Inorganic waste classifier using artificial intelligence

LAB University of Applied Sciences
Bachelor of Engineering, Information and Communication Technology
2021
Isabel Ferri Mollá

Abstract

Author(s) Ferri Mollá, Isabel	Publication type Thesis, UAS	Completion year 2021
	Number of pages 54	
Title of the thesis **Inorganic waste classifier using artificial intelligence**		
Degree Information and Communication Technology		
Name, title and organisation of the thesis supervisor Ismo Jakonen, Senior Lecturer, Media Technology		
Name, title and organisation of the client		

Abstract

Artificial intelligence is a field that has experienced incredible growth in recent years. However, one of the milestones that marked a before and after in this field is deep learning and the emergence of neural networks, algorithms capable of, given a set of data, extracting features and patterns from them, as well as the possibility of applying them to classify or obtain information from new data not previously known.

The objective of this thesis was to create a neural network capable of classifying waste according to the material it is made of, thus facilitating the task of recycling these objects. To achieve the objective, a convolutional neural network was created as well as a dataset for training, which was correctly labelled and reviewed.

Keywords

Classifier, artificial intelligence, waste, image recognition, neural networks

Contents

1 Introduction

Technology, and more specifically computers, is one of the most useful and powerful tools available to us today. It has been evolving in a fast way and it has been introduced into our lives to the point of becoming practically indispensable in our daily routine. There is no doubt that computing is capable of helping us to solve many of today's problems.

Among today's problems, one of the ones that affect us most and should concern us when we look through the future is ecology. It is undeniable that we are using the resources of our planet at a faster rate than nature can support. Therefore we need to change now and start planning measures that large companies and governments can carry out, but also small actions that we can all do to contribute our bit.

However, although recycling is an action that many citizens around the world carry out, it has some nuances and small differences in each country. At least that was the problem I encountered during my exchange in Finland. Although most of the containers were similar, some changed and some objects were thrown in different containers in Spain and Finland. This is the case of milk cartons that in Spain go in the yellow container, the plastic one, and in Finland, they are recycled in the cardboard container. Also in Spain, there is a blue container for paper and cardboard, while in Finland paper and cardboard are separated and go into different containers. These are just some of the variations between countries.

As mentioned earlier, computer science is a growing and enormous field, and among the fields it encompasses, one of the ones that are growing and gaining more importance is artificial intelligence, and within it, deep learning. Deep learning algorithms are capable of given a set of input data, learning the characteristics of that data, and generating responses to new data that were unknown until now. The growth in the Artificial Intelligence field is amazing and it was unimaginable years ago. So why not use deep learning to help people learn how to recycle and do it properly, and thus contribute to improving the major climate problem we face in the coming years?

This thesis aims to develop a waste classifier that, given an image of a recyclable object, can recognize which type of material it is done with, so it would be easy to say in which bin it should be recycled. However the task of creating a neural network that can work properly is not easy. It needs an appropriate dataset to get good accuracy, as well as finding the proper hyperparameters to get a good result.

During the next pages, an introduction to the earth’s ecological problem which this waste classifier is faced to help will be explained and after that, a theoretical background about artificial intelligence will be explained. Finally, the practical case and all the technologies used to build it as well as the results received will be presented too.

2 Environment

2.1 Earth's ecological problem

Regarding the United Nations foundation's list of five global issues to watch in 2021, the second concept that appears in this list is the accelerating progress on the sustainable development goals. This concept, sustainable development, appeared in 1987 in the Brundtland report produced by different countries of the United Nations, and it was described as *the need to ensure that development meets the needs of the present without compromising the ability of future generations to meet their own needs*. (World Commission on Environment and Development 1987,16.)

According to this United Nations foundations list, it is undeniable that the environmental problem of the planet is one of the greatest challenges that humanity will have to face in the coming years. But to understand better the ecological problem faced and why it affects people, it is needed to understand what ecology is.

Ecology is the science of the study of interactions between individual organisms and their environments including the relationship with members of their same species and members of other species (Stanford encyclopedia of philosophy 2005). This concept is intricately linked to the environment one, which is the *set of physical, chemical, biological, and social components that can cause, direct or indirect, and short or long-term effects on living beings and human activities* (United nations conference on the environment, Stockholm 1972).

Although these concepts are becoming increasingly topical and we have all heard about them to a greater or lesser extent, the reality is that humans show an increasingly destructive consumption which causes us to need every year more resources than the earth can provide and replenish. This overexploitation leads to an ecological debt so that, according to a study carried out in 2016 by the organization Global Footprint Network, if we maintain the current rate of consumption, then 1'75 planets will be needed to assume the ecological impact that humanity produces annually. This fact reduces, even more, the earth's regeneration capacity. (Global Footprint Network 2016.)

2.2 Reduce, Reuse, and Recycling

As it has been explained before, the current environmental situation is unsustainable, so people must act in consequence, trying to get the aforementioned sustainable development. To achieve this goal, it is necessary to achieve a growth where resources were efficiently used, and some of the actions, we can carry out are those known as 3R: reduce, recycle, and reuse. These 3Rs focus on decrease the amount of waste we generate, but to do this properly it is important to identify each of these concepts to be able to put them into practice. (Nava Bautista et al.)

Reduce is the fact of reducing the number of goods we consume, this means be responsible at the time we are doing our purchases. It is possible to put this concept in practice buying these things we really need, looking for products made with recyclable materials and with as less amount of packaging as possible, as well as choosing these products which have been made nearer us. In addition to this, another way of reducing the amount of co2 we produce can be reducing the amount of electricity we use unnecessarily (turning off the lights, etc.) or using more public transport or bicycle. (Nava Bautista et al.)

With the action of reuse, we try to give things as most utility as possible before discarding them, with this action, we try to give the product a second life. For example, we can use a packing box to store other things once we have received and used the product containing. Other possible actions which can be used to reduce are repairing our products as well as donating objects we do not need instead of discarding them. (Ohio Valley Waste Service 2017.)

The last R is recycling which means transform used materials from useless products to get new products that can be used afterward. With these actions, the materials are incorporated into a cycle trying to get new materials with less energy and less generated waste and it is especially important to get a balance between production, consumption, and waste. To do this, the materials must be separated so that is easier their transformation, so the help of everybody is needed to achieve this balance goal. With this simple action, it can be possible to reduce the use of raw materials needed to process the new product. (OXFAM Intermon.)

Figure1 shows in a more visual way some actions which should be done to apply the 3 Rs.

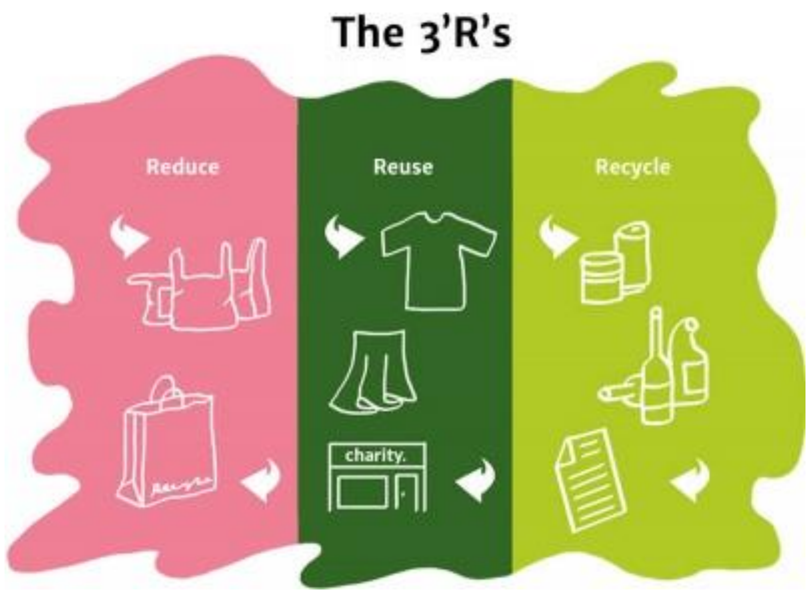


Figure 1. Reduce, Reuse and Recycle (solar Schools)

2.3 Circular economy

The way how humans processing of our waste is not sustainable, usually the economy is viewed in a linear sense which line is take-make-consume-dispose, this model requires a huge number of natural resources which overlap the planetary boundaries. However natural resources on our planet are limited so if we think in the long term future the economic model must be changed to make it sustainable. (Garcia Garcia 2020.)

If we think of an ecosystem, we can observe how herbivores feed on plants, the carnivores of herbivores, and in turn when they die, they become nutrients for the soil that feeds the plants which are eaten by the herbivores, as we can observe this is a cycle that succeeds indefinitely and is sustainable in time. This is the main idea in a circular economy, transform the linear model into a circle that can be long-term sustainable, making the raw material of new products come from the waste of other used products, and trying to avoid single-use products by reusing and repairing existing products. (World Economic Forum 2019.)

In order to do that, collaboration is needed, it is necessary to redesign the products by the companies, thinking of them as products of various uses, easily degradable, and made of reusable materials that can be used to make other products once the life cycle of the current product has ended. In addition to this, an infrastructure that allows the collection of objects after the first uses for reuse or recycling is necessary too. But not only companies should put on their side, citizens must also make sure they return these products in the right place for subsequent recycling and reuse. (Garcia Garcia 2020.)

This means a big effort both economically and socially since it implies a change in the economic model and in the mentality of the citizen. However, it can imply in a long-term point of view not only environmental benefits but economic and social too. (Garcia Garcia 2020.)

That is why some of the European Union countries have begun projects to accomplish the circular economy objective. This is the case of Holland which has a project to reduce the 50% of their waste by 2030 and to get a 100% circular economy and waste-free by 2050. (Construncia 2020.)

In the case of Finland, they have the objective to reach a circular economy too, some of their actions through these objectives are the commercial incentives as the packaging return machines which can be found in some markets, or the commitment to raise awareness of future generations through education. (Haapio, 2019.) Other countries like Spain have an ecological project too, trying to reduce the material consumption by a 30%, the waste generation in a 15% or the Co2 emission under 10.000.000 tonnes. (Construncia 2020.)

But the change is growing slowly because of some factors as can be high public awareness, political barriers, and lack of public and private investment. According to an article written by a Spanish MEP, nowadays only *12% of the European secondary materials and resources are returned to the economy*. (Maldonado 2020.)

2.4 Recycling apps

As can be seen, in the next years Europe and countries all over the world are going to face a huge challenge to reach sustainable development and economy, to do this task a little bit easier, technology can be a great ally since it is an enormously powerful tool capable of making human lives easier. That is why some code developers have been doing some applications to help and make people easier to collaborate in the task of creating a sustainable world. In Finland, some of these applications made to help people to recycle and collaborate with the sustainable economy are between others

Cannit- https://www.canit-app.com/what-canit

It is an app to avoid the bottles and cans waste inside parks and public spaces. As in Finland, the fact of returning cans and bottles at the supermarket means receiving a small amount of money, this app communicates to people who do not have time to go to supermarkets to return this type of waste with people who go looking for cans and bottles in the street to return them. Thanks to this app the first group post where they leave each residue to make sure the seconds can find it and return it for recycling. (Cannit.)

Gambit- https://www.gambitgroup.fi/recycling-made-easy/

This app allows people to see the recycling stations in Vantaa's region. With the use of this app, it is possible to look for the direction of your nearest waste recycling point. (Gambit.)

Recycle Finland- https://appadvice.com/app/recycle-finland/1095762900

This application is done with the main objective of informing the user about the emplacement of the different recycling points around Finland. So, it allows the user to see which recycle point is nearer to him or her and the exact position of these recycling centers. (AppAdvice 2016.)

Other countries have some other types of recycling app as the Spanish AIRE(https://www.ecoembes.com/proyectos-destacados/chatbot-aire/) from ecoembes enterprise (which is in charge of the recycling sort in Spain). This is an intelligent recycling chat box that helps people to find in which container they must throw each type of waste looking at the Spanish recollection system. (Ecoembes 2018.)

3 Artificial intelligence

3.1 Introduction to AI

Artificial intelligence is usually defined as *The automation (by a machine) of activities that we associate with human thinking such as decision-making, learning*, *etc*. (Bellman 1978). However, the task of defining properly what is Artificial intelligence is quite hard.

Maybe it is reasonable to begin looking at what is intelligence because, in the end, artificial intelligence is just a piece inside the field of intelligence. Intelligence is a concept without a specific definition, most philosophers and experts coincide in considering it as a capacity, which enables the entity holding it to understand ideas, reasoning, classify, be autonomous and adaptative, etc. (Ardila 2011.)

So according to this definition, humans are not the only ones who can be considered intelligent, there are other types of intelligence as can be the natural one or artificial intelligence. As it has been seen Artificial Intelligence is a very big field of study which has inside some others as Machine learning or deep learning. In figure number 2 it is possible to see some of the subsets which compose Artificial Intelligence.

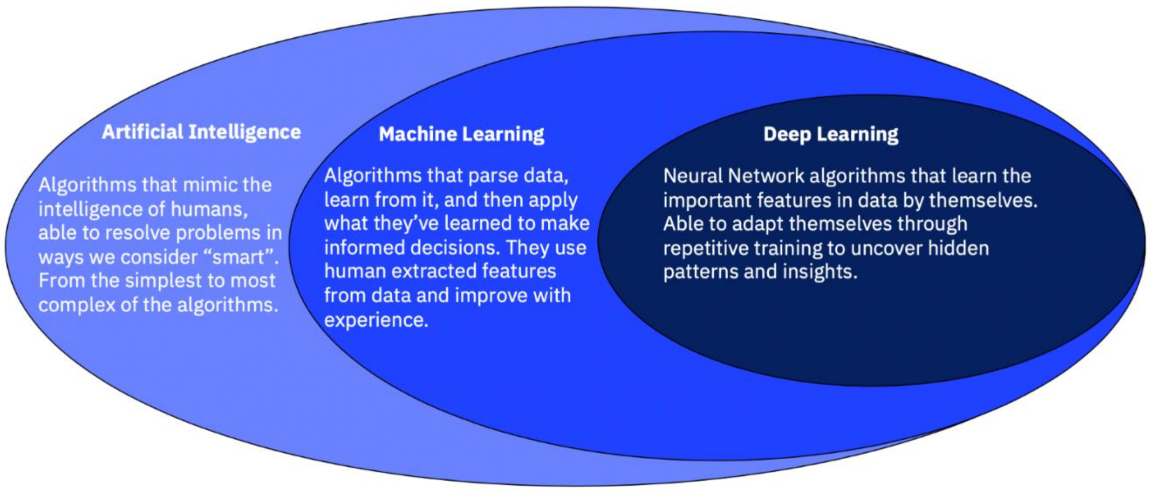

Artificial Intelligence
Algorithms that mimic the intelligence of humans, able to resolve problems in ways we consider "smart". From the simplest to most complex of the algorithms.
Machine Learning
Algorithms that parse data, learn from it, and then apply what they've learned to make informed decisions. They use human extracted features from data and improve with experience.
Deep Learning
Neural Network algorithms that learn the important features in data by themselves. Able to adapt themselves through repetitive training to uncover hidden patterns and insights.

Figure 2. Specialization of AI algorithms (IBM 2019)

In the research in the field of AI, there are two paradigms with different approaches about how intelligence should be seen. Next can be found a short description of each of these paradigms.

Symbolic, top-down approach

This approach consists of getting the knowledge and logic for experts of specific areas, use inference and specify rules to get new insides. This approach assumes that intelligence can be got through rules and logic operations, it presumes that the world can be understood in the terms of structured representation. The main advantage of this approach is that it is simple, however, it has problems to generalize and be used in changing context problems, because usually, the set of rules which suits a problem doesn't suit another one. Heuristic search is an example of an algorithm in this approach. (Bajada 2019.)

Connectionist, bottom-up approach.

This is one of the first approaches and right now this approach is behind the latest hits. Its main idea is to decompose complex things into simpler calculation units and interconnect them to solve the original problem, in this case, data is the center. The most popular technique in this approach is the Artificial Neural Network (ANN), which is used in deep learning. In it, several nodes called neurons are interconnected and receive different weights, each one of these neurons receives inputs and produces an output which can be the input of another neuron. These techniques are usually very effective in minimizing errors and it doesn't need a model of the world, just enough amount of sample data from which the model can be inferred. (Bajada 2019.)

3.2 History of AI

Although in the last years, Artificial intelligence has been a very concurrent concept and there have been huge advances in this field, it is a quite old concept. To find its beginning is necessary to go back to 1900 with one of the most notable events coming in 1936 with the birth of the Turing machine, the model on which today's computers are based. In 1950 Turing himself published his article computing machinery and intelligence in which he argued that computers could imitate human behavior. (De la Torre 2019.)

However, the term Artificial intelligence appeared in 1956, it was said during the Dartmouth conference by John McCarty, Marvin Misky and Claude Shannon, at that moment Artificial intelligence was described as *the science and ingenuity of making intelligent machines, especially intelligent computing programs*. (McCarty et al. 1956.) In 1958, Frank Rosenblatt introduced the perceptron concept. This perceptron is a neural network unit that receives input data and does some computations to detect some features in this input data. (De la Torre 2019.)

