data points.
- f is the chosen activation function.
- $w_i(k)$ is the weight value at in discrete time k where i goes from 0 to m.
- $x_i(k)$ is the input value at in discrete time k where i goes from 0 to m.
- b is the bias.

The selection of the activation function f for an artificial neuron depends on the nature of the problem it aims to solve. Typically, the activation function is chosen from a set of options, the most common are the *Step function, Linear function, and Non-linear (Sigmoid) function* [16].

3.6.3 Topology of Artificial Neural Networks

An ANN is made up of arrangement and connections of the artificial neurons. Every network is composed of layers split in three main parts: the first part is a single layer called *input layer* and is the only one to be exposed to external signals.

The second part consists of a range hidden layers (that can also be 0), that extracts relevant features or patterns from the received signals.

Finally, the last part is the output layer which is the final layer of the network.

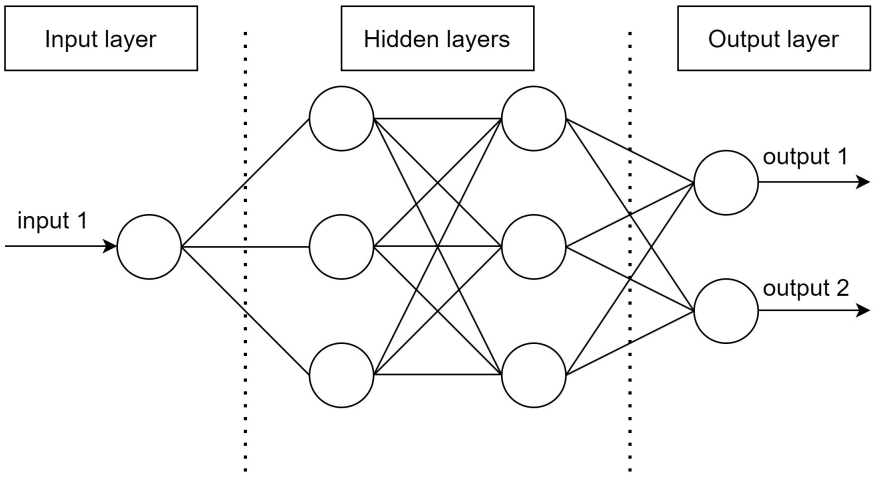

Figure 3.7: Example of Artificial Neural Network

The arrangement of individual artificial neurons and their connections patterns is referred to as the topology or graph of an artificial neural network. The range of possibilities in how these connections are established leads to a variety of potential topologies, which can be categorized into two fundamental classes:

1. Feedforward topology (FNN)
2. Recurrent topology (RNN)

The core difference between these two is that in the first (feedforward) the graph is acyclic meaning it has no loops, therefore information goes exclusively from input to output, while the latter (recurrent) allows to have loops as it's a semi-cyclic graph which means that information can also go in the opposite direction.

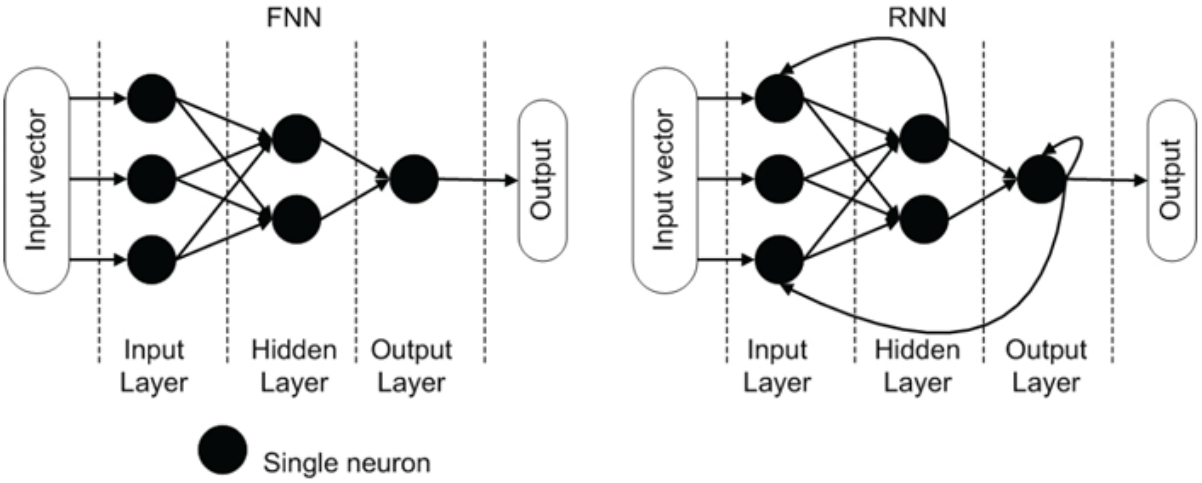

Figure 3.8: Feed-forward and recurrent topologies [16]

3.6.4 Learning methods

The process of neural network learning involves iteratively modifying its weights and biases (referred to as free parameters) to achieve the intended output. This adjustment occurs through training, where the network is exposed to a specific set of guidelines, often referred to as the learning algorithm. The most common training algorithm are:

1. Gradient Descent Algorithm
2. Back Propagation Algorithm

Gradient Descent is an algorithm used in Artificial Neural Networks to train the network by adjusting its parameters (weights and biases) in case the actual output is different from the target output or *expected value*. By doing so it is possible to guide the network towards making better predictions and minimize mistakes. Here's a brief explanation of the different steps:

1. The network's performance is measured using a loss function, which quantifies how far off its predictions are from the actual target values.
2. Gradient Descent calculates the gradient of the loss function with respect to the network's parameters. The gradient points in the direction of the steepest increase in the loss.
3. The algorithm adjusts the parameters in the opposite direction of the gradient. This means moving the parameters in a way that reduces the loss.

4. The size of each adjustment is controlled by a parameter called the learning rate. A smaller learning rate takes smaller steps, while a larger one takes bigger steps.

5. The process is repeated iteratively. In each iteration, the parameters are adjusted based on the gradient and learning rate.

6. The process continues until a stopping criteria is met, for example a fixed number of iterations.

Backpropagation algorithm utilizes the methods of mean square error and gradient descent to realize the modification to the connection weight of network. After identifying an error (the gap between the desired and actual outcomes), this error is then sent backward through the network, moving from the output layer to the input layer via the hidden layers [17]. Backpropagation process can be described in this way:

1. Forward propagation of operating signal: the input signal progresses through the network, starting from the input layer, passing through the hidden layer, and finally reaching the output layer. Throughout this phase, the weight and offset values of the network remain unchanged, and the state of each neuron layer only impacts that of the subsequent layer. In case that the expected output can not be achieved in the output layer, it can be switched into the back propagation of error signal.

2. Back propagation of error signal: the process involves defining the error signal as the discrepancy between the actual output and the expected output of the network. This error signal is then transmitted in a layer-by-layer manner, originating from the output end and extending to the input layer. As this error signal travels backward, it influences the network's weight values. Through continuous adjustments of the weight and offset values, the network aims to cover the gap between its actual output and the desired output [18].

Chapter 4

Functioning of Artificial Neural Networks

The journey of developing an effective Artificial Neural Networks encompasses a complex procedure, which doesn't only include the training phase, but also the process of taking key decisions that shape the neural network's architecture, performance, and ultimate efficacy. This chapter is going to be explore the process of creating a working ANN by breaking it down in a number of steps which will lead to a trained and tested network. Keep in mind that these steps are meant to be general as they can vary depending on the problem and tools used, that's why it will be also covered how the problem is defined and the choices made within each step.

As it was stated, the primary goal of this preliminary study is to explore a strategy for safeguarding the intellectual property associated with a network trained using supervised learning algorithms.

The strategy for protecting AI systems that we have devised is based on the idea that intellectual property may reside in the training process of the network, and thus, in the components that describe that process. For this reason, an analysis will be conducted to identify those key components that should be protected during both the training and the overall development process.

4.1 Key steps to developing a Neural Network

As previously highlighted, there isn't a universally standardized approach to distinctly define every phase of designing and developing ANNs, however it is possible to list the most common and used steps to do so, which include different methods and techniques. Here's a quick break down of the main phases:

- **Defining the problem:** Clearly state the problem you want the ANN to solve.
- **Dataset preparation:** Gather and clean a relevant dataset.
- **Setting network architecture:** Decide on the network's topology as well as its structure which include layers and number of neurons.
- **Training**: This is the process of feeding the ANN with data and adjusting its weights and biases until it learns to perform the desired task.
- **Testing**: This is the phase in which the ANN performance is evaluated to determine how well it generalizes new data.

Let's now delve into each step in order to analyze it.

4.2 Defining the problem

When working on an ANN's development, it's crucial to precisely outline the task the network will undertake, articulating it with accuracy for several reasons. Firstly a well-defined problem is important to work as a guideline as it sets a clear direction for the entirety of the project as well as helping to avoid ambiguity and confusion.

Furthermore, having a clear scope of the problem to deal with, ensures that the network's architecture, training data, and evaluation metrics align precisely with the intended task. The nature of the problem influences the type of ANN architecture, activation functions, and optimization algorithms to be employed. That's why, as it is shown above (3.5.3), having a strong grasp of the problem and being capable to breaking it down into suitable elements is fundamental in order to avoid a biased developing of the model.

