Libera Università Internazionale degli Studi Sociali

Dipartimento di Impresa e Management

Bachelor's degree in
Management and Computer Science

"*Artificial Intelligence methodologies for COVID-19 detection in chest CT and X-ray images.*"

Candidate : Giorgio Bientinesi

Supervisor : Prof. Marco Querini

Course : Artificial Intelligence and Machine Learning

Academic Year

2020-2021

Index

Abstract

The first case of COVID-19 occurred in China, specifically in the Wuhan region in December 2019. After that, the virus spread worldwide, and, at the time of writing this article, the total number of confirmed cases is over 173,609,772 with over 3,742,653 deaths.

Machine learning algorithms built on breast images can be used as a decision support mechanism to help radiologists speed up the diagnostic process.

This work represents a comprehensive analysis to assess 1) which of the 4 most important types of classification algorithms, namely Convolutional Neural Network, Support Vector Machine, Random Forest and Logistic Regression, performs best in the problem addressed and 2) which type of image, namely X-ray or CT scan is best suited for the analysis.

We used two datasets, one containing X-ray images and one allowing CT-scan. Both datasets are composed of about 10,000 images, well balanced in the 2 categories (COVID and Non-COVID).

In order to obtain the best possible model type, we conducted an extensive parameter hyper-tuning phase for each of the model types mentioned, testing the architectures on both the X-ray and CT-scan datasets.

We concluded by confirming the general hypothesis that the algorithm that performs better in terms of accuracy is the Convolutional Neural Networks, which have reached a test accuracy of 95% and a training accuracy of 99%.

However, the more relevant result obtained by the research is the superiority of CT-scan images over X-ray.

Lastly, a User-friendly platform was developed. This WebApp, based on the best model obtained from the entire analysis, is designed to allow the user (doctors) to upload a breast CT-scan image and obtain, within few seconds, accurate classification of the lung in "COVID-19 affected" or "Healthy".

Section 1. Introduction

SARS-Cov-2 is a severe acute respiratory syndrome that has been assigned the name COVID-19, a combination of the family name of the virus to which it belongs "Coronavirus" and the year in which it was discovered, 2019.
It represents a new strain of coronavirus as it had never before been identified in humans. Although the early development of this virus is still the subject of extensive research, the most popular theories support the fact that it was transmitted to humans by bats, natural hosts of viruses belonging to the COVID family19.

Symptoms of COVID-19 vary based on the severity of the disease. An infected individual may be asymptomatic, and therefore present no symptoms, or have fever, cough, sore throat, weakness, fatigue, and muscle pain, and in severe cases, pneumonia, acute respiratory distress syndrome, and other life-threatening complications.

Before going in-depth with the key analysis of this paper, it is necessary to contextualize the topic with statistical data, to fully understand its medical and totalitarian relevance.

To date, there have been 162,773,940 confirmed cases of COVID-19, including 3,375,573 deaths, reported to WHO [1].
The first cases of COVID19 were identified in December 2019 in the Asian continent, specifically in China's Wuhan region, the first region to face a medical emergency. From there until the following months, the virus took hold with great speed, spreading with force mostly in the Western continents, bringing the most developed countries of the world to their knees medically, and subsequently socially and economically. Below is a current chart of the situation, divided by continents.

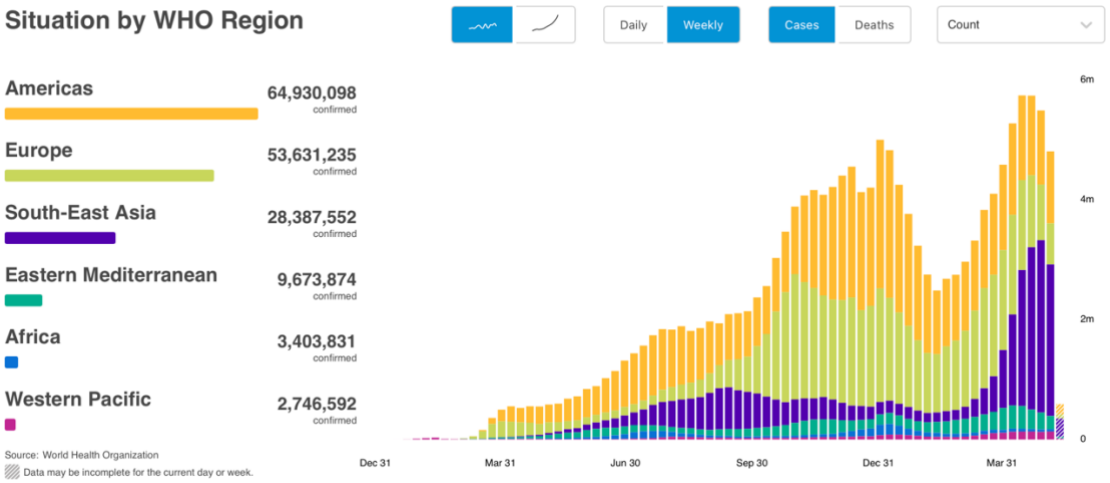

Figure 1. COVID-19 confirmed cases by continent

Although current data on the process of the fight against the virus show positivity about the future, mostly due to the fact that the entire world scientific community is united in research and development of better methods of treatment and prevention for more than a year, make estimates on the total disappearance of the virus and a potential "date" of return to normality is unrealistic, both for the continuous evolution of the virus that by definition often finds the development of variation to survive and because reaching the certainty of medicinal methods in the short term is against the protocols of scientific research.

Analysis of such a phenomenon can take hold in different fields. Important studies can invade the medical field, and therefore dwell on the definition of the virus, its origins, and its development. Other studies of equal relevance may concern the social and psychological consequences that the virus has brought, until you get to talk about the economic aspect, undoubtedly the hottest for future years and new generations.
This paper wants to focus on the medical aspect of the subject, and in particular on the detection of the virus in its early stages in an individual.

Detecting a patient's positivity immediately and effectively is undoubtedly a critical step in combating the virus. Individuals need to receive timely treatment to avoid developing more advanced symptoms, and they must also be properly isolated from the public to limit the spread of the disease;

Currently, Reverse Transcription Polymerase Chain Reaction (RT-PCR), which can detect SARS CoV-2 RNA from respiratory samples (such as nasopharyngeal or oropharyngeal swabs), is the gold standard screening method for detecting cases of COVID-19.
However, this method suffers from two fundamental problems, namely the limited availability of test kits, and the time required to get the result (from a few hours to a day or two) [2]. Not long ago were also introduced a new type of nasopharyngeal swab, called "antihygienic" (PCR) that, although faster, lacks accuracy and cost.

Thus, there is a growing need for faster and more reliable screening techniques that could be further confirmed by RT-PCR testing.

Some studies, which we will discuss in the following sections, have highlighted the potential of some artificial intelligence models in the classification of medical images, in particular X-ray and CT-scans radiographic, intending to detect the presence of the virus effectively in term of both time and accuracy. If, in fact, on the one hand, this tool is to come to the aid of the classic tools to reduce the timing of detection, it also must ensure an important relevance of classification score. This paper aims to describe a meticulous analysis which is an expansion of the analyzes already carried out.
In particular, the primary objective is to evaluate 1) which type of images between X-ray and CT-scan is best suited to this type of analysis and 2) which type of model, including CNN, Support Vector Machine, Logistic Regression, and Random Forest performs best with the selected data type.
By doing so, we will have a complete and comprehensive view of what artificial intelligence can give in this regard.

The paper will develop as follows: Section 2 will explain what "classification model" actually means and what are, to date, the most important contributions that this technology has brought to the world of medicine.
In section 3, we will analyze current studies on the development of classification models for pulmonary identification of Covid19. A literature review of the key topic of the paper will therefore be made.

In section 4 we will go into theoretical depth on the types of models used. We will therefore explain the technical functioning, the advantages and disadvantages of the 4 most famous image classification models, to have a greater awareness of the work done.

Section 5 will describe the dataset used and the methodology used to process it, to make it suitable for the models.

Section 6 represents the core of our paper. The models and their related results will be described.
Finally, after the conclusions, a user-friendly platform will be presented, designed for hospitals. It will be built on the best model found and is tought be used by anyone as an aid in identifying Covid-19 on individuals.

In dealing with such a topic, it is important to define the context of the speech. In the next section, we will define what is meant by “classification model” and we will briefly analyze the previous uses of this technology in the medical field.

Section 2. Medical applications

Although the technical functioning of the most important automatic classification methods is far from intuitive (and we will see it in section 4), the theoretical concept underlying all models of artificial intelligence is very "human". Let's take, for example, a binary classification model. A dataset is created and the images are targeted according to the category, let's say 0 and 1. At this point, the dataset is fed to the model, which, depending on the type, analyzes the images in their most intrinsic characteristics, applies mathematical models, and through the discovery of "patterns" that differentiate, on a large scale, an image 0 from an image 1, can distinguish the two types of an image with a certain level of accuracy, measured on a part of the dataset, called the test set.

The protagonist model in this field, currently, is the Convolutional Neural Networks, a piece of the Deep Learning, a fundamental branch of the world of machine learning[3], but other types of models can play very important roles in the field. This topic will be the heart of this paper. For now, let's focus on the major contributions that automated image classification

has made to the world of medicine through the review of the Article "A review of the application of deep learning in medical image classification and segmentation" [4].

2.1) Ultrasound detection of nodules

Classification models have obtained important research results in the medical field by exploiting ultrasound images.
The use of automatic tools for the detection of nodules takes hold in practically all parts of the body. Among these the most important are certainly the breast, the liver and the skin.

A particular type of neural network, called ResNet (Residual Neural Network) has been used for the recognition of **breast** melanoma in dermoscopic images in the study and reached an AUC [5] of 80%.

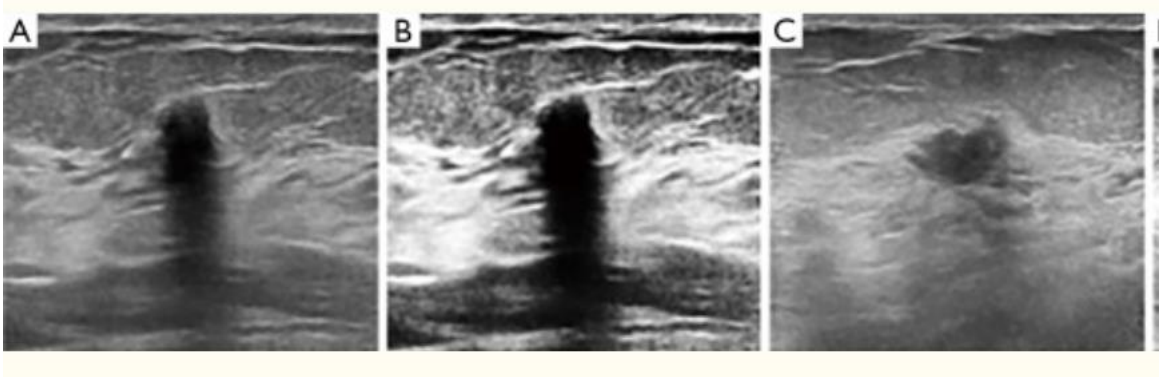

Figure 2. Ultrasound image of breast nodules

Moreover, According to relevant data statistics, the detection rate of **pulmonary** nodule has increased 5 times in recent years. With the development of deep learning technology, a series of deep learning methods are emerging to detect pulmonary nodules.

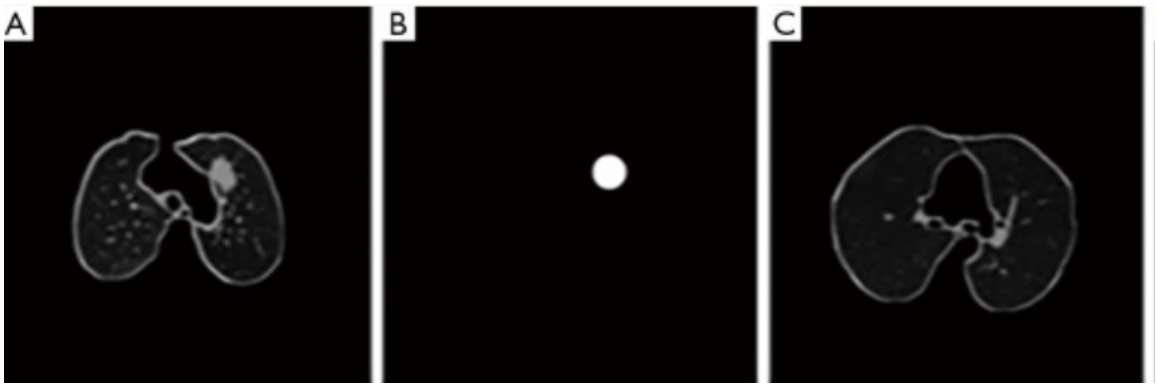

Figure 3. An example of a pulmonary nodule image

2.2) Early diagnosis of Alzheimer's disease (AD)

The study of automatic classification is very widespread as regards the detection of diseases or disorders in the brain area, and among these one of the most important is undoubtedly the treatment of Alzheimer's disease in its various processes.

Deep learning-based diagnosis of AD is primarily based on segmentation of the hippocampus, cortical thickness, and brain volume in brain MRI images. Sarraf et al. [6] trained AD samples for sMRI and fMRI using the well-known LeNet-5 framework in CNNs, respectively, with an accuracy of 98.84% and 96.85%, respectively.

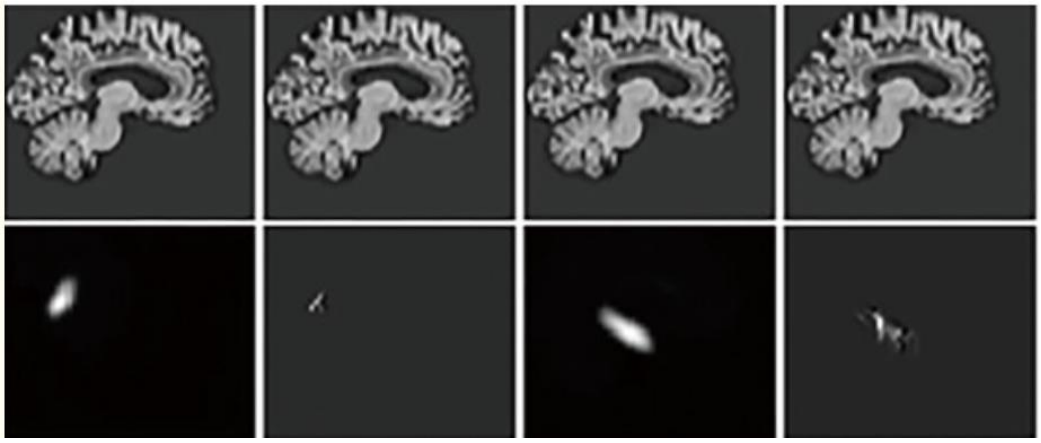

Figure 4. Hippocampus image resulted by the segmentation.