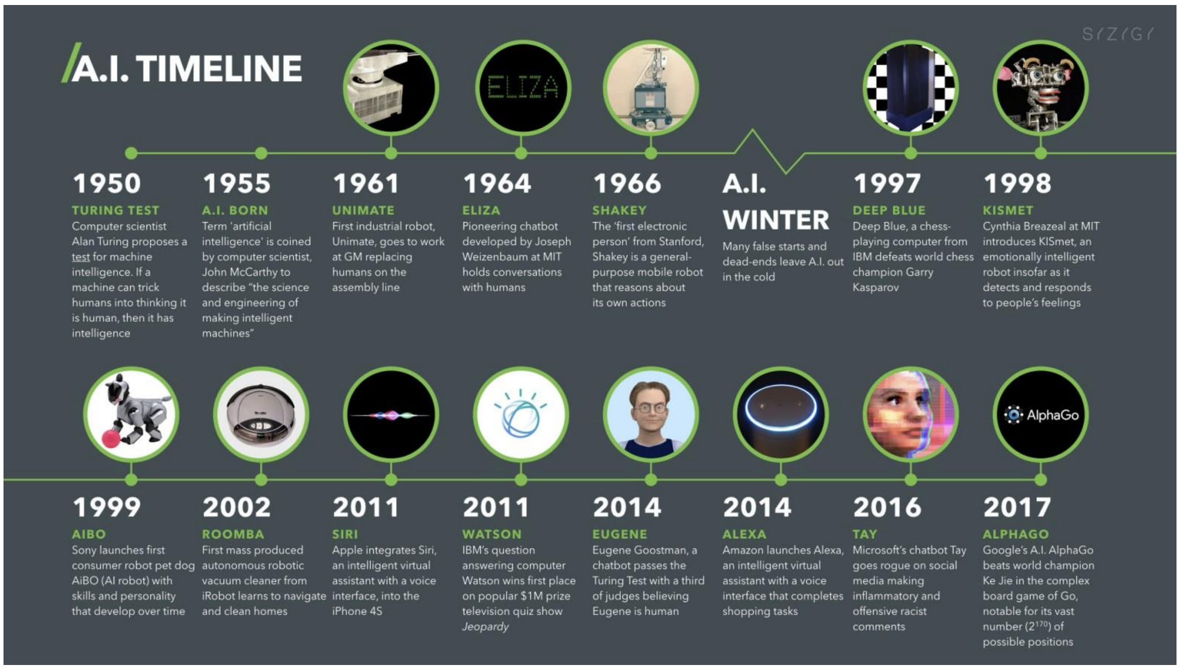
However, although artificial intelligence was already being talked about in the mid-20th century, its "golden age" began in the 1990s. One of the most talked-about milestones that helped to enshrine artificial intelligence happened in 1997, when the supercomputer called deep blue, created by IBM, managed to defeat the then world champion, Gari Kasparov. This event also helped to raise the profile of the field. In addition to this, during the 90’s decade, intelligent agents emerged. (Brezal 2017.) An intelligent agent is *a software entity that, based on its own knowledge, performs a set of operations aimed at satisfying the needs of a user or another program, either on its own initiative or because one of them requires it* (Vargas-Quesada 1999).

Despite the events narrated in the 20th century, it is during the second decade of the 21st century that artificial intelligence begins to develop enormously. In 2011, the computing system “Watson” from IBM, which was able to learn while it works and could interact with humans in natural language, won the TV show Jeopardy! against the world’s champions in this contest. (Brezal 2017.)

But the big explosion arrived in 2012 when the first commercial products that understood speech were introduced, and after that, the first image recognition applications appeared. During these years, Google created a supercomputer capable of learning through YouTube to identify cats as well as faces and human bodies.

Another important fact around 2012 is the AI-supported virtual assistants with deep learning algorithms launch (in June 2012 Google introduced its virtual assistant, Google Now, while in April 2014 Microsoft introduced its own virtual assistant, Cortana). As it is possible to see in figure 3, over the last few years, artificial intelligence has seen a big and fast growth, achieving milestones such as the victory of Google's artificial intelligence alpha go over the world go champion Se-Dol in 2016. (Brezal 2017.)

This huge and fast evolution makes us think that we are on the threshold of the fourth industrial revolution, the incorporation of all this technology into the industry. In it, artificial intelligence, as well as the internet of things or home automation promise a crucial role since the point that experts and big enterprises as IBM ensures that the fourth industrial revolution (4.0 industry) will become a reality from the hand of artificial intelligence. (IBM.)



1950 TURING TEST
Computer scientist Alan Turing proposes a test for machine intelligence. If a machine can trick humans into thinking it is human, then it has intelligence

1955 A.I. BORN
Term 'artificial intelligence' is coined by computer scientist, John McCarthy to describe "the science and engineering of making intelligent machines"

1961 UNIMATE
First industrial robot, Unimate, goes to work at GM replacing humans on the assembly line

1964 ELIZA
Pioneering chatbot developed by Joseph Weizenbaum at MIT holds conversations with humans

1966 SHAKEY
The 'first electronic person' from Stanford, Shakey is a general-purpose mobile robot that reasons about its own actions

A.I. WINTER
Many false starts and dead-ends leave A.I. out in the cold

1997 DEEP BLUE
Deep Blue, a chess-playing computer from IBM defeats world chess champion Garry Kasparov

1998 KISMET
Cynthia Breazeal at MIT introduces KISmet, an emotionally intelligent robot insofar as it detects and responds to people's feelings

1999 AIBO
Sony launches first consumer robot pet dog AiBO (AI robot) with skills and personality that develop over time

2002 ROOMBA
First mass produced autonomous robotic vacuum cleaner from iRobot learns to navigate and clean homes

2011 SIRI
Apple integrates Siri, an intelligent virtual assistant with a voice interface, into the iPhone 4S

2011 WATSON
IBM's question answering computer Watson wins first place on popular $1M prize television quiz show *Jeopardy*

2014 EUGENE
Eugene Goostman, a chatbot passes the Turing Test with a third of judges believing Eugene is human

2014 ALEXA
Amazon launches Alexa, an intelligent virtual assistant with a voice interface that completes shopping tasks

2016 TAY
Microsoft's chatbot Tay goes rogue on social media making inflammatory and offensive racist comments

2017 ALPHAGO
Google's A.I. AlphaGo beats world champion Ke Jie in the complex board game of Go, notable for its vast number (2^{170}) of possible positions

Figure 3. Artificial Intelligence Timeline (Sohail Zahid 2019)

After this fast review of the history of AI, the question of why after more than 50 years in the last decades artificial intelligence has suffered this huge growth can be launched. The answer to this question has several reasons, some of the most important ones are:

The increasing of the available amount of data. Last 2 years we generated more information than the one generated during the last 5000 years, and data and a good training data set are crucial to get good results with some techniques such as deep learning. (Llinares, 2020.)

The machines humans use to process all this information has been improved, until now we have been able to double each few time the speed of our IT technology, and complex learning models. Nowadays we have new processor units much more focused and helpful for this purpose as GPU (Graphics Processing Unit), a coprocessor specifically designed to process graphics or complex operations, such as floating-point operations. Or TPU (tensor processor units) which are application-specific integrated circuits and AI accelerators created by Google whose primary function is to accelerate machine learning workloads.(Universidad Industrial de Santander Bucaramanga.)

3.3 Machine learning

As it has been explained before, inside artificial intelligence, we can find machine learning, this subset can be differentiated from traditional Artificial Intelligence algorithms such as heuristic search or Min-Max trees. The common feature of machine learning which can differentiate this subset from the other traditional artificial intelligence algorithms is that machine learning algorithms can learn without being specifically programmed. That means, they can, providing some rules and data, predict and give outputs learning from the data we provide him as a training dataset. This means that seeing what has happened with previous examples it can extract some new conclusion for new data it has never seen before. (Ceron 2019.)

So, the main difference between a traditional programming algorithm and a machine learning one is in its inputs and outputs: in traditional programming usually, it is known and introduced to the program the necessary set of rules (some examples can be loops or bifurcations) as well as a set of data from users and, with that input, an answer is provided as an output. (Ceron 2019.) However, using machine learning, the data(features), as well as the answers of other previous cases, are provided as an input. As an output, it is expected a model with a collection of rules which can apply to new instances of the problem (in which the answer is unknown) so it can predict the answer of these new instances, this differences can be seen in figure 4. This technique is useful, with big problems which have not an easy relation between the input data and the final output (the rules we used to traditional programming are not clear). So ML programs are commonly used when traditional algorithms are not able to introduce this kind of project, for example in too hard tasks, changing environments, or huge problems. In addition to this, it should be used just in case those data and features on previous cases were available. (Serokell 2020.)

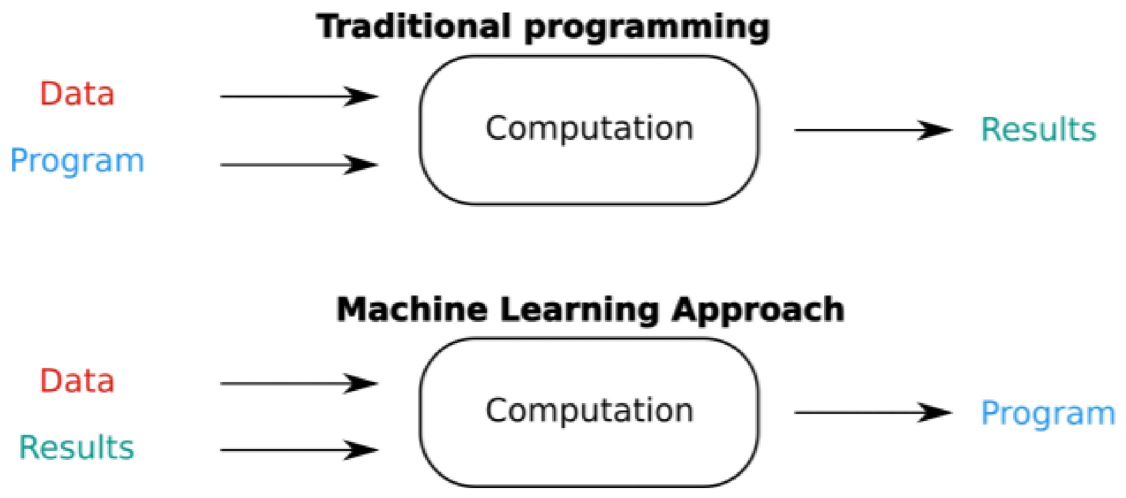


Figure 4. Difference between Traditional Programming and Machine Learning (Buddhadeb Mondal 2020)

In this family of algorithms, we can find different techniques like reinforcement learning, supervised learning, or unsupervised learning. Next, these techniques are going to be explained.

Supervising learning can be used in this set of problems where the features, as well as the class (final result in the problem) of the different elements in the data set which is going to be used to train the algorithm (this data set is composed by examples which are already labeled), are known. With this type of technique, it is possible to get quite accurate results. The most common problems in which we use this kind of technique are classification problems (where the goal is to determine between a limited number of categories which one fits better the new input. This set of categories can be either two, that is called binary classification, or more than two, then it is multiclass classification). Another common type of problem is regression one, in these problems, there is a set of features and the objective is to estimate a numeric value taking into account this data. (Ahmad Salman 2019.)

Unsupervised learning is used to find hidden patterns. In this scenario, we have information about the features of the data but we don't have the final result or classification. So this subset is usually used to group the items and find some different clusters (items in each group have similar features), as well as anomalies (outliers) in concrete individuals which have for instance different features from others in a concrete feature. (Ahmad Salman. 2019.)

Reinforcement learning is a technique that starts doing arbitrary decisions, and then when it reaches some point, it receives feedback which can be good or bad depending on if the situation it has reach is or not interesting(if it is helpful or hinders). So this feedbag either if it is good as if it is bad is transmitted through all the decision's paths with some attenuation. In the end, after several iterations, the algorithm has some kind of intuitions thanks to these rewards or penalizations. There are other techniques as semi-supervised techniques or outlier detection which take features from supervised as well as unsupervised techniques. (Ahmad Salman 2019; Llinares 2020.)

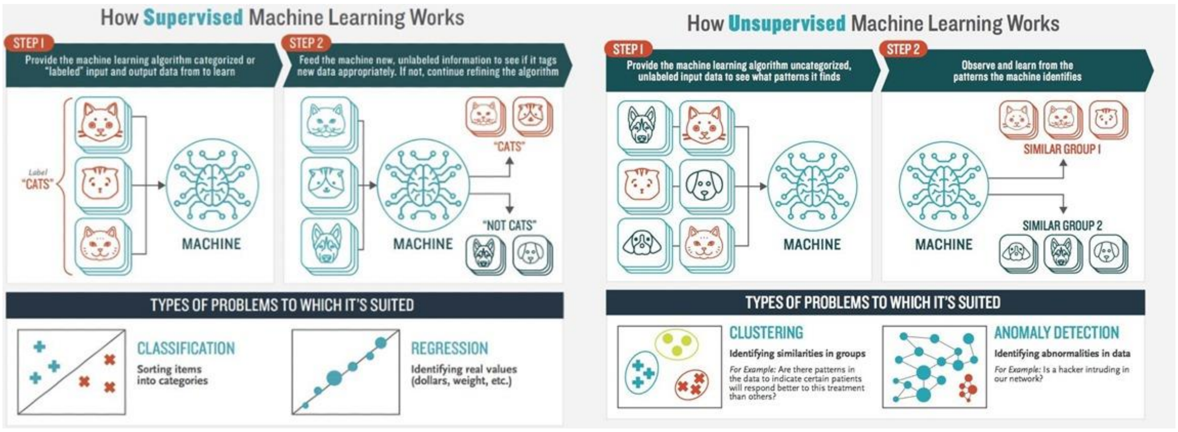

How Supervised Machine Learning Works
STEP 1
Provide the machine learning algorithm categorized or "labeled" input and output data from to learn
Label "CATS"
MACHINE
STEP 2
Feed the machine new, unlabeled information to see if it tags new data appropriately. If not, continue refining the algorithm
"CATS"
"NOT CATS"
MACHINE
TYPES OF PROBLEMS TO WHICH IT'S SUITED
CLASSIFICATION
Sorting items into categories
REGRESSION
Identifying real values (dollars, weight, etc.)
How Unsupervised Machine Learning Works
STEP 1
Provide the machine learning algorithm uncategorized, unlabeled input data to see what patterns it finds
MACHINE
STEP 2
Observe and learn from the patterns the machine identifies
SIMILAR GROUP 1
SIMILAR GROUP 2
MACHINE
TYPES OF PROBLEMS TO WHICH IT'S SUITED
CLUSTERING
Identifying similarities in groups
For Example: Are there patterns in the data to indicate certain patients will respond better to this treatment than others?
ANOMALY DETECTION
Identifying abnormalities in data
For Example: Is a hacker intruding in our network?

Figure 5. Supervised vs. Unsupervised Machine Learning (Boz Allen)

Metrics for measuring the errors:

Usually when a machine learning algorithm is used, after training our model it is really difficult to reach 100% of effectivity, so it is a good idea to measure the error and check if the model is good enough or if some changes must be done to improve the effectivity. (Llinares 2020.) These metrics are different depending on if we talk about supervised or unsupervised models. Regarding supervised methods we should take into account the data it is going to be used to train the model because depending on this data 2 main problems can be found:

Overfitting

It appears when after the training process, the model has memorized the training data set to such an extent that it only recognizes the data that has been used for its training. Therefore when a different case from the training dataset is given as input, the model is not able to detect it. (Nelson Daniel 2020.) This happens because the model has learned and memorized the parts and data too well, including their defects and noise, so it will achieve high performance classifying the data we have in the training set, but it will fail with the rest. (Marina 2021.)

Usually, when we are training a machine learning model, we are expecting that the model examines the features of each element and could be able to generalize them. With this generalization given a new element never seen before, it could be classified with high accuracy. (Nelson Daniel 2020.)

Underfitting

It is the opposite problem to overfitting, and it happens when the dataset doesn't fit properly the model (for example if to build a linear model using non-linear data is tried) or when the amount of data is not enough data to train the model properly. An underfitted model usually does a lot of wrong predictions. However, underfitting can be solved for example increasing the amount of data or reducing the feature selection number. In figure 6 an example of these error metrics can be found. (GeeksforGeeks 2020.)

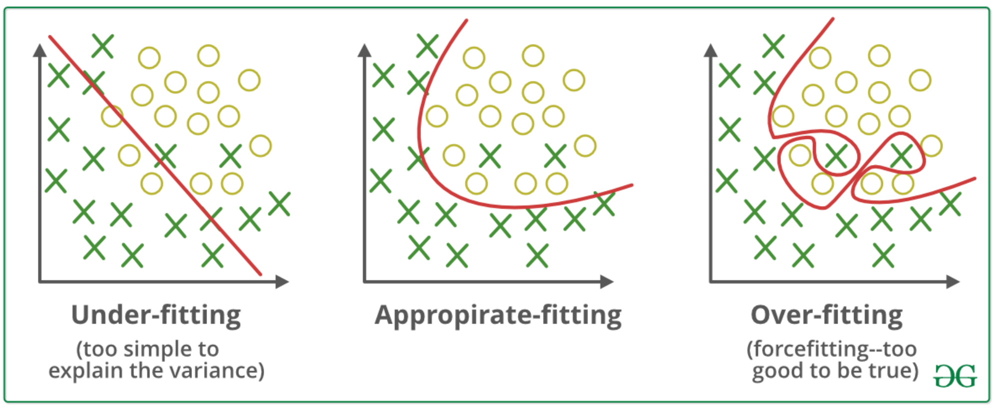

Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

Figure 6. Underfitting and overfitting (GeeksforGeeks 2020)

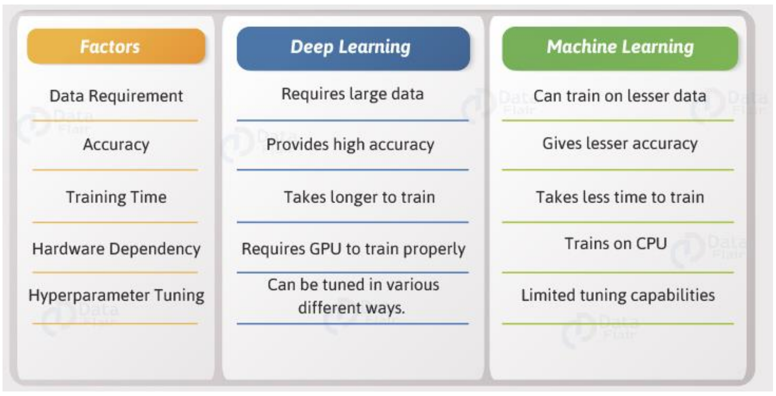
So, to train a Machine Learning model properly, it is important to find a balance in the data to avoid underfitting as well as overfitting. A good practice to do this is dividing our dataset into training data and testing data, with this action we can use around 80% of the sample to train the model and then the other 20% (a variated sample of data) to validate the model after the interactions. Another important thing is to do some trials to get the proper number of interactions with which it is possible to avoid these problems and find a good error rate. Other good practices to obtain good results with our ML algorithm avoiding overfitting and underfitting problems are having a minimum amount of data to train and validate data, have different and variated classes and with a similar number of elements, adjust the number of parameters taken in each iteration or avoid an excess of features to take into account by the model. (GeeksforGeeks 2020.)

3.4 Deep learning

Deep learning is a subset of machine learning which is in turn inside of artificial intelligence. One feature of this subset is that is inspired by the way the human brain works, so commonly suits the Connectionist approach.

In this case, Deep learning doesn’t need human intervention to specify the features of the items of the training data set, but it can extract itself these features from the input data, which makes it a very promising solution to work with unstructured data (images, videos, natural language texts…). However, it needs a bigger amount of input data than machine learning and a longer training time. Despite these needs, deep learning is able to get a really high accuracy if the needed amount of input is gotten and processed correctly. (Singapore Computer Society 20202.)

