1 2 90

UNIVERSIDADE D
COIMBRA

Ainhoa Sagardoy Zaro

COMPARISON OF MACHINE LEARNING ALGORITHMS IN ICON RECOGNITION

Dissertação orientada pelo/a Professor Nuno Miguel Mendoça da Silva Goçalves e pelo Doutor Leandro Moraes Valle Crus e apresentada ao Departamento de Engenharia Electrotécnica e Computadores da Faculdade de Ciência e Tecnologia da Universidade de Coimbra, no âmbito do programa ERASMUS.

Julho de 2019

ABSTRACT

The main aim of this project consists in the study and comparison of Machine Learning algorithms using them for icon recognition in Graphic Code.

Graphic Code is a new Machine Readable Code recently developed by the VisTeam (Institue of Systems and Robotics) that combines the capacity of sending a big amount of information with good aesthetic, improving the capacity when compared with QRcodes, for instance.

The methods used for the comparison are the Support Vector Machine and the Convolutional Neural Network. These two algorithms have been chosen because of their ability to recognise different figures and separate them correctly.

KEYWORDS

Machine Learning, Support Vector Machines, Convolutional Neural Networks, Graphic Code, icons.

RESUMO

O objetivo principal deste projeto consiste no estudo e comparação de algoritmos de Machine Learning utilizando-os para reconhecimento de ícones em aplicações com códigos de leitura por máquina (machine readable codes).

O Graphic Code é um novo código de leitura por máquina recentemente desenvolvido pelo VisTeam (Instituto de Sistemas e Robótica) que combina a capacidade de enviar uma grande quantidade de informação com uma boa estética, melhorando as capacidades de códigos com os QRcodes, por exemplo.

Os métodos que serão comparados são o Support Vector Machine e o Convolutional Neural Network. Esses dois algoritmos foram escolhidos por causa de sua capacidade de reconhecer diferentes figuras e separá-las corretamente.

PALABRAS-CHAVE

Machine Learning, SVM, CNN, Graphic Code, ícones.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The topic chosen for this project consists in the study and application of machine learning techniques having the aim of developing a model for icon automatic recognition in Graphic Code.

Machine Readable Codes (MRC) have supposed a very useful tool for sending information. The most used are barcodes, normally implemented to products recognition, and QR codes which allow to send more amount of information.

Despite the QR codes have solved the problem of the amount of information that can be sent, it is still necessary to implement codes with a good appearance.

Graphic Code is a Machine Readable Code originated due to the necessity of sending information combining the possibility of transmitting it with good aesthetic.
The idea of this new information sender technique is being developed by “VIS Team” at University of Coimbra collaborating with “Imprensa Nacional-Casa da Moeda” (INCM) company.

TrustStamp project intends to develop verification tools to be applied on INCM trust stamps, to confirm authenticity and protect products against counterfeiting. The verification of the authenticity of the trust stamp will be done using a mobile application (smartphones and tablets), due to its high ubiquity and personal ease of use.

In this project, the use of Graphic Code was extended in different fields such as Augmented Reality or products recognition. For the second one, they have been created specific codes for tobacco and wine.

There are three different styles which Graphic Code can be made of: black and white pixels, color pixels and icons. The study that is going to be realised in this project is about Graphic Code made with icons using Machine Learning techniques to be able to recognise and differentiate several of them.

Nowadays Artificial Intelligence (AI) is being very used due to the fact that it allows machines to manage a lot of information in a very effective way.

Machine Learning (ML) is a field of Artificial Intelligence which creates algorithms with which the machines can learn certain behaviours to later implement them automatically.

1.1. Contribution and motivation

There are many ways to implement figure recognition. In this project it is going to be studied one of these techniques: Machine Learning algorithms. This choice was taken because the Artificial Intelligence is growing up very fast and a lot of companies are implementing it in many ways. Besides, it is possible to reach very good results using different parameters with these algorithms.

The study that is going to be developed in this project contributes for the comparison between methods in the recognition of icons with the aim of being able to implement it in the future in the Graphic Code coding process.

Personally, when this idea was proposed I found it very interesting for the fact of being able to perform a study in a field that nowadays is being implemented in many fields. Besides, I think that working with a company in a project of real application is a good opportunity to learn and improve my skills.