2.3) Detection of diabetic retinopathy

The primary method of studying related fundus diseases using deep learning techniques is to classify and detect fundus images, such as diabetic retinopathy detection and glaucoma detection.
The best results were obtained through a combination of CNN and complex data preparation achieving an accuracy greater than 97%.

2.4) Pathological gastric cancer

Pathological diagnosis is the "gold standard" of various methods of cancer diagnosis, which plays an important role in the medical field.
The application of deep learning mainly includes early tumor screening and benign and malignant tumor diagnosis.

The study described in the paper "Deep convolutional neural networks for automatic classification of gastric carcinoma using whole slide images in digital histopathology" describes a CNN-based diagnostic system capable of classifying gastrin cancer from benign to malignant with a sensitivity greater than 97% [7].

Section 3. Related Works

The global emergency has introduced the need for many scholars around the world to move quickly and critically. The studies related to the subject are numerous, we will analyze some of the most relevant and interesting. In particular, we will focus on studies concerning the process of distinction between a healthy patient and a patient affected by Covid, since our dataset, already well constructed and targeted, allows us to ignore the possible misclassification of a lung affected by Covid rather than by any other lung disease.

Taban Majeed1 et al [8] investigated the topic by building their dataset from 3 resources, arriving at approximately 3300 Chest-X ray images, balanced across the 2 categories.
Acting by hypothesis, they trained 12 pre-built convolutional neural networks, which differ in internal structure. In particular, they opted for the following architectures: AlexNet, VGG16, VGG19, ResNet18, ResNet50, ResNet101, GoogleNet, InceptionV3, SqueezeNet,

Inception-ResNet-v2, Xception and DenseNet201. Bearing in mind that it is beyond the scope of this paper to describe in detail the structure of each of these models, we limit ourselves in stating that the architecture of most of them are the result of years of studies by the best universities in the world or the most important companies in the industry.
The results obtained were evaluated for Sensitivity and specificity, and the study showed that the ResNet18 architecture, proposed by Microsoft in 2015, obtained the best performance, with a Sensitivity of 95.28% and a specificity of 98.72% .

Shayan Hassantabar et al [9] conducted a 2-step analysis.
Specifically, they used 3 deep learning-based methods for the detection and diagnosis of Covid19 on patients. This analysis also includes the use of Chest-Xray images. For the diagnosis of the disease, they developed two algorithms including deep neural network (DNN) on the fractal feature of the images and convolutional neural network (CNN) methods on the whole images, demonstrated that the latter analysis outperformed the former in both accuracy (93.2% vs. 83.4%) and sensitivity(96.1% vs. an 86%).
After that, they showed through the use of a CNN built with 11 layers, 3 of which are convolutional, it is possible to identify with an accuracy of 83.4% the infected region of the lung, following the process of image segmentation.

The study by Zhang et al [10] occurred about 1 year ago, and thus boasts of a small number of available data, but is interesting to name as it differs from other studies by the procedure in the whole, rather than in features of the network architecture.
They used a ResNet-18 CNN architecture as the backbone network and they added two heads on it, the classification one, and the anomaly detection one, which generated scores for detecting anomaly images (COVID-19). Surprisingly, they achieved an accuracy of 96% on detecting COVID-19 cases, and an accuracy of 70.65% on detecting non-COVID-19 cases.

The analysis conducted by Halgurd S. Maghdid et al [11] approaches, on a theoretical level, the analysis conducted by this paper, in that an evaluation is conducted on both X-ray Chest data and CT images.
Again, the chosen model encompasses the field of neural convolutional networks, serving a simple 1-layer convolutional architecture.

However, the model is trained on two very poor datasets, and the accuracy level of 94% for X-ray and 94.1% for CT evaluated on two test sets of not even 50 images suggests a very weak classification in terms of reliability.

Interesting relevances come from Prabira Kumar Sethy et al [12]. For the first time in our literature review, we consider a model that is not entirely built with neural networks.
In this case, the suggested classification model i.e. is a resnet50 plus SVM and achieved an accuracy, FPR, F1 score, MCC, and Kappa of respectively 95.38%,95.52%, 91.41%, and 90.76% for detecting COVID-19. Analysis was performed using Chest-Xray imaging.

Other relevant analyses move away from the binary classification of Covid and Non-Covid images to make room for a multi categorical classification, including image sets of unhealthy lungs subjected to different respiratory diseases than COVID-19.

Md. Zabirul Islam et al [13] proposed an interesting combination of a convolutional neural network (CNN) and long-term memory (LSTM) to automatically diagnose COVID-19 from X-ray images. In this system, CNN is used for deep feature extraction and LSTM is used for detection using the extracted feature. The system, tested on a dataset of approximately 5000 Chest-Xray images, showed very positive results, achieving 99.4% accuracy, 99.9% AUC, 99.2% specificity, 99.3% sensitivity, and 98.9% F1-score.

Finally, we consider the work described in Morteza Heidari et al [14].
Their methodology involves that pre-processed Chest-Xray images are fed into three input channels of a convolutional neural network (CNN) model based on transfer learning to classify chest X-ray images into 3 classes of COVID-19-infected pneumonia, other community-acquired pneumonia without COVID-19, and normal (non-pneumonia) cases. The dataset included 8474 chest radiographic images, divided by 415, 5179, and 2,880 cases in three classes, respectively.
This scheme yielded an overall accuracy of 94.5% in 3-class classification, 98.4% sensitivity, and 98.0% specificity in the classification phase.

The strength of this study lies in the way the images were processed. Using a CAD(computer-aided diagnosis) method, two stages of image preprocessing are applied.

The first one aims to remove most of the aperture regions and the second one aims to process the original image using a histogram equalization algorithm and a bilateral low-pass filter.

Section 4. Models explanation

In this section, we are going to explain the technical functioning of all 4 models that will be used and evaluated during our analysis. It is believed that to fully understand our results and therefore to be able to make the relative conclusions, it is obligatory to know what is behind a specific mathematical structure of a model. Before that, we define the concept of a supervised model, a definition common to all 4 types of models used in our analysis.
In statistical learning theory, the supervised learning problem is formulated as follows. We are given a set of training data {(x1,y1)... (xl,yl)} in Rn × R sampled according to an unknown probability distribution P(x,y), and a loss function V(y,f(x)) that measures the error, for a given x, f(x) is "predicted" instead of the true value y. The problem is to find a function f that minimizes the expectation of the error on the new data i.e. find a function f that minimizes the expected error: $\int V(y,f(x))\, P(x, y)\, dx\, dy$.
In statistical modeling we should choose a model from the space of hypotheses, which is closest (concerning some measure of error) to the underlying function in the objective space [15].

We will start with CNN networks, then we will talk about Support Vector Machine, Logistic Regression, and Random Forest.

4.1 Convolutional Neural Network

4.1.1 Background

The concept of Neural Convolutional Networks was born in the early 1980s due to discoveries in the medical field about the cerebral cortex.
During those years, numerous medical studies shed light on the fact that the human brain processes visual stimuli not in their entirety, but rather by dividing them into small parts, analyzing them with different neurons, and then assembling them. This happens because different types of neurons process parts of what is seen in different ways: some neurons can

process images for horizontal lines, others for vertical lines, and not all have the same dimensionality of the reception field.
This characteristic gave a strong impetus to the development of technologies capable of recognizing and classifying images, and over time, thanks to both to the continuous increase in the computational power of machines and the greater availability of images on which to train models, CNNs have become able to match the recognition capacity of the human being, and in some cases to exceed it.

Although, as we said, this particular type of neural network was born in the 80s, the turning point came at the end of the 90s, with the presentation by the researcher Yann le cun. [16] of the famous architecture LeNet5 [17]. This architecture differed from the models used up to that time due to the presence of 2 types of new blocks, or the convolutional block and the layer Pooling. Through the use of these, the neural network expanded the classic architecture of the Multilayer perceptron, solving the problems of the latter in 1) processing large amounts of images and 2) recognizing the same element if it appears in two different positions in different images. We will now explain in detail the technical functioning and the role of all the layers involved in the architecture.

4.1.2 Convolutional layer

As the name suggests, the architecture of the CNN is characterized by the presence of hidden layers, called convolutional layers. These, as already mentioned, are the fundamental part that differentiates a CNN from a standard neural network, and have the function to identify patterns within the images. Like all other types of layers, the convolutional layer receives an input, transforms it, and passes it to the next layer in the form of output. This transformation operation is called the convolution operation.
Patterns are identified within the layer by filters, whose size is variable. But what is a pattern? As we know, images are composed of different shapes.
A pattern is nothing else than a shape, so it can be related to edges, corners, squares, or geometric shapes. The simplest shapes are identified in the initial part of the network, until they arrive, with increasingly complex filters, in the entirety of the subject represented in the image to be classified.

The size of the filter is variable and must be chosen during the construction of the convolutional layer. Mathematically, a filter is a relatively small matrix for which the number of columns and rows initialized with random numbers is defined. It is important to clarify that a computer sees an image as a two-dimensional array of pixels. In the case of Grayscale images, for example, the value of a pixel varies from 0(black) to 256(white).
Suppose now that the filter is 3x3 in size. Upon receiving the image input, the convolutional layer divides the entire pixel array into many 3x3 blocks. When the filter encounters the first block of divided pixels, the product of the matrices is performed and recorded. This happens for each pixel matrix.

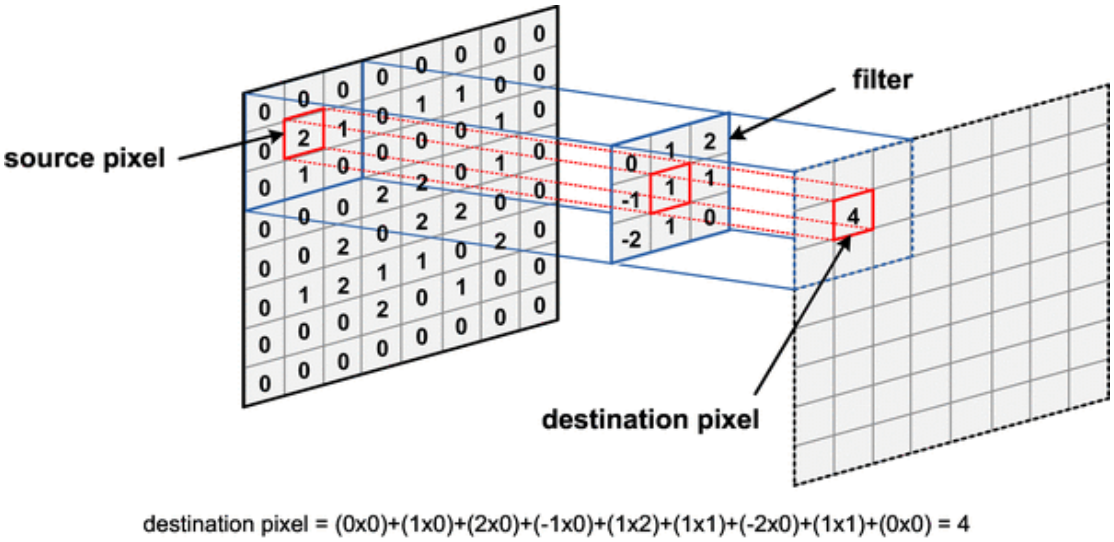

Figure 5. Convolutional layer 3x3[18]

After the block has convolved the entire set of 3x3 pixel matrices, we get a new representation of the input, i.e. the output of the first convolutional layer. Each convolutional layer can be composed of filters of different nature, with the function of identifying different patterns. Each time a new layer is created, the recognized pattern becomes more and more complex.
For clarity of terminology, a filter can also be called a kernel, and the output of the image processed by the kernel is called a feature map.
We do not have to define the filters manually: instead, during training, the convolutional layer will automatically learn the most useful filters for its task, and the layers above will learn to combine them into more complex patterns and will discard, through the use of an activation function, the neurons deemed less useful for classification.

The main benefits brought by this layer are 2 [19]:

- *Sparse connectivity*: In contrast to Full connected layers, in CNNs, only a few weights are available between two adjacent layers. Thus, the number of weights or connections needed is small, while the memory required to store these weights is also small.
- *Sharing of weights*: There are no weights assigned between two neurons of neighboring layers in the CNN since all weights operate with one and all pixels of the input matrix. Learning a single set of weights for the entire input will significantly decrease the required training time and various costs since there is no need to learn additional weights for each neuron.

4.1.3 Max pooling layer

The Pooling layer is inserted between the convolution layers. This layer aims to reduce the number of parameters and computations in the network, controlling overfitting by progressively reducing the spatial size of the network. Several types of pooling methods are available for utilization in various pooling layers. These methods include tree pooling, gated pooling, average pooling, min pooling, max pooling, global average pooling (GAP), and global max pooling.

The most frequent operation is called "Max Pooling", and as the name implies, it extracts only the maximum from a pool. This is done with the use of filters that flow through the input; and at each step, the maximum parameter is taken and the rest is eliminated, reducing the dimensionality of the network.

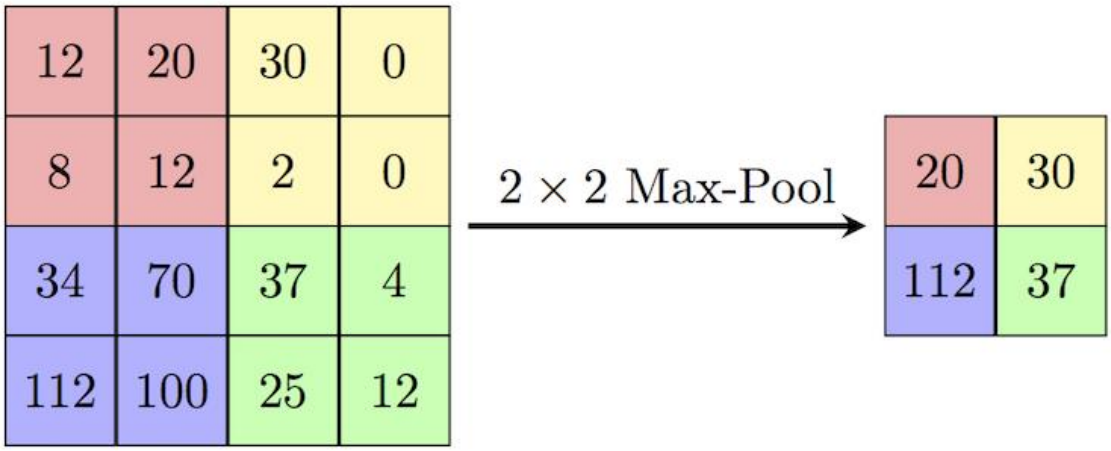

Figure 6. Max pooling example

4.1.4 Activation functions

The activation function is a node that is put at the end of or between neural networks. They help decide whether or not the neuron activates. [20].