Defining the problem help to evaluate the type and amount of data needed for the training and testing phases is essential to gather relevant data and creating representative datasets in the later stages of development. For example in the research carried out in the paper *Automatic Speech Classifier for Mild Cognitive Impairment and Early Dementia*[19] authors propose to use the autoencoders for the detection of cognitive decline,

however typically it is necessary to have large training datasets for the performance of deep learning models. In this specific case study the context didn't allow to meet the requested amount of data, therefore it was adopted a data augmentation approach to enlarge the size of the input dataset to have better results.

This underlines the importance of comprehending not only the importance of the individual problem, but also its contextual surroundings, along with available the tools to address it.

4.3 Dataset preparation

A dataset is a structured collection of data that represents a particular set of observations, often related to a specific problem or domain. It's used for training and testing phases so it is important to consider some fundamental factors when choosing it as the purpose, size, quality, format and availability. Some common types of datasets used to develop ANNs through supervised learning algorithms include:

- Image datasets: contain images, which can be used for tasks such as image classification and object detection.
- Text datasets: contain text, which can be used for tasks such as natural language processing and sentiment analysis.
- Audio datasets: contain audio, which can be used for tasks such as speech recognition and music classification.
- Video datasets: contain videos, which can be used for tasks such as video classification and object tracking
- Time series datasets datasets: contain time series data, which can be used for tasks such as forecasting and trend analysis. Evaluation helps prevent overfitting (3.5.2) and guides the training process. It can considered as an integral part of the training phase which make sure that the model doesn't memorize the pattern that links input data and the target value.
- Sensor datasets: These datasets contain sensor data, which can be used for tasks such as environmental monitoring and health monitoring,

In order to be prepared, a dataset must be gathered first. Collecting data can be done from different sources for example public datasets, private datasets and data generated by sensors, keeping in mind that it should be as large and diverse as possible to help the neural network to learn a variety of patterns and to generalize to new situations.

A dataset can contain errors, missing values, outliers, and inconsistencies. In order to fix these issues it is necessary to go through a cleaning process that ensures the dataset reliability.

Preparing the data for modeling is done by preprocessing it, which consists in standardizing formats, scaling numerical values, encoding categorical variables, and handling missing values. As we've seen in the previous paragraph, in some contexts datasets may be not large enough to efficiently train a model, therefore, if applicable, it's possible to use the data augmentation technique to increase the training dataset through slightly modified copies of existing data or newly created synthetic data [19].

It is fundamental for a ANN developed through a supervised learning algorithm to have data correctly and fully labeled to ensure a better training process. The label can be for example a category, a value, or a timestamp.

Finally the dataset is typically splitted into three main parts:

- Training Set: it constitutes the largest part of the dataset as it is used to train the ANNs parameters. As it is mentioned before it contains the input data and corresponding known target values to make the ANNs grasp patterns and relationships between the two.

- Validation Set: it is a smaller part of the dataset used to fine-tune hyperparameters and evaluate the model's performance during the training phase.

- Testing Set: After the model is fully trained, the ANN is evaluated on a completely separate and unseen part of the dataset called testing dataset that it hasn't encountered during training or validation. In this way it's possible to assess the ANN performance on new cases.

The split ratios for training, validation, and testing sets are arbitrary, but most of the times the biggest slice of data is used for training while the smaller slices are used to validate and test. These ratios can vary based on the size of the dataset and the specific problem. It's important to ensure that the data in each subset is representative of the overall distribution. Randomization is often used to prevent bias and ensure a fair representation. Dataset splitting helps assess the ANN's ability to generalize to unseen data and aids in selecting optimal hyperparameters and architecture during training.

A data set can be public or private, however, due to the nature of our proposal this does not matter as we intend to protect a set of components together, and, given the impact of dataset on the ANN outcome, it is crucial to take it into consideration.

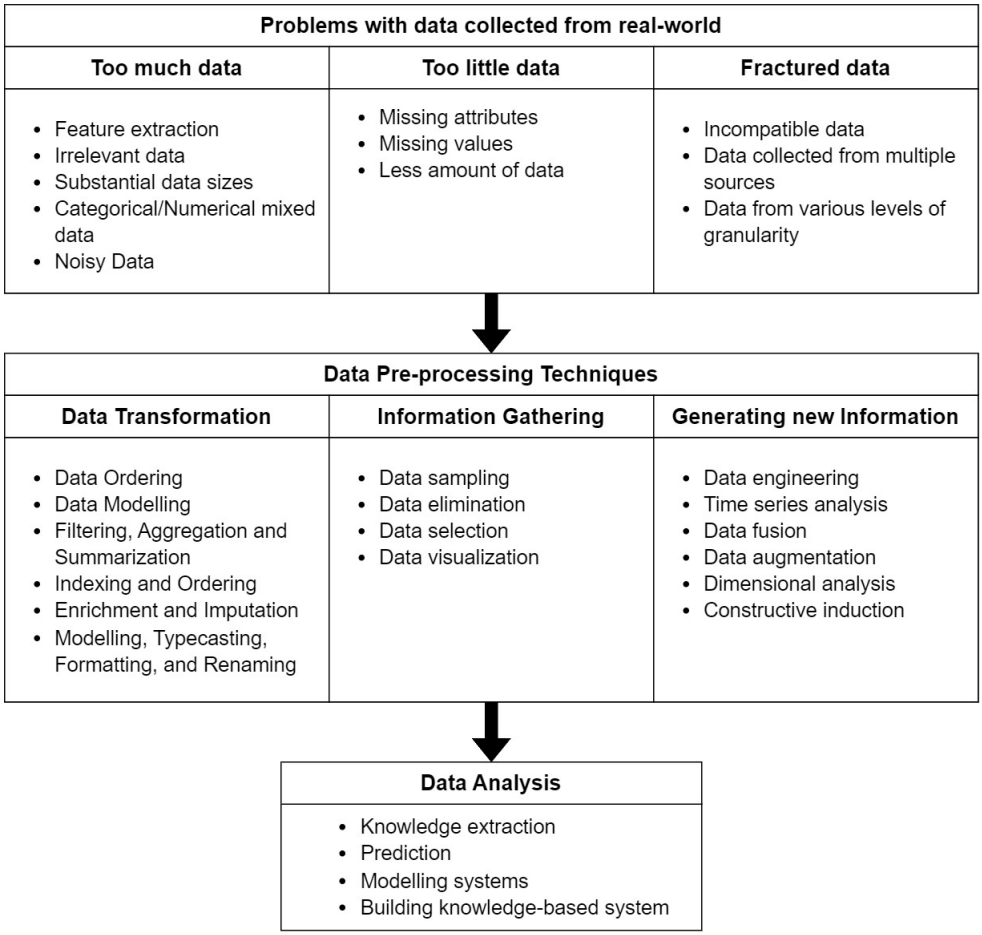

Figure 4.1: Some possible problems with the data [20]

4.4 Setting network architecture

As it is already mentioned, for network architecture we intend the arrangement of the artificial neurons, their number and their connection patterns. Therefore setting an ANN architecture consists in determining these parameters:

- Number of neurons in each layer
- Number of layers (single-layer, multi-layer)
- Topology (feedforward, recurrent)

These choices are strictly bound on the type of problem we're dealing with and the quantity of available data for the training phase. Some of the most common types of artificial neural network based on their architecture and sizing are:

- Single-Layer Perceptron: is a simple feedforward neural network architecture composed by a single layer. The perceptron model is primarily used for binary classification tasks where the decision boundary is linearly separable.
- Multi-Layer Perceptron: MLPs are a type of multi-layer feedforward neural network which can be include one or more hidden layers. They can handle complex non-linear relationships and are commonly used for various tasks such as classification, regression, and function approximation.
- Convolutional Neural Network: These multi-layer feedforward neural networks are used mostly for computer vision tasks, such as image classification, object detection, and image recognition. They do this by exploiting the spatial structure of the data, which means that they can learn to identify patterns that are related to each other in space.
- Long Short-Term Memory Network: LSTM networks are a type of recurrent neural networks which can either be single-layer or multi-layer. They are a special type of RNNs which is capable of learning long-term dependencies specifically designed to address the vanishing gradient problem, which is an issue that makes it difficult for the network to learn. Single layer LTSMs are simpler and easier to train but obviously cannot be as complex as a multi-layer LTSMs which can learn more complex pattern but are harder to train. Typical tasks carried out by LTSM are NLP speech and recognition.
- Gated Recurrent Unit Network: GRU networks are similar to LTSMs, in fact they are recurrent artificial neural networks that can either be single-layer or multi-layer. Just like LTSMs they mitigate the vanishing gradient problem but they implement a simpler internal structure and design.

The architecture of an ANN strongly impacts its ability to learn. A *shallow* neural network (with a simpler architecture) may be more suitable for less complex tasks, while a *deep* neural network (with a more sophisticated architecture) might be the solution for carrying out more complex tasks. In any case, the ANN's architecture is one of the core components that strongly affects training, and thus the weights; this means that it strongly affects how the ANN perform its instructions, and therefore needs to be taken into account for intellectual property protection.

4.5 Training

The training phase of Artificial Neural Network development plays a core role in enabling networks to learn and make accurate predictions. During the training phase, the ANN

undergoes an iterative process, gradually refining its parameters to minimize the discrepancy between its predictions and the true outputs. Before getting into the training phase it is necessary to set the network's parameters that will collectively define the behavior and performance of the ANN.

Typically, the main network parameters are the following:

- Initial weights: as we've seen they are numerical values associated with the connections between artificial neurons that will be adjusted during training. The choice of weight initialization method depends on various factors, including the network architecture, the activation functions used, and the specific problem being addressed. Some common methods are simple random initialization or Xavier Glorot initialization, in any case most of the time experimentation and empirical evaluation are necessary to determine suitable weights for a specific network.