2. STATE OF THE ART

2.1. Machine Readable Codes

The use of Machine Readable Codes (MRC) has become essential nowadays due to the fact that with them it is possible to represent the information by an image and then recover it with simple and fast methods.

Over the years, different types of MRC have appeared implementing new improvements.

The first popular MRC was the 1D barcode such a Universal Product Code (UPC). It is characterised by its capacity of sending short and basic information and it is mainly used for products identification. These codes had a very important success because of their ability to read the information really fast and in an easy way.

Later it was implemented the 2D barcode, being the Quick Response Code (QR) the most known and used. The improvement of QR was the capacity to transmit more information than the 1D barcodes. However, one of the biggest disadvantages of QR codes is the appearance. They seem like a group of pixels represented in a random and disorganised way and due to the fact that they have a very strict regular structure, it makes the opportunity to create more aesthetic codes impossible.

To solve the appearance problem there are other codes, known as Picture-Embedded QR Codes (PEQRC) that embed a picture in the code whereas they keep decoding capability.

The problem is that, on these methods, they are used only very few pixels to encode the data so the amount of coded information decreases enormously.

Graphic Code has appeared to improve the last mentioned codes because it combines the capacity of transmitting a lot of information guaranteeing a good appearance. [1]

2.1.1. Universal Product Code

Universal Product Code is considered the informatic tool to capture information automatically and systematized. It is composed of a group of parallel black lines and white spaces with variable thickness that contain different kind of information. [2]

These codes are recognised by special devices which are called sensors and that are capable to obtain the information. After it, this information is decoded, verified, compared and accepted by a data base to make a logic decision. This code is normally used to identify products but it can be also useful in different fields as sales analysis, production controls, logistics etc. [3]

8 414533 043847

Figure 1. Universal Product Code

2.1.2. Quick Response codes

The main improvement of the 2D codes is that the information can be coded vertically and horizontally so it is possible to code much more information than in 1D codes. Therefore, it is possible to use these codes in many different fields of the society and send varied information such as images, documents or web sites links. Besides, QR codes can be also decoded even if there is some damage in the code and in all directions. [4] [5]

Even though the QR codes are very good to code the information, they do not have a good appearance. To improve this problem they have been implemented some solutions as Picture-Embedded QR Codes, that represent the pixels like an image and not randomly.

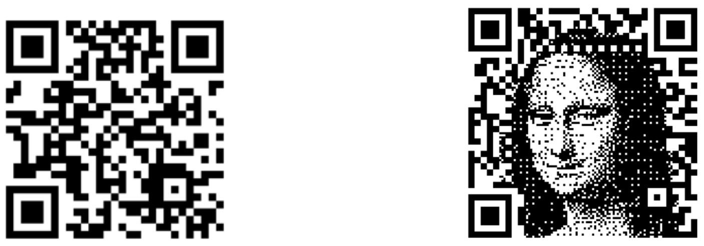


Figure 2. (a) QR Code (b) PEQRC

The problem of these new codes is that they use very few pixels to code the image so it is not possible to send a lot of information. [1]

To solve and improve all the advantages and disadvantages of the mentioned Machine Readable Codes, “Vis Team” has proposed a new codification method: Graphic Code.

2.1.3. Graphic Code

Graphic Code is a new steganography technique that combines the image visual appeal with the communication power of words. It means that this method is able to create a MRC that guarantees aesthetics while increasing the amount of coded data. This technique consists of an appropriate primitive graphic units distribution throughout a certain image. It will be assumed that these primitives are black and white pixels. Such primitives, the graphic units, are distributed throughout the code into clusters called cells. Some cell paterns are previously associated with characters, forming what is called a dictionary, the primary element of encoding a decoding.