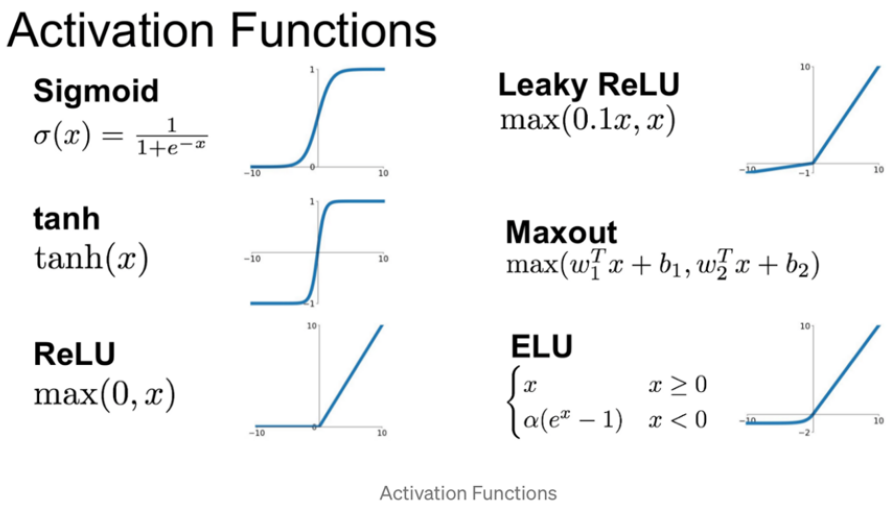

Figure 7. Types of Activation functions

Among them, the most widely used in CNN construction is The ReLU function.

One of the major advantages of ReLU over other activation functions is that it does not activate all neurons at the same time. From the image of the ReLU function above, we will notice that it converts all negative inputs to zero and the neuron is not activated. This makes it very computationally efficient, as only a few neurons are activated at a time. It does not

saturate in the positive region. In practice, ReLU converges six times faster than the tanh and sigmoid activation functions.

One disadvantage that ReLU has is that it is saturated in the negative region, which means that the gradient in that region is zero. With the gradient equal to zero, during backpropagation all weights will not be updated, to solve this, we use Leaky ReLU. Also, ReLU functions are not zero-centered. This means that to get to its optimal point, it will have to use a zig-zag path which can be longer.

An equally important role in an analysis like the one described by our paper is covered by the final "sigmoid" activation function, placed after the last layer, i.e. the Dense Layer.

The sigmoid function takes one value as input and produces another value between 0 and 1. It is non-linear and easy to work with when building a neural network model. The good part about this function is that it is continuously differentiable over different values of z and has a fixed output range.

The sigmoid function is used mostly in binary classification models as part of the output layer to capture the probability ranging from 0 to 1. In cases of multi categorical classification, the "softmax" function is preferred [21].

4.1.5 Dense Layer

The Dense layer is the final layer of the network and is the only one not "peculiar" to the CNN network.

As the name suggests, at this stage the layers are fully connected (dense) by the neurons in one layer of the network. Each neuron in a layer receives input from all the neurons in the previous layer, so they are densely connected.

In other words, the dense layer is fully connected, meaning that all neurons in one layer are connected to those in the next layer.

A densely connected layer provides learning features from all combinations of the previous layer's features, and thus serves to somatize all the information found and give the classification result [22].

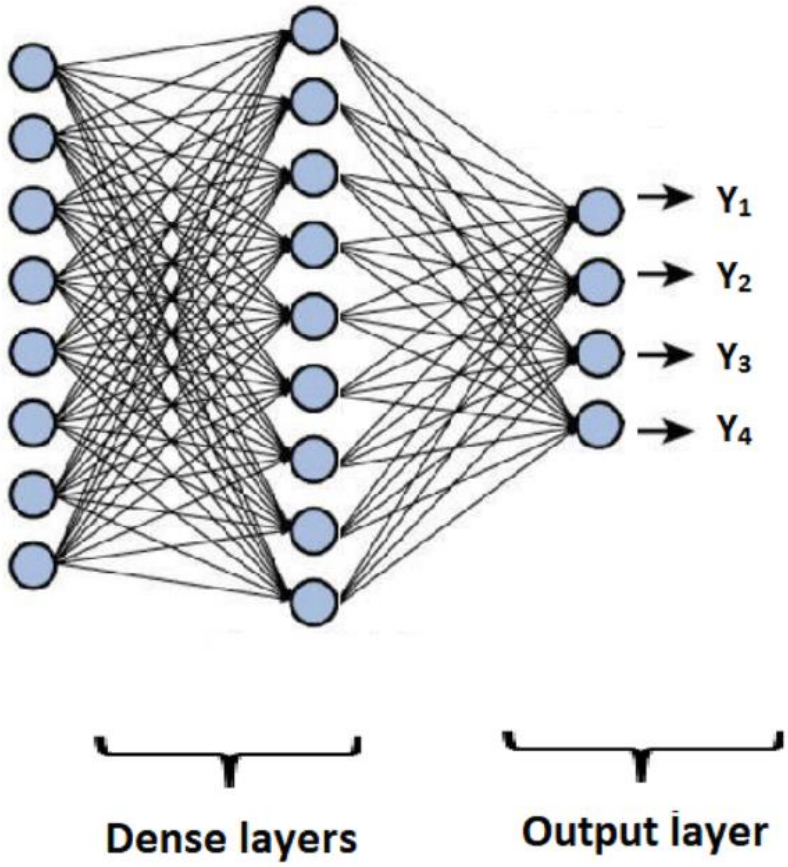

Figure 8. Dense layer graphical representation [23]

4.1.6 Loss Functions

The final classification is obtained from the output layer, which is the last layer of the CNN architecture. To evaluate the error and accuracy of the model, we use Some functions, called loss functions, which test the results are random samples of the labeled dataset.

This error reveals the difference between the actual and predicted result. Subsequently, it will be optimized through the CNN learning process at each sliding of an epoch.

Different types of loss functions are employed in various types of problems. Let us see some of them.

- *Cross-Entropy or Softmax loss function*: This function is commonly used to measure the performance of the CNN model. Its output is the probability $p \in \{0, 1\}$. We will use this in our analysis.

- *Euclidean loss function*: This function is widely used in regression problems. It is also the so-called mean square error.

- *Hinge loss function*: This function is commonly used in problems related to binary classification and is most commonly associated with Support Vector Machine models.

4.1.7 The Overfitting problem

During the training phase of a CNN network, the risk of overfitting. There are many methods to reduce this problem. Among them, the most common are Dropout, BatchNormalization, and Data Augmentation.

- *Dropout*: Dropout is a regularization method that approximates the training of a large number of neural networks with different architectures in parallel.
  During the training phase, a definied number of layer outputs are randomly ignored or "dropped out". This has the effect of making the layer look similar and be treated as a layer with a different number of nodes and connectivity than the previous layer.
  The dropout has therefore the effect of making the training process noisy, forcing nodes within a layer to probabilistically assume more or less responsibility for inputs.
  This conceptualization suggests that perhaps dropout interrupts situations in which network layers adapt to correct errors in previous layers, making the model more robust.

- *Data Augmentation*: Practicing Data Augmentation involves expanding the size of your dataset by adding slightly modified copies of images which already compose the dataset [24].

- *Batch Normalization*: Following a Gaussian distribution, this method allows subtracting the mean and dividing by the standard deviation of the output at each layer. The normalization is a very common practice in the pre-processing phases in all models of artificial intelligence and presents itself as an excellent solution in handling the difficulties due to the enormous amount of data handled.

Below an image of the entire architecture of a classic CNN.

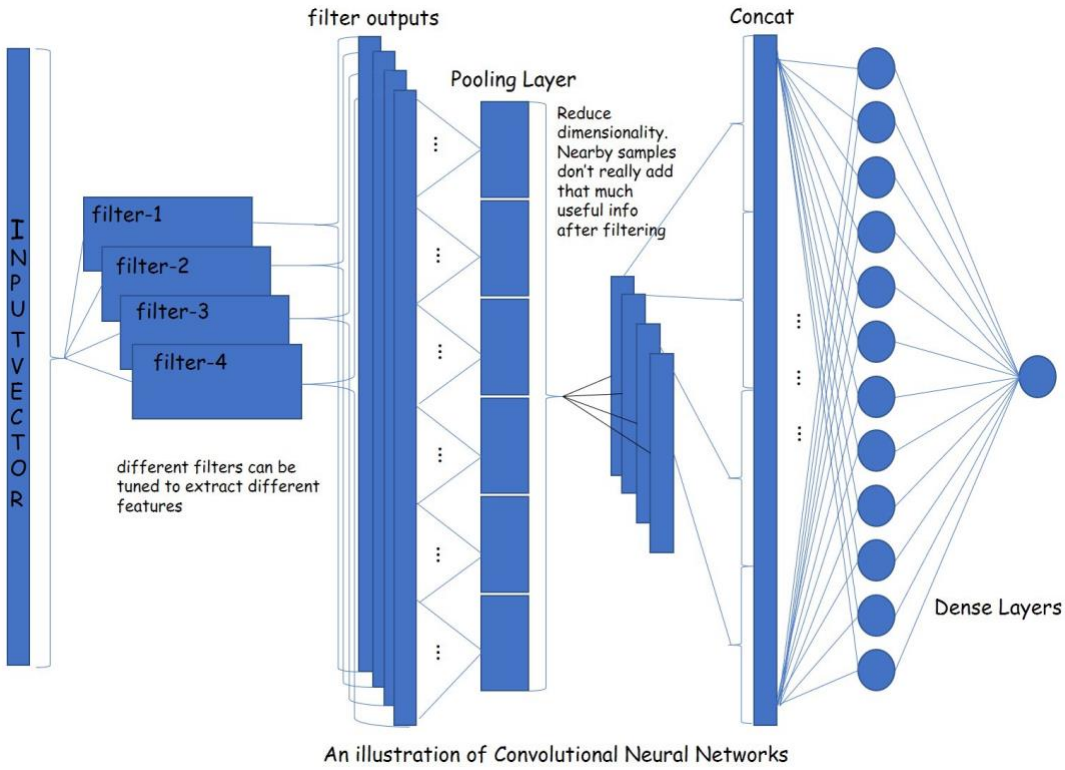

Figure 9. CNN full architecture [24]

4.2 Support Vector Machine

4.2.1) Background

The Support Vector Machine is a type of supervised artificial intelligence model used mostly for binary classification problems, invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963.
Although indicated for problems with datasets of small to medium size, this type of model can perform both linear and non-linear regressions.

In the case of binary classification, such as the one that is the heart of this paper, the Support Vector Machine is based on the idea of finding a hyperplane that best divides a dataset into two classes. To understand how it works, it is good to define some key concepts:

- *Hyperplane or linear decision boundary*: For a classification task with only two spatial dimensions x1 and x2 (or x and y), a hyperplane is depicted as a line that separates and

classifies a data set. At 3 dimensions, a hyperplane is represented as a plane. With more than three dimensions it is generically referred to as a hyperplane.

- *Support Vector*: these are the data points closest to the hyperplane. These points depend on the data set being analyzed and if they are removed or modified they alter the position of the dividing hyperplane. For this reason, they can be considered the critical elements of a data set.

- *Margin*: this is defined as the distance between the support vectors of two different classes closest to the hyperplane. At the middle of this distance the hyperplane is drawn, or a straight line in case you are working in two dimensions.

Having identified the elements that make up the SVM puzzle, let's go into more detail about its technical operation.

The Support Vector Machine aims to identify the hyperplane that best divides the support vectors into classes. To do this it has 2 ways, namely to look for a linearly separable hyperplane and, if this is not possible, to look for a non-linear one.

4.2.2) Linearly separable hyperplane

SVM Look for a linearly separable hyperplane or decision boundary that separates the values of one class from the other. If there is more than one, look for the one that has a higher margin with the support vectors, to improve the accuracy of the model.

Theoretically, you can draw an infinite number of straight lines to separate 2 categories of items. The problem is to find which of the infinite number of straight lines is optimal, i.e., the one that generates the least classification error on a new observation.
The further away from the hyperplane, our data points are, the more confident we are that they have been classified correctly. Therefore, we want our data points to be as far away from the hyperplane as possible while remaining in the correct part.

So, when new test data is added, the model decides the class we assign to it. This is what the Support Vector Machine does in practice when it finds a linearly separable hyperplane.

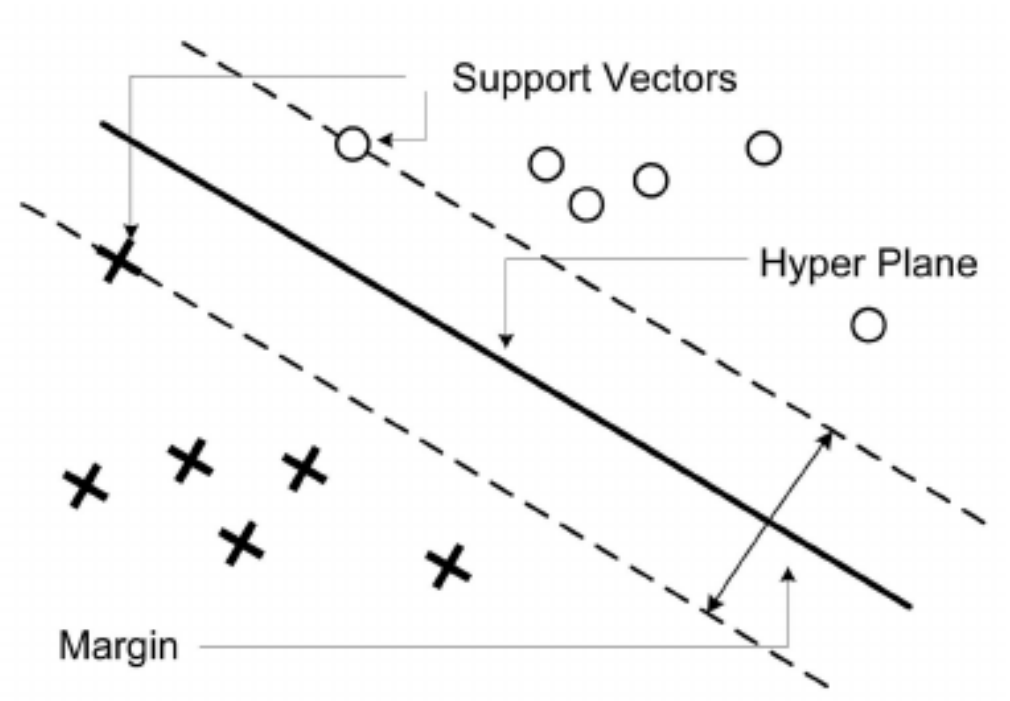

Figure 10. Linearly separable Hyperplane [26]

At the mathematical level, the issue looks as follows.
The optimal hyperplane can be defined as a multidimensional scalar product in compact form:

$$\vec{w}\vec{x} + w_0 = 0$$

where w is the weight vector, x is the input feature vector, and w0 is the bias. In a 2-dimensional problem, the separation hyperplane is found with:

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

The points above the hyperplane, and representing a class, satisfy the next condition:

$$w_0 + w_1 x_1 + w_2 x_2 > 0$$

and any point that is below the hyperplane belongs to the other class, which is satisfied by the next condition:

$$w_0 + w_1 x_1 + w_2 x_2 < 0$$

Wanting to include in these conditions also the limits of the edges of the classes, it is possible to adjust the coefficients or weights w1 and w2 in the following form:

$w_0 + w_1 x_1 + w_2 x_2 \geq 1$, per y = + 1

$w_0 + w_1 x_1 + w_2 x_2 \leq -1$, per y = - 1

The two results above represent the margin boundaries, shown in the picture as dashed.