- Biases: As we've seen above (3.6.2) biases are a network parameter independent of the input data helping the network to grasp complex patterns and generalize data more efficiently. They are scalar values, each associated to a specific layer and affect every neuron in that layer. They condition ANNs training by determining how easily a neuron are activated based on its inputs: A higher bias value makes it more likely that a neuron will activate (produce an output) even if the weighted inputs are not very large. Conversely, a lower bias value makes it less probable that a neuron will activate, requiring the weighted inputs to be relatively larger for activation to occur.

- Activation Functions: these functions introduce non-linearities to the network by transforming the weighted sum of inputs from the previous layer. They determine the output or activation level of a neuron. The choice of an activation function is made based on the nature of the problem, network architecture (deep or shallow), the characteristics of the data and the computational efficiency. As for others parameters, it's often necessary to experiment different activation functions to define which one works better for a specific ANN.

In the following subsections it'll be shown the above-mentioned example of an ANN training which will be exemplified enough to grasp the main concepts necessary to make some considerations. The network that will be developed is a classifier which determines whether a given image is a cat or a dog.

4.5.1 The neural network

Let's suppose our artificial neural network is multi-layer and feedforward and it is composed by:

- An input layer with a single node i.
- A hidden layer with two nodes, respectively j_1, j_2.
- An output layer with two nodes, respectively o_1, o_2.
- $w_1, ..., w_n$ are the weight associated with connections.
- b_1, b_2 are the biases.
- The outputs 0 and 1 to were chosen to represent cats and dogs, respectively.

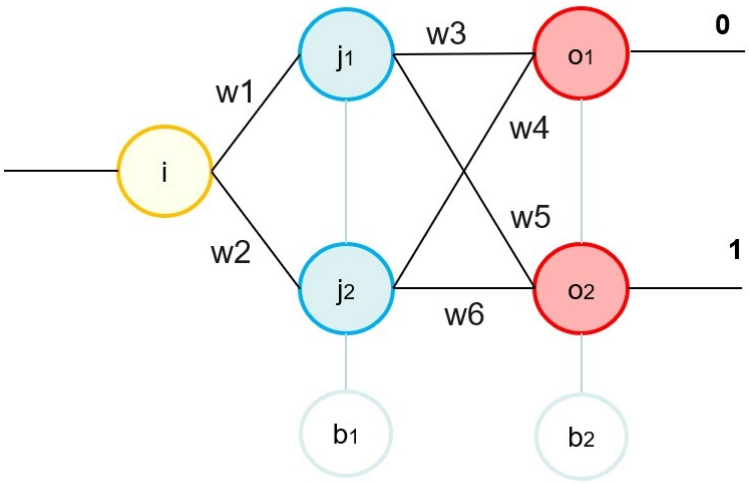

Figure 4.2: Initial network

4.5.2 A result example

The objective of this training is to attain a specific outcome. When presented with an input image depicting either a dog or a cat, the artificial neural network should have the capability to generate an output in the form of a prediction. This prediction serves the purpose of indicating whether the image contains a dog or a cat, thus demonstrating the network's ability to discern between the two categories.

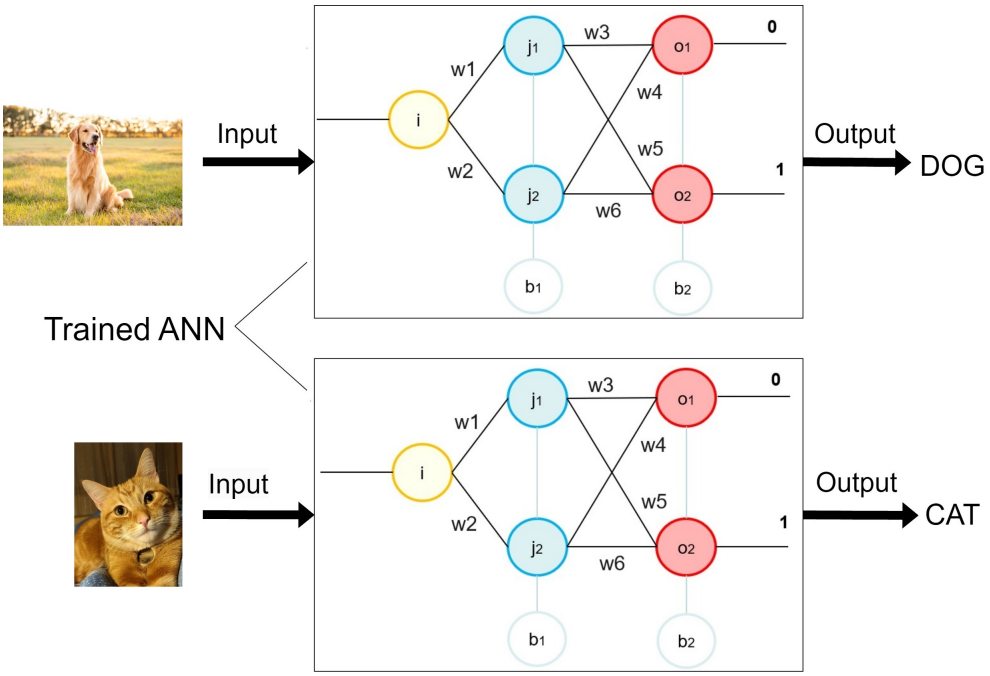

Figure 4.3: An example of the expected result

4.5.3 Training process

The selected method for conducting the training phase will be the Backpropagation algorithm (explained in section 3.6.4). Let's proceed by enumerating and providing explanations for each step:

1. **Weights and biases initialization:** In order to set the initial values for these parameters, we will adopt a basic random initialization approach.

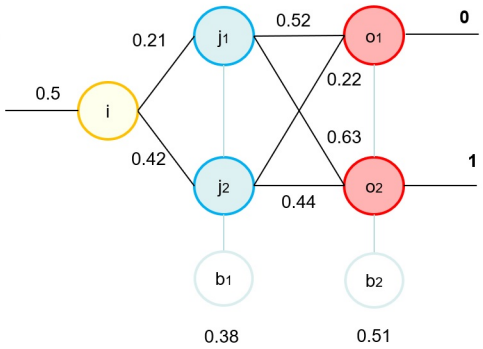

Figure 4.4: The initialized parameters in the ANN

2. **Forward propagation:** During this phase the input signal progresses through the network starting from the input layer, then passes through the hidden layer, and finally reaches the output layer.

   In this process we calculate the value of each node while the weights and biases remain unchanged, as shown in 3.6.2 the formula includes a summation which is going to be the argument for the chosen activation function. Let's go through summation of the $k-th$ node using the following generalization sequence:

$$j_k = a_1 * w_1 + a_2 * w_2 + ... + a_n * w_n + b_k$$

   For clarity let's remind that a represents the value propagated from the adjacent node, w denotes the weight of the traversed connection, and b signifies the bias.

   We will employ the sigmoid function as our chosen activation function, which is defined as follows:

$$f(j_k) = \frac{1}{(1 + e^{(-j_k)})}$$

   The computed result will be the propagated to the following adjacent nodes. Let's compute the value of j_1 and its propagated value through the activation function:

$$j_1 = 0.21 * 0.5 + 0.38 = 0.485$$

$$f(j_1) = \frac{1}{(1 + e^{(-0.485)})} = 0.619$$

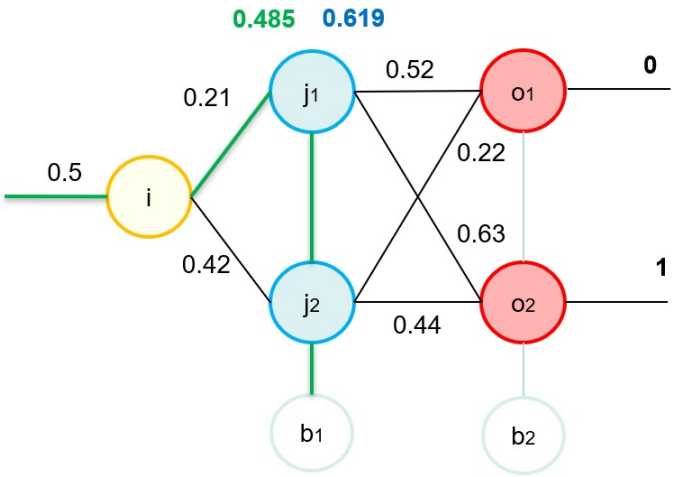

Figure 4.5: ANN with j_1 value computed. The green value is the result of the summation whilst in blue we find the propagated value computed through the activation function

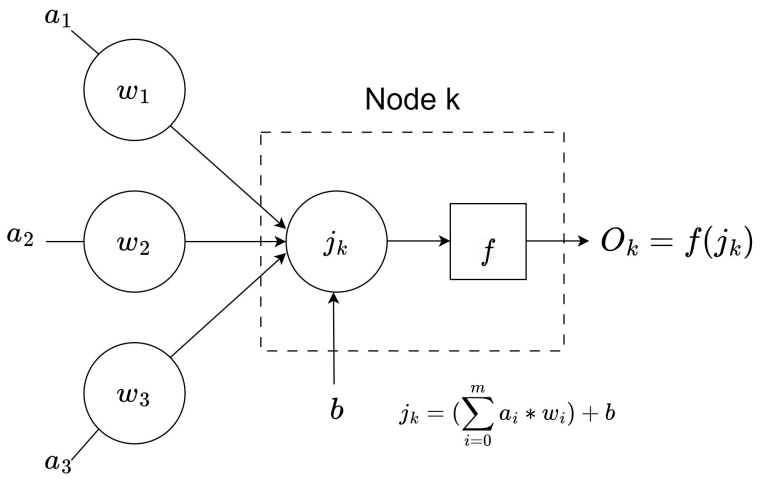

Figure 4.6: Inside the artificial neuron firstly summation is computed and then the activation function elaborate the result to produce an output

Within this process, the state of each neuron layer exclusively influences the subsequent layer's state. This iterative process traverses the entire artificial neural network, encompassing every node, until it reaches the output layer.