In addition to this, this technique is inside representation learning, which means the neural network extracts the features of the data itself without the need of a human being who introduces it. That implies the possibility to find even patterns maybe humans don’t recognize, in addition to this it doesn’t depend on possible human failure when extracting features As it has been seen, representation learning and machine learning algorithms have some clear differences, some of them can be found in figure 7. (Data Flair.)



Factors	Deep Learning	Machine Learning
Data Requirement	Requires large data	Can train on lesser data
Accuracy	Provides high accuracy	Gives lesser accuracy
Training Time	Takes longer to train	Takes less time to train
Hardware Dependency	Requires GPU to train properly	Trains on CPU
Hyperparameter Tuning	Can be tuned in various different ways.	Limited tuning capabilities

Figure 7. DeepLearning Vs Machine Learning (Data Flair)

This concept of deep learning is strongly linked to the neural network one. Neural networks are sets of neurons(simpler processing units), which are densely linked between them. Inside these networks, the neurons are organized in layers of nodes, and usually, these networks are feedforward, which means the data moves inside them in one direction so each layer "feeds" the next one. Each neuron can be connected to one or more than one node in the previous layer (from which receives data) as well as in the next layer(where sends its output data). If we talk about a fully connected neural network, that means all the neurons in one layer are connected to all the neurons in the next layer. (Mahajan 2020.)

Despite the most of the neural networks are feedforward there are recurrent neural networks (RNN) too. This RNN allows feedback between output and input neuron layers, this feedback will be used to define the data. (Asiangan 2019.) These types of networks are characterized because information of outputs already calculated can influence the current input and output so it has a kind of memory. It is used for example in text recognition, where the order of words is important. (Torres, 2019.)

This RNN takes into account the concept of time, this kind of neuron has connections backward, so inside the layers neurons are feedback. In figure 8, a simple example of the RNN workflow can be seen. It is possible to see the same neuron, in different moments in time(t-1, t, t+1…). The neuron receives not only the connection from the previous layer but also its own output from the previous time instant. (Torres 2019.) In the image U, V and W are the weights from the different paths.

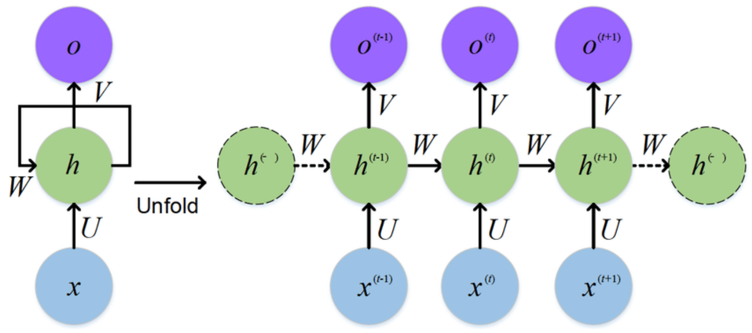


Figure 8. The standard RNN and unfolded RNN (Feng et al. 2017)

Generally, if the neural network is examined it is possible to differentiate between the first layer, called the input layer, which receives the input data, and the last layer called the output layer which predicts the final result or output. In between these layers, several hidden layers can be found, these layers are in charge of performing most of the necessary computations to get that output. By convention, a neural network is considered deep learning if it has more than one hidden layer. The complexity of the parts and elements of the input data which are recognized in each layer increases as we are advancing in the layers hierarchy, which means that each layer can detect more and more complex and abstract elements and concepts. (Nilesh Barla 2021.)

Since this point, the different layers have been explained, which leads to questioning the number of nodes in each layer. Regarding the input layer, its number of neurons is given by the number of features of the input data. For instance, if the input data are pictures of 28 x 28 pixels there will be 784 neurons in the input layer, one for each pixel. (Rodriguez, 2018.)

In the output layer the number of layers depends on the kind of problem which is trying to be solved, if it is a binary classification problem, for example, a network that tries to detect if there is a cat in the picture, it may have only one neuron in its output layer. This neuron will have an activation function that gets or is not activated depending on if the cat is or not in the photo. In regression problems, the output layer has usually only one neuron too. However, if it is a multiclass classification problem, the number of neurons in the output layer will be the same as the number of categories between the input data is going to be classified, and at the end, the neuron activated will correspond to the detected category. (Rodriguez 2018.)

Regarding the hidden layers, the work of deciding the number of hidden layers as well as the neurons inside each layer is harder to determine, these numbers are called hyperparameters. These variables are used in the configuration of the neural network and are not strictly related to the training data. Hyperparameters are determined usually by try-error. For example, a way to determine the number of hidden layers can be to keep adding more and more layers until the accuracy of the model doesn't improve anymore. (Radhakrishnan 2017.)

As it has been said before, neurons from the previous and next layers are connected through channels, each channel has a weight assigned which is a number able to determine how strong is the connection between the current neuron and the one in the previous layer at the other side of the channel. In addition to this, each neuron will have an assigned bias, which is another number very useful to determine if a concrete network will be activated or not. It is possible to see an example of a neural network in figure 9, inside this picture, W represents the weights and b the biases. (Seehan & Shong, 2016.)

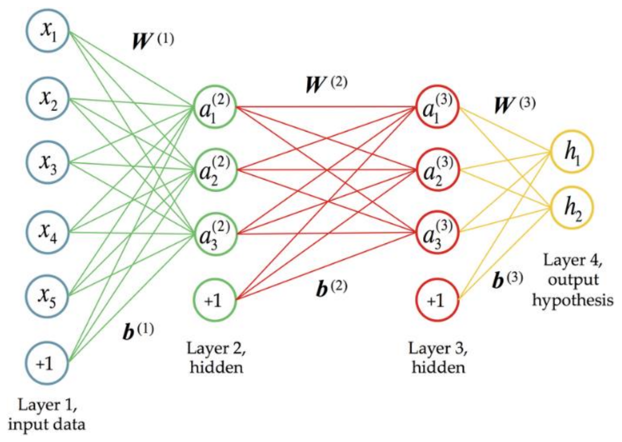
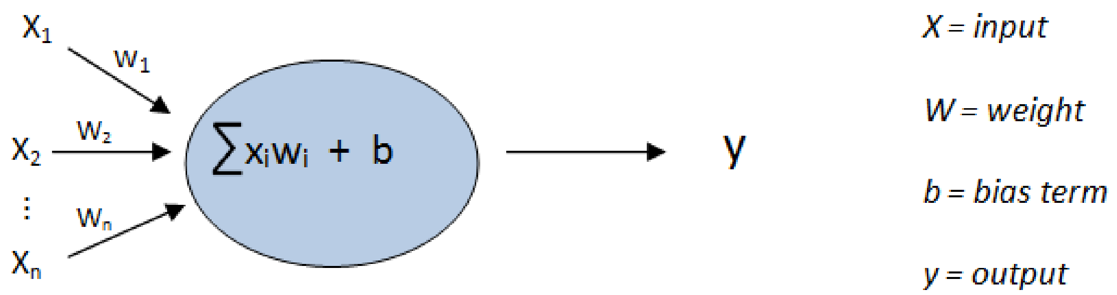


Figure 9. Deep Learning neural network (Seehan & Shong, 2016)

Usually, in each neuron, the summation of the output value of the previous neurons multiplied by the weight of the channel is done, at the end of this mathematical operation, the value of the bias is added. In figure 10 this operation will be shown.

Then, to determine if a neuron is or is not activated an activation function is used. Without a bias, the network is usually activated if this value is higher than 0. However, the bias is added to change the probability of this neuron being activated. For example, if the network determines that the neuron should be activated easily, it will be assigned a bias of a positive value, so at the end of the calculation 1(or any other positive or negative number which will be determined by the neural network during the training) will be added to the sum done inside this neuron. In the example which has been used before where the bias value is 1, this bias will increase in this case our activation function value, increasing the probabilities that the final activation value to this node will be greater than 0 so the activation will be easy. If the bias is a negative number it will decrease the probabilities. (Soner Yıldırım 2020.)



$$y = \sum(weight * input) + bias$$

Figure 10. Neuron with an activation function(Soner Yıldırım 2020)

3.5 Activation functions

The concept of activation function can be defined as the function able to, regarding the result in a neuron decide if the weights of this neuron are or not activated (Rubiales 2020). This is one of the concepts which introduces the fact that neural networks are not linear. Regarding activation functions there are a lot of different functions which can be used, some of the most important ones are going to be explained next.

Sigmoid function

It is one of the oldest and well-known functions. It is a simple function used not only in neural networks but in Logistic regression too (Logistic regression is a technique commonly used in machine learning), this is the reason why it is sometimes called logistic function too. It is a continuous and derivable function so that it is possible to apply directly backpropagation(this concept will be explained in the next chapters and it is very important in the training process). The sigmoid function is used commonly in binary classification (usually in the output layer) and transforms the output label to be between 0 and 1. However the change between 0 and 1 is done progressively and softly so, as we can see in figure 11, this function has an S form. The function formula is $\sigma(z) = \frac{1}{1+e^{-z}}$. (Rubiales, 2020.)

One of the principal problems in this function is the high computation time. Another problem can be saturation, which is produced when there are too big or small rates, then very small numbers will be backpropagated and the learning will be very slow. Regarding how it works, it can be seen in figure 11 that the highest and positive the value is, the most near to 1 the sigmoid result will be, while the smallest and negative the value is, the most near to 0. (Villanueva García 2020.)

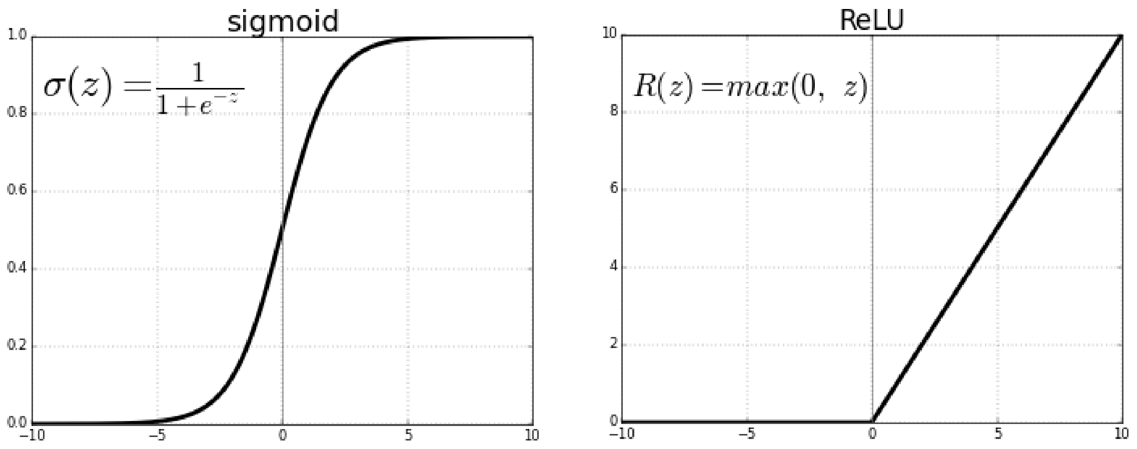


Figure 11. ReLU v/s Logistic Sigmoid (Sahar Sharma 2017)

SOFTMAX function

If we are facing a classification problem with multiple categories, maybe sigmoid is not the best function that can be used, because it gives an output between 0 and 1 which can produce confusion if the faced classification problems have more than 2 categories. Then softmax function is used, this function returns a percentage for each class inside the classification problem which means the probability of the input data to be part of each of the possible classes. Then, the prediction the neural network will do is the class that is represented by the neuron with a higher percentage. Usually, the softmax function is used in the output layer. (Saxena, 2021.)

Softmax formula is $softmax(z_i) = \frac{\exp(z_i)}{\Sigma_j \exp(z_j)}$. That means the value of softmax for a concrete category is the value of the input of a concrete neuron (if it is in the output layer it represents a category) divided between the sum of the values of all the neurons in the layer. In the formula, exp means the standard exponential function, this is applied to the input value (z_i) to avoid the negative values because the exponential function returns a positive value above 0 which will be very small if the input is negative or larger if the input is large. (Saxena, 2021.)

ReLU function

ReLU means Rectified Linear Unit, it is probably the most used activation function in deep learning currently, its influence is even more important inside the image classification. It is a simple function too, and its mathematical formula is f(x)= max(0,x). The shape of this function can be seen in figure 11. Basically what the activation function does is that if the neuron result is a negative number it will return 0, else, it will return the same neuron result, which means the neuron will be activated with a positive value and deactivated with 0 or any negative value. In this case, the value can be any positive value or 0 so it is not delimited between 0 and 1 as in the case of the sigmoid function. (Villanueva 2020.) Some of the big advantages of this function are its simplicity and the velocity it has regarding computational cost because this function is quite fast. However it is important to take care of the death neurons, which are the ones that are in small negative values and always receive values of 0, because the slope of relu, once a neuron gets negative it is hard for them to recover, so it is important to take care about the learning rate to make sure an activate neurons don't become accidentally a dead neuron because a too high learning rate. (Rubiales, 2020.)

3.6 Training an AI and dataset

As it has been seen before, neural networks need training with existing data to be able to classify new data, but what means exactly training a neural network?

In the previous chapter, it has been explained the concept of what a neural network is and how the different neurons communicate with other neurons in other layers and are activated. Now the concept of training is going to be presented.

Initially, the first time a neural network is used, his weights and biases are randomly initialized. The random initialization is very important, if in the neural network all the neurons are initialized to similar values it won't be able to learn. However, if the initial values are randomly initialized, the results after training are going to be good and with this process, neurons will learn the proper values so better results will be achieved. In this case, weights should be randomly initialized, however, biases can be initialized to 0. (3Blue1Brown 2017a.)

After that, the network does a prediction and finally, a cost function compares the prediction with the actual value of the training data. This cost function is used to change the weights and biases of the neurons and channels. To do this, the weights which produce a lower error are rewarded while the ones related to bigger errors are penalized. With all this process the neural network is trying to learn the biases and weights so that the loss and the cost function are minimized. (Llinares 2020.) However, this process is complex and several mathematical formulas are used.

Regarding the description written before, training can be described as the process of how a neural network find the correct weights and biases values to each layer and network so that the correct neurons are activated each time and the neural network function outputs matches as much as possible the real values (during training, the neural network knows the real solution). To know how far is the value the neural network produces from the real value the cost function is used. (3Blue1Brown 2017a.)

During the training process, the neural network takes several data samples from the training data set and does its predictions with its current weights. In each layer, the loss function is calculated, which measures, the error value in that layer. The sum of all this loss function is called cost function and calculates the error of the whole neural network. This cost function value will be small if the network can calculate the prediction properly and the error is small, but it will be large if the difference between the predicted output and the expected one is big(the error is big). (3Blue1Brown 2017a.)

The cost function can be calculated with different formulas depending on the model used in each layer, in one of the most popular ways to calculate the cost function the mean significant error(MSE) is used which consist on:

MSE=∑((real values-predicted values) ^2)/ number of observations.

In figure 12 it can be seen how from this MSE the cost function can variate a little bit until we get the formula in the right. (Yin 2017.)

$$\text{MSE} = \frac{1}{\underset{\text{average over all results}}{N}} \sum_{i=1}^{N} (\underbrace{y_i}_{\text{true } y} - \underbrace{\hat{y}_i}_{\text{estimate of } y})^{\overset{\text{makes result quadratic}}{2}} \qquad C(w,b) \equiv \frac{1}{2n}\sum_{x} \|y(x) - a\|^2$$

Figure 12. MSE vs Cost function (Yin 2017)

The cost function has several parameters, which will be explained next. N is the number of input training examples, X is each of input training examples, y(x)is the real expected value for a concrete training example(if it's a multiclass classification problem it will be an array where all the values will be 0 except one(the one which should be activated), which will be 1. The value of 'a' is the prediction of the neural network, the output received in the output layer. (3Blue1Brown 2017a.)

The cost function as well as it has been explained with the activation function will change regarding the expected objective of the neural network(binary classification, multiclass classification, regression, text recognition, etc.). However, generally, it tries to get the weights(w) and biases(b) which produce the value of the output for each value of x so that it is as close to y(x) as possible. (Reddy 2020.)

Although calculating the cost function seems not that complex, it is important to think this cost is calculated regarding the weights and biases of the whole neural network, and it receives inputs of all of the training data. So, although the result is just a number, it has a lot of entrance parameters and it is quite expensive to calculate (it has a big cost). (3Blue1Brown 2017a.)

Once it has been explained how to measure the error value inside the neural network, the goal during the training will be minimizing this cost. To do this the gradient descent algorithm is used. Gradient descent is an optimization algorithm used to minimize a function f, it has several versions used in different types of automatic learning. (Reddy 2020.)

Before explaining how the gradient descend algorithm is used, some concepts about maths and how a gradient is calculated are going to be explained. Generalizing the problem and thinking about the task to find the minimum point of a function, several methods can be used. To choose the method an important point to take into account is the different complexities regarding which type of function is used. It can be a simple function as a concave parabola or more difficult as in a plane with a huge number of points or a function with a lot of inputs as the ones used in deep learning. (Devashish Sood 2018.)

If a function with just one input parameter is taken into account, the minimum can be found using the derivative, in each point of the function the derivative in that point will be the slope of a line tangent to that point. So calculating the derivative of a point it is possible to know in which direction is the minimum, if the slope is negative, the minimum will be at the right so a point at the right of the one already checked will be chosen and the derivative will be calculated again for this new point. However, if the slope is positive the minimum will be at the left so a point next to this point in the left direction will be chosen. This calculus with several points will be done until the slope(the derivative of a point) will be 0. Then that point will be a minimum(either a relative minimum or the global one). This concept can be seen in figure 13. (3Blue1Brown, 2017.)