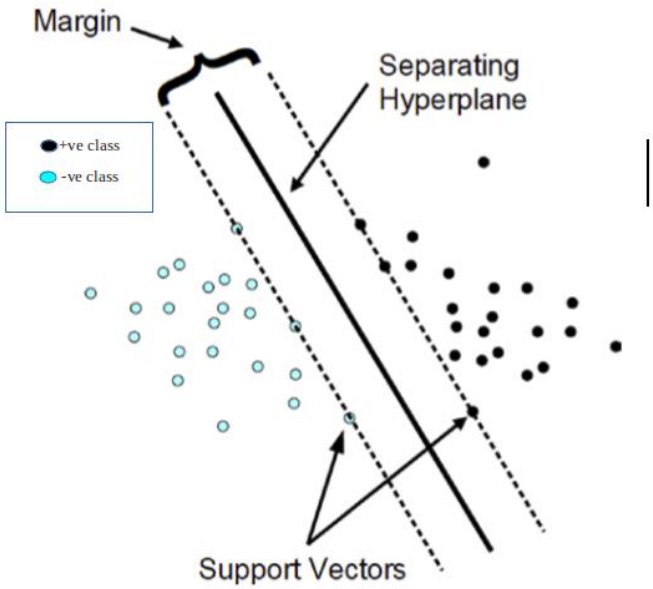

Figure 11. SVM margins

4.2.2) Non-linear separable Hyperplane

An alternative use for SVM is the kernel method (or kernel trick), which allows us to model higher-dimensional nonlinear models.

Indeed, there are situations where categories can be distinguished, but not by a simple straight line.

In this case, a new dimension z must be created and must be set in order to be calculated in a way that is convenient for the case.

For example, if the data can be suitely separated through a circle the third dimension will be $z = x^2 + y^2$.

The third dimension will give us a 3-dimensional space, and make it easy to distinguish the 2 classes. In doing so, a kernel method is used. Let's delve into the mathematical side of things.

In general, the Kernel can be defined as:

$K(x, y) = < f(x), f(y) >$

where K is the kernel function, x, y are n-dimensional input vectors, f is used to map the input from n-dimensional space to m-dimensional space (higher level than level n), While <x, y> denotes the scalar product. If the new space we want is $z = x^2 + y^2$, the scalar product in that space will result as from the following expressions:

$$x \cdot y = x_1 \cdot x_2 + y_1 \cdot y_2 + z_1 \cdot z_2$$

$$x \cdot y = x_1 \cdot x_2 + y_1 \cdot y_2 + (x_1^2 + y_1^2) \cdot (x_2^2 + y_2^2)$$

Using a nonlinear kernel (as in this case) we can obtain a nonlinear classifier without completely transforming the data: we just change the scalar product to that of the space we want and SVM will help us find the best hyperplane [27].

In conclusion, we can define SVM as a viable classification alternative to neural networks due to its strong effectiveness in high space dimensions and its versatility of use in different problems and data structures.
However, this type of model, as opposed to CNN's and the models that we will see in the following sections, is flawed by the lack of the "Probabilistic" component; as it is designed SVM does not allow us to assign a probability to a given category as the element is classified only according to the integer it belongs to. Moreover, SVM cannot boast of great transparency in the training phase.

4.3 Random Forest

The Random Forest algorithm is a supervised learning algorithm that represents a type of ensemble model, which uses bagging as the ensemble method and the decision tree as the individual model.

This means that a random forest combines many decision trees into one model. Individually, the predictions made by the decision trees may not be accurate but combined, the predictions will be closer to the result on average.

The final result returned by the Random Forest is nothing more than the average numerical result returned by the different trees in the case of a regression problem, or the class returned by the largest number of trees in the case the Random Forest was used to solve a classification problem.

To understand how Random Forests work, it is necessary to become familiar with decision trees. Decision trees are predictive models that use a set of binary rules to compute a target value. Two types of decision trees are classification trees and regression trees. The former, the type we will use in our analysis, aim to assign an item to a certain category, the latter, on the other hand, work more on continuous data, and therefore have output numerical or percentage values.

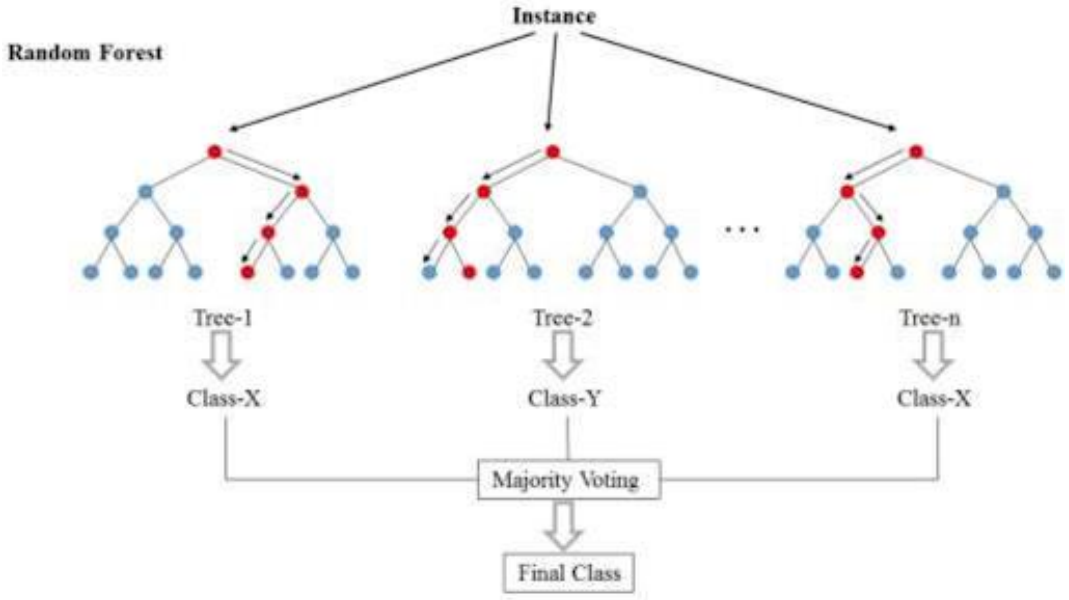

Figure 12. Example of a binary Random Forest

Let's see how it works in detail.

As has been previously anticipated, random forests, like decision trees, can be used to solve classification and regression problems but can overcome the disadvantages associated with individual decision trees while retaining the benefits.
The random forests model computes a response variable (in our case the classification of positivity to COVID19) by creating many different decision trees and then placing each object to be modeled (in our case the object is an image pixel) in each of the decision trees. The answer is then determined by evaluating the response from all the trees. In other words, if 1000 trees are grown and 900 of them predict that a particular pixel is COVID19 and 100 predict that it is Non-COVID19, the predicted output for that pixel will be COVID19.
Because the random forest predictions are derived using a forest of trees, it is not possible to easily illustrate how the predictions are made. To illustrate the process it would be necessary to draw all the trees for each prediction which would result in hundreds of decision tree diagrams for each model. This practice is very complex, but it could be very useful if we work on numerical variables. In fact, it could be used to attribute importance to each variable, but for what concerns our analysis it is superfluous to look pixel by pixel (we will return to this point in the conclusions).

There are two steps involving random selection that is used when forming trees in the forest:

1) The *random selection*, with replacement, of data from the training areas provided to construct each tree. For each tree, a different subset of the training data is used to develop the decision tree model, and the remaining third of the training data is used to test the accuracy of the model. The data used for testing are often called "out-of-bag" samples. This practice is common to all artificial intelligence models.
2) The *Random sampling* used to determine the split conditions for each node in the tree. At each node in the tree, a subset of the predictor variables is randomly selected to create the binary rule. The number of randomly selected predictor variables can be set by the user or the choice can be left to the random forest algorithm. Using a randomly selected subset of the predictor variables to split each node results in less correlation between trees and consequently a lower error rate. Although smaller subsets of predictor variables will reduce the correlation between trees, it will also result in trees with less predictive power than trees constructed using more predictor variables. It is important to select the number of variables that provide sufficiently low correlation with adequate predictive power. Fortunately, the optimal range for the subset of predictor variables is quite large and there are easy tests that can be performed to select an optimal subset size. Fortunately, the optimal range for the subset of predictor variables is quite large and there are easy tests that can be performed to select an optimal subset size [28].

   As with the rest of the models explained, we will discuss the parameters in more detail in Section 5.

Among the main advantages of the Random Forest we highlight the greater consistency in terms of overfitting compared to the rest of the decision trees, and its elasticity both on the fact that it can work even with a dataset characterized by missing or null values, and on the fact that it does not require prior normalization of the data.

However, this algorithm lacks computational efficiency and therefore requires a lot of computing power and time in training phases.

4.4 Logistic Regression

Logistic regression is a classification method belonging to the family of supervised learning algorithms. It allows generating a result that represents a probability that a given input value belongs to a certain class, using precise statistical methods.

In the case of binomial logistic regression, the probability that the output belongs to one class will be P, while it belongs to the other class 1-P (where P is a number between 0 and 1 because it expresses a probability).

Like all regression analyses, logistic regression is a predictive analysis that is used to measure the relationship between the dependent variable (i.e., what we want to predict) and one or more independent variables (our characteristics) by estimating probabilities using a logistic function.

These probabilities are then transformed into binary values (which take on values between 0 and 1) to make a prediction. We then assign the result of the prediction to the class it belongs to, based on whether or not it is close to the class itself.

If, for example, we obtain a value of 0.8 after applying the logistic function, we will say that the input has generated a positive class and will be assigned to class 1 (vice versa if it had obtained a value < 0.5).

Logistic regression overcomes the problem in the classification range of linear regression related to the fact that a prediction can fall outside the range (0,1). To do this, this model makes use of the logistic function.

$$p(X) = \beta_0 + \beta_1 X. \qquad p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}.$$

Also called a sigmoid function, the logistic function is an S-shaped curve that can take any real-valued number and map it to a value between 0 and 1, excluding extremes. Let's go over it in mathematical detail. As we can see from the image, the function consists of :

1) The mathematical symbol *e*, which is nothing more than the base of natural logarithms
2) The regression parameters *β0* and *β1*.

They are unknown and must be estimated from the available training data. The more general maximum likelihood method is preferred since it has better statistical properties. The basic intuition behind using maximum likelihood to fit a logistic regression model is as follows: we seek estimates for β0 and β1 such that the predicted probability pˆ(xi) of default for each

individual, matches as closely as possible the observed default state of the individual. In other words, we seek to find βˆ0 and βˆ1 such that fitting these estimates into the model for p(X) yields a number close to one for all individuals who belong to a given category, and a number close to zero for all individuals who belong to another [29].

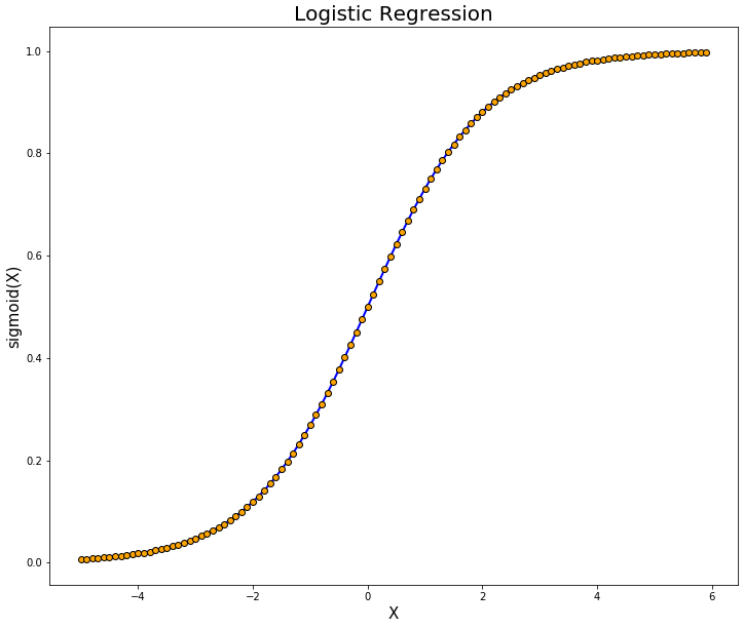

Figure 13. Logistic function

Logistic regression is definitely the simplest model we have discussed. This can be viewed with both a positive and negative eye. While in fact, the "Linear" provenance of the mathematics on which it is based makes the algorithm very easy to implement and efficient, it sometimes limits it in accuracy.

Section 5. Data and Methodologies

At a general level, the work was performed in the following steps:

1) *Data collection,* i.e., both X-Ray and CT images already pre-labeled in COVID and Non-COVID.

2) *Identification* of the 4 models to be used

3) *Unfolding* of each model. For each model, the data was pre-processed in a manner consonant with the algorithm and then trained.

4) *Parameter Optimization*. For each model, through optimization techniques, we found the best-performing parameters in terms of accuracy.

5) *Analysis* of results and conclusions

In this section, we are going to explain what in the world of artificial intelligence happens around model training. First, the two datasets used, namely the X-ray dataset and the CT dataset, will be described.
Secondly, the pre-processing of the dataset will be explained. This part is essential because it ensures the perfect functioning of the models and it happens, obviously, before the training phase.
Finally, we will address the process of parameter optimization. As we have seen, each type of algorithm is based on different mathematical models and is therefore characterized by different parameters. By attempting different combinations of these parameters we can compare different types of structures within the models and choose, in the end, the one that performs best.