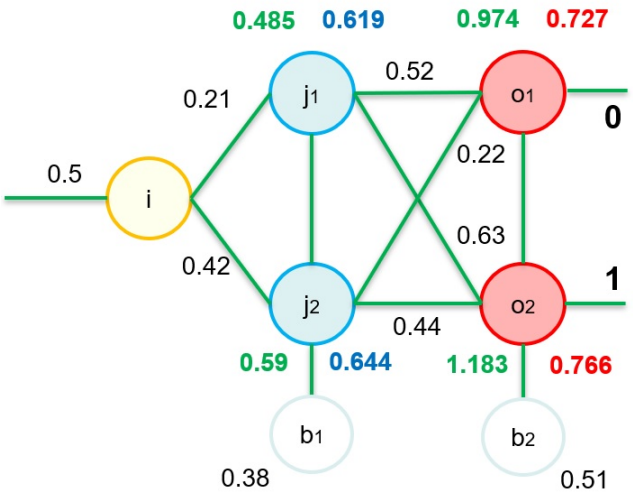

Figure 4.7: Every value is calculated until the output layer, the output value propagated is the red one

At this point the final output values, respectively 0.727 and 0.766 (colored in red in the figure 4.7), are going to be compared to the target values in order to check the discrepancy between the obtained value and the expected one.

Let's assume, for the sake of simplicity, that we've fed a dog picture into the network. Our desired outcome is for the propagated value from the upper output node in this network to be as close to 0 as possible, while the output value generated by the lower output node should be close to 1. Conversely, if the inserted picture was a cat, the ideal output for the upper node should be as close as possible to 1, and for the lower node, it should be close to 0. Note that this example has been based on the assumptions made earlier, namely that 1 corresponds to a dog and 0 to a cat.

3. **Backpropagation**: During this phase, we are going implement the process described in the previous example, which involves defining the error signal as the difference between the actual output and the expected output of the network. The error signal we're about to calculate will follow the reverse path of the feedforward phase. This means that it will propagate from the output layer, extending back to the input layer, moving in the opposite direction, and impacting the weight values of the network in order to adjust them. The aim of this process is to reduce the discrepancy between the output and the target value as much as possible.

   Let's go through each point of this phase, with the assumption that the fed image was a dog:

   (a) Calculation of the error: In order to calculate the error for each training example we're going to using a metric called Mean Squared Error (MSE). The formula for a single example is:

   $$E_k = \frac{1}{2}(output - target)^2$$

   Therefore our ANN errors are respectively:

   $$E_1 = \frac{1}{2}(0.727 - 0)^2 = 0.264$$

   $$E_2 = \frac{1}{2}(0.766 - 1)^2 = 0.027$$

   Now we want to calculate the total error by summing up the individual MSE values for all training examples to obtain the total MSE.

   $$E_{tot} = \sum_{i=1}^{n} E_n = E_1 + E_2 = 0.291$$

(b) **Backpropagation of Error:** It involves computing the error gradient for each layer in the network by propagating the error backward from the output layer to the input layer. The error gradient quantifies how much each parameter (weights and biases) in the network should be adjusted to minimize the chosen error metric (in this case, Mean Squared Error). So basically we're going to measure the impact of each weight on the total error starting from the output layer through the hidden layer and finally the input layer.
Let's start from w_6 (figure 4.2). Through the partial derivative of E_{tot} on w_6 (called the gradient) we calculate the impact of w_6 on E_{tot}. The formula, using the chain rule, is the following:

$$\frac{\delta E_{tot}}{\delta w_6} = \frac{\delta E_{tot}}{\delta out_{o2}} * \frac{\delta out_{o2}}{\delta in_{o2}} * \frac{\delta in_{o2}}{\delta w_6}$$

$$E_{tot} = E_1 + E_2 = \frac{1}{2}(out_{o1} - target)^2 + \frac{1}{2}(out_{o2} - target)^2$$

$$\frac{\delta E_{tot}}{\delta out_{o2}} = 0 + 2 * \frac{1}{2}(out_{o2} - target_{o2})^{2-1} * -1$$

$$\frac{\delta E_{tot}}{\delta out_{o2}} = out_{o2} - target_{o2} = 0.766 - 1 = -0.234$$

$$out_{o2} = \frac{1}{1 + e^{-in_{o2}}}$$

$$\frac{\delta out_{o2}}{\delta in_{o2}} = out_{o2}(1 - out_{o2}) = 0.766(1 - 766) = 0.179$$

$$in_{o2} = w_8 * out_{j2} + w_7 * out_{j1} + b_2$$

$$\frac{\delta in_{o2}}{\delta w_8} = out_{j2} = 0.644$$

$$\frac{\delta E_{tot}}{\delta w_6} = -0.234 * 0.179 * 0.644 = -0.027$$

The final updated w_6 value is:

$$w_{6updated} = w_6 - \epsilon \frac{\delta E_{tot}}{\delta w_6}$$

We will take as learning rate value 0.5 for simplicity:

$$w_{6updated} = 0.44 - 0.5 * -0.027 = 0.427$$

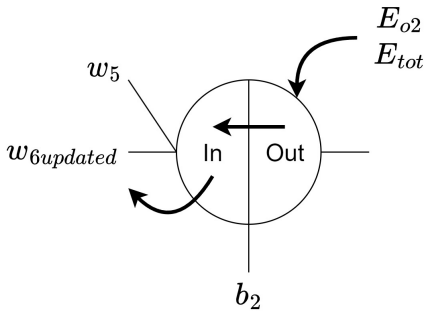

Figure 4.8: Visual representation of this phase

To update the biases associated with the layer, we would use a similar formula:

$$b_{2updated} = b_2 - \epsilon \frac{\delta E_{tot}}{\delta b_2}$$

Using the chain rule:

$$\frac{\delta E_{tot}}{\delta b_2} = \frac{\delta E_{tot}}{\delta a_2} \cdot \frac{\delta a_2}{\delta b_2}$$

Applied on the initial formula:

$$b_{2updated} = b_2 - \epsilon \cdot \frac{\delta E_{tot}}{\delta b_2} = b_2 - \epsilon \cdot \left(\frac{\delta E_{tot}}{\delta a_2} \cdot \frac{\delta a_2}{\delta b_2} \right)$$

Where a_2 represents the output of the neuron in the output layer. It's the result of applying the sigmoid activation function to the weighted sum of inputs and the bias associated with that neuron.
In mathematical terms:

$$a_2 = \sigma(weighted\ sum + b_2)$$

Weighted sum represents the sum of the weighted inputs coming into the neuron from the previous layer and σ is the activation function (sigmoid in this case).

These calculations will be repeated for every weight and bias, throughout the entire artificial neural network. It's important to note that this parameter adjustment process is performed multiple times until the difference between the predicted output and the target value is considered acceptable. Consequently, this process will cycle back to the forward propagation phase as needed.

4.5.4 Validation

Validation during training involves using a separate dataset to assess and monitor the model's performance, guide hyperparameter tuning, and potentially trigger early stopping to optimize the model's generalization ability. It is a critical step in the development of a robust and effective neural network model. The key aspects of validation during training are as follows:

- Validation Dataset: A portion of the available data is set aside and designated as the validation dataset. This dataset is not used for training the model. A portion of the available data is set aside and designated as the validation dataset. This dataset is not used for training the model.

- Periodic Evaluation: After each training epoch (a complete pass through the training data), the model's performance is evaluated using the validation dataset. The model's predictions are compared to the known target values in the validation dataset to compute evaluation metrics, such as accuracy, loss, or other relevant metrics.

- Performance Monitoring: The evaluation metrics obtained from the validation dataset provide insights into how well the model is learning during training. It helps to monitor the model's ability to generalize to unseen data and identify potential issues like overfitting (when the model performs well on training data but poorly on new data).

- Hyperparameter Tuning: Validation data is also used for hyperparameter tuning. Hyperparameters are settings that are not learned by the model but are set prior to training, such as the learning rate or batch size. By systematically adjusting hyperparameters and evaluating the model's performance on the validation data, data scientists can identify the hyperparameters that result in the best model performance.

- Early Stopping: Validation data plays a crucial role in early stopping. If the model's performance on the validation dataset begins to degrade (e.g., loss increases or accuracy decreases), while the performance on the training dataset continues to improve, it may indicate overfitting. Early stopping involves halting the training process when such degradation is detected on the validation data. This helps prevent the model from overfitting and ensures that the model generalizes well to new, unseen data.

4.5.5 Cross-Validation

One of the most common types of validation techniques is known as *cross-validation*. It involves partitioning the training dataset into several subsets referred to as *folds*. In the training procedure, the model is trained using all *folds* except one, which serves as the *validation set*. This procedure is iterated multiple times, with each fold taking turns as the *validation set* while the remaining folds are used for training.