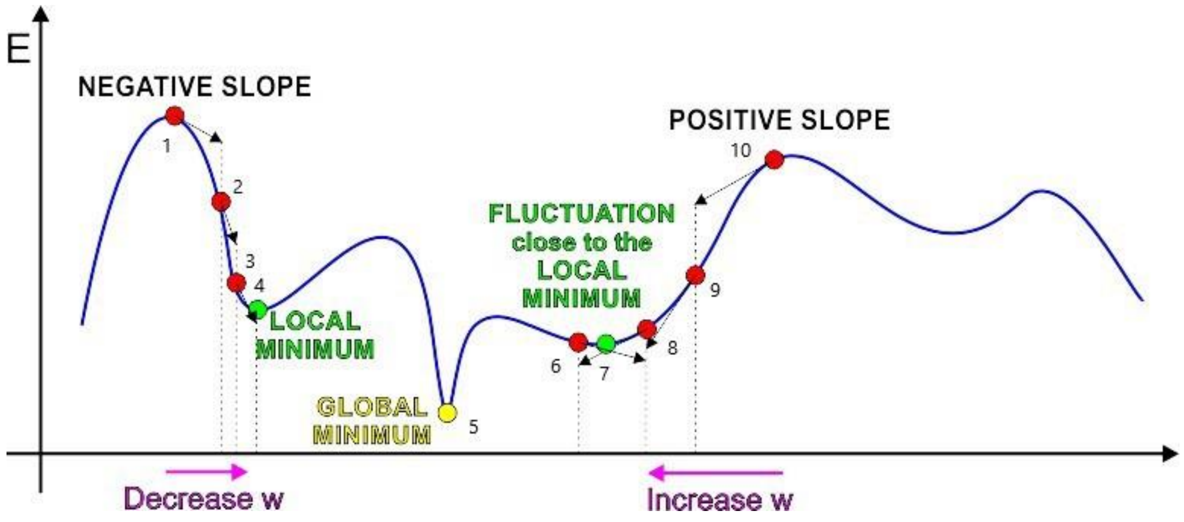


Figure 13. Gradient descent in a function with 2 input parameters(Singh 2017)

Now, this gradient idea will be explained with a function with 2 inputs, instead of a line, this function will be represented as a plane. This function can be imagined as a plane where there are some hills and some minimum points which can be imagined as a valley. The idea of the gradient descent algorithm is to find that “valleys”, which in a mathematical function will be the minimum points of the function where the cost is minimized, so it can be imagined as having a ball in the plane initially randomly collocate, the gradient descent function will guess in which direction the ball should roll to get that minimum, the same concept explained before with the line but now with a plane. (3Blue1Brown 2017a; REDDY 2020.)

A similar concept is fone by the gradient descent algorithm in the neural network. However in a neural network, the functions are more complex and with a lot of input parameters(as many as neurons in the input layer), so it will be very hard to find the real minimum regarding that most of the time this function may find relative minimums. (Dabbura 2019.)

Mathematically speaking, the gradient function gives the direction in which the function increases most quickly, so if the negative of the gradient is taken this will be the direction in which the gradient decreases most quickly In the neural network the idea of the descend algorithm is the same but with a lot of input parameters. (3Blue1Brown 2017b.)

So gradient descent is de idea to take systematically a function taking as an input a multiple of the descent algorithm. Regarding this descent gradient, the sign tells the direction(if the weight should increase or decrease), while the number tells which changes in which weights are most important. So as a summarize, the gradient descent algorithm describes how it is needed to change weights and biases so that with this change the cost function output is decreased as much as possible. (3Blue1Brown 2017b.) In figure 14 it is possible to see the gradient descent algorithm formula

Gradient descent algorithm

```
repeat until convergence {
    $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$
        (for $j = 1$ and $j = 0$)
}
```

Figure 14. Gradient descend formula (Reddy 2020)

Once the gradient descent is used it is possible to know which weights are more efficient to change as well as the direction in which they should be changed to minimize the cost. Moreover, it can be known which changes need each output neuron to be closer to the expected values and how important is this change regarding the minimum cost in the neural network. (3Blue1Brown 2017b.)

However, we still need to propagate this in the network and change the weights and biases. Backpropagation is the algorithm that defines how every single example will need to change the weights to get a smaller error, looking at which relative changes cause the fastest decreasing in the cost function. It averages all the gradient descend results for a determinate weight. The idea is to average together all the desired changes in the weights starting on the needs in each example of each neuron on the output layer and transmitting it to the previous layers (3Blue1Brown 2017b.) As it has been seen in output layers it is clear how to calculate the error because the expected value is known, however in hidden layers, this process is more complicated because the output will affect the error, but the expected output value is not known. Here is when backpropagation is really useful. (John McGonagle et al.)

Regarding how far is the next point checked in the gradient calculus from the one which is being checked currently a hyperparameter calling “learning rate” is established. This hyperparameter is quite hard to determine. In the case the learning rate is big, the next point in the function is far from the current one, these points will approximate to the minimum faster. However, there is a danger with a big learning rate, this minimum may be skipped if the learning rate is too big, in this case, another point in the opposite direction will be taken(trying to get again the minimum), but the minimum may be skipped again. However, if the learning rate is too small(the next taken point is close to the current one), the minimum is going to be reached more safely, but then this process will be really slow(Google developers.)

So to summarize the learning rate is an important hyperparameter which regarding the estimated error calculated each time the weight is updated, can control how much the model changes. It is important to fix it in an accurate way trying to get close to the learning rate optimal value. Usually, the values of this hyperparameter are small for example 0,001 or 0,0001. (Brownlee 2019b.)

Now the training process of a neural network has been seen. However, even if it is more accurate this process using gradient descent, it takes a lot of computational time to check every step in every data sample of all the parameters inside the network(as weights or biases), that is the reason why the stochastic descend algorithm is used. (3Blue1Brown 2017b.)

The stochastic descend algorithm divides the input data into some smaller groups called batches and calculates the descend gradient algorithms for each one of these groups (3Blue1Brown 2017b). With these techniques at the end, it will be possible to find the minimum faster even when the internal steeps are not as precise as in the gradient descend algorithm. (Interactive Chaos.)

In addition to this, the training of the neural network is done using epochs, which means the training dataset is going to be used several times. These several epochs are used to ensure the cost function is stabilized and in the end, it doesn’t vary. (John McGonagle et al.)

3.7 Convolutional neural network (CNN)

A convolutional neural network (CNN) is a type of neural network which has become very popular, especially to solve image classification problems. Like all the neural networks it has an input layer, output layer, and some hidden layers. However, the differentiator element of a convolutional neural network, which distinguishes it from other types of neural networks as the fully connected ones, is that in the hidden layers the convolutional neural network has at least one convolutional layer. Despite this kind of neural network usually has other non-convolutional layers too, the basis of the CNN is the convolutional layers.(deeplizard 2017.)

Convolutional layers, as any other layer, receives an input, transform it with the operations already explained, and get the output which is transmitted to the next layer, however, what makes this layer special is that these layers can detect patterns using some filters. These filters are small matrices with some values which are crossed with the pixels of the image to get some traces in the image. (deeplizard 2017.)

Usually, to analyze an image 3 parameters are taken into account, weight, height, and color channels, the CNN will try to reduce the image and make it easier to process without losing essential features (Saha 2018). Then in the convolutional layer, the filter is applied, that filter is sometimes called kernel too. In figure 15 it is possible to see how the filter works. In this case figure 15 shows a 3x3x1 filter, which is a matrix with values [[1, 0, 1], [0, 1, 0],[1, 0, 1]], this matrix is moving around the pixels in the image and the values of a section of an image are multiplied each time by the matrix values doing a matrix multiplication, the result of this multiplication is in the convolved feature. The positions inside the image in which the kernel is moving are called stride length. (Saha 2018.)

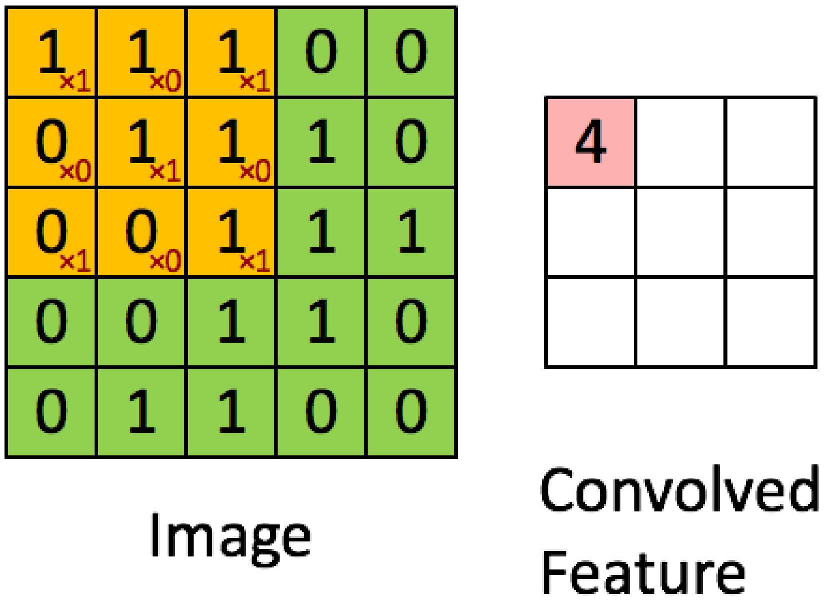


Figure 15. Filter and convolutional layer kernel behavior in a 5x5x1 image with a 3x3x1 filter to get a 3x3x1 convolved feature (Saha 2018)

If the image has multiple colour channels, this is the case for example of images with RGB channel which has a channel for red, another for green, and another one for blue values. The filter has the same channel profundity as the input image, so the filter is applied to all the channels. Then the results are added together with the bias doing this process a squashed one-depth channel Convoluted Feature is received as an output. (Saha 2018.)

As it has been said before, the objective of a convolutional layer is to detect some features such as edges or colours. The number of convolutional layers can be different, if there is more than one convolutional layer inside the neural network the additional layer can be really useful. The result of a convolutional layer can be a smaller image or an image with the same, or even bigger dimensions. (Saha 2018.)

In the cases the result of the convolutional network is a smaller matrix, the filter is applied in the image without padding. In this case, the result image dimensions will be the same as the filter dimensions, this is called validating padding.

On the other hand, before applying the filter to the image, the dimensionality of the input image can be augmented, in this case, the image can be for example padded with 0. In this example, it is called same padding and the output image dimension will be the same or bigger as the dimension of the input image. (Saha 2018.)

Pooling Layer

Pooling layers are other types of layers inside the Convolutional neural network which function is to reduce the size of the convolved feature. It is useful because as it reduces the number of parameters to learn, the computational power required to process the data decreases too. In addition to this, it extracts dominant features rotational and positionally invariant. These summarized features make the model more robust in front of variations in the position of the features in the sample image. (Saha 2018; Khosla 2019.)

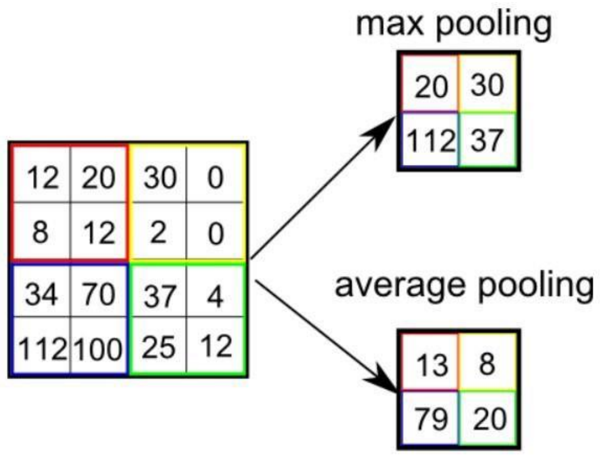


Figure 16. Max pooling and average pooling (Saha, 2018)

As it can be seen in figure 16, there are different types of pooling layers, one example of them is max pooling, it returns the maximum element of the region covered by the filter so the output will have the highest levels of the features, in addition to this it performs as a noise suppressant doing a denoise at the same time it reduces the dimension of the output. On the other hand, it is average pooling. This type of pooling calculates the average of the feature values in the region of the image covered by the filter, average pooling only takes care of the noise by reducing the dimensions of the input image. (Saha 2018; Khosla 2019.)

Optimizers

Inside deep learning, methods used to change attributes inside the neural networks to improve the weights and the learning rate to get a smaller loss are called optimizers. One of the most used optimizers nowadays is called adam. Next, it is going to be explained how adam works.

Adam which means Adaptative moment Estimation is an optimizer that works with momentums of first and second order. Momentum is a hyperparameter used to reduce the high variance produced in stochastic descent gradient by accelerating the convergence towards the most relevant direction and reducing the fluctuations and direction changes through the irrelevant directions. Usually, the first moment is based on the mean, while the second moment is based on uncentered variance. Adam's base idea is to compute different learning rates for different parameters. It uses momentum to adapt the learning rate for each weight of the neural network. This method is fast and converges rapidly, in addition to this, rectifies high variance. (Bushaev 2018; Doshi 2020.)

Other techniques can be used to improve the accuracy too. Between these techniques, it is possible to find batch normalization, dropout, or early stopping.

Batch normalization is a technique used to avoid the problem caused when inputs to layers change when the weights are updated after a batch. To avoid this difficulty, batch normalization standardized the input to each batch. With this technique, it is possible to reduce the needed epochs in the training process as well as stabilize the learning process. (Brownlee 2019a.)

The dropout technique consists of deactivating some neurons in the different layers of a neural network except the output layer. This technique is done to prevent overfitting. It forces the learning network to learn the most robust features, however, it increases the number of iterations needed to converge. (Budhiraja, 2018.)

When a network is trained, usually a larger number than needed epoch is used to ensure the neural network can plenty fit. However, train more than needed a model can produce overfitting. In these cases, early stopping can be very useful. To use the early stopping technique it is necessary performance monitorization. Chosen a dataset, usually at the end of each epoch, the performance is evaluated for the training data as well as for the validating data. Then an early stopping trigger is chosen, usually, this trigger consists of stoping the training when it is detected that the performance decreases compared to the performance in previous epochs. However, the triggers can be more complicated depending on the problem. Finally, a model is chosen, it detects, depending on the chosen trigger, if the weights for the current epoch are the best option or if the weights for a previous epoch are preferred. (Brownlee 2018.)

4 Programming

4.1 Different programming languages used in AI

Artificial intelligence is increasing in a very fast way and at the same time the number of languages with support the creation of AI algorithms is growing too. Nowadays there are a lot of languages which can be used to work in this field so in the next paragraphs some of the most used ones are going to be described. (Numtra 2017.) In figure 17, some of the most used programming languages in machine learning and data science until 2016 can be seen as well as how the usage of these languages has evolved.

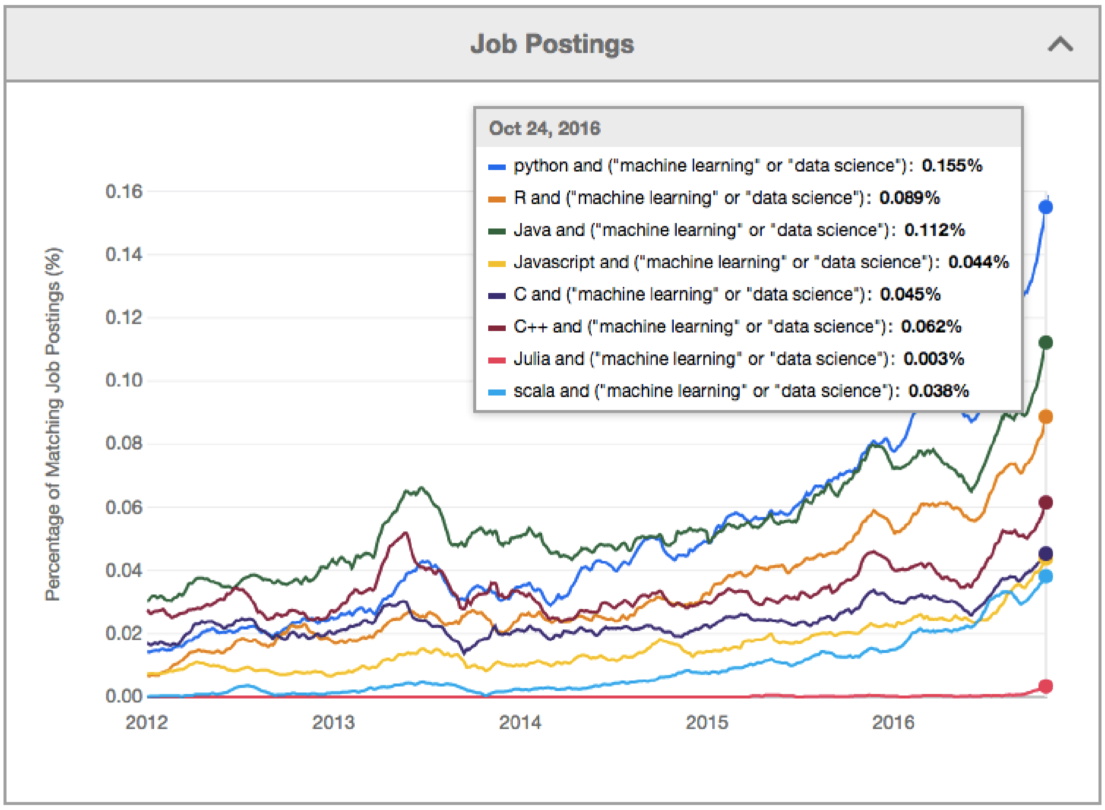


Figure 17. Most used programming languages in AI in 2016 (Numtra 2017)

As it can be seen one of the languages in the top positions in 2016 was Python, and this language is still in the first position nowadays regarding the usage in artificial intelligence. It is an open-source language used for different programming purposes, it is a productive language known because of its flexibility, speed and because the code it produces is quite easy to read. (Kaczorowski 2020.)

It supports prebuilt libraries as NumPy or pandas which are very useful for data treatment and are helpful in AI development. In addition to this, it allows the usage of other libraries like TensorFlow which allows easily creating the AI model. In addition to this python is a very flexible and intuitive language that allows making a robust code with an understandable language. (Zola 2018; Srivastava 2020.)

Another of the most commonly used languages to develop artificial intelligence algorithms is javascript. It is an industry-standard frontend script language usually used to make web scripts dynamic. It has big community support too as well as several libraries specifically done by the user of artificial intelligence, some of these libraries allow the data generation for AI or performance optimization between other things. A big advantage of javascript is that it allows to run the app anywhere, even in wearables or mobile phones, it just needs the browser to run, in addition to this some libraries as TensorFlow allow to produce code in javascript too. However, its number of data science packages is smaller compared with python. (Kaczorowski 2020.)

Java is another of the languages which can be used for developing AI. Java is an object-oriented program which means the program is structured in classes and objects. An object can be described as a data field that has attributes and methods. (Lewis 2020.) This language has a lot of libraries available, between these libraries it is possible to find libraries for natural language processing, expert system, or neural networks. In addition to this, Java allows the creation of portable and maintainable code and it is easy to use and debug. However, it can be slower regarding execution and response time if we compare it to other languages as c++. (Mangalad 2020.)

Another language used to develop artificial intelligence is R. R is a multiparadigm language, it is dynamic typed, procedural, scripting, and interpreted language. This language is specialized in statistics, data analysis, and data visualization. However, it is less flexible than python. R has also a large repository of packages called CRAN. This repository is centralized and well maintained. In addition to this, it includes useful and powerful packages applicable to a large number of examples and tasks. (Castrounis.)