5.1 Dataset

The COVID-19 dataset [31] consists of Non-COVID and COVID cases of both X-ray and CT images. The associated dataset is augmented with several augmentation techniques to generate approximately 17099 X-ray and CT images. The dataset contains two main folders, one for X-ray images, which includes two separate subfolders of 5500 Non-COVID images and 4044 COVID images. The other folder contains the CT images. It includes two separate subfolders of 2628 Non-COVID images and 5427 COVID images. The dataset is therefore well-balanced.
It is important, to better understand the data you are working with, to clarify what they are and the differences between the 2 types of data that make up the dataset:

1) *X-rays* are a type of radiation called electromagnetic waves. X-ray imaging creates images of the inside of your body. The images show parts of your body in different shades of black and white. This is because different tissues absorb different amounts of radiation. The calcium in your bones absorbs most of the X-rays, so your bones appear white. Fat and other soft tissues absorb less and appear gray. Air absorbs less, so the lungs appear black. The most familiar use of X-rays is checking for fractures (broken bones), but X-rays are also used in other ways such as identifying lung disease, as in our case.

Figure 14. COVID-19 X-ray

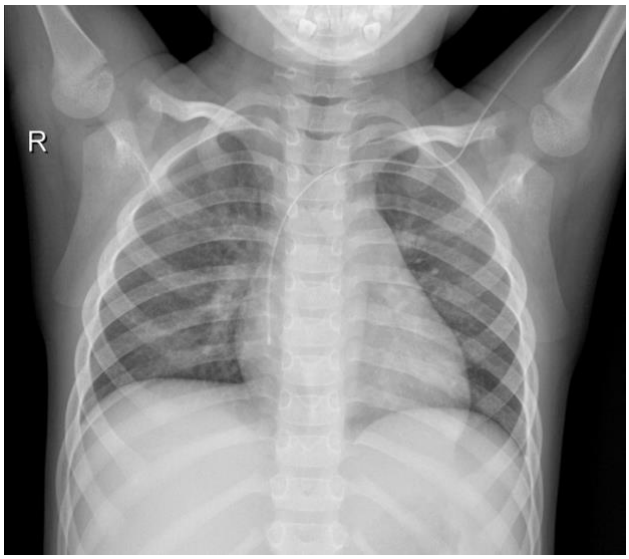

Figure 15. Non-COVID-19 X-ray

2) *CT scan*, or computed tomography (CT or CAT scan), uses computers and rotating X-ray machines to create cross-sectional images of the body. These images provide more detailed information than regular X-ray images. They can show soft tissue, blood vessels, and bone in various parts of the body.

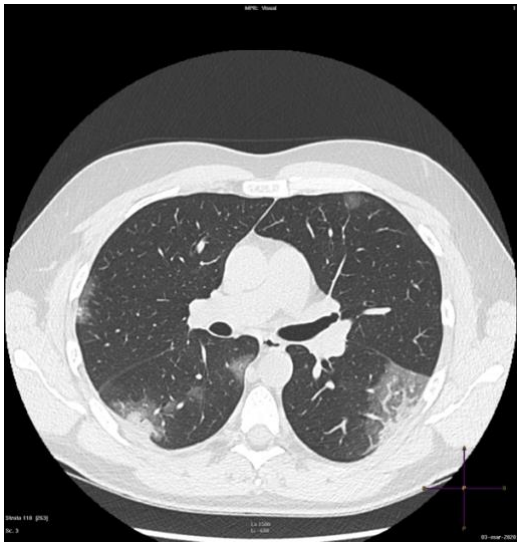

Figure 16. COVID-19 CT SCAN

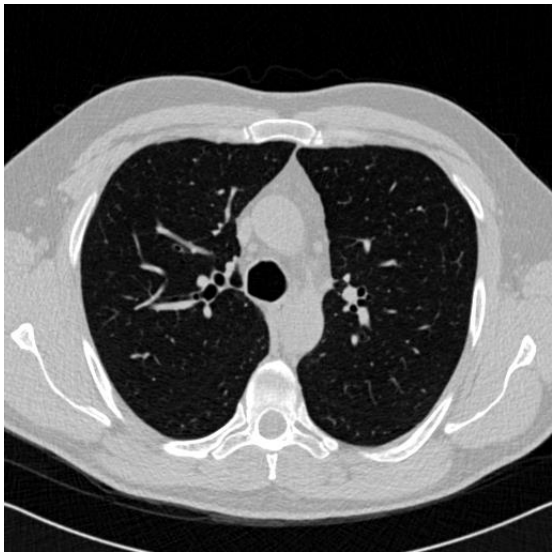

Figure 17. Non-COVID-19 CT SCAN

For all models, the division of the data into train and test data was done with a 0.2 split. This means that the models were trained with 80% of the dataset, and accuracy scores were measured on the remaining 20%.

5.2 Pre-processing

Many data scientists today see data preparation as the key to increasing their ability to efficiently use data to optimize business processes or, primarily, to enable new and innovative business models.

A good data preparation process is a crucial step in building a predictive model and focuses on organizing and preparing data to get the maximum benefit from analyzing that data. This saves time in searching for information as the data will be more consistent, valid, and free of errors of any kind.
In our case, the dataset does not present "noise" or particular errors to be handled or removed, but undoubtedly each model requires a certain pre-processing of the images to read, study and predict them. If it is true that intelligence models simulate and emulate human intelligence, at the same time they need their key to read. Each model works differently, but at the base of everything, there is the premise that a machine reads an image as a sequence of pixels. The data processing part is common for all image classification models, and to describe them all separately would be useless and redundant.
Let us now go into the details of the issue.

The pre-processing of the data for all of our models are composed by few but vital steps.

Taking into account the fact that the images are arranged in a folder divided by the 2 categories, what we wanted to obtain was a merged dataset composed of sequences of targeted arrays.

To achieve this, using the library OpenCV [32] and then the library NumPy [33] we created a function that would perform, sequentially, the following steps:

1) *Read* the directory of each image, taking into account the category to which it belongs.

2) *Transform* the image into an array with values limited to the Greyscale(0-255)
3) *Resize* the array in fixed dimensions (in our case 150,150) to obtain a homogeneous and not too heavy dataset
4) *Create* a list for which each element was composed of the array of the image plus the label, represented by the category of belonging (0 if COVID, 1 if Non-COVID).

Then this list is "*shuffled*" to create maximum randomness in the order of the elements, and then divided into Features (the array) and Label (the category 0 and 1). The model will come trained recognizing the 80% random Features given their specific Label, in order then to be tested on the remaining 20% of the Features in the moment of attribution of the unknown labels.

The only difference to highlight is the presence of the *flatten*() function in the case of SVM, Random Forest and Logistic regression, which is not present in the data processing in the CNN model.

More insights and details are present within the code (see last section).

5.3 Parameter Hypertuning

5.3.1 Premise

In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. We define hyperparameters as the chunk that contains the data that governs the training process itself.

Specifically, the training application handles three categories of data as it trains the model:

1) *Input data* (also called training data), defined as a collection of individual records (instances) that contain the features important to the machine learning problem. These data are used during training(train) to configure the model and make accurate predictions about new instances of similar data(test). In our case, it is the images.

2) *The parameters* of your model, which are variables that the chosen machine learning technique uses to fit the data. For example, a convolutional neural network (CNN) consists of processing nodes (neurons), each with an operation performed on the data as it travels through the network. When the CNN is trained, each node has a weight value that tells the model how much impact it has on the final prediction. These weights are an example of the model parameters.
3) *Hyperparameters*, defined as the variables that govern the training process itself. For example, part of setting up a deep neural network is deciding how many hidden layers of nodes to use between the input layer and the output layer, and how many nodes each layer should use (we'll look at this in more detail in the next section). These variables are not directly related to the training data. They are configuration variables [34].

Our optimization work focused on the hyperparameters of the model.
The logic behind this lies in trial and error. For each model, we prefixed a set of different parameters depending on the type of algorithm, and we tried all possible combinations of these parameters, analyzing the results. In this way we will have a complete vision of how the algorithm can behave in the problem, obtaining an optimized architecture.
In the world of artificial intelligence there are 2 main tuning techniques:

1) *Grid search*: Grid search is a traditional way to perform hyperparameter optimization. It works by exhaustively searching through a specified subset of hyperparameters.

The advantage of grid search is that it is guaranteed to find the optimal combination of the given parameters, as it tests every single combination. On the other hand, however, it has the disadvantage of being very expensive in terms of time and computation.

2) *Random Search*: Random search differs from grid search mainly in that it searches the specified subset of hyperparameters randomly instead of exhaustively. The main benefit is the decrease in processing time but we are not guaranteed to find the optimal combination of hyperparameters [35].

For Support Vector Machine, Logistic Regression and Random forest we used a particular type of Grid search, namely the Halved GridSearch . This funcion reduce the computational cost of the operation by running a first selection of candidates using a subset of the entire dataset and then proceeds iteratively in the selection increasing the size of the initial subset at each step. All

this testing the model 3 times through cross validation, in order to have even more consistent results.
Let's see in detail how the optimization for each type of model developed.

5.3.2 CNN

The optimization process to find the best CNN architecture, the same of course for both X-ray and CT photos, involved the realization of 18 different types of compositions, obtained through the combination of 3 different elements: the number of convolutional layers, the size of the layers and the additional number of Dense layers. We highlight "additional" because a dense layer is mandatory for the model to work. The combination data were as follows:
1) Number of *convolutional layers*: 1,2,3.
2) *Size of layers* (both convolutional and dense): 32,64,128.
3) Number of *additional Dense layers*: 0,1.

The choice of such parameters is justified by the average size of the dataset. For the problem posed, in fact, we believe that among the structures that vary from a total number of layers equal to 2 up to a maximum of 5 there can be one that represents the right compromise between precision and non-overfitting.

5.3.3 SVM

1) *Kernel* --> The function of the kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. These functions can be different types. For example linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid.
2)*Gamma* --> defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’. The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors.
3) *C* --> The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

Specifically, the combinations tested in our study were created using the following features:

1) Kernel of type *RBF*(Radial basis function) with gamma values [1e-3, 1e-4] and C = [1, 10, 100, 1000].
2) *Linear* kernel with default gamma value (0.2857) and C = [1, 10, 100, 1000].

A total of 12 different types of architecture were therefore tested, both for X-ray and CT dataset.

5.3.4 Random Forest

For what concerns the Random forest model, the combinations tested are more numerous, and with them the variety of parameters tested. Let's see them one by one and explain their usefulness:

1) *min_samples_split*: it specifies the minimum number of samples required to split an internal node. **The possible values set have been** 2,5,10.

2) *min_samples_leaf*: specifies the minimum number of samples required to be at a leaf node. The possible values heard were 1,2,4.

3) *estimators*: number of trees in the forest. The possible values are 300 and 400.

4) *Bootstrap*: When training, each tree in a random forest learns from a random sample of the data points. The samples are drawn with replacement, known as bootstrapping, which means that some samples will be used multiple times in a single tree. The idea is that by training each tree on different samples, although each tree might have high variance with respect to a particular set of the training data, overall, the entire forest will have lower variance but not at the cost of increasing the bias. In this case, there is, of course, no value to be set, but rather the model can be evaluated with and without this sampling methodology.

5) *Depth*: indicates the maximum depth of each decision tree that makes up the forest. The set values are 3 and 5.

We, therefore, train 72 different types of architectures, both for the X-ray and CT datasets.

5.3.5 Logistic Regression

Finally, we describe the fine-tuning process concerning the logistic regression model. In particular, we denote 3 parameters:

1) *C*: we have already found it in SVM. It is the inverse of the regularization force and must be a positive float. As in support vector machines, smaller values specify stronger regularisation. In our case we analyse C = [1,5,10,20].

2) *solver*: Algorithm to use in the optimization problem. There are different types, such as 'newton-cg', 'lbfgs', 'liblinear', 'sag', and 'saga'. Our optimization problem uses the 2 most popular ones, namely "liblinear" and "saga".
3) *penalty*: Used to specify the norm used in the penalty. The solvers 'newton-cg', 'sag', and 'lbfgs' only support l2 penalties. If 'none' (not supported by the liblinear solver), no regularisation is applied. We set the 2 most popular types, namely l1 and l2.
In total, therefore, we tested 16 different types of structures for this linear model.

Section 6. Models and Results

6.1 Premise

We finally get to the heart of our research, that of the results. However, before describing all the results that came out, the best structures found for both datasets and all 4 types of models, and making our comparisons and conclusions, let us explain how we evaluated our models.
The metric used to select the best structure for each proposed model was the accuracy of the test set.
Once this was found, the best models were evaluated to have a more complete view taking into account other metrics.
In the case of convolutional neural networks, Validation_loss was also considered while for Support Vector Machine, Random Forest and Logistic Regression we evaluated Precision, Recall, F1 score, and Confusion Matrix.
Let us quickly illustrate these metrics, but first defining the concepts of True Positive, True Negative, False Positive, and False Negative.
1) *True Positive*: Image correctly classified as positive at COVID-19.
2) *True Negative*: Image correctly classified as non-COVID-19 positive.
3) *False Positive*: Image classified as COVID-19 positive but which in reality is not.
4) *False Negative*: Image classified as COVID-19 negative but is.

We can now define our metrics.

1) *Accuracy*: is the most intuitive performance measure and is simply a ratio of the correctly predicted observation to the total number of observations.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{All Samples}}$$

2) *Precision*: is the ratio of correctly predicted positive observations to the total number of predicted positive observations.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

3) *Recall*: is the ratio of correctly predicted positive observations to all observations in actual class

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

4) *F1-score*: is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

$$\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

5) *Confusion Matrix*: A confusion matrix is a summary of prediction results on a classification problem. It tracks, in a matrix form, the True positive, the False Positive, the True Negative, and the False-negative.

		Predicted Class	
		Normal	Attack
Actual Class	Normal	True Negative (TN)	False Positive (FP)
	Attack	False Negative (FN)	True Positive (TP)

The models were tested with a validation split varying between 0.2 and 0.3. Let us analyze the results obtained algorithm by algorithm.

6.2 Convolutional Neural Network

As we have explained above, the architecture of the best convolutional neural network differs from the others by a combination of 1) Layer size, 2) number of dense layers added, and 3) number of convolutional layers.

The results obtained were easily analyzed using Tensorflow [37]. This tool allows models to be monitored at any time in their life cycle employing clear and simple graphical representations.

The models were evaluated on both the train and test set results. This made it possible to assess the credibility of each model and the possible risk of overfitting. The best model, of course, is the one that performs best in the test set.

All models were fictitious at 15 epochs, as it was seen that beyond about the 12th epoch the accuracy remained constant, and using a batch size of 64 units, suitable for the size of the dataset.

After each convolutional layer, the chosen activation function is the "relu" function, while the "sigmoid" function follows the dense layers. The optimizer chose, in each fit, is the "adam" optimizer. A dropout layer is added before the multiconnected layer.

We distinguish the results by dataset.

6.2.1 X-Ray

The results obtained are shown below.