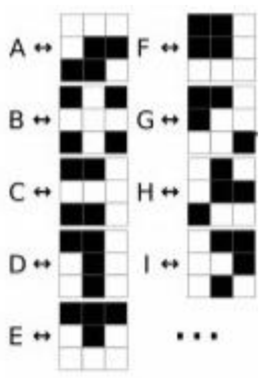


Figure 3. Dictionary

The encoding and decoding processes are symetrical and both receive as input the same dictionary and grid of pixels, defining the code resolution and the area of the image that will be used. Moreover, the encoding receives message to be encoded and, when necessary, a base image (Figures 4 and 5). [6]

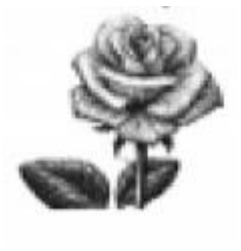


Figure 4. Base Image

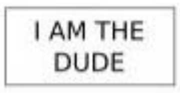
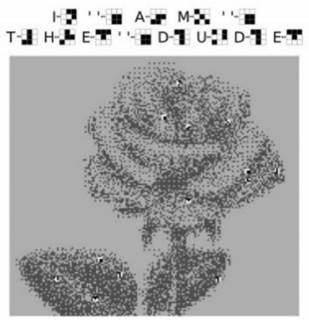
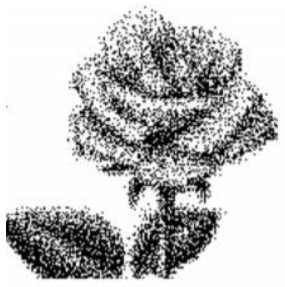

I AM THE
DUDE

Figure 5. Sent message

Furthermore the decoding also receives the graphic code itself and returns the recovered message.



(a)



(b)

Figure 6. (a) Coded message, (b) Coded image

These two phases (coding and reconstruction) are going to be explained in depth in Chapter 3.

2.2. Machine Learning Methods

In the last years, the use of Machine Learning methods is becoming more popular for image recognition due to the fact that they are able to learn complex patterns and differentiate images with a lot of details. That is why it has been decided to make a deep study about the application of these methods to recognise some icons that in the future could form the Graphic Code. [7]

2.2.1. Machine Learning

Machine Learning (ML) is a scientific discipline in Artificial Intelligence field that provides to computers the ability to perform a task without being explicitly programmed. It is focus on developing algorithms that work correctly without the necessity of defining the data that is going to be used. [8]

Typically, it can be said that ML consists in developing an algorithm prediction to solve a specific problem. Algorithms can learn from the data with the purpose of finding patterns between them and after it form a model able to predict and classify the elements. This model is the way in which it is possible to connect the entry data with the related labels. There are two phases in every ML model: training and predicting. The training phase is where the model is created. This model is going to try obtain a relation between the training data and the labels. In the second phase the model is used to make predictions with data that is not labeled. [7]

In mathematical terms, the expression of the model could be:

$$y = wx + b$$

Where:

- y : Labels
- x : Entry example
- w : Slope of the line that is called *weight*. It is one of the parameters that the model has to learn during the training process
- b : It is called *bias* and it is the other parameter that the model has to learn.

The ideal values for weights and bias are learned in the training phase. There are different ways to optimize the predictions based on the received data. The algorithms can be clasiffied generally in three big groups [9] :

- **Supervised learning** : The models are created by a set of data that contains the desired output, the label.
- **Unsupervised learning** : The data that is used by this models does not include the labels so the algorithm has to classify the information by themselves

- **Reinforcement Learning** : In this case the models learn by themselves thanks to the rewards and penalties that obtain with their acctions.

In this project it is going to be studied deeply the supervised learning algorithms.

It was mentioned before that the aim of the training phase is to get the ideal values for the weights and bias. With the supervised learning, this parameters are obtained using a lot of labeled examples and deciding values to minimise the "*loss"*. The *"loss"* is a value that indicates the error probability obtained in a model based on the number of images that have not been correctly identified during the training phase. It is necessary to adjust all the parameters in this phase to reduce the loss as much as possible.[10]

It is also important to mention the "*overfitting"* concept that happened when the model is adjusted too much to the training data and it is not able to make correct predictions with new examples. [7]

2.2.2. SVM and Neural Networks

In this project it is going to be studied two different supervised algorithms normally used to identify images.

The first one is the Support Vector Machine (SVM) algorithm and the second one the Artificial Neural Network algorithm, in particular the Convolutional Neural Network (CNN) one.

They have been choosen because although they are both efficient in image recognition, the development of each one is different. SVM algorithm has very good properties to separate few different icons with a hyperplane. However, CNN algorithm can differentiate more icons comparing many