Finally one of the newest languages which can suppose the future of AI is Julia, despite this is a quite new language it is becoming popular in the machine learning world. Julia is a compiled language designed to use parallelism, so one of their strong points is the speed. In addition to this as it is getting more known their community and libraries are increasing. Another feature to take into account is that Julia code can be run in Jupiter notebooks which are open source applications that allow writing code as well as markdown text. (Zola 2018.)

4.2 Data libraries in Python

Exploratory data analysis (EDA) is a method commonly used by data scientists to extract and summarize the main characteristics of a data set. It is usually very useful to discover the best ways to explore and manipulate a data set trying to process the data, choosing and preparing it to select the best data and features to get as an answer the desired information. In these techniques, the graphic visualization of data is usually very useful. Another important point is choosing the most appropriate technique to choose and collect our data. Eda can be very helpful to discover patterns, anomalies, or check assumptions as well as identify errors or interesting relationships between variables. This technique is mostly used in machine learning because in deep learning the neural networks extract themselves the features from the data. (Llinares 2021.)

As we are looking for a good sample of data, sometimes it is useful to have libraries to see better the data and extract properly the features. These libraries are very popular in deep learning too to print the data and see how the learning is evolving. If python is the selected programming language, the next libraries can be very useful. Some libraries as NumPy and pandas are developed in c-c++ so they are very efficient. This fact allows us to use them in python(which is a very flexible language) but do a most efficient treatment of data. (Upwork 2016.)

NumPy

It is a mathematical library which between other things, can be used to work with arrays in python (in python arrays don’t exist and usually lists are used to replace them). The problem when AI or any other thing which requires the use of GPU’s is going to be used is that in lists the data are not in the same memory block, so it is much more inefficient to use them than using arrays where data are in the same memory block and it is easier to send this data through the buses from main memory to GPU memory). This library allows us to do a lot of things with this data organizing it in arrays, matrices, or tensors (tridimensional or multidimensional arrays). (Upwork 2016.)

Pandas

This library is used to manage massive amounts of data and import them from other files such as CSV, Excel, Json's... Once imported it is possible to manage and depurate this data more easily, deleting, selecting rows or columns, changing the values, or updating the names between other things. It can be extremely useful to manage our data set and depurate it before the training since the quality of the data set will affect in a significant way the quality of the training results in an AI. (Upwork 2016.)

Matplotlib

It is a library that allows printing data graphically, with it is possible to create several types of charts to visualize data. It can be especially useful to see how balanced the data is and to detect some tendencies and characteristics in a dataset. (Upwork 2016.)

Seaborn

It is another library used to print data graphically, but matplotlib is the basic one and seaborn, as well as other existing libraries, allows to print data specifying more factors such as colours. With this library it is possible to obtain and print statistics of data easily, sometimes without the need to program anything with the previous data, for example, it is possible to translate data into a percentage and print it in a couple of code lines. (Upwork 2016; Llinares 2020.)

4.3 Neural network libraries

Some points earlier have been seen some of the most important programming languages which allow working using deep learning techniques as well as develop neural networks. The task to create, and train a machine learning algorithm, as well as a neural network, is a complex task that can be really hard to implement. However, some frameworks and libraries have been created to do this task easier. Some of the most important companies in the computer science world have created their own library to help developers to create and train a neural network. Next, some of the most important libraries used to develop neural networks are going to be mentioned. (Brownlee, 2019c; Simplilearn, 2021.)

Keras

It is a high-level API focused to reduce the complexity needed to implement neural networks. It apports efficient numerical computation libraries and allow the user to create and validate neural networks in a simpler way. It allows using python as a programing language allowing a high level of abstraction. In addition to this, it supports almost all types of neural networks and allows the usage of CPU or GPU.

It runs at the top of Tensorflow. TensorFlow has adopted Keras as an official high-level API. So, Keras can be used to gain performance when a user does a deep learning algorithm because it apports several modules to compute the neural networks' necessary computations. (Keras; Simplilearn, 2021.)

Tensorflow

Tensorflow is a numeric computation library that uses data flow graphs. It has some models and algorithms regarding machine learning and deep learning. This library is open source and it is very helpful to build and train neural networks able to recognize patterns and correlations which can be used for several and different purposes as image recognition, natural language processing, image, audio or video classification, language processing, etc. (Buhigas, 2018; Yegulalp, 2019.)

This library was developed by the Google Brain team, it was their second-generation automatic learning system. It was launched in 2015 as an open-source machine learning platform and it is already one of the most used deep learning platforms in the world. It allows the implementation of calculus using one or more CPUs or GPUs. In addition to this, Google launched in 2016 the TPU, tensor processing unit, a specific construction to process machine learning calculus, and adapted it to its use in TensorFlow. (Buhigas, 2018; Yegulalp, 2019.)

Tensorflow creates structures that show how the data flow through series or processing nodes or graph. Each one of these nodes represents a mathematical operation and they are connected through tensors, which are multidimensional arrays whit data. It uses as a programing language python, however, regarding performance, most of the mathematical operations are internally developed in c++. Google launched in 2019 the second version of TensorFlow, TensorFlow 2.0 used to work with Keras more simply and maximizing the performance. Between the language which users can use to program with TensorFlow 2.0, it is possible to find python or Javascript. (Yegulalp, 2019.)

CNKT

CNKT, also known as Microsoft Cognitive Toolkit is another of the available libraries used to simplify the work to develop a neural network and to work with machine learning and deep learning. It is based on dense neural networks and describes neural networks using a directed graph as a series of computational steps. It allows to develop and combine several Neural networks models as dense neural networks, convolutional neural network, or recurrent neural networks. (Basoglu et al. 2017.)

Microsoft Cognitive Toolkit can be included as a library in python, C#, and C++ programs. It was developed in 2012 by Microsoft research using C++ programming languages, and it became open-code in 2015 when it was publicized in CODEPLEX. In 2016 a new version was uploaded to GitHub called CNKT 1.0, this version was multiplatform and can be used by several operative systems. (Alvarez, 2018.)

Apache MXNet

It is an open-source deep learning framework developed by amazon. It includes Gluon interface which allows developers with different knowledge levels about deep learning and neural networks to develop neural networks using the Amazon Web Services environment. (Amazon.)

It allows, as most of the frameworks of this type, the creation of different kinds of neural networks (recurrent, convolutional, dense, etc.) regarding different objectives as image classification or natural language processing between others. It has deep integration to python but supports other languages too such as Julia, Scala, C++, or R, among others. (MXNet.)

5 Practical case

5.1 Introduction

Now that some of the main concepts of artificial intelligence and especially deep learning have been explained, it is time to see how this technology can really be useful. This practical case will aim to demonstrate that all these concepts can be put into practice by using this technology to help people solving problems in their daily lives. In this case, the objective is to create a neural network able to recognize the material from which the waste is made given a dataset of waste images as input data. Subsequently, this neural network could be used in the waste classification process to help in the task of recycling. The faced problem is a multiclass classification problem because as a final output it is expected to receive a category between the 3 possible categories which are going to be fixed, which are paper, metal, and glass. In this case, the input data will be images.

In this case, the use of deep learning can suit adequately the problem because the input dataset is composed of unstructured data, in this case, images, so it will be really complex to extract the features by hand. It is quite hard to say regarding an image that features differentiate the object in the picture in that way that looking at the image it is possible to decide if it is a material done with paper, metal, or glass. So this problem is hard to solve using traditional programming. Regarding artificial intelligence, as the extraction of features is difficult, machine learning will be harder to use too. Regarding all of this information, it seems justified the use of deep learning to face this problem.

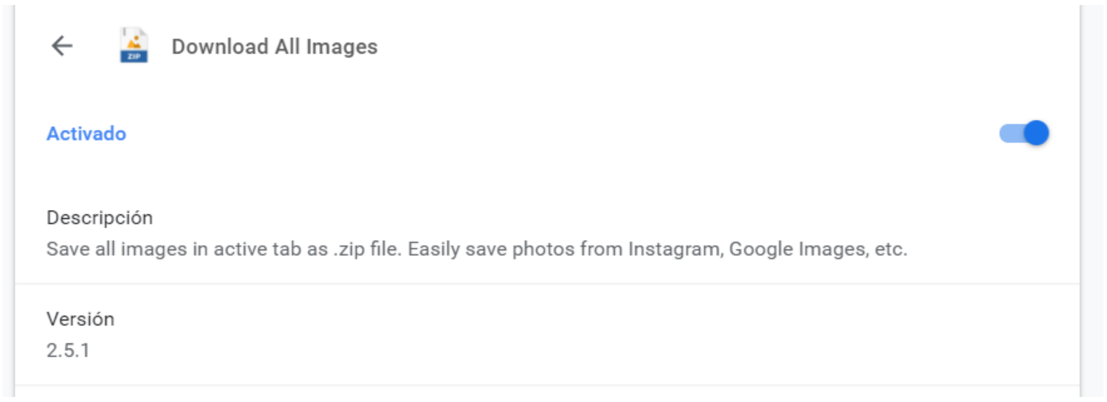
To get this objective a big amount of images of inorganic waste have been found, they are going to be used as a training dataset. These images will be classified and balanced. Then a neural network will be created, in this case, a convolutional neural network is going to be used. As it has been said before, the initial situation is a multiclass image classification and as it has been explained in the previous point, convolutional neural networks are very used especially to solve image classification problems, so the created neural network will have at least a convolutional layer. After the creation of the neural network, the dataset will be used to train it so, in the end, the generated model can be able to given a new image, differentiate the material of the object, and consequently in which bin it should be thrown.

5.2 Dataset creation

As it has been explained before one of the most critical points when we talk about the creation and training of a neural network is the dataset. Deep learning algorithms can extract themselves the features of the input data. However, to get an acceptable accuracy, they need a lot of images and samples in the training dataset.

However, not all the data are valid. It is necessary that the data could be correctly classified into training and testing images. Usually, the proportion used is 80% training and 20% testing. In addition to this, it would be a good practice if that data can be balanced, which means that there are a proportional number of data, in this case, images, inside each category.

In this practical case, there have been some problems during the training with the dataset. Initially, a dataset with around 12.000 images had been recollected from different sources and it has classified into 5 classes paper, glass, metal, plastic, and carton. To create this some images have been downloaded from the internet. To do this, it has been used "download all images", an extension from chrome that allows the user to download all the images of a webpage, google images, Instagram, etc. and store them into a zip. In figure 18, it is possible to see the extension data. After the download, the images have been selected and separated into different categories.


Download All Images

Activado

Descripción
Save all images in active tab as .zip file. Easily save photos from Instagram, Google Images, etc.

Versión
2.5.1

Figure 18. Download all images extension

In addition to this, some photos have been taken with a smartphone, concretely a Xiaomi mi A2 lite with a 12mp camera. Other photos have been gotten mixing some already existing datasets which have been found. These data sets are, on the one hand, a git repository with 2576 images taken and distributed in 594 paper images, 403 carton images, 482 plastic images, 410 metal images, and 137 random trash images. On the other hand, a Kaggle dataset has been used. Kaggle is a platform with several challenges in the field of artificial intelligence where people upload their solutions and receive punctuation. This platform is useful even if the user doesn't want to participate in the challenge because it allows downloading hundreds of datasets regarding different topics so that users can practice and learn machine learning on their own. (Aprende IA, 2018.)

In this case, this dataset obtained from Kaggle is composed of 22500 images classified in recyclables and organic. In this case, only some of the recyclable images have been introduced in our dataset, approximately around a 30% of the total amount of images in the dataset. The images have been checked and classified by hand in each of the already mentioned 5 categories in which the dataset that has been done for this practical case will be divided.

After collecting all the abovementioned images, it has been tried to balance all these categories, which means to try that all the categories have a similar amount of data. Finally, several folders with data have been checked to remove confusing images.

However, after collecting all the images to do this dataset, and use it to train the created neural network the results weren't as good as expected. It was a clear case of overfitting having a high accuracy between 90 and 98% recognizing the images of the training data set but just a 55% of accuracy rate when it tried to predict the class of new images. Regarding this case, and after several trials changing some hyperparameters a conclusion was reached, it seemed that despite having a big dataset with images of different waste and different materials, the images were too much different between them. As they have been got from different sources, most of the images had a different background. In addition to this, the number of elements in some images was different. For example, some photos had only one element(like a bottle or a can), but others had a group of elements(several bottles or cans in the same image). Furthermore, some images were taken near the objects but in other images, the waste was quite far.

Regarding all this data the conclusion reached was that probably there was too much variability in the images as there were pictures of very different objects from different materials in different environments, and using different perspectives. Too much variability to allow the neural network to obtain a good pattern with only 12.000 images, probably to make this complex dataset work properly much more images would be needed.

As it seemed clear that the objective to get good accuracy with the built dataset was probably not an option, some more similar images were selected from the original dataset with the objective to build a simpler and smaller dataset. In this case, around 2000 images were selected. In these images, there were only one object and all the images had a white background. Initially, it was tried to split the dataset into the same 5 categories we had before for the original big dataset.

The images were classified into two folders, one for training and the other one for testing. Inside each of these main folders, there were five subfolders, each corresponding to one of the categories that the neural network was intended to recognize. At that moment the categories were plastic, carton, paper, glass, and metal.

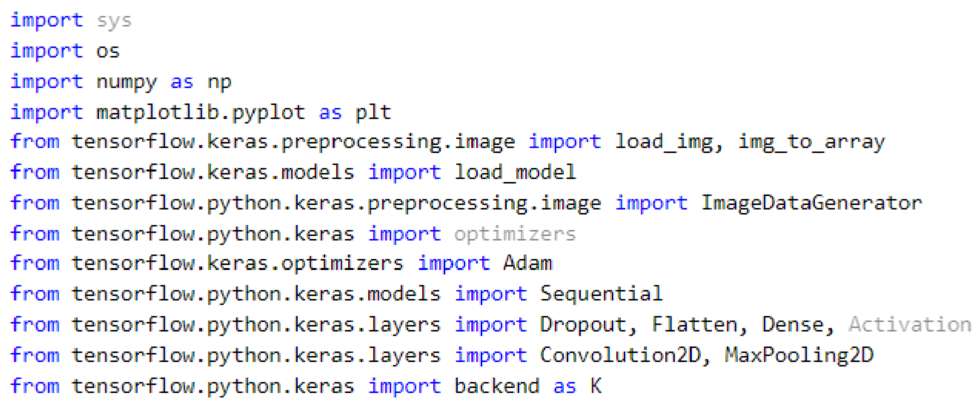
However, after training the artificial intelligence with this dataset the accuracy level was quite low, just around 65% in new images and 72% in training images. Despite this, there was a good fact in this trial, the amount of overfitting was better (it was possible to notice this because the percentage between the accuracy with the images in the testing dataset and the one get with new images was not so different).

After this, some hyperparameters were changed and it was decided to remove two of the categories. After these changes at the end of the last trial, it was gotten around a 95% of accuracy with images of the training dataset and around an 85% of accuracy with new data.

The final structure of the dataset used to get these results was to divide the total amount of images in a training folder with 80% of images and a testing folder with 20%. Inside each of these two folders, there were 3 subfolders named, paper, metal, and glass. In each one of the subfolders was stored the images with the objects made with each material respectively.

5.3 Tools and libraries

Now that it has been explained the data that is going to be used, which is really important to get a good accuracy result, is time to think about how the neural network is going to be done and used. In this case, the language which is going to be used is python. As has been already explained there a lot of different languages which support the creation of AI algorithms. However, in this case, python has been chosen because it is one of the most used programming languages in the world so there is a lot of material to learn how to use it. In addition to this, it is an open-source language, it has a lot of flexibility and its syntaxis is easy to read as well. Some of the most important companies in the world use it and it allows to include and use in the projects a lot of libraries to help the user to manage data as NumPy or pandas as well as specific libraries used in neural networks as Keras or TensorFlow.



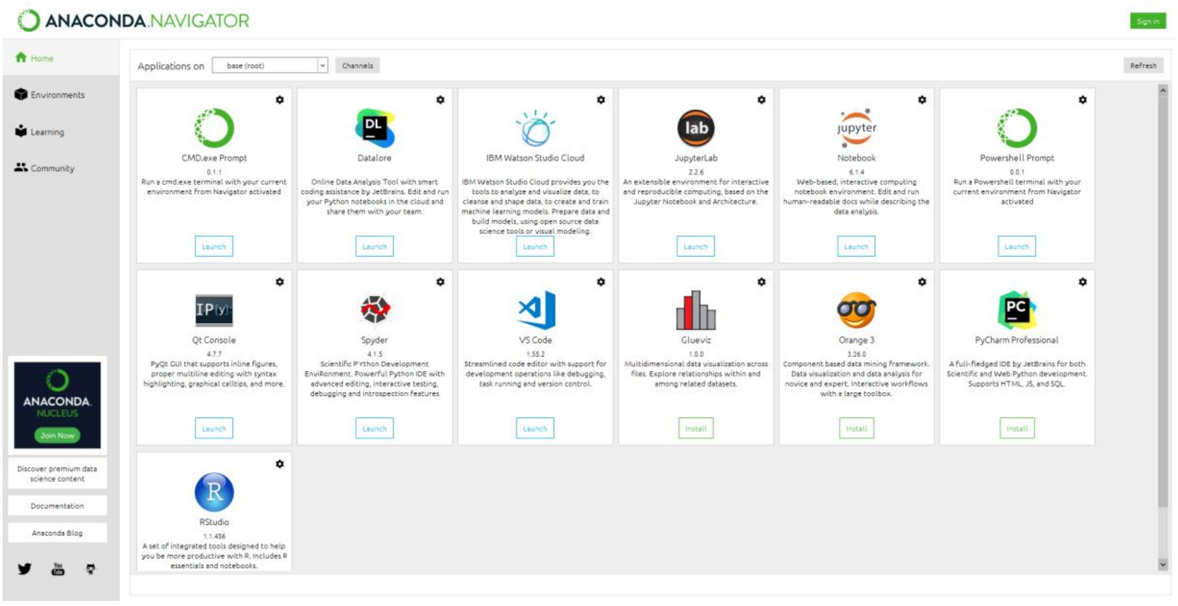
```
import sys
import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.models import load_model
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from tensorflow.python.keras import optimizers
from tensorflow.keras.optimizers import Adam
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dropout, Flatten, Dense, Activation
from tensorflow.python.keras.layers import Convolution2D, MaxPooling2D
from tensorflow.python.keras import backend as K
```

Figure 19. Imported libraries inside the neural network code

As it can be seen in Figure 19, to manage the data easily NumPy library is going to be imported and used to create vectors and matrices of big dimensions which could hold the data. This library will help to manage and operate with this data as well. Then matplotlib will be imported too to print some graphics, thanks to this library it will be easier to visualize the data and see fastly between other things how the training is going or which percentage of data have been correctly classified.