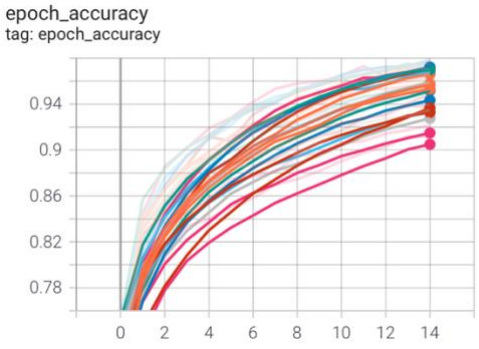

Figure 18. Models accuracy on train set

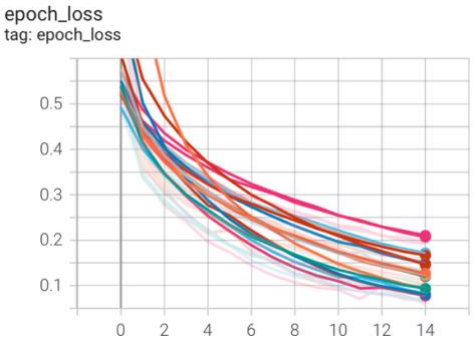

Figure 19. Models loss on train set

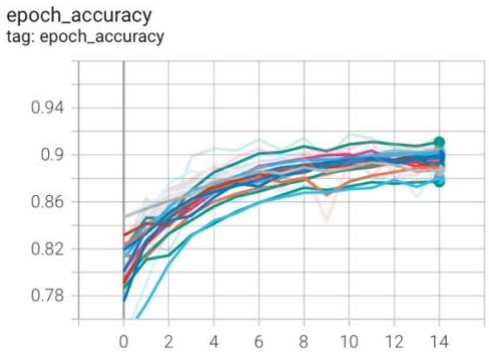

Figure 20. Models accuracy on test set

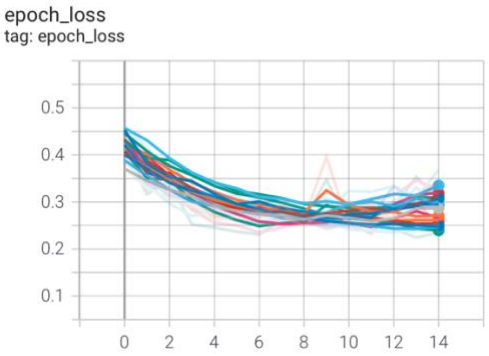

Figure 21. Models loss on test set

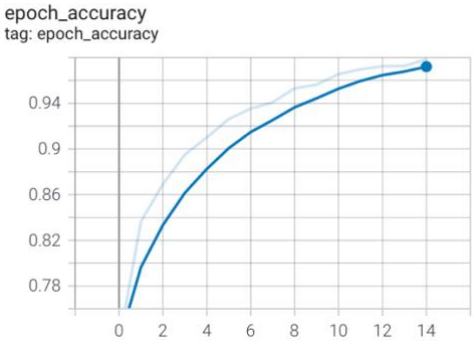

Figure 22. Best train-model Accuracy

Figure 23. Best train-model loss

As can be seen from the images, the best structure for the train set is characterized by the presence of an additional dense layer, a convolutional layer, and a layer size of 64. The accuracy on the train set is 95%, with a validation loss of less than 0.1. However, if we compare this model with the test set, we notice that the validation loss increases a lot (greater than 0.35) and the accuracy reaches relatively low levels, about 88%.

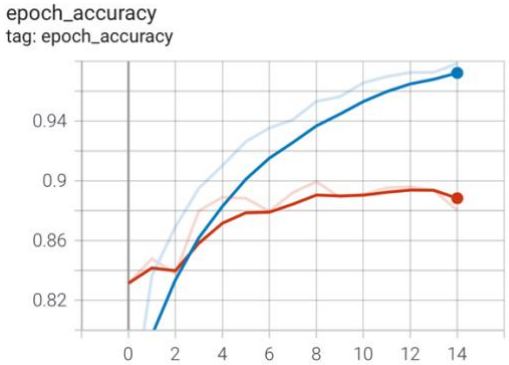

Figure 24. Best train-model accuracy comparison

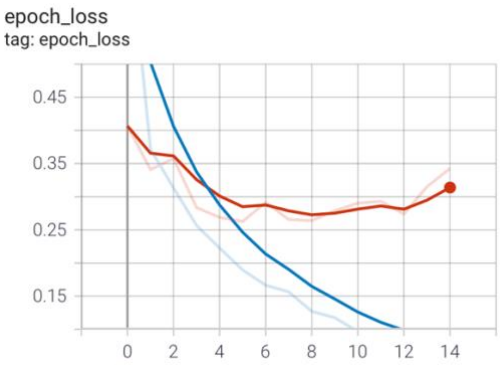

Figure 25. Best train-model loss comparison

Let us now analyze the best performator in the test set. The best structure in this regard consists of 3 convolutional layers, 1 added dense layer, and a layer size of 128. The complete structure is shown below.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 128)     1280
_________________________________________________________________
activation (Activation)      (None, 148, 148, 128)     0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 128)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 128)       147584
_________________________________________________________________
activation_1 (Activation)    (None, 72, 72, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 128)       0

conv2d_2 (Conv2D)            (None, 34, 34, 128)       147584
_________________________________________________________________
activation_2 (Activation)    (None, 34, 34, 128)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 36992)             0
_________________________________________________________________
dense (Dense)                (None, 128)               4735104
_________________________________________________________________
activation_3 (Activation)    (None, 128)               0
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
_________________________________________________________________
activation_4 (Activation)    (None, 1)                 0
=================================================================
Total params: 5,031,681
Trainable params: 5,031,681
Non-trainable params: 0
```

In this case, the accuracy obtained is 91%, with a validation loss of 0.3.

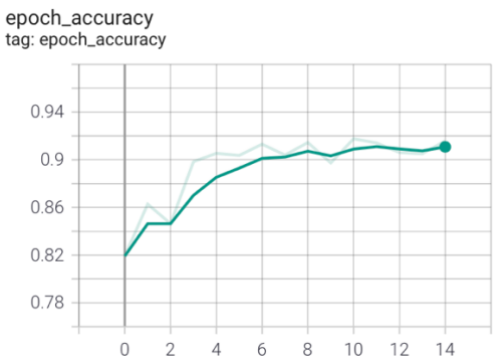

Figure 26. Best test-model accuracy

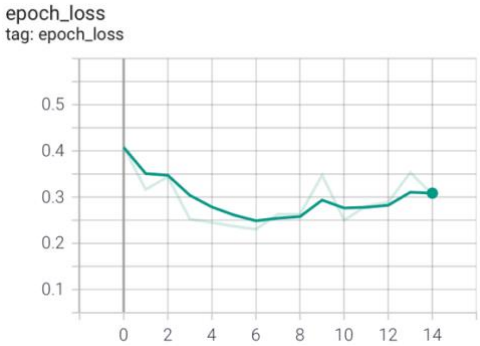

Figure 27. Best test-model loss

When compared to the respective result on the train set, the model gains a few points on both accuracy and validation loss but remains fairly consistent.

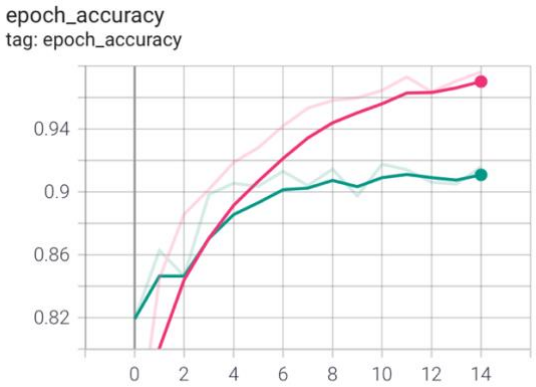

Figure 28. Best test-model accuracy comparison

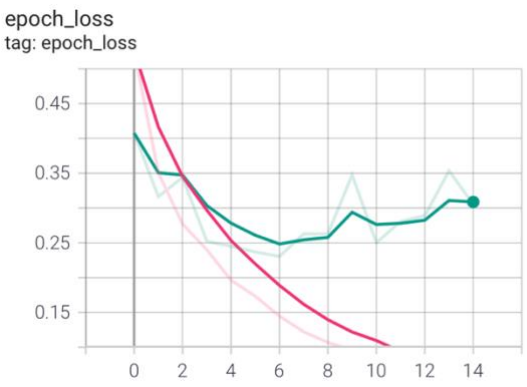

Figure 29. Best test-model loss comparison

6.2.2 CT

The results on the CT dataset are more promising. Let's see how the models performed on the train and test set.

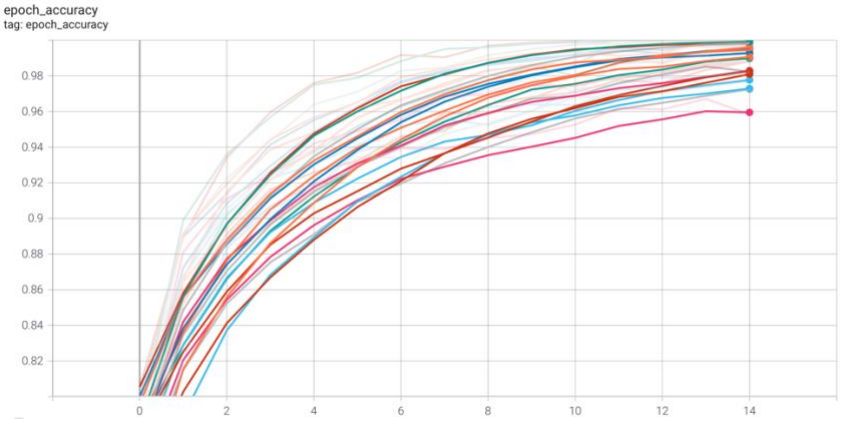

Figure 30. Models accuracy on train set

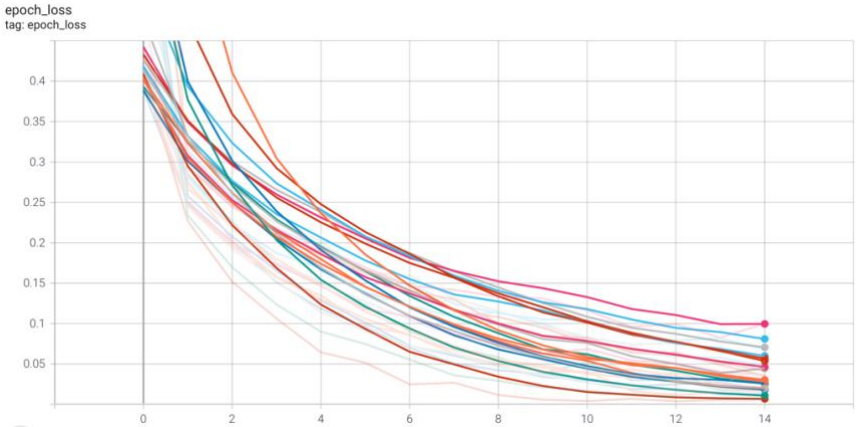

Figure 31. Models loss on train set

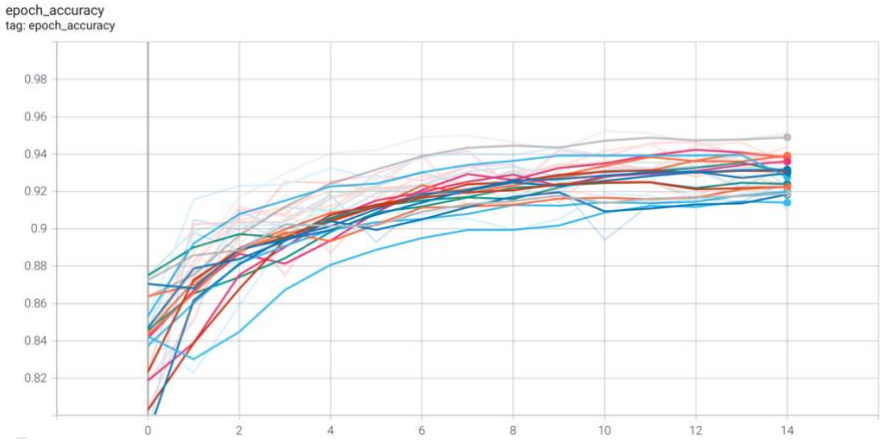

Figure 32. Models accuracy on test-set

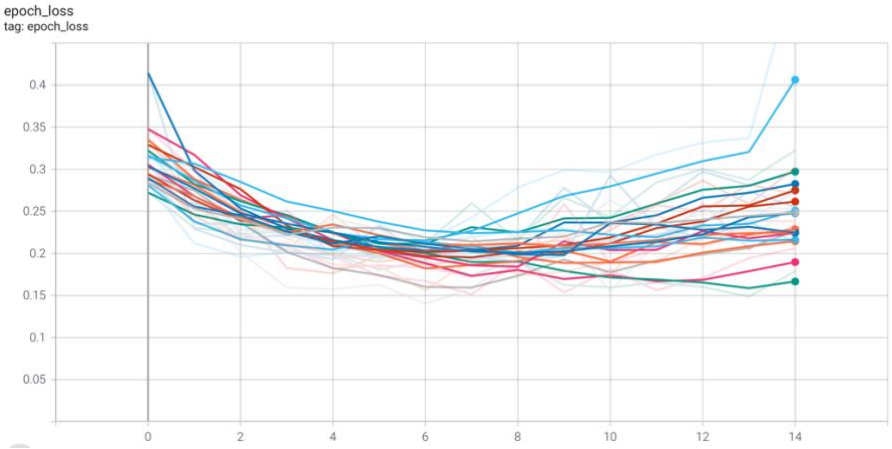

Figure 33. Models loss on test-set

The best model on the train set is composed of 1 convolutional layer, 0 additional dense layers, and a layer size of 128.

Reaching an accuracy close to 100% in the train set (with validation close to 0), the model is overfitting, since, if tested on the test set, it reaches an accuracy of about 92% and a loss close to 0.25.

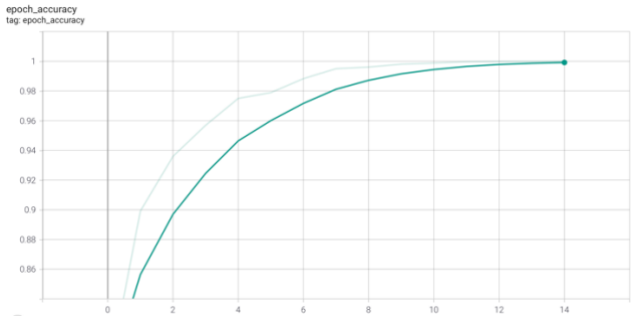

Figure 34. Best model in train set accuracy

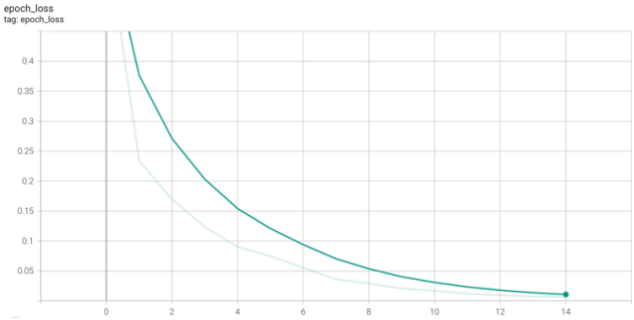

Figure 35. Best model in train set accuracy

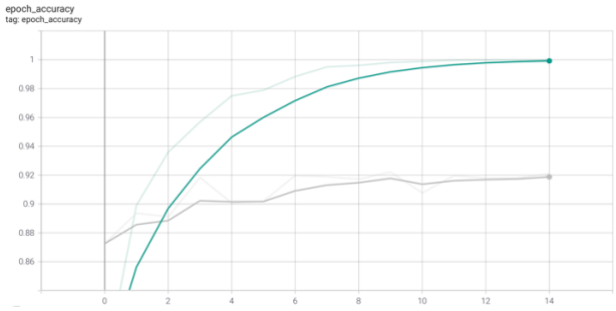

Figure 36. Best train model accuracy comparison

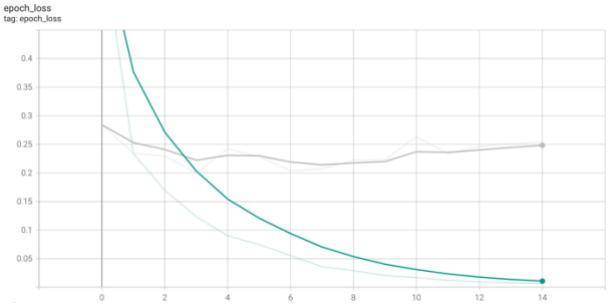

Figure 37. Best train model loss comparison

Very interesting results come from the test set.

The best structure, consisting of 3 convolutional layers, an additional dense layer, and a layer size of 64, achieves an accuracy of 95%, with a validation loss of about 0.22.

The whole structure is shown below.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 148, 148, 64)      640
_________________________________________________________________
activation_5 (Activation)    (None, 148, 148, 64)      0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 74, 74, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 72, 72, 64)        36928
_________________________________________________________________
activation_6 (Activation)    (None, 72, 72, 64)        0
```

```
max_pooling2d_4 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 34, 34, 64)        36928
_________________________________________________________________
activation_7 (Activation)    (None, 34, 34, 64)        0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 18496)             0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                1183808
_________________________________________________________________
activation_8 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65
_________________________________________________________________
activation_9 (Activation)    (None, 1)                 0
=================================================================
Total params: 1,258,369
Trainable params: 1,258,369
Non-trainable params: 0
```