The average performance of the model across the various folds serves as an estimate of how well it would perform on unseen data. This approach offers a more reliable estimation of the model's performance compared to using the entire training dataset as the *validation set*, as it reduces the risk of *overfitting*.

For instance, consider having a training dataset with 100 examples. You could partition this dataset into 4 folds, with each fold containing 25 examples. During each iteration, the model would be trained on 75 examples and evaluation on 25 examples. This process would be reiterated 4 times, utilizing a distinct fold as the *validation set* in each iteration. The mean performance of the model across these 4 folds would then serve as an approximation of its performance when faced with unseen data.

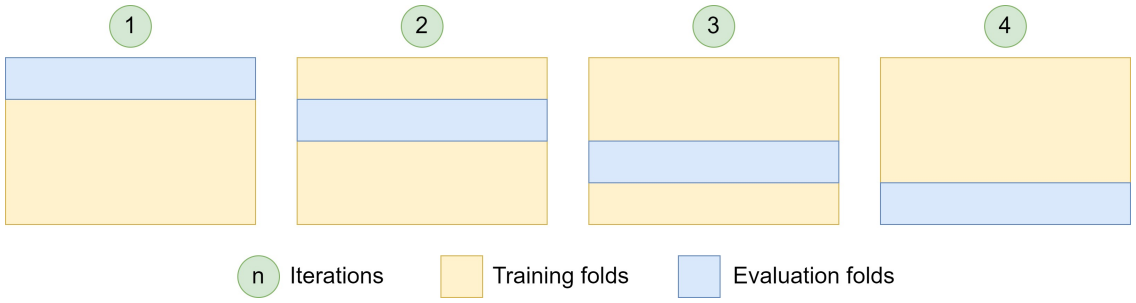

Figure 4.9: An example of cross-validation

4.6 Testing

The testing phase serves the purpose of assessing the model's ability to generalize to new data. This evaluation is done using a dedicated subset of the dataset called *test set*. The *test set* consists of a set of data that the model has never seen during its training or validation stages. This ensures that the model has no prior knowledge of the test data and cannot rely on memorization.

The primary objective of the *test set* is to provide objective estimation of the model's performance and if all the testing metrics meet the desired criteria, the neural network is ready to proceed to the deployment phase. It's important to note that the outcomes of testing are highly dependent on the specific problem being addressed, and what might be considered acceptable performance for one application might be insufficient for another.

The selection of appropriate validation methods depends on the type of application[21]:

- Approximation testing methods: These methods are used to evaluate the accuracy of a model's predictions. They do this by comparing the model's predictions to the actual values. Some common approximation testing methods include mean squared error (MSE), root mean squared error (RMSE), and mean absolute error (MAE).
- Classification testing methods: These methods are used to evaluate the accuracy of a model's predictions for categorical variables. They do this by comparing the model's predictions to the actual labels. Some common classification testing methods include accuracy, precision, recall, and F1 score.
- Forecasting testing methods: These methods are used to evaluate the accuracy of a model's predictions for future values. They do this by comparing the model's predictions to the actual values. Some common forecasting testing methods include mean absolute percentage error (MAPE), Theil's U statistic, and the mean squared error (MSE).

4.7 Considerations

The result of the process explained in the previous section is none other than a function capable of taking in an input, which in this case is an image, and processing it to produce an output, that in this case is the prediction, much like a Turing Machine.

In a Turing Machine, the process of computation is defined by a source code, which consists of a sequence of instructions that guide the machine's behavior. This code is a fundamental and protectable element.

On the other hand, in neural networks, the source code does not explicitly defines the operation that the ANN do to perform a specific task, rather it describes the setup of a neural network. This is because an ANN functioning is what we described as a *black box*, therefore its operation and functioning are difficult to interpret, if not impossible.

How could be possible to protect an ANN just like Turing Machine instructions are protected through their source code?

It is important to note that the ability of a neural network to make predictions or decisions is the result of autonomously applying certain instructions by the network when facing generalized cases. These instructions are not explicitly encoded but emerge through the process of learning from data, the outcome of which is a set of weights that, on their own, are merely numbers. However, when considered in conjunction with other elements, they describe the functioning of the network and, consequently, its instructions. Therefore, one could argue that protecting the key components contributing to this learning process, and its outcome (i.e., the weights), means safeguarding the instructions that the neural network performs to compute the learned function, much like how source code protects the instructions carried out by a Turing Machine.

On the basis of these assumptions, the conceived proposal is to protect the following key elements that could constitute intellectual property of data-driven AI systems:

- Data set
- Network architecture
- Technical implementation
- Weights

The next chapter of this thesis will delve more into this proposal in order to explain how those key components together would make it possible to protect an ANN and why this idea seems to be more effective than what is actually protected by IP laws in the state of the art in Italy. In addition to this, this final chapter is also delving into different jurisdictions in order to have an idea of how IP for AI systems is conceived in some other countries of the world.

Chapter 5

Legal protection of Artificial Intelligence

In the previous chapter, the functioning of an artificial neural network trained using a supervised learning algorithm was explained in detail. From this description it is observable how the result of this process is essentially a function capable of processing an input and producing an output, like the operation of a Turing Machine. However, in the context of ANNs and AI data-based systems in general, the absence of "traditional" source code that prescribes their instructions and functionality results in a lack of a protectable and tangible component for safeguarding their intellectual property.

As of today, both in Italy and worldwide, there is no legal framework that addresses the protection of artificial neural networks by specifically distinguishing the elements that have to be protected. In the following chapter, a more in-depth explanation will be provided for the proposal made to protect ANNs, which, as previously anticipated, consists of the collective protection of four core elements. These elements are believed to adequately represent the instructions and, consequently, the functioning of the network, thus they should be protected through intellectual property rights to ensure legal safeguards for these systems.

5.1 Artificial neural networks and Turing Machines: a comparison

Like it was stated in 4.7, both artificial neural networks and Turing Machines can be viewed as systems that perform computations by processing input and generating an output, but they do so quite differently. In the case of ANNs the input might be an image, a text, an audio and anything that can be represented as a numerical vector. On the other hand a Turing Machine can take as an input anything which can be represented as a string of symbols from the Turing Machine's tape alphabet.

In both cases they process this data in order to produce the output. Let's see a visual comparison of an ANN that classify images of cats and dogs, and a Turing Machine which calculates a second degree polynomial with coefficients equal to 1:

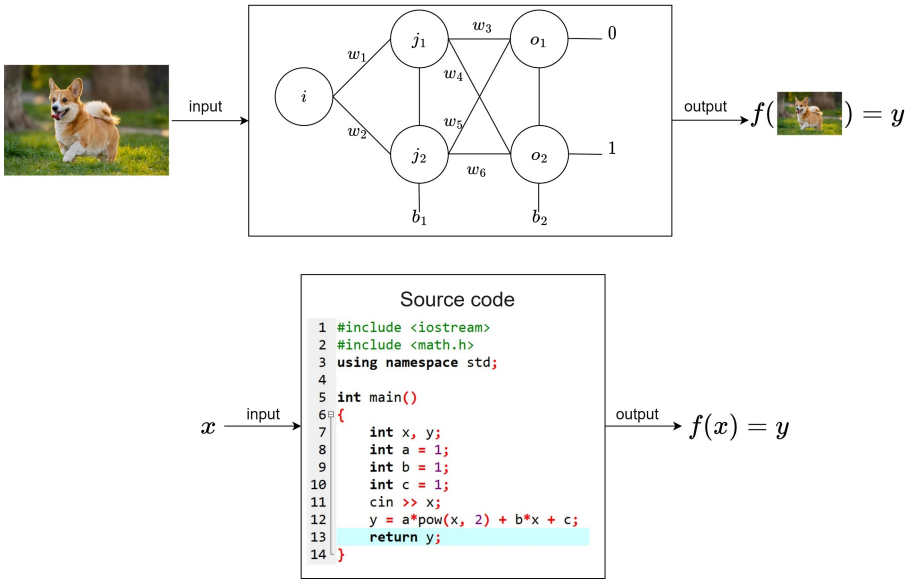

Figure 5.1: Comparison between a dog&cat classifier and a Turing Machine

As we can see from figure 5.1, fundamentally, an ANN produce mathematical functions like Turing Machines, but obviously in a different way. This conceptual similarity is important because it allows us to understand that a neural network can perform calculations and take actions on data much like a Turing Machine would. However it's important to note a key distinction previously discussed: the Turing Machine has a tangible **source code** which represent a detailed description of the transition rules and instructions that the machine follows to perform computations, so basically its **behaviour**.

In the case of ANN, the source code is quite different as it does not **explicitly** describes how the model operates following a well-defined set of instructions, instead instructions are learned automatically by the network during the training process, without

being manually coded in the source code. This is exactly why the objective of this study is to develop a standardized method for accurately representing the functioning of these models. This representation can then be protected in a manner similar to how existing laws protect the instructions of a Turing Machine, which are embedded in source code.

5.2 ANN intellectual property

In order to address the legal protection of neural network models, it is necessary to grasp a clear understanding of what should specifically protected by the *intellectual property* in ANNs in order to implement adequate measures to safeguard against unauthorized access and use.

The idea proposed in this thesis involves grouping these components together as a unified entity that is eligible for intellectual property rights protection, in order to legally safeguard and maintain the originality and value of an artificial neural network:

- Dataset
- Network architecture
- Technical implementation
- Weights

Let's go through all of them to justify this choice.

5.2.1 Network architecture

By network architecture it is meant:

- The number of layers and the number of nodes per layer.
- The topology as intended in the point 3.6.3.