Regarding the libraries which are going to use in python to build and train this neural network, TensorFlow is going to be used as well as Keras. These libraries are going to simplify the task of creating the different layers in the neural network as well as will allow simplifying the code to train that neural network.

About the environment used, anaconda will be used. It is a data science platform that allows simplifying package management, it uses a package system called conda which is able to run, install and update dependency packages(Conda 2017). Anaconda has a big amount of packages installed automatically. In addition to this, it is possible to install many other packages. It includes a graphical user interface too, which can be used sometimes as an alternative to a command line. Anaconda Navigator allows using conda as well as run several applications which are available by default. In figure 20 it is possible to see the graphical interface of anaconda's navigator. (Anaconda 2021.)



CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

Datalore
Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

IBM Watson Studio Cloud
IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

JupyterLab
2.2.6
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.1.4
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

Qt Console
4.7.7
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
4.1.5
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

VS Code
1.55.2
Streamlined code editor with support for development operations like debugging, task running and version control.

Glueviz
1.0.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.26.0
Component based data mining Framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

PyCharm Professional
A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

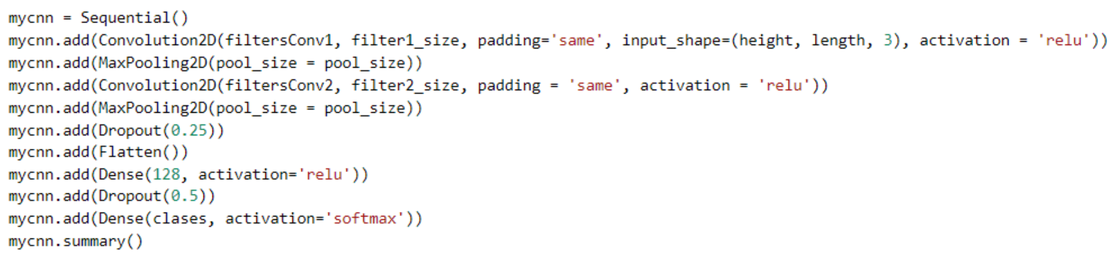
Figure 20. Anaconda navigator

Between the application which can be found in anaconda's navigator in this practical case, Jupyter Notebook will be used, it allows the creation of documents that contains text in markdown format as well as code. In addition to this, code can be run modularly on the same platform or be downloaded to execute it as a python file. In this practical case, jupyter has been used principally because it allows the user to run the code modularly, and it was very useful to debug because it was easier to find in which part of the code there where a failure(the failure appeared when this concrete part of the code was run).

After the code had been developed it was downloaded as a python file with .py extension and executed with a CLI. As training a neural network is a complex process that needs a lot of computational power and takes usually a lot of time, the local machine has been configured with the GPU packages as well as the CUDA's and NVIDIA controllers and tools needed to use the GPU with TensorFlow, with the usage of the GPU the computations could be done faster.

5.4 AI creation

Once the dataset and the environment have been prepared it is time to talk about how the neural network has been created. In this case, a convolutional neural network has been created. It has been chosen because they are specially used in image classification problems, as the one faced in this practical case. The reason why they are commonly used in this classification problem is that convolutional neural networks can detect simple characteristics such as borders, edges, etc. These networks can give a faster and more accurate answer and reduce the number of necessary hidden layers.



```
mycnn = Sequential()
mycnn.add(Convolution2D(filtersConv1, filter1_size, padding='same', input_shape=(height, length, 3), activation = 'relu'))
mycnn.add(MaxPooling2D(pool_size = pool_size))
mycnn.add(Convolution2D(filtersConv2, filter2_size, padding = 'same', activation = 'relu'))
mycnn.add(MaxPooling2D(pool_size = pool_size))
mycnn.add(Dropout(0.25))
mycnn.add(Flatten())
mycnn.add(Dense(128, activation='relu'))
mycnn.add(Dropout(0.5))
mycnn.add(Dense(clases, activation='softmax'))
mycnn.summary()
```

Figure 21. Neural network creation

As it can be seen in figure 21, the neural network has several layers. In this case, the model has been called mycnn (my convolutional neural network). First of all Sequential() has been used, this is one of the Keras available models, and indicates that the neural network layers will be in sequential order, one behind another.

After that, a convolutional layer is added. Keras simplify too the creation of layers in a deep learning model. In this case, this first convolutional layer has 32 filters with a size of (3,3). This size specifies the height and width of the convolutional window. With the padding = ‘same’ parameter, it is being indicated that there will be padding in the right, left, top, and down of the image. The input_shape parameter specifies the height, length, and number of color channels in the input image. In this case, the images will have 200*200 px and 3 color channels. Finally, the activation function is indicated, Keras allows to import some activation functions, in this case, relu will be used. The values of the variables and hyperparameters used can be seen in Figure 23. (TensorFlow 2021.)

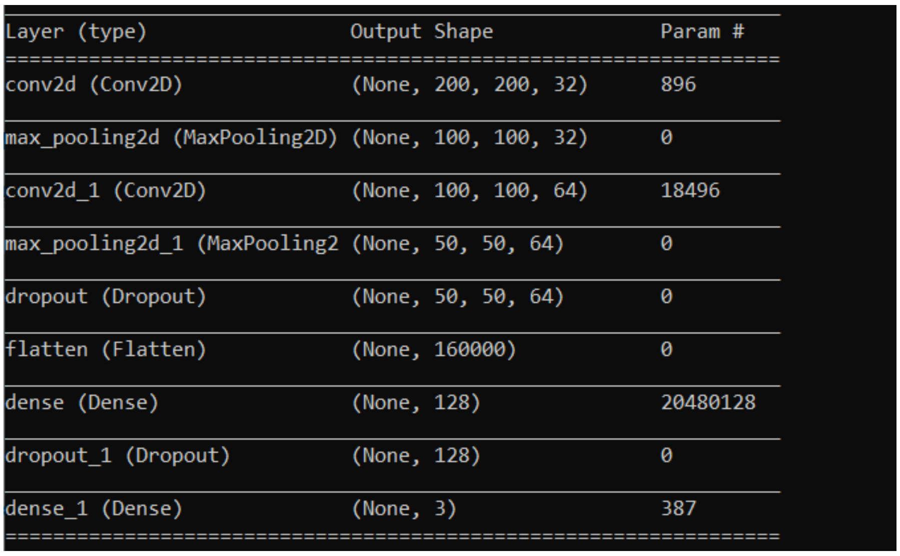
After this first convolutional layer, a pooling layer is applied to reduce the spatial dimensions of the input image regarding the next convolutional layer input. It can reduce the overhead regarding calculus in the next layer and despite this reduction can produce the loss of some information, it can help to prevent overfitting too. In this case, max-pooling (which has been explained in the theoretical case) is going to be used, and it will have a pool of (2,2). (Lopez Briega 2016.)

Then, another convolutional layer has been used, this time with 64 filters with (3,3) size. Consequently, after that convolutional layer, another pooling layer has been used too. In this second pooling layer, max-pooling has been used too. Then a dropout layer has been generated which deactivates randomly some of the neurons in each one of the dense layers with the objective to avoid overfitting. (Oliva Rodriguez 2018.)

After that, it is possible to find a flatten layer, it is needed when convolutional layers have been used and the next layers are fully connected (Keras tell them dense layers). This flatten layer adapts the output of the previous layers which has a tuple with several parameters, in this case, three numbers, to the input for the fully connected layer which needs a tuple with only one parameter.

To perform this operation, the values of the parameters of the previous layer are multiplied. In this case, it can be seen in figure 22 that the output of the flatten layer is 160000 which is the result achieved if the multiplication 50*50*64 (which are the output values of the previous layer) is done.

Next, a fully connected hidden layer has been introduced with 128 as output dimension, and its activation function will be relu. Finally, after another dropout layer is added, the output layer is created with 3 neurons and using a Softmax activation function, which is very useful in multiclass classification problems. With the command mycnn.sumary() it can be seen the structure of the neural network.



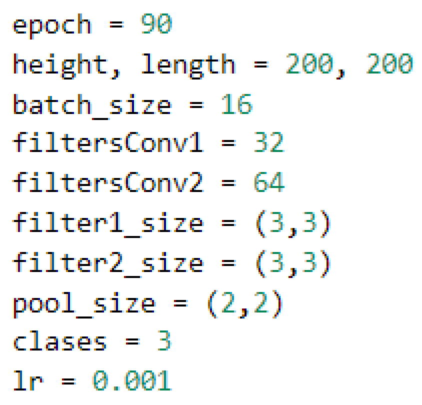
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 200, 200, 32)      896

max_pooling2d (MaxPooling2D) (None, 100, 100, 32)      0

conv2d_1 (Conv2D)            (None, 100, 100, 64)      18496

max_pooling2d_1 (MaxPooling2 (None, 50, 50, 64)        0

dropout (Dropout)            (None, 50, 50, 64)        0

flatten (Flatten)            (None, 160000)            0

dense (Dense)                (None, 128)               20480128

dropout_1 (Dropout)          (None, 128)               0

dense_1 (Dense)              (None, 3)                 387
=================================================================
```

Figure 22. Neural network parameters

Figure 22 shows the return of mycnn.sumary() command. As it can be seen, it has been created a quite big neural network with 2 convolutional layers and 2 fully connected laters which has a total of 20499011 parameters.

5.5 Training process

After the creation of the neural network model, it is time to train it with the already created dataset, this dataset will be imported at the beginning of the script. Regarding the training, an important aspect to take into account is the value of the hyperparameters, these hyperparameters are values such as the number of hidden layers, the number of neurons of each layer, or the learning rate. They are values difficult to choose and can vary depending on the training dataset, so to ensure the values fit correctly our objective there is no other option than testing different values and look for the ones that allow the neural network to get higher accuracy. In image 23 some of the variables with data used to train and built the neural network can be seen. In this case, there are variables such as the learning rate, the number of epoch chosen, the batch size, or the number of filters in each convolutional layer.

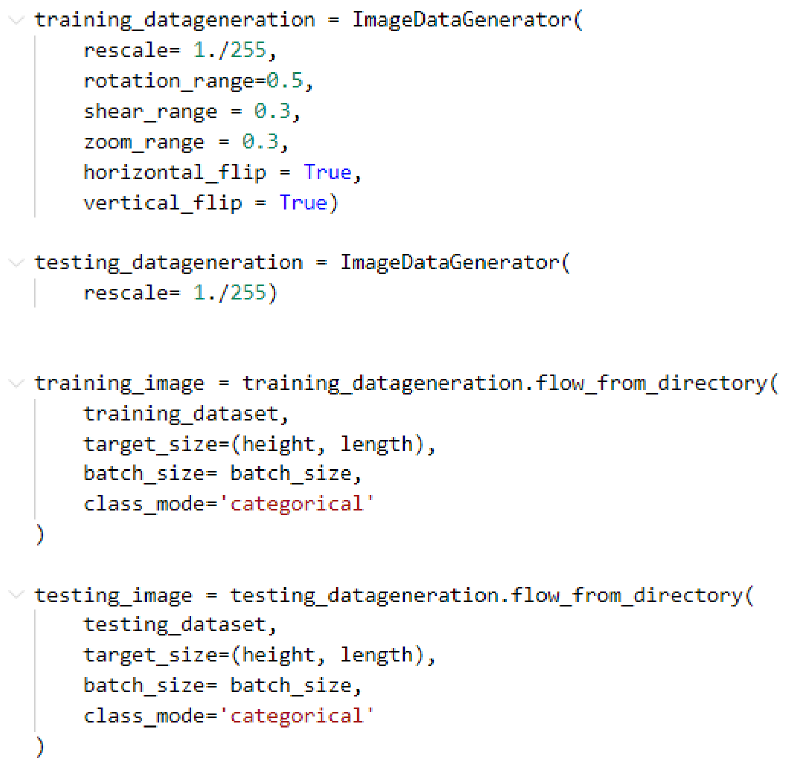


```
epoch = 90
height, length = 200, 200
batch_size = 16
filtersConv1 = 32
filtersConv2 = 64
filter1_size = (3,3)
filter2_size = (3,3)
pool_size = (2,2)
clases = 3
lr = 0.001
```

Figure 23. Used variables and hyperparameters

As in this case, the dataset which is going to be used at the end is quite small regarding the high complexity of the problem which is trying to be solved, a technique called data augmentation is going to be used. This technique consists of increasing the number of images in the dataset by modifying each original image, for example taking an image and rotating it, scaling it, flipping it, making zoom… with this technique it is possible to generate more images and augment the dataset. (Utrera Burgal 2019.)

Keras has a class called Image Data Generation which allows normalizing the image so that all the images in the dataset have the same form and size which can be recognized by the neural network. In addition to this, it allows doing data augmentation. In image 24 it can be seen the parameters used to augment the dataset.

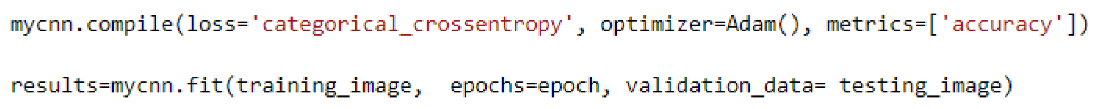


```
training_datageneration = ImageDataGenerator(
    rescale= 1./255,
    rotation_range=0.5,
    shear_range = 0.3,
    zoom_range = 0.3,
    horizontal_flip = True,
    vertical_flip = True)

testing_datageneration = ImageDataGenerator(
    rescale= 1./255)


training_image = training_datageneration.flow_from_directory(
    training_dataset,
    target_size=(height, length),
    batch_size= batch_size,
    class_mode='categorical'
)

testing_image = testing_datageneration.flow_from_directory(
    testing_dataset,
    target_size=(height, length),
    batch_size= batch_size,
    class_mode='categorical'
)
```

Figure 24. Data augmentation using ImageDataGenerator

As it can be seen in the image training, the data set has been modified and augmented to get more data, while the testing dataset has been just rescaled. Once we have defined the image generator, the dataset can be charged using the flow_from_directory operation in which some parameters such as the data directory, the image size, the batch size, or the class mode will be used. In this case, categorical means that 2D one-hot encoded labels will be returned, which means that there are mutually exclusive classes, a paper object can't be glass at the same time. (Utrera Burgal 2019.)

After the data has been properly imported, the model is defined as it has been explained in the previous point, and then the training process starts. But before it can start, the model should be complied. In this step, the model is generated specifying the loss function, the optimizer, and the metrics which will be shown in each step.



```
mycnn.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])

results=mycnn.fit(training_image,  epochs=epoch, validation_data= testing_image)
```

Figure 25. Model generation and training

As it can be seen in the compile method the chosen loss function is categorical cross-entropy, this function is the one commonly chosen in multiclass classification problems, where a tested image can only belong to one of the possible classes. It is designed to quantify the difference between two possible distributions. As optimizer Adam has been chosen, it is a different algorithm to the descent stochastic method and calculates an exponential moving which is the result of the average of the gradient and the squared gradient. Finally, the last parameter shown in the compile operation is the metrics evaluated by the model during the training and testing, in this case, it has been configured to use the accuracy.

Finally, it is time to train the model. In this case, thanks to TensorFlow and Keras start the training is quite simple. It is done with the fit function, using as input data the training dataset, fixing the number of epoch to 90 (the epoch variable value can be seen in figure 23), and using as validation data the testing dataset.

After the training process the accuracy of the model with the training dataset as well as the accuracy of the model obtained with new data validation(validation data), has been checked and the evolution of this data has been printed using matplotlib library. As it has been seen in the theoretical part, matplotlib is a library used to create static, animated, and interactive visualizations in python. In this practical case, it can be very useful to represent graphically the data and how it varies while the training is taking place, with the information shown graphically, it is easier to get an idea about how the training process has been developed. To print the data about the evolution of the accuracy during the training process the methods plot, title, legend, xlabel, ylabel, and show from the matplotlib library have been used, the data results are shown in figure 26.

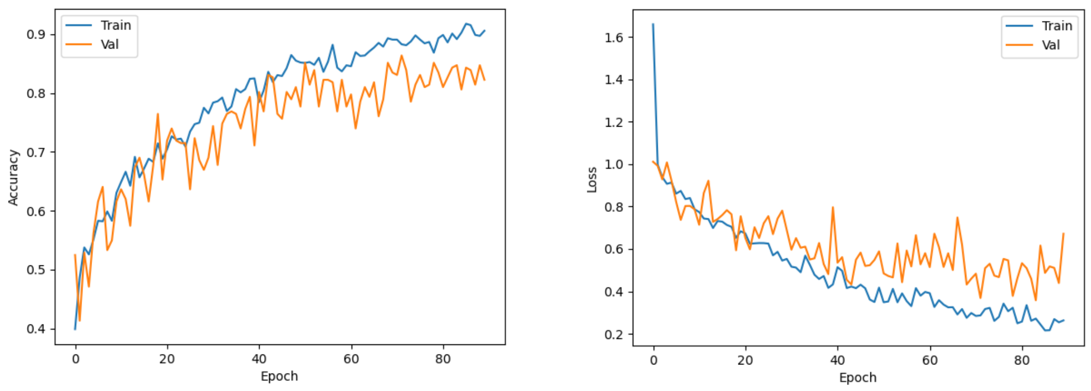
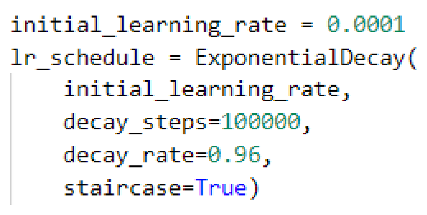


Figure 26. Accuracy and loss results.

In figure 26 two graphics can be seen, the first one represents the evolution of the accuracy during the different training epochs, and the second one the evolution of the loss value. Inside the graphic, it is possible to find with a blue line the data regarding the training dataset and in orange the results obtained with new images, the validation images. At the end of the training, the obtained values are for data in the training dataset an accuracy of around 95%, and regarding the validation images, new images which have never been seen before by the neural network, an accuracy of around 85%.

Several trials have been done changing the different hyperparameters to find the ones which provide better results. In one of the latest trials, it was tried to add more data in the validation dataset. In this case, the data was divided into the same three categories as before glass, paper, and carton, and 180 epochs were used. In addition to this, in this case, a technique called learning rate decay has been used. This technique consists of decreasing the learning date value during the training, as has been explained before, the learning rate is a hyperparameter that controls how much the estimated error affects the changes in the model when weights in the model are updated. In this case, a quite small learning rate is used, its value is around 0.0001 and the decay is done with a decay rate of 0.96. In figure 27 it is possible to see the code used to decay the learning rate.