Good results also on the "overfitting" side. This architecture, when tested on the train, increases its accuracy by only 3 percentage points, proving to be consistent and efficient.

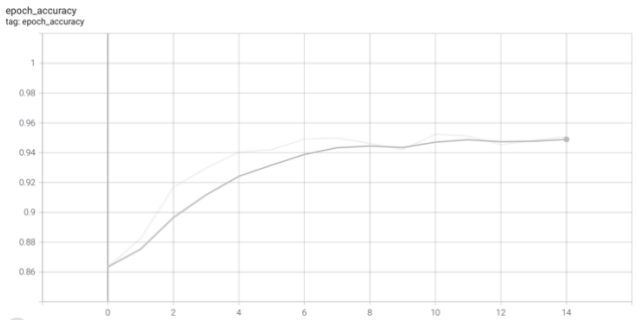

Figure 38. Best model on test set accuracy

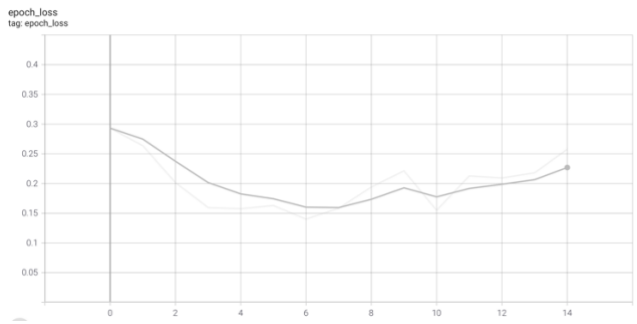

Figure 39. Best model on test set loss

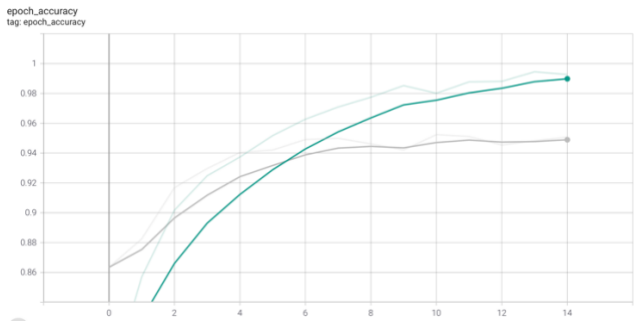

Figure 40. Best model on test set accuracy comparison

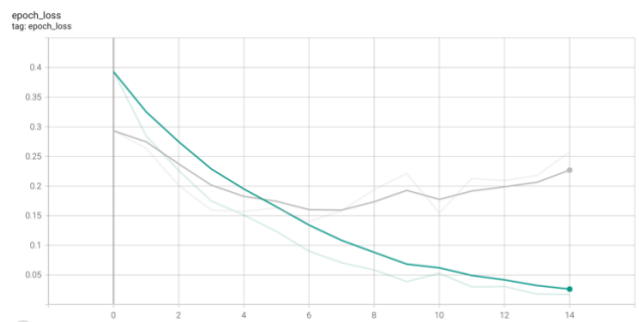

Figure 41. Best model on test set loss comparison

6.3 SVM

The Support Vector Machine, surprisingly, achieved the worst results. For this algorithm, and the next 2 (Random Forest and Logistic Regression) we analyzed the models that performed best only in the test set, as we have shown a strong propensity to overfitting in the train set, and it is, therefore, useless to consider a large number of structures of poor credibility.
In general, for the Support Vector Machine, we noticed that the linear kernel is more reliable than the others, as it is better at generalizing the problem.

6.3.1 X-ray

As far as the X-ray dataset is concerned, the parameter optimization showed a superiority of the structure composed of a linear kernel and C=10.
Let's see it in more detail through the image.

```
SVC(C=10, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Figure 42. SVM best structure

The results obtained in metrics are as follows.

METRICS	**TRAIN**	**TEST**
Accuracy	0.991	0.796
Precision	0.994	0.803
Recall	0.990	0.855
F1-Score	0.992	0.828

Confusion Matrix**:**

	TP	**FN**
Train	3208	23
Test	583	230
	FP	**TN**
Train	41	4363
Test	158	938

6.3.2 CT

On the other hand, for the CT dataset, the optimised parameters were C=1000 and linear kernel. Again, the CT dataset performed better than the X-ray dataset.

```
SVC(C=1000, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Figure 43. SVM best model structure

The results obtained are the following.

METRICS	**TRAIN**	**TEST**
Accuracy	1.0	0.864
Precision	1.0	0.795
Recall	1.0	0.801
F1-Score	1.0	0.798

As we stated above, the SVM has showed everything but consistency.

Let’s see the Confusion matrices for both train and test set.

	TP	**FN**
Train	3217	16
Test	672	139
	FP	**TN**
Train	38	4364
Test	115	983

6.4 Random Forest

Very good results were obtained by the Random Forest decision tree algorithm.
As we have seen in the Literature Review section, neural networks are legitimately preferred in this type of problem. However, we have shown that Random Forest can still be an excellent candidate, again in the CT dataset analysis.
Let's see the core.

6.4.1 X-Ray

The parameters optimized in this phase were more numerous. The best model in the X-ray dataset was found to have a minimum samples leaf = 1, minimum samples split = 2, number of estimators = 300, and the presence of Bootstrap. The full structure is shown below.

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=300,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

Figure 44. Random Forest best model structure

Regarding the metrics, we formulate a table as we did for the SVM and we picked the Confusion Matrix for both the train and test set.

METRICS	TRAIN	TEST
Accuracy	0.992	0.866
Precision	0.996	0.876
Recall	0.991	0.895
F1-Score	0.993	0.885

Confusion Matrix:

	TP	**FN**
Train	3217	16
Test	672	139
	FP	**TN**
Train	38	4364
Test	115	983

6.4.2 CT

The best results were obtained by training the model on CT images.

In particular, the optimal structure has the following characteristics: min samples leaf = 2, min samples split = 5, no Bootstrap. The complete structure is shown below.

```
RandomForestClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=2, min_samples_split=5,
                       min_weight_fraction_leaf=0.0, n_estimators=400,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

Figure 45. Random forest best model structure

As far as the results are concerned, we show excellent results in the test-set, almost on a par with convolutional neural networks.

METRICS	TRAIN	TEST
Accuracy	0.999	0.947

Precision	1.0	0.916
Recall	0.999	0.930
F1-Score	0.999	0.923

Confusion Matrix:

	TP	**FN**
Train	4360	0
Test	1021	46
	FP	**TN**
Train	1	2083
Test	38	506

6.5 Logistic Regression

The section on our results concludes with logistic regression. Not very popular in the field of image classification, this algorithm proved to be a compromise between computational difficulty and accuracy. However, it ranks as less powerful than both Convolutional Networks and Random Forest.

6.5.1 X-ray

The best structure obtained for X-ray image classification is characterized by the presence of liblinear solver, C=5, and penalty = "l2".
Let's see it in detail, and the results obtained.

```
LogisticRegression(C=5, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

Figure 46. Logistic Regression best model structure

METRICS	TRAIN	TEST
Accuracy	0.991	0.805
Precision	0.995	0.811
Recall	0.990	0.871
F1-Score	0.992	0.840

Confusion Matrix:

	TP	**FN**
Train	3236	21
Test	560	227
	FP	**TN**
Train	144	4337
Test	38	978

6.5.2 CT

Like all the algorithms used, the CT dataset proved to be already prone to classification in terms of accuracy. However, in this case, it lacks Precision, Recall, and F1 score.