The network architecture determines the model's capacity to represent and approximate complex input-output mappings. An appropriate architecture can help prevent *overfitting*, where the model becomes too specialized in the training data and performs poorly on unseen data. Furthemore, the choice of network architecture can significantly impact computational efficiency during training as it influences *hyperparameter tuning*. Also the architecture is designed to work for a specific task. This means that different network architectures are made to handle specific types of supervised learning tasks.

Keep in mind that protecting the network architecture this doesn't mean that, for example, we're forbidding to use convolutional neural networks with k nodes and n layers, but we're trying to seek legal protection by not allowing to use a certain neural network architecture combined to other components in order to carry out our very same function.

5.2.2 Data set

The data set used to train, validate and test an artificial neural network and it is crucial for preserving the integrity and intellectual property associated with the network.

Firstly, if a data set contain sensitive or proprietary information, such as personal data, trade secrets, or valuable research, it also must be kept secret as a single component for ethical reasons within the laws of the existing legal framework about data sets. Either way, if it's public the idea remains to protect it as a unit with other components.

The data set influences some of the main phases of an ANN development:

- Training: The data set provide the ANN with examples of patterns and relationships within the data. The network learns to recognize and generalize from these patterns to make predictions or classifications by changing its weights. This is essential to actually achieve a trained model that deeply understand the complex relationship between the labeled data and the expected value, and therefore impact training.

- Validation: A diverse and representative data set helps the ANN generalize its learning to new, unseen data. Without sufficient data, the network may overfit, meaning it learns to memorize the training data but cannot make accurate predictions on new data. For this reason, during validation the data set influences key steps like *hyperparameter tuning*, performance monitoring and others aspect listed in the point 4.5.4.

- Testing: A part of the data set is dedicated to testing, which allows to actually assess the model's generalization performance.

This means that it is a fundamental element for the ANN as strongly impacts how the network is learning because it provides the examples on which the ANN learns to perform its instructions. Also protecting just the other three elements is not enough as one could use the same network architecture and the same technical implementation and a different data set to train the network, and casually obtain the same weight matrix as a result. Of course this is unlikely to happen, but still there is a chance, and if that's the case it actually would not be fair for the competitor, therefore, the data set also helps to maintain the uniqueness of the model.

5.2.3 Technical implementation

Technical implementation refers to the comprehensive set of specific decisions, configurations and processes that dictates implementation process of an artificial neural network. It encompasses a range of elements including:

- Initialization and Regularization Techniques: Deciding on the initialization method for the network's weights and biases, such as random initialization or using pre-trained weights.
- Choice of Optimization Algorithm: Selecting an optimization algorithm suitable for supervised learning, such as *stochastic gradient descent*; The chosen algorithm determines how the network's weights are updated during training to minimize the loss function.
- Hyperparameter Tuning: Determining the values of hyperparameters specific to the supervised learning algorithm, such as learning rate, batch size, number of epochs, or regularization strength. Adjusting these hyperparameters can significantly impact the model's performance.
- Loss Function Selection: Choosing an appropriate loss function that corresponds to the specific supervised learning problem, such as *mean squared error* (MSE showed in 4.5.3). The loss function measures the discrepancy between predicted and true values during training.
- Model Evaluation: Determining the evaluation metrics to assess the performance of the trained model on test or validation data depending on the supervised learning task.

Protecting the technical implementation involves safeguarding these choices and configurations that are fundamental for the ANN functioning which may include solutions or technical approaches tailored for the other components that enhance the effectiveness and efficiency of the ANN.

5.2.4 Weights

The last fundamental component is the weight matrix of trained neural networks. The weights represent the internal parameters of the network that are updated during training to optimize the network's behavior. If taken alone, they are merely numbers, however, along with other components involved in the training process, they describe the functioning artificial neural network, reflecting the specific configuration that the network has learned from the data during the training process. Let's clarify:

As it was already explained, the weights represent the coefficients that define the relative importance of connections between neurons in the network. Each weight influences the contribution of an input neuron to the next neuron. By changing the weights, the network can adjust the importance of different input features and their influence on the output results. During the training process, weights are adjusted based on the learning algorithm used as the goal is to find an optimal configuration of weights that minimizes the error between the network's predictions and the desired output values. These weights determine the strength and significance of the connections, influencing how information flows through the network during the inference or prediction phase. This means that they encapsulate the network's ability to generalize from the training examples and make predictions or classifications on unseen or new input data. Therefore, weights can be considered as the end result of the training process, along with the choices and techniques, as they reflect the network's final ability to take an input, processing it, and producing an output.

The main reason why they need to be protected with other attributes is that, in contrast to traditional source code, where the computer realize a function by following a well-written set of instruction, the weight information associated with the connections between neurons does not possess the capability to direct a machine to execute a particular process, therefore it is not capable of instructing the machine to do anything because, as it was stated result weights alone mean nothing but mere numbers, so they need to be combined with other elements to form a unique component able to suitably represent an ANN.

5.2.5 Further explanations

It is important to underline the importance of considering the intellectual property of a neural network as a combination of these elements and not taking them individually for various reasons. Firstly, it is possible for some of these elements to be the same across different artificial neural networks. This is because, for specific applications, certain elements such as topology, the number of nodes and layers, or even hyperparameters like the activation functions may well-known and notably optimal choices.

Secondly, when taken individually, these four components do not offer robust legal protection for an artificial network as each component, on its own, presents limitations in safeguarding the intellectual property and uniqueness of an ANN.

Take the weight matrix in artificial neural networks, for example. As mentioned earlier, weights are essentially data structures. Copyright law, on the other hand, primarily protects the specific way an idea is expressed and not the concept behind the idea. This means that copying the exact set of weights, with only minor adjustments, may not constitute a copyright infringement. Furthermore, replicating the weights with slight modifications might not significantly impact the neural network's performance. In

fact, this procedure may not even violate any copyrights. This underlines the necessity of protecting these attributes together.

OwnerImpl.jad (decompiled version of Oracle OwnerImpl.class)
[spacing adjusted for comparison]

```
    public synchronized boolean deleteOwner(Principal principal, Principal
principal1) throws NotOwnerException, LastOwnerException {
        if(!isOwner(principal))
            throw new NotOwnerException();


        Enumeration enumeration = ownerGroup.members();
        Object obj = enumeration.nextElement();
        if(enumeration.hasMoreElements())
            return ownerGroup.removeMember(principal1);
        else
            throw new LastOwnerException();
    }


    public synchronized boolean isOwner(Principal principal) {
        return ownerGroup.isMember(principal);
    }
}
```

OwnerImpl.java (Android version)
[spacing adjusted for comparison]

```
    public synchronized boolean deleteOwner(Principal principal, Principal
principal1)
                                throws NotOwnerException, LastOwnerException
{
        if(!isOwner(principal)) throw new NotOwnerException();
        Enumeration enumeration = ownerGroup.members();
        Object obj = enumeration.nextElement();
        if(enumeration.hasMoreElements()) {
            return ownerGroup.removeMember(principal1);
        } else {
            throw new LastOwnerException();
        }
    }

    public synchronized boolean isOwner(Principal principal)
    {
        return ownerGroup.isMember(principal);
    }
}
```

Figure 5.2: Exhibit Copyright-N in the Oracle Inc. vs Google case [22]

Oracle America, Inc., owns a copyright in Java SE, a computer platform that uses the popular Java computer programming language. In 2005, *Google* acquired Android and sought to build a new software platform for mobile devices. To allow the millions of programmers familiar with the Java programming language to work with its new Android platform, *Google* copied roughly 11,500 lines of code from the Java SE program [23].

The exhibit in Figure 5.2 was presented as evidence in the *Oracle vs. Google* case. This exhibit is an example of what a copyright infringement of software source code might entail: the codes are identical and, as a result, they have an identical functionality. The case of an ANN would be different because, as it was stated, the source code does not explicitly include instructions, however this proposal is seeking to find a similar protection. Let's can provide a simplified hypothetical example to illustrate the concept of ANN's protection with the given components:

Take two developers, A and B, both working on artificial neural networks for image recognition, for simplicity let's say the ANN illustrated in the previous chapter, a dog and cat classifier. They both have similar ANN architectures but B doesn't know how to progress further. Now, let's consider a scenario where B, without A's permission, copies the entire configuration of her ANN's including data set, architecture, technical implementation into his project, which ends up giving him the exact same weight matrix. In this case, B has essentially replicated A's network fundamental design, replicating its functionality, just like it was happening in the source code in figure 5.2 in Oracle vs Google case, which would constitute an intellectual property infringement.

As previously mentioned, my goal was to make a proposal to take care of the lack of regulations in Italy that directly deal with the protection of ANNs, and the framework provided regarding the elements to defend seems to be a good generalization of what actually needs to be safeguarded in an ANN.

5.3 AI intellectual property in Italy: State of the art

The significant growth and widespread adoption of artificial neural networks have raised significant legal issues concerning their protection and intellectual property. Particularly in Italy, where regulations related to emerging technologies like AI may not be adequately defined, it is crucial to carefully examine the existing legal framework to identify some of the regulatory gaps.

This section aims to explore the context of intellectual property for artificial neural networks in Italy, focusing on how existing laws can be applied to such AI systems. We will begin by analyzing relevant forms of intellectual property protection, such as copyright, patents, and trade secrets, and evaluate their suitability in addressing the specific challenges posed by artificial neural networks. Additionally, we will thoroughly examine the current state of artificial neural networks in Italy to further underscore the pressing need to introduce new proposals, as previously mentioned, in order to provide adequate legal protection for ANNs and, consequently, the intrinsic know-how associated with these models.