```
initial_learning_rate = 0.0001
lr_schedule = ExponentialDecay(
    initial_learning_rate,
    decay_steps=100000,
    decay_rate=0.96,
    staircase=True)
```

Figure 27. Learning rate decay

Doing all these changes before mentioned the accuracy has increased with the validation values as well as whit the training ones. Regarding these new values, the accuracy with the images in the training dataset is around 98,8% while the accuracy obtained with the validation data is around 87,3%. In figure 28 a graphic showing these results can be found.

In addition to this, the results were very surprising because the neural network in this point is able to detect images of really different types of objects made with paper, glass, or metal. Moreover, it is not only able to detect pictures of straight objects but bent and deformed objects too. For instance, it was able to detect metal cans if they are in a good state but also if they are bent, the same thing happened with paper, it was able to detect a straight sheet of paper but also a ball of paper as it can be seen in figure 30.

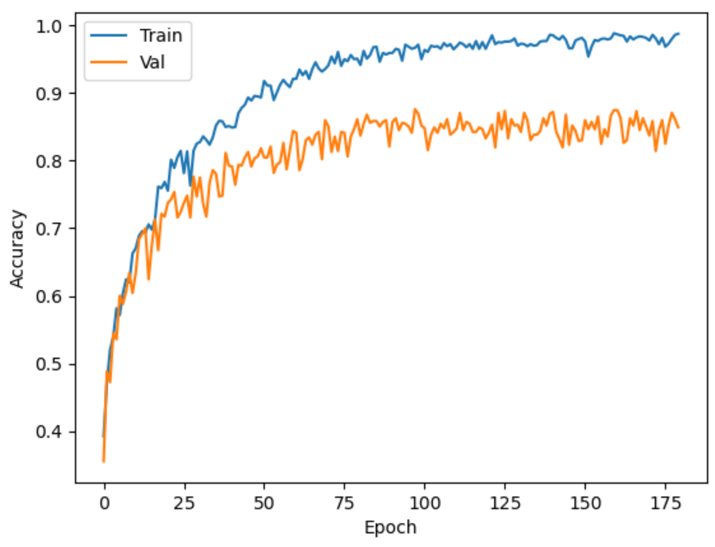
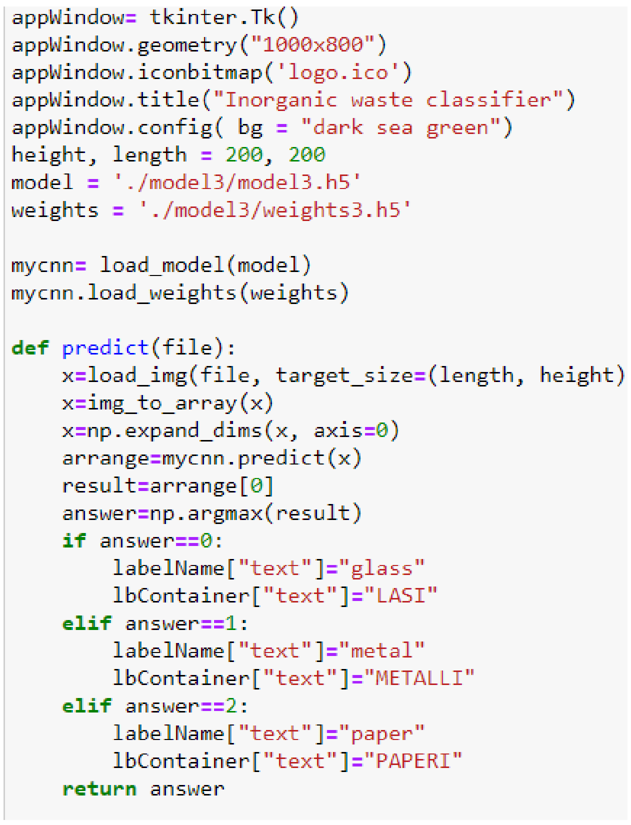


Figure 28. Accuracy results after improvements

5.6 User interface

Once the model was gotten, it was stored in a .h5 file, the obtained weights during the training were stored in another .h5 file too. Then a small and very simple user interface was created. To do this a python library was used, it is called Tkinter and is commonly used to create user interfaces in python. Another used python library was cv2 which is a free and open-source python library used to process images in real-time. Inside the script used to create this application, the model is loaded and used to do the prediction. We can see the code used to load the model and do the prediction in figure 29.



```
appWindow= tkinter.Tk()
appWindow.geometry("1000x800")
appWindow.iconbitmap('logo.ico')
appWindow.title("Inorganic waste classifier")
appWindow.config( bg = "dark sea green")
height, length = 200, 200
model = './model3/model3.h5'
weights = './model3/weights3.h5'

mycnn= load_model(model)
mycnn.load_weights(weights)