The best structure found is different from that of X-ray in that C=1, but remains constant in solver = liblinear and penalty =l2.

```
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

Figure 47. Logistic Regression best model structure CT

METRICS	TRAIN	TEST
Accuracy	0.999	0.857
Precision	1.0	0.768
Recall	0.999	0.787
F1-Score	0.999	0.777

Confusion Matrix:

	TP	**FN**
Train	4325	0
Test	981	121
	FP	**TN**
Train	1	2118
Test	108	401

Section 7. Results analysis, conclusions and potential developments

This article presented a critical analysis of several different architectures of 4 of the most important types of classification algorithms, originally proposed for natural image analysis, to help radiologists discriminate COVID-19 disease based on chest X-ray or CT scan images. After all, the analyses are done and the results obtained, there are some considerations to be made.

The most important thing this research has demonstrated is the superiority of CT-scan imaging over X-ray imaging in prediction accuracy. This demonstration is, in our opinion, a big step towards a potential development not only on COVID-19 disease but on the whole research linking the world of artificial intelligence to the world of medicine. As we have seen in the literature review section, all the studies that have been done have always used X-ray images. I, therefore, believe that the replacement, or at least the accompaniment, of CT analysis, can be established as an important research tool.

The results obtained have demonstrated, as expected, greater reliability of the Convolutional Neural Networks in respect of the other types of algorithms. This model not only outperforms the others in terms of accuracy, but also of consistency and resistance to the overfitting problem. We have seen that the problem faced certainly requires a high complexity of mathematical analysis, and the models, even if optimized, of Support Vector Machine, Logistic Regression, and the majority of the Random Forest have shown to be somewhat weak

and not robust in the prediction. However, we highlight great results from some Random Forest structures on the CT dataset, which almost reach those obtained by the best CNN architectures.

After all, the best results are characterized by accuracy of 95% on the test set, and 99% on the train set. Recall that the datasets used are well balanced, and we can therefore call our results very credible and robust. COVID-19 respiratory disease has brought the entire world population to its knees, socially, economically and in terms of health. A tool that can identify the presence of this virus in seconds with an accuracy close to that of current tools can be a vital weapon in the war we are fighting. Let's bear in mind that, in the world we live in today, thinking of replacing a doctor with an artificial intelligence model is still a distant idea, but soon such tools may play a more important role than just support.

A few points need to be touched upon regarding potential future developments and improvements in research.

The efficiency of artificial intelligence models is closely linked to the amount of data on which they are trained. Analyses in this regard have been underway for just over a year, and working on two datasets of around 10,000 images each is already a good result. However, the more time passes, the more images will be available, and it will be possible to think of achieving an accuracy close to 100%. Furthermore, images derived from practices such as data augmentation (section 4) on the one hand increase the availability of images and improve the model, but they are certainly not as powerful in generalization as real images are.

Other interesting studies can be carried out not only on the identification of the presence of the virus but precisely on the identification of the affected areas. We have seen some studies, such as that carried out by Taban Majeed et al [39], which are very powerful and useful for research into vaccines and treatments.

Finally, we attach further importance to the hyper tuning of models. This paper describes a good optimization practice, but we can say that it is never "enough". With more computational power, and time, we could have tested many additional types of architectures, perhaps with different optimizers and different batch sizes (in the case of CNN).

Working with artificial intelligence models is never a finished job. The developments that such technologies can have in the medical field, but not only, are endless. Looking for more and more efficient solutions is the basis of this subject, and talking about future developments as a topic with a certain end date is vain and wrong.

COVID-19 Identification software

Creating models capable of simulating and emulating human intelligence by limiting the work to writing code and graphical representations of results suppresses the potential of these tools. If research into artificial intelligence solutions is important, so is its widespread sharing. The deployment phase, the phase in which anyone, and not just a programmer or a data scientist, can use the program, is probably the phase that makes the world of artificial intelligence one of the most appreciated and fascinating fields worldwide.
All of our work has been aimed at finding a faster and cheaper alternative methodology to the tests used for the detection of COVID-19, and it was therefore decided to create a user-friendly platform, within the reach of doctors and healthcare professionals.

This platform allows the user to upload an image in .png or .jpeg format of a CT-scan of the lung, and detects, in a few seconds, whether or not it is positive for the virus.
In terms of functionality, it is very simple and intuitive**:**

1) The user selects an image from his PC, with a maximum limit of 200 MB per image.
2) Once the image has been selected, it is displayed on the platform.
3) At this point, all the user has to do is click on the classification button in the sidebar, wait a few seconds and receive the result.

The model behind which the platform works is the best model obtained from our research. In particular, we are talking about the Convolutional Neural Network model applied to CT-Scan images described in section 6.2.2. At the level of background functionality, the functionalities are as follows:

- If no image is selected, simply wait.
- If the image is uploaded, check that it is of a suitable type (png, jpeg).
- Image *preprocessing*: retrieve image pixels by matrix, convert to GrayScale, resize to 150x150, reconstruct the array in 4D.

This is done to make the image consonant with the operation of the trained model.

- *Prediction* of the obtained array. A value between 0 and 1 is obtained. If the value is <0.5, the image is classified as Coronavirus positive, otherwise, it is classified as healthy.

  Below are some representative images of the platform.

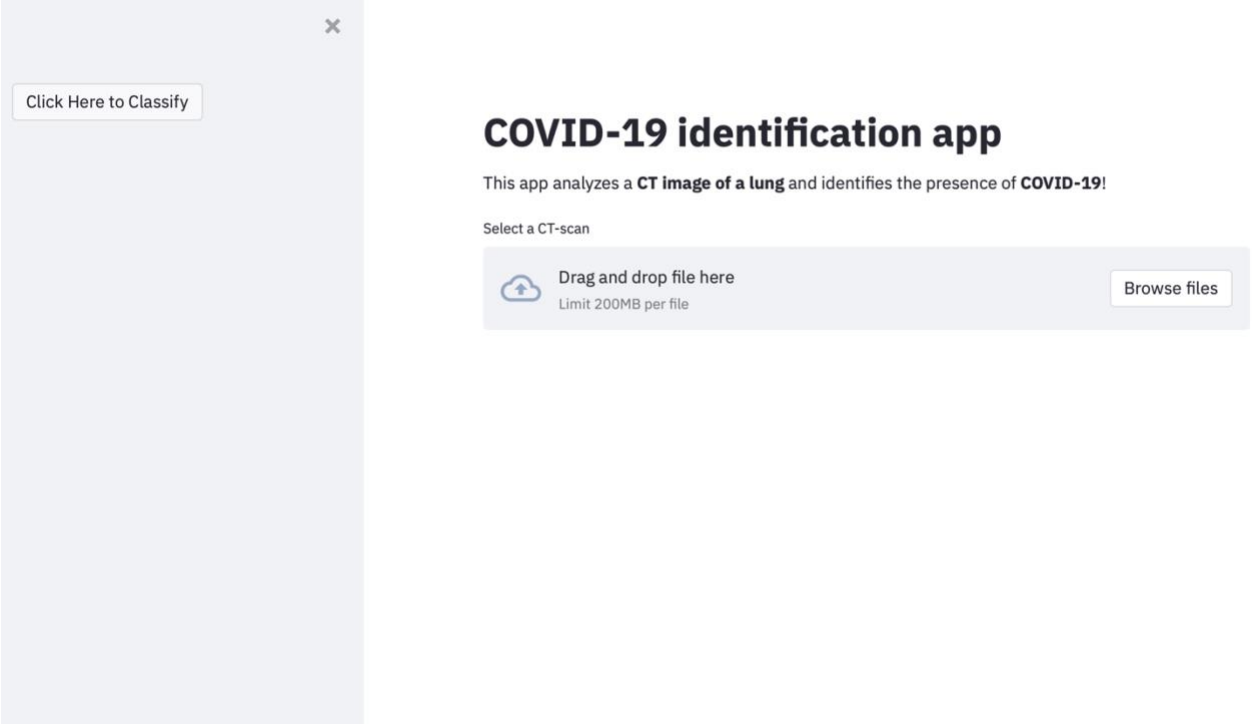

Figure 48. WebApp Home

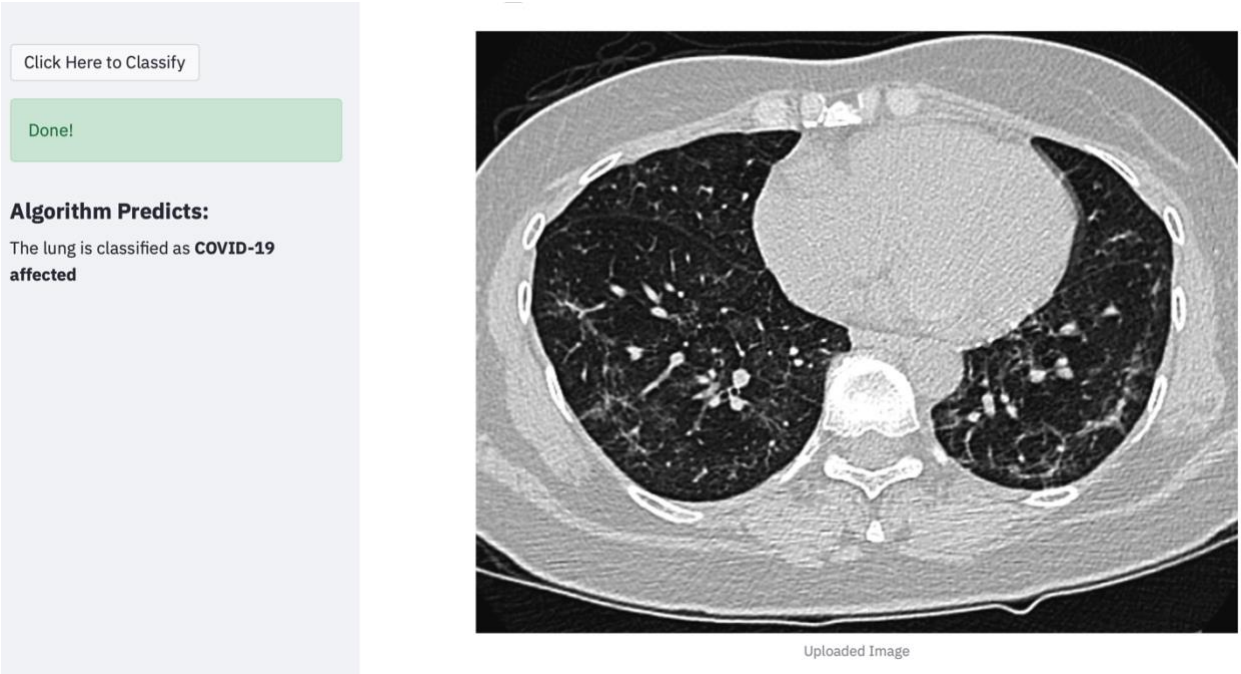

Figure 49. COVID-19 CT-scan prediction

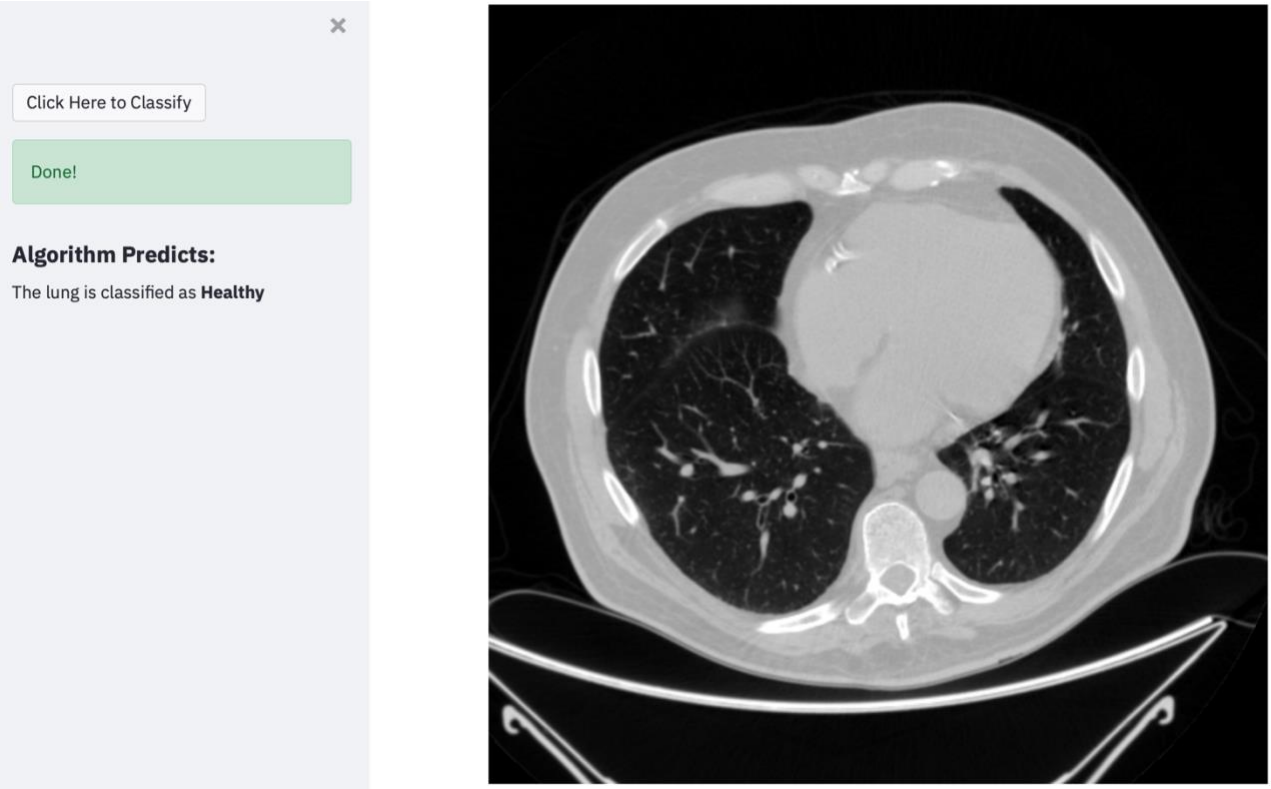

Figure 50. Healthy CT-scan prediction

The platform was built using the streamlit library [39]. The pre-processing phase was made possible through the libraries PIL(for obtaining the pixels and converting them into grayscale) [40] and Numpy(for converting them into arrays). The prediction model, previously trained, was saved and connected via tensorflow [41].

The platform is **deployed** at :
https://share.streamlit.io/giorgiobientinesi/covid-19_webapp/main/Webapp-COVID-19.py

And the entire **source code** is available at**:**
https://github.com/Giorgiobientinesi/COVID-19_WebApp
https://github.com/Giorgiobientinesi/CT-and-X-ray-COVID-image-classification

References

1) https://covid19.who.int

2) Majeed, T., Rashid, R., Ali, D., & Asaad, A. (2020). Covid-19 detection using cnn transfer learning from x-ray images. *medRxiv*.
3) Hassantabar, S., Ahmadi, M., & Sharifi, A. (2020). Diagnosis and detection of infected tissue of COVID-19 patients based on lung X-ray image using convolutional neural network approaches. *Chaos, Solitons & Fractals*, *140*, 110170.
4) Cai, L., Gao, J., & Zhao, D. (2020). A review of the application of deep learning in medical image classification and segmentation. *Annals of translational medicine*, *8*(11).
5) https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

6) Sarraf, S., Tofighi, G., & Alzheimer's Disease Neuroimaging Initiative. (2016). DeepAD: Alzheimer's disease classification via deep convolutional neural networks using MRI and fMRI. *BioRxiv*, 070441.
7) Sharma, H., Zerbe, N., Klempert, I., Hellwich, O., & Hufnagl, P. (2017). Deep convolutional neural networks for automatic classification of gastric carcinoma using whole slide images in digital histopathology. *Computerized Medical Imaging and Graphics*, *61*, 2-13.
8) Majeed, T., Rashid, R., Ali, D., & Asaad, A. (2020). Covid-19 detection using cnn transfer learning from x-ray images. *medRxiv*.
9) Hassantabar, S., Ahmadi, M., & Sharifi, A. (2020). Diagnosis and detection of infected tissue of COVID-19 patients based on lung X-ray image using convolutional neural network approaches. *Chaos, Solitons & Fractals*, *140*, 110170.
10) Zhang, J., Xie, Y., Li, Y., Shen, C., & Xia, Y. (2020). Covid-19 screening on chest x-ray images using deep learning based anomaly detection. *arXiv preprint arXiv:2003.12338*, *27*.
11) Maghdid, H. S., Asaad, A. T., Ghafoor, K. Z., Sadiq, A. S., Mirjalili, S., & Khan, M. K. (2021, April). Diagnosing COVID-19 pneumonia from X-ray and CT images using deep learning and transfer learning algorithms. In *Multimodal Image Exploitation and Learning 2021* (Vol. 11734, p. 117340E). International Society for Optics and Photonics.
12) Sethy, P. K., Behera, S. K., Ratha, P. K., & Biswas, P. (2020). Detection of coronavirus disease (COVID-19) based on deep features and support vector machine.
13) Islam, M. Z., Islam, M. M., & Asraf, A. (2020). A combined deep CNN-LSTM network for the detection of novel coronavirus (COVID-19) using X-ray images. *Informatics in medicine unlocked*, *20*, 100412.
14) Heidari, M., Mirniaharikandehei, S., Khuzani, A. Z., Danala, G., Qiu, Y., & Zheng, B. (2020). Improving the performance of CNN to predict the likelihood of COVID-19 using chest X-ray images with preprocessing algorithms. *International journal of medical informatics*, *144*, 104284.
15) Jakkula, V. (2006). Tutorial on support vector machine (svm). *School of EECS, Washington State University*, *37*.
16) https://it.wikipedia.org/wiki/Yann_LeCun

17) https://en.wikipedia.org/wiki/LeNet

18) https://www.researchgate.net/figure/The-convolution-operator-in-a-CNN

19) Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., ... & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of big Data*, *8*(1), 1-74.
20) https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17

21) https://towardsdatascience.com/activation-functions-in-neural-networks-83ff7f46a6bd).

22) https://heartbeat.fritz.ai/classification-with-tensorflow-and-dense-neural-networks-8299327a818a
23) https://www.mdpi.com/2076-3417/10/4/1245/html
24) https://www.tensorflow.org/tutorials/images/data_augmentation
25) https://towardsdatascience.com/a-comparison-of-dnn-cnn-and-lstm-using-tf-keras-2191f8c77bbe
26) https://www.researchgate.net/figure/SVM-for-a-linearly-separable-binary-classification-problem_fig1_6992544).
27) https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f).
28) http://wgrass.media.osaka-cu.ac.jp/gisideas10/viewpaper.php?id=342)
29) James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112, p. 18). New York: springer.
30) https://towardsdatascience.com/logistic-regression-explained-9ee73cede081)
31) El-Shafai, Walid; Abd El-Samie, Fathi (2020), "Extensive COVID-19 X-Ray and CT Chest Images Dataset", Mendeley Data, V3, DOI: 10.17632/8h65ywd2jr.3
32) https://opencv.org
33) https://numpy.org
34) https://cloud.google.com/ai-platform/training/docs/hyperparameter-tuning-overview
35) https://towardsdatascience.com/hyperparameter-tuning-c5619e7e6624
36) https://www.mydatamodels.com/how-good-is-your-machine-learning-algorithm
37) https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/Callback
38) Majeed, T., Rashid, R., Ali, D., & Asaad, A. (2020). Covid-19 detection using cnn transfer learning from x-ray images. *medRxiv*.
39) https://streamlit.io
40) https://pillow.readthedocs.io/en/stable
41) http://tensorflow.org