5.3.1 Copyright

For what concerns software, Italy relies on the Italian Copyright Law (Law no. 633/1941), known as the "Legge sul Diritto D'Autore," to safeguard original works of authorship, which encompass various forms of creative expression, including software. This legal framework primarily extends protection to the tangible and well-defined code resulting from a programmer's creative efforts. It covers the literal representation of the code, namely the specific lines of code that constitute the program.

However, when it comes to AI, specifically artificial neural networks, the notion of a conventional source code becomes complex and elusive. ANNs operate in a dynamic manner, and identifying a traditional source code may not be straightforward.

In such instances, the focus of protection may shift towards specific components like it was previously illustrated, rather than the mere executable code produced by a compiler. This highlights the need for a sophisticated legal approach designed to protect the intricately linked intellectual property associated with ANNs.

5.3.2 Patents

A patent represents a territorial form of intellectual property rights, implying that its protection extends solely to the jurisdiction where the relevant application was filed and subsequently approved.

Generally, patents are granted by national patent offices, such as the UIBM in Italy or the USPTO in the United States. Alternatively, they can be established through international agreements, as is the case for Italy, which adheres to European international conventions. This prominent example regards the European Patent Convention (EPC), where the patent application procedure is overseen by the European Patent Office (EPO). Italy, as a member of the European Patent Convention (EPC), participates in the EPO system.

In accordance with Article 45, paragraph 2 of the Industrial Property Code (Legislative Decree, 10 February 2005, no. 30 - IP Code)[24], an AI system, similar to mathematical methods and computer programs, does not qualify as an invention and is therefore not eligible for patent protection, unless specific criteria are met [25]:

- It produces a **technical effect** as it is intended in 2.3.3.
- It is original and the result of an intellectual creation.

EPO seems to confirm this version through the *Guidelines for Examination* at the section 3.3.1 [26]:

> *Artificial intelligence and machine learning are based on computational models and algorithms for classification, clustering, regression and dimensionality reduction, such as neural networks, genetic algorithms, support vector machines, k-means, kernel regression and discriminant analysis. Such computational models and algorithms are per se of an abstract mathematical nature, irrespective of whether they can be trained based on training data. Hence, the guidance provided in G-II, 3.3 generally applies also to such computational models and algorithms*

Upon closer examination, the legal provision stipulates that these algorithms are excluded from patentability only *as such*, implying that inventions related to *software* can legitimately obtain protection when they interact with *hardware systems* and control certain functions. This is known as the theory of **technical character**: even inventions implemented through a computer can be patented if they are capable of producing an additional technical effect that goes beyond the normal interaction between hardware and software [27].

Furthermore, to better understand whether an AI system is patentable or not, EPO presents two examples of patentable and non-patentable creations in the previously mentioned 3.3.1 section. The first - patentable - example is:

> *Artificial intelligence and machine learning find applications in various fields of technology. For example, the use of a neural network in a heart monitoring apparatus for the purpose of identifying irregular heartbeats makes a technical contribution.*

On the other hand, a non-patentable subject example is:

> *Classifying text documents solely in respect of their textual content is however not regarded to be per se a technical purpose but a linguistic one.*

This implies that the effectiveness of IP protection is closely tied to the functionality of the model, which can be a significant limitation when attempting to protect our proprietary artificial neural networks, again underlining the necessity to find a combination of attributes that enable to protect an AI system regardless of its final purpose.

5.3.3 Trade secrets

In Italy, as an alternative to seeking patent protection, AI systems can be safeguarded as confidential trade secrets. Trade secrets fall under the scope of articles 98 and 99 of the IP Code:

- According to article 98, protection is granted to business information and technical-industrial experience, including commercial information and experience, subject to the legitimate control of the owner. To be protected as a trade secret, the information must meet the following three criteria [24]:
  - *It must be secret*: The information must not be known to the public or to those who can reasonably be expected to have access to it.
  - *It must have economic value*: The information must have some commercial value, such as by giving the owner a competitive advantage.
  - *It must be kept secret*: The owner must take reasonable steps to keep the information secret, such as by restricting access to the information and using confidentiality agreements.
- Article 99 says: In accordance with Article 98 and without prejudice to unfair competition regulations, the lawful owner of the information and business expertise has the right to prevent third parties from acquiring, disclosing, or using such information without consent, unless the third party independently obtained it.

Therefore, upon meeting all the required, you can protect your ANN as a trade secret. However, it is important to note that trade secret protection can be lost if the ANN becomes public. This can happen if the ANN is published, disclosed to others, or if you fail to take reasonable steps to keep it secret. This implies that if I choose to define four attributes that make up the source code of my ANN and seek protection under this particular law, these attributes must have a basis for direct or indirect protection.

5.4 AI intellectual property outside Italy

5.4.1 European Union

In the European Union, the European Patent Office is one of the two organs of the European Patent Organisation. It grants European patents covering the Contracting States to the European Patent Convention and several other states that have concluded extension and validation agreements with the EPO.

As it was already stated, EPO mentions that mathematical methods play an important role in the solution of technical problems in all fields of technology. However, they are excluded from patentability under Art. 52(2)(a) when claimed as such (Art. 52(3)) [26]. The point 3.3.1 which specifically concern AI and ML, affirm that such computational models and algorithms are per se of an abstract mathematical nature, therefore they are patentable exclusively if they carry out a technical effect just like it was explained in 5.3.2, which was in fact referring to the European convention.

5.4.2 United States of America

The USA regulation is quite further than the italian and european regulations. In October 2020 the United States Patent and Trademark Office (USPTO) published a report titled *Public Views on Artificial Intelligence and Intellectual Property Policy*. The report takes a comprehensive look at a wide variety of stakeholder views on the impact of AI across the IP landscape and provides AI context, legal background, and public comment synthesis for each of the questions presented in the two requests for comments. The USPTO has used the report to focus on issues for continued exploration and stakeholder engagement to bolster the understanding and reliability of IP rights for AI and other ET.

Firstly, the USPTO's mentioned report provide a definition of AI invention to identify broadly the elements and attributes of an AI that may be subject to patentability, and this is the result:

> *Among the responses, four common answers arose:*
>
> 1. *The various elements disclosed in the question constitute a non-exclusive list of elements of an AI invention.*
> 2. *AI can be understood as computer functionality that mimics cognitive functions associated with the human mind (e.g., the ability to learn).*
> 3. *AI inventions can be categorized (in no particular order) as follows:*
>   (a) *inventions that embody an advance in the field of AI (e.g., a new neural network structure of an improved machine learning (ML) model or algorithm)*

(b) inventions that apply AI (to a field other than AI)

Secondly, this report stated that the inventor of an AI work, which is defined as the individual or, if a joint invention, the individuals collectively who invented or discovered the subject matter of the invention by 35 U.S.C. § 100, must be human:

> *Title 35 of the United States Code is replete with language indicating that the inventor of a patent application must be a natural person. For example, 35 U.S.C. § 101 states, "Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter ... may obtain a patent therefore, subject to the conditions and requirements of this title" (emphasis added). "Whoever" denotes whatever person, a person being a human being—a natural person.19 By the use of "whoever," § 101 limits patent protection to inventions and discoveries by natural persons.*

Finally, it states that the USPTO will continue to "evaluate AI inventions on a case-by-case basis", but will generally consider whether the invention meets the same requirements as any other patentable invention, which means that an invention, in order to be patentable has to have these three main requirements:

- Novelty
- Non-obviousness
- Utility

Furthermore, the USPTO stated that will take into account factors such as:

- The AI system's contribution to the invention
- The degree of human participation
- The AI system's capacity for independent thought or action

On the basis of these statements we can pinpoint a couple pros and cons to this systems in the light of patent systems we've analyzed until now.

The US system allows patenting AI algorithms regardless its final purpose, which means it is not necessary to carry out the so called technical effect. Furthermore the case-by-case evaluation gives a certain degree of flexibility while developing an AI model, which translates into encouraging innovation and investment in AI technology. Additionally, patenting AI inventions can help to prevent others from copying or using the inventions without permission. This can help to protect the interests of the developers and owners of AI systems.

On the other hand patenting AI inventions could lead to monopolies. If a single company is able to patent a key AI technology, it could prevent other companies from

using that technology. This could give the patent holder a significant advantage in the market. Additionally, there is the question of who should be credited as the inventor of an AI invention. Under current patent law, only natural persons can be named as inventors. This means that an AI system cannot be named as an inventor, even if it played a significant role in the invention. This could lead to disputes over who is entitled to the patent rights.

In conclusion, ANNs are patentable provided they meet the same requirements as any other invention, this has both positive and negative implications which we just listed. According to USTPO the patentability of narrow AI systems is likely to continue to be a topic of discussion and debate in the near future, therefore it will surely include further modifications and rearrangements.

5.4.3 Canada

Canada is a world leader in the field of artificial intelligence. According to the report *Processing Artificial Intelligence: Highlighting the Canadian Patent Landscape* carried out by the Canadian Intellectual Property Office (CIPO), Canadian researchers and institutions accounted for 1.8%, or 1,516, of the 85,144 AI inventions patented worldwide between 1998 and 2017 [28].

The Canadian Intellectual Property Office (CIPO), a Special Operating Agency of Innovation, Science and Economic Development Canada (ISED), is responsible for the administration and processing of Intellectual Property (IP) in Canada. According to CIPO's report, assessing innovation in the AI field is challenging due to its diverse techniques applied across numerous industries, therefore even if patented inventions aren't a perfect measure, they serve as a valuable proxy for assessing innovation in this specific technology sector.