def predict(file):
    x=load_img(file, target_size=(length, height)
    x=img_to_array(x)
    x=np.expand_dims(x, axis=0)
    arrange=mycnn.predict(x)
    result=arrange[0]
    answer=np.argmax(result)
    if answer==0:
        labelName["text"]="glass"
        lbContainer["text"]="LASI"
    elif answer==1:
        labelName["text"]="metal"
        lbContainer["text"]="METALLI"
    elif answer==2:
        labelName["text"]="paper"
        lbContainer["text"]="PAPERI"
    return answer
```

Figure 29. App window creation, loading of the model, and prediction function

In figure 29 it is possible to see how the window in the application has been created using Tkinter. Then a title and an icon have been added. Regarding the icon, it has been done using Adobe Illustrator. About the prediction function once the model is loaded the image is taken and the predict function in the model (mycnn.predict(x)) is used to come up with the corresponding category. The answer from the model is given in an array, inside this array each position represents a category and has a value, so a function called argmax from numpy library is used. This function returns the position in which the maximum value in the array was. Then this position is translated to the corresponding category obtaining the name of the predicted category.

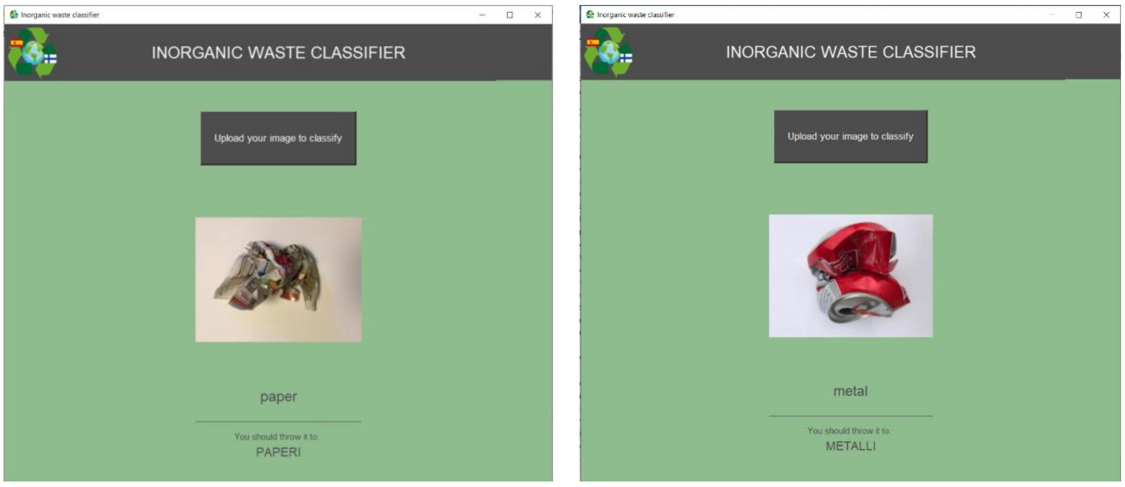

Inorganic waste classifier
INORGANIC WASTE CLASSIFIER
Upload your image to classify
paper
You should throw it to:
PAPERI
Inorganic waste classifier
INORGANIC WASTE CLASSIFIER
Upload your image to classify
metal
You should throw it to:
METALLI

Figure 30. User interface of the app with two different predictions

In figure 30 the interface of the app is shown. As it can be seen it has a toolbar with the title of the application and the logo, then it has a button that opens a file explorer. This file explorer allows the user to choose a picture from his own computer to classify. Finally, when the image is uploaded it is shown in the application and a label obtained from the model prediction explained before appears telling the user the material of the object in the image. Then, below this label, a message telling the user the bin where this waste should be thrown appears too.

6 Summary

During this thesis, there have been explained several concepts regarding artificial intelligence. I have started talking about basic ideas of artificial intelligence, and I have been concretizing until reaching deep learning through the explanation of machine learning and some concepts related to this field. Finally, I explained the concept of neural networks, how they work as well as different types of neural networks.

Then the practical case has been introduced not before giving some brushstrokes on the main programming languages related to artificial intelligence and some libraries useful for the development of the same in Python. Concerning the practical case, I have explained the results I have obtained and how I have reached them.

In this thesis practical case, I did a neural network that can classify the waste in several categories regarding the material the object of each image was done with. To do this a convolutional neural network has been created.

Between the problems I faced during the development of the practical case, it happened that the problem of classifying each kind of inorganic waste was too complex. Despite I spent time recollecting a dataset with 12.000 images of inorganic waste divided into 5 labeled categories (plastic, paper, carton glass, and metal), due to the complexity of the problem, the accuracy I got with this dataset at the very beginning was not good. So I decided to reduce the scope of the problem. I reduced the number of classes of materials in which the objects can be classified as well as I choose only a few images that were more similar between them inside the big 12000 images original dataset.

Regarding the final results, I can say the objective of creating a neural network able to recognize different waste done with different materials has been accomplished. In addition to this, I was very surprised with the final results because it allows recognizing more variability in the waste than expected. In addition to this, a simple user interface has been done so people without big knowledge about technology could use it as well.

Although with this new scope and regarding the complexity of the current problem I got a good accuracy, between 87% and 98%, in the future, I would like to be able to solve the ambitious first idea of recognizing more variety of waste and classify it in a bigger number of categories. However, to get that objective a huge number of images should be get. Recollecting the necessary amount of data would take a really huge amount of time for a particular or a student, but maybe it can be an easier task for enterprises which has access to big databases with images.

Another change I would like to do in the future with more time could be the improvement to the application user interface. I would like to improve the application by adding options and screens to the user-friendly interface that allows clients to use the created model of the neural network more easily.

At the end I had the chance to learn during this thesis because artificial intelligence was a completely new field for me. In addition to this, I could get more familiarized with Python which is a programming language I had not used that much.

References

3Blue1Brown. 2017a. Decenso de gradiente, es como las redes neuronales aprenden| Aprendizaje profundo, capítulo 2. Retrieved on 7 May 2021. Available at https://www.youtube.com/watch?v=IHZwWFHWa-w&list=PLZHQOb-OWTQDNU6R1_67000Dx_ZCJB-3pi&index=2

3Blue1Brown. 2017b. ¿Qué es la retropropagación y qué hace en realidad? In Aprendizaje profundo, Capítulo 3. Retrieved on 8 May 2020 Available at https://www.youtube.com/watch?v=Ilg3gGewQ5U

Advanced factories. 2017. Los expertos dicen que la Inteligencia Artificial será la base de la nueva revolución industrial. Retrieved on 2 March 2021. Available at https://www.ad-vancedfactories.com/los-expertos-dicen-la-inteligencia-artificial-sera-la-base-la-nueva-revolucion-industrial/

Ahmad, S. 2019. Supervised Learning vs Unsupervised Learning vs Reinforcement Learn-ing. In Tech Doodlin. Retrieved on 26 March 2021.Available at http://www.techdoo-dling.com/2019/07/supervised-unsupervised-reinforcement-learning.html

Alvarado, I. Redes Neuronales. Retrieved on 6 May 2021. Available at https://ml4a.github.io/ml4a/es/neural_networks/

Alvarez, J. M. 2018, March 12. ¿Qué es Microsoft Cognitive toolkit? Retrieved on 12 May 2021. Available at http://blog.josemarianoalvarez.com/2018/03/12/que-es-microsoft-cogni-tive-toolkit/

Amazon. Apache MXnet en AWS.In Amazon Web Services. Retrieved on 12 May 2021. Available at https://aws.amazon.com/es/mxnet/

AMP Tech. 2018. ¿Cómo funciona softmax? Retrieved on 7 May 2021. Available at https://www.youtube.com/watch?v=ma-F0RsMAjQ

Anaconda. 2021. Anaconda.The World’s Most Popular Data Science Platform. Retrieved on 10 May 2021, Available at https://www.anaconda.com/

Aprende IA. 2018.Todo lo que tienes que saber sobre Kaggle. Retrieved on 10 May 2021. Available at https://aprendeia.com/todo-lo-que-tienes-que-saber-sobre-kaggle/

Aprendemachinelearning. 2017. Qué es overfitting y underfitting y cómo solucionarlo. Re-trieved on 1 April 2021. Available at https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/#:~:text=Overfitting%20in-dicar%C3%A1%20un%20aprendizaje%20%E2%80%9Cexcesivo,comprender%20nue-vos%20datos%20de%20entrada.

Ardila, R. 2011. Inteligencia. ¿qué sabemos y qué nos falta por investigar?. In SCIELO. Retrieved on 2 March 202. Available at http://www.scielo.org.co/scielo.php?script=sci_art-text&pid=S0370-39082011000100009

Artola Moreno, A. 2019. Clasificación de imágenes usando redes neuronales convolucio-nales en Python. Universidad de Sevilla. Thesis. . Retrieved on 13 Abril 2021 Available at http://bibing.us.es/proyectos/abreproy/92402/fichero/TFG-2402-ARTOLA.pdf

Bajada, J. 2019. Symbolic vs Connectionist A.I. In Trowards Data science. Retrieved on 24 March 2021. Available at https://towardsdatascience.com/symbolic-vs-connectionist-a-i-8cf6b656927

Barl, N. 2021. Understanding Representation Learning With Autoencoder: Everything You Need to Know About Representation and Feature Learning. In Neptuneblog. . Retrieved on 26 April 2021. Available at https://neptune.ai/blog/understanding-representation-learning-with-autoencoder-everything-you-need-to-know-about-representation-and-feature-learning

Basoglu, C., Barrett, S., Ahlers, D., & Crepaldi, T. (2017). The Microsoft Cognitive Toolkit—Cognitive Toolkit—CNTK. In Microsoft. ? Retrieved on 12 May 2021. Available at https://docs.microsoft.com/en-us/cognitive-toolkit/

Boz Allen, H. A quick guide to how machines learns. Retrieved on 4 April 2021. Available at https://www.boozallen.com/content/dam/boozallen_site/sig/pdf/publications/machine-in-telligence-quick-guide-to-how-machines-learn.pdf

Brezal F. 2017. Breve historia de la inteligencia artificial: el camino hacia la empresa. In CESCE. . Retrieved on 23 March 2021.Available at https://www.cesce.es/es/-/asesores-de-pymes/breve-historia-la-inteligencia-artificial-camino-hacia-la-empresa

Brownlee, J. 2019. Understand the Impact of Learning Rate on Neural Network Perfor-mance. In Machine Learning Mastery website. Retrieved on 8 May 2021. Available at https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

Brownlee, J. 2018. A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks. Retrieved on 14 May 2021. Available at https://machinelearningmas-tery.com/early-stopping-to-avoid-overtraining-neural-network-models/

Brownlee, J. 2019a. A Gentle Introduction to Batch Normalization for Deep Neural Networks. Retrieved on 14 May 2021. Available at https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

Brownlee, J. 2019b. Your First Deep Learning Project in Python with Keras Step-By-Step. Retrieved on 12 May 2021. Available at https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/

Budhiraja, A. 2018. Learning Less to Learn Better—Dropout in (Deep) Machine learning. Retrieved on 14 May 2021. Available at 2021.https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

Buhigas, J. 2018. Todo lo que necesitas saber sobre TensorFlow, la plataforma para Inteligencia Artificial de Google. In Puentes Digitales. Retrieved on 12 May 2021. Available at https://puentesdigitales.com/2018/02/14/todo-lo-que-necesitas-saber-sobre-tensorflow-la-plataforma-para-inteligencia-artificial-de-google/

Bushaev, V. 2018Adam—Latest trends in deep learning optimization. Retrieved on 14 May 2021Available at https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c

Canit. Retrieved on 19 February 2021. Available at https://www.canit-app.com/fi/mika-canit

Castrounis, A. Python vs R for AI, Machine Learning, and Data Science. In InnoArchiTec. Retrieved on 11 May 2021. Available at https://www.innoarchitech.com/blog/python-vs-or-r-artificial-intelligence-ai-machine-learning-data-science-which-use

Ceron, R 2019. AI, Machine Learning and Deep Learning: what's the difference?. In IBM. Retrieved on 9 march 2021. Available at https://www.ibm.com/blogs/systems/ai-machine-learning-and-deep-learning-whats-the-difference/

Cerrillo, A. 2019. Crece la deuda ecológica. In La Vanguardia. Retrieved on 7 February 2021. Available at https://www.lavanguardia.com/natural/20190730/463772770926/consumo-cambio-climatico-sobreexplotacion-recursos-planeta.html

Conda. 2017. Conda documentation. Retrieved on 10 May 2021. Available at https://docs.conda.io/en/latest/

Construncía. 2020. Which countries are leading the change in circular economy?. Retrieved on 15 February 2021. Available at https://www.construcia.com/en/noticias/which-countries-are-leading-the-change-in-circular-economy/

Corp media. Las 3r de la ecología: reducir, reutilizar y reciclar. Retrieved on 10 February 2021. Available at https://corp-promotores.es/magazine/reducir-reutilizar-reciclar/

Dabbura, I. 2019. Gradient Descent Algorithm and Its Variants. Retrieved on 8 May 2021. Available at https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

Data Flair. Deep Learning vs Machine Learning – Demystified in Simple Words. Retrieved on 26 April 2021. Available at https://data-flair.training/blogs/deep-learning-vs-machine-learning/

De la Torre, D. 2019. ¿Cómo nació la Inteligencia Artificial?. In Blogthinkbig. Retrieved on 21 March 2021.Available at https://blogthinkbig.com/historia-como-nacio-inteligencia-artificial

Decena, E. 2019. Entendiendo las redes neuronales PART 1. Retrieved on 6 May 2021. Available at https://medium.com/@eddydecena/entendiendo-las-redes-neuronales-part-1-fca3adf78c5b

deeplizard. 2017. Convolutional Neural Networks (CNNs) explained. Retrieved on 8 May 2021. Available at https://www.youtube.com/watch?v=YRhxdVk_sIs

Devashish S. 2018. Backpropagation concept explained in 5 levels of difficulty. Retrieved on 27 April 2021. Available at https://medium.com/coinmonks/backpropagation-concept-explained-in-5-levels-of-difficulty-8b220a939db5

Doshi, S. 2020. Various Optimization Algorithms For Training Neural Network. Retrieved on 14 May 2021. Available at https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

Ecoembes. 2018. Ecoembes lanza AIRE, el primer asistente virtual de reciclaje. Retrieved on 26 February 2021. Available at https://www.ecoembes.com/es/ciudadanos/sala-de-prensa/notas-de-prensa/ecoembes-lanza-air-e-el-primer-asistente-virtual-de-reciclaje

Economía circular. Economía Circular. Retrieved on 16 February 2020. Available at http://economiacircular.org/wp/economia-circular/#:~:text=La%20econom%C3%ADa%20circular%20es%20la,aspectos%20ambientales%2C%20econ%C3%B3micos%20y%20sociales.&text=Por%20lo%20tanto%2C%20la%20econom%C3%ADa,del%20uso%20de%20los%20recursos.

Feng, Weijiang & Guan, Naiyang & Li, Yuan & Zhang, Xiang & Luo, Zhigang. 2017. Audio visual speech recognition with multimodal recurrent neural networks. 681-688.

10.1109/IJCNN.2017.7965918. . Retrieved on 3 May. Available at https://www.researchgate.net/publication/318332317_Audio_visual_speech_recognition_with_multimodal_recurrent_neural_networks

Gambit. Retrieved on 19 February 2021. Available at https://www.gambitgroup.fi/recycling-made-easy/

García García P.P. 2013. Reconocimiento de imágenes utilizando redes neuronales artificiales. Thesis. Universidad Complutense de Madrid. Retrieved on 2 May 2021 Available at https://eprints.ucm.es/id/eprint/23444/1/ProyectoFinMasterPedroPablo.pdf

Garcia Garcia, A. 2020. Análisis comparativo de experiencias en Economía Circular en Europa. Universitat Politècnica de València Campus d' Alcoi. Thesis. Retrieved on 15February 2020. Available at https://riunet.upv.es/bitstream/handle/10251/154139/Garc%C3%ADa%20-%20An%C3%A1lisis%20comparativo%20de%20experiencias%20en%20Econom%C3%ADa%20Circular%20en%20Europa.pdf?sequence=1]

Gavrilova, Y. 2020. Artificial Intelligence vs. Machine Learning vs. Deep Learning: Essentials. In Serokell. . Retrieved on 11 March 2021.Available at https://serokell.io/blog/ai-ml-dl-difference

GeeksforGeeks.2020. Underfitting and Overfitting in Machine Learning. Retrieved on 4 April 2021. Available at https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

general information

Github. 2017. Gradient Descent—ML Glossary documentation. Retrieved 7 May 2021, Available at https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html

Google Developers. Reducción de la pérdida: Tasa de aprendizaje. Retrieved on 8 May 2021. Available at https://developers.google.com/machine-learning/crash-course/reducing-loss/learning-rate?hl=es-419

Google.Descripción general del ajuste de hiperparámetros. Retrieved on 5 May 2021. Available at https://cloud.google.com/ai-platform/training/docs/hyperparameter-tuning-overview?hl=es-419

Haapio, S. 2019. Finland pioneers circular economy to ensure prosperity in the future. In ThisisFINLAND. Retrieved on 13 February 2021 Available at https://finland.fi/business-innovation/finland-pioneers-circular-economy-to-ensure-prosperity-in-the-future/

IBM cloud education. 2020. Exploratory Data Analysis. Retrieved on 4 April 2021. Available at https://www.ibm.com/cloud/learn/exploratory-data-analysis

IBM,What is Industry 4.0? . Retrieved on 21 March 2021.Available at https://www.ibm.com/topics/industry-4-0

Interactive Chaos website. Stochastic Gradient Descent. Retrieved on 8 May 2021.Available at https://interactivechaos.com/es/manual/tutorial-de-machine-learning/stochastic-gradient-descent

Jaime Maestre, R. 2019. Economía circular en Europa: situación y cómo está evolucionando. In World Economic Forum . Retrieved on 15 February 2021. Available at https://es.weforum.org/agenda/2019/06/economia-circular-en-europa-situacion-y-como-esta-evolucionando/

John McGonagle, Shaikouski, G., Williams, C., Hsu, A., Khim, J., & Miller, A. Backpropagation. In Brilliant Math & Science Wiki. Retrieved 7 May 2021. Available at https://brilliant.org/wiki/backpropagation/

Justel Pizarro, J .T. 2019. Detección y clasificación de diferentes formas eritrocitarias anómalas mediante redes neuronales profundas.Universitat Politècnica de Catalunya. Thesis. Retrieved on 25 Abril 2021 Available at https://upcommons.upc.edu/bitstream/handle/2117/173454/Memoria+TFG+Joaqu%C3%ADn+Justel+Pizarro.pdf?sequence=1

Kaczorowski, M. 2020. Python vs. JavaScript. Which Is Better For Your Machine Learning Project? Retrieved 10 May 2021, Available at https://www.ideamotive.co/blog/python-vs.-javascript

Keras. Keras: The Python deep learning API. Retrieved on 12 May 2021. Available at https://keras.io/

Khosla, S. 2019. CNN | Introduction to Pooling Layer. In GeeksforGeeks. Retrieved on 10 May 2021. Available at https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

Kobran, D., & Banys, D. Activation Function. Retrieved on 6 May 2021. Available at https://docs.paperspace.com/machine-learning/wiki/activation-function

Lewis, S. 2020. What is object-oriented programming (OOP)? In Search app website. Retrieved on 11 May 2021. Available at https://searchapparchitecture.techtarget.com/definition/object-oriented-programming-OOP

Llinares Pellicer, J. 2021. Introduction to neural networks and deep learning. In Universitat Politècnica de València. Retrieved on 5 March 202. Available at https://polifor-mat.upv.es/access/content/group/GRA_14411_2020/Basic%20slides/10%20-%20Intro-duction%20to%20neural%20networks%20and%20deep%20learning.pdf

Lopez Briega, R. 2016. Redes neuronales convolucionales con TensorFlow. Retrieved on 13 May 2021. Available at https://relopezbriega.github.io/blog/2016/08/02/redes-neu-ronales-convolucionales-con-tensorflow/

López Camacho, Y. D. Unidad de Procesamiento Tensor de Google Tensor Processing Unit . In Universidad Industrial de Santander Bucaramanga, Colombia. Retrieved on 21 March 2021.Available at http://wiki.sc3.uis.edu.co/images/f/f8/GR14.pdf

Mahajan, P. 2020. Fully Connected vs Convolutional Neural Networks. In The Startup Me-dium. Retrieved on 3 May 2021. Available at https://medium.com/swlh/fully-connected-vs-convolutional-neural-networks-813ca7bc6ee5

Mahajan, Pooja. 2020. Fully Connected vs Convolutional Neural Networks. In The Startup Medium. Retrieved on 10 May 2021. Available at https://medium.com/swlh/fully-connected-vs-convolutional-neural-networks-813ca7bc6ee5

Maldonado, A. 2020. La nueva economía circular en Europa. In Navarra Capital. Retrieved on 15 February 2021. Available at https://navarracapital.es/la-nueva-economia-circular-en-europa/#:~:text=La%20econom%C3%ADa%20circu-lar%20es%20un,de%20los%20productos%20se%20extiende

Mangalad, A. S. 2020. Java in artificial intelligence: How is it used? In Big Data Made Sim-ple. Retrieved on 11 May 2021. Available at https://bigdata-madesimple.com/java-artificial-intelligence-uses/

Marina. 2021. Overfitting. Qué es, causas, consecuencias y cómo solucionarlo. In Grupo Atico34. Retrieved on 1 April 2021.Available at https://protecciondatos-lopd.com/empre-sas/overfitting/

Mazur M. A Step by Step Backpropagation Example. Retrieved on 27 April 2021. Available at https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Meza Ruiz, I. V. 2016. Descenso por gradiente (Gradient descent). . Retrieved on 27 April 2021. Available at https://turing.iimas.unam.mx/~ivanvladimir/posts/gradient_descent/

Mirchandani, R. 2020. Five global issues to watch in 2021. In United Nations Foundation. Retrieved on 5 February 2021. Available at https://unfoundation.org/blog/post/five-global-issues-to-watch-in-2021/

Mondal, B. 2020. Difference between Traditional Programming and Machine learning. In Kaggle. Retrieved on 4 April 2021. Available at https://www.kaggle.com/getting-started/150361

MXNet. Apache MXNet. Retrieved on 12 May 2021. Available at https://mxnet.apache.org/versions/1.8.0/

National Gepgraphic. Breve historia visual de la inteligencia artificial. Retrieved on 18 March 2021. Available at https://www.nationalgeographic.com.es/ciencia/breve-historia-visual-inteligencia-artificial_14419

Nava Bautista, J. V., Carapia Carapia, Ana Laura & Vidal-García, Francisca. Las tres R: Una opción para cuidar nuestro planeta. In INECOL Instituto Nacional de Ecología. Retrieved on 10 February 2021. Available at https://www.inecol.mx/inecol/index.php/es/2013-06-05-10-34-10/17-ciencia-hoy/413-las-tres-r-una-opcion-para-cuidar-nuestro-planeta

Nelson, D. 2020. What is Overfitting? In Unite.ai. Retrieved on 1 April 2021. Available at https://www.unite.ai/what-is-overfitting/

Nicholson, C. A Beginner's Guide to LSTMs and Recurrent Neural Networks. In Pathmind Retrieved 4 May 2021. Available at http://wiki.pathmind.com/lstm

Numtra. 2017. Top Languages and Packages for Machine Learning. Retrieved on 14 May 2021. Available at https://numtra.com/top-languages-packages-machine-learning/

Ohio Valley Waste Service.Inc. 2017. Reduce, Reuse, Recycle: what does it mean?.. Retrieved on 10 February 2021. Available at https://www.ohiovalleywaste.com/ohio-valley-waste-news/reduce-reuse-recycle-what-does-it-mean-3049

Oliva Rodriguez, A. 2018. Desarrollo de una aplicación de reconocimiento en imágenes utilizando Deep Learning con OpenCV. Universitat Politècnica de València. Thesis. Retrieved on 9 April 2021. Available at https://m.riunet.upv.es/bitstream/handle/10251/107243/OLIVA%20-%20Desarrollo%20de%20una%20aplicaci%C3%B3n%20de%20reconocimiento%20en%20im%C3%A1genes%20utilizando%20Deep%20Learning%20con%20O....pdf?sequence=1&isAllowed=y

OXFAM Intermón. Reducir, reutilizar, reciclar: descubre las claves de un mundo más sostenible. Retrieved on 11 February 2021. Available https://blog.oxfamintermon.org/reducir-reutilizar-reciclar-descubre-las-claves-de-un-mundo-mas-sostenible/

Radhakrishnan, P. 2017. What are Hyperparameters? And How to tune the Hyperparameters in a Deep Neural Network? In Towards Data Science. Retrieved on 5 May 2021. Available at https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a

Reddy, N. 2020. How Does the Gradient Descent Algorithm Work in Machine Learning? Retrieved on 8 May 2021. Available at https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/

Rodriguez, V. 2018. Conceptos básicos sobre redes neuronales. In Vicente Rodríguez blog. Retrieved on 5 May 2021. Available at https://vincentblog.xyz/posts/conceptos-basicos-sobre-redes-neuronales

Rubiales, A. 2020. Explicación Funciones de activación y práctica con Python. Retrieved on 10 May 2021. Available at https://rubialesalberto.medium.com/explicaci%C3%B3n-funciones-de-activaci%C3%B3n-y-pr%C3%A1ctica-con-python-5807085c6ed3

Ruder S. 2016. An overview of gradient descent optimization algorithms. Retrieved on 27 April 2021. Available at https://ruder.io/optimizing-gradient-descent/

Saha, S. 2018. A Comprehensive Guide to Convolutional Neural Networks—The ELI5 way. Retrieved on 8 May 2021. Available at https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Sangam. 2019. Difference between Feed Forward Neural Network and RNN. Retrieved on 4 May 2021. Available at https://www.aisangam.com/blog/difference-between-feed-forward-neural-network-and-rnn/.

Saxena, S. 2021, April 5. Introduction to Softmax. In Analytics Vidhya. Retrieved on 6 May 2021. Available at https://www.analyticsvidhya.com/blog/2021/04/introduction-to-softmax-for-neural-network/

Seehan S, shong Yung S. 2016. Deep learning for polulation genetic inference. In PLOS Computational Biology. Retrieved on 27 April 2021. Available at https://www.researchgate.net/publication/299474560_Deep_Learning_for_Population_Genetic_Inference

Shah, D. 2018. AI, Machine Learning, & Deep Learning Explained in 5 Minutes. In Becomin Human. Retrieved on 8 April 2021. Available at https://becominghuman.ai/ai-machine-learning-deep-learning-explained-in-5-minutes-b88b6ee65846

Sharma, S. 2017. Activation Functions in Neural Networks. Retrieved on 6 May 2021. Available at https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Silvennoinen, J. 2016. Recycle Finland. In AppAdvice. Retrieved on 14 February 2021. Available at https://appadvice.com/app/recycle-finland/1095762900

Simplilearn. 2021. What is Keras and Why it so Popular in 2021. Retrieved on 12 May 2021. Available at https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-keras

Singapore Computer Society. 20202. Simplifying the difference: machine learning vs deep learning. Retrieved on 26 April 2021. Available at https://www.scs.org.sg/articles/machine-learning-vs-deep-learning

Singh, V. 2017. An intuitive explanation to Gradient Descent. Retrieved on 9 May 2021. Available at https://viveksingh-heritage.medium.com/an-introduction-to-gradient-descent-54775b55ba4f

Solar Schools. Reduce, Reuse, Recycle. Retrieved on 26 April 2021. Available at https://www.solarschools.net/knowledge-bank/sustainability/reduce-reuse-recycle

Srivastava, P. 2020. Why use Python for AI and Machine Learning? In TechNative. Retrieved on 10 May 2021. Available at https://technative.io/why-use-python-for-ai-and-machine-learning/

Stanford University Center for the Study of Language and Information. 2005. Ecology. Retrieved on 5 February 2021. Available at https://plato.stanford.edu/entries/ecology/

TensorFlow. 2021. Tf.keras.layers.Conv2D. In TensorFlow Core v2.4.1. . Retrieved on 13 May 2021. Available at https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D?hl=es

Torres, J. 2019. Redes Neuronales Recurrentes. Retrieved on 4 May 2021, Available at https://torres.ai/redes-neuronales-recurrentes/

Turing A.M. 1950. Computing machinery and intelligence. Retrieved on 1 April 2020. Available at https://www.csee.umbc.edu/courses/471/papers/turing.pdf

Unesco. Sustainable Development. Retrieved on 6 February 2021. Available at https://en.unesco.org/themes/education-sustainable-development/what-is-esd/sd#:~:text=The%20concept%20of%20sustainable%20development,to%20meet%20their%20own%20needs.%E2%80%9D

Uniqtech. 2018. Understand the Softmax Function in Minutes. Retrieved on 6 May 2021. Available at https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d

Upwork. 2016. 15 Python Libraries for Data Science. Retrieved on 6 April 2021. Available at https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d

Utrera Burgal, J. 2019. Tratamiento de imágenes usando ImageDataGenerator en Keras. Retrieved on 13 May 2021. Available at http://enmilocalfunciona.io/tratamiento-de-imagenes-usando-imagedatagenerator-en-keras/

Verma, A. 2016. Most Popular Programming Languages For Machine Learning And Data Science. In Fossbytes. Retrieved on 10 May 2021. Available at https://fossbytes.com/popular-top-programming-languages-machine-learning-data-science/

Villaluenga Moran, J. L. 2019.OpenStreetCam. reconocimiento automático de objetos en imágenes mediante machine learning. Universitat oberta de Catalunya. Thesis. Retrieved on 23 April 2021. Available at http://openaccess.uoc.edu/webapps/o2/bitstream/10609/88287/6/jvillaluengaTFG0119memoria.pdf

Villanueva García, J. D. 2020. Redes neuronales desde cero (I)—Introducción. Retrieved on 6 May 2021, from https://www.iartificial.net/redes-neuronales-desde-cero-i-introduccion/

Weaver, F. 2016. Eliminating waste in a circular economy. In ThisisFINLAND. Retrieved on 13 February 2021. Available at https://finland.fi/business-innovation/eliminating-waste-in-a-circular-economy/

World Commission on Environment and Development. 1987. In Bruntland Commission. Retrieved on 6 February 2021. Available at https://sustainabledevelopment.un.org/content/documents/5987our-common-future.pdf

Yegulalp, S. 2019. What is TensorFlow? The machine learning library explained. Retrieved on 12 May 2021. Available at https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html

Yıldırım, S. 2020. 5 Must-Know Activation Functions Used in Neural Networks. In Medium. Retrieved on 6 May 2021. Available at https://towardsdatascience.com/5-must-know-activation-functions-used-in-neural-networks-8c5052757750

Yin, L. 2017. A Walk-through of Cost Functions. Retrieved on 8 May 2021.Available at https://medium.com/machine-learning-for-li/a-walk-through-of-cost-functions-4767dff78f7

Yu, D., Wang, H., Chen, P., & Wei, Z. 2014. Mixed Pooling for Convolutional Neural Networks. 364–375. In Tongji University, Shanghai, P.R. China. Retrieved on 9 May 2021. Available at https://www.researchgate.net/publication/300020038_Mixed_Pooling_for_Convolutional_Neural_Networks

Zahid , Sohail. 2019. A Brief History of AI. Retrieved on 26 April 2021. Available at https://sohailzahid.com/2019/02/02/a-brief-history-of-ai/

Zenrobotics. Robots for your waste stream. Retrieved on 26 February 2021. Available at https://zenrobotics.com/

Zhao, M. 2020. Symbolic AI vs Connectionism. Retrieved on 24 March 2021. Available at https://becominghuman.ai/symbolic-ai-vs-connectionism-9f574d4f321f

Zola, A. 2018. The 5 Best Programming Languages for AI. In SpringBoard. Retrieved on 10 May 2021. Available at https://www.springboard.com/blog/ai-machine-learning/best-programming-language-for-ai/