However this very same article specifies how, as opposed to the US patent system, the Canadian Patent Act states that a patent can only be granted for the *physical embodiment of an idea*, hence computer programs are not considered to be patentable subject matter. So how do they patent their AI systems?

In Canada, the patentability of AI inventions is determined based on the subject matter eligibility criteria stated in the Patent Act. The Act defines an *invention* as any new and useful art, process, machine, manufacture, composition of matter, or improvement thereof. However, it excludes *mere scientific principles or abstract theorems* from patentability.

In order to clarify this matter, Canadian courts have provided guidance on the eligibility of computer-implemented inventions. For example, in the Schlumberger case, it was established that implementing a non-patentable algorithm using a computer to do mathematical calculations does not make the algorithm patentable exploiting the fact

that a patent can be granted for the physical embodiment of an idea. As a matter of fact, the Federal Court of Appeal (FCA) ruled that the inventive aspect must go beyond a *mere scientific principle or abstract theorem*. Furthermore, in the Amazon.com case, relating to the one-click online ordering system, the FCA emphasized the importance of *purposive construction of patent claims* in the subject matter analysis. It stated that patentable subject matter should have *physical existence* or *manifest a discernible effect* or change.

In addition to this, CIPO issued practice notices providing further guidance on the matter. According to these notices, when a claimed computer element is essential to solving a problem addressed by the invention, the claim is considered **statutory subject matter**. In fact, the CIPO considers whether the claimed elements address a *computer problem* rather than being merely part of an *operating environment*.

The Manual of Patent Office Practice (MOPOP) provides additional guidance suggesting that controlling a computer's operations to achieve a technological result is considered statutory. Additionally, a claim element that is common general knowledge but essential to the invention is not automatically excluded.

In conclusion, until a dedicated regulatory framework for AI, such as the proposed Artificial Intelligence and Data Act (AIDA), is enacted into law, there are specific steps that can enhance the likelihood of AI inventions receiving patent approval in Canada. These steps include:

- Clearly explain the technical problem solved by the inventors.
- Embed the creation within a physical object.
- Characterize - preferably - the problem as a "computer problem".
- Provide detailed technical information about the hardware and software used to solve the problem to support the eligibility of claims defined by the Patent Act.

Chapter 6

Conclusions

In this thesis, our primary aim was to delve into the realm of AI intellectual property, with particular attention to the Italian legal framework, focusing on the legal protection of AI systems, specifically artificial neural networks trained using supervised learning algorithms. Our mission was to dissect these AI systems' critical components that govern their functionality and, in doing so, establish a robust foundation for their legal safeguarding.

Through an in-depth analysis of AI intellectual property, interesting findings have emerged, significantly advancing the understanding of legal protection for AI systems, particularly artificial neural networks.

Firstly, the groundwork for hypotheses was laid with a generic background. This encompassed an overview of Italy's intellectual property legal framework for software, an exploration of the functioning of Turing Machines, and a comprehensive introduction to Artificial Intelligence and artificial neural networks. This contextual knowledge was instrumental in our subsequent analysis of ANNs' functionality, enabling the identification of core components responsible for their functioning, and therefore that demand legal protection as a single unique component.

Furthermore, the research revealed the intricacies of AI ownership. Comparisons were drawn between AI systems and traditional source code, emphasizing the conceptual similarities with Turing Machines while underscoring the unique challenges associated with safeguarding ANN intellectual property due to the fundamentally distinct nature of source code.

The core components comprising network architecture, data set, technical implementation, and weights collectively constitute the foundation of an artificial neural network's functionality. These attributes, as elucidated, fundamentally dictate the final outcome of the ANN. Recognizing their pivotal role, it becomes evident that they are the key areas demanding legal protection as these components not only shape the network's behavior but also represent the fundamental principles for safeguarding the innovations and investments made in AI development.

Finally, I conducted an in-depth study of existing laws, both at the national level in Italy and internationally, in order to provide a comprehensive overview of how these core components, and AI in general, could be legally protected in the current state of the art. This analysis revealed the challenges and opportunities within the current regulatory framework and underscored the need to develop more suitable laws to preserve the integrity and innovation within the field of Artificial Intelligence.

For the future, an ongoing commitment to developing more suitable laws and regulations, taking into account the nuances of AI, is essential. Furthermore, collaboration among legal experts, computer scientists, and AI ethics professionals will be crucial in maintaining a balance between innovation and rights protection.

Bibliography

[1] A. M. Turing *et al.*, "On computable numbers, with an application to the entscheidungsproblem," *J. of Math*, vol. 58, no. 345-363, p. 5, 1936.

[2] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[3] Council of Europe, *History of Artificial Intelligence*, `https://www.coe.int/en/web/artificial-intelligence/history-of-ai`.

[4] J. Lewis, S. Schneegans, T. Straza, *et al.*, *UNESCO Science Report: The race against time for smarter development*. Unesco Publishing, 2021, vol. 2021.

[5] WIPO, *Intellectual Property and Frontier Technology*, `https://web.archive.org/web/20220402064804/https://www.wipo.int/about-ip/en/frontier_technologies/`.

[6] G. Sartor, *L'intelligenza artificiale e il diritto*. G. Giappichelli Editore, 2022.

[7] R. Y. Choi, A. S. Coyner, J. Kalpathy-Cramer, M. F. Chiang, and J. P. Campbell, "Introduction to machine learning, neural networks, and deep learning," *Translational vision science & technology*, vol. 9, no. 2, pp. 14–14, 2020.

[8] M. Cord and P. Cunningham, *Machine learning techniques for multimedia: case studies on organization and retrieval*. Springer Science & Business Media, 2008.

[9] IBM, *What is machine learning?* `https://www.ibm.com/topics/machine-learning`.

[10] S. Badillo, B. Banfai, F. Birzele, *et al.*, "An introduction to machine learning," *Clinical pharmacology & therapeutics*, vol. 107, no. 4, pp. 871–885, 2020.

[11] L. Belenguer, "Ai bias: Exploring discriminatory algorithmic decision-making models and the application of possible machine-centric solutions adapted from the pharmaceutical industry," *AI and Ethics*, vol. 2, no. 4, pp. 771–787, 2022.

[12] T. Hellström, V. Dignum, and S. Bensch, "Bias in machine learning–what is it good for?" *arXiv preprint arXiv:2004.00686*, 2020.

[13] D. Pedreschi, F. Giannotti, R. Guidotti, A. Monreale, S. Ruggieri, and F. Turini, "Meaningful explanations of black box ai decision systems," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 9780–9784.

[14] M. Carabantes, "Black-box artificial intelligence: An epistemological and critical analysis," *AI & society*, vol. 35, no. 2, pp. 309–317, 2020.

[15] O. Eluyode and D. T. Akomolafe, "Comparative study of biological and artificial neural networks," *European Journal of Applied Engineering and Scientific Research*, vol. 2, no. 1, pp. 36–46, 2013.

[16] K. Suzuki, *Artificial neural networks: methodological advances and biomedical applications*. BoD–Books on Demand, 2011.

[17] N. S. Gill, *What is an Artificial Neural Network?* `https://www.xenonstack.com/blog/artificial-neural-network-applications`, 2023.

[18] J. Li, J.-h. Cheng, J.-y. Shi, and F. Huang, "Brief introduction of back propagation (bp) neural network algorithm and its improvement," in *Advances in Computer Science and Information Engineering: Volume 2*, Springer, 2012, pp. 553–558.

[19] F. Bertini, D. Allevi, G. Lutero, D. Montesi, and L. Calzà, "Automatic speech classifier for mild cognitive impairment and early dementia," *ACM Transactions on Computing for Healthcare (HEALTH)*, vol. 3, no. 1, pp. 1–11, 2021.

[20] K. Maharana, S. Mondal, and B. Nemade, "A review: Data pre-processing and data augmentation techniques," *Global Transitions Proceedings*, vol. 3, no. 1, pp. 91–99, 2022.

[21] *Neural Networks Tutorial*, `www.neuraldesigner.com/learning/tutorals/testing-analysis`, 2023.

[22] J. C. Mitchell, *Opening expert report of john c. mitchell regarding copyright*, Northern District of California, Case 3:10-cv-03561 WHA, 2012.

[23] *Google LLC v. Oracle America, Inc., Certiorari to the United States court of appeals for the Federal Circuit*, Supreme Court of the United States, No. 18–956, 2021.

[24] Decreto Legislativo, 10 febbraio 2005, n. 30, *Il codice della proprietà industriale (cpi)*, 2005.

[25] E. Fabrizi, F. Ferrara, and G. Marino, *Law Over Borders Comparative Guide: Artificial Intelligence*, `https://www.globallegalpost.com/lawoverborders/artificial-intelligence-1272919708/italy-1602230361`, 2023.

[26] European Patent Office, *Guidelines for Examination in the European Patent Office*. European Patent Office, Directorate Patent Law 5.2.1, 2023.

[27] European Patent Office, *Guidelines for Examination of European Patent Applications*, https://new.epo.org/en/legal/guidelines-epc/2023/g_ii_3_3_1.html, 2023.

[28] Canadian Intellectual Property Office, *Processing artificial intelligence: Highlighting the canadian patent landscape*, https://ised-isde.canada.ca/site/canadian-intellectual-property-office/en/processing-artificial-intelligence-analysis-canadian-perspective, 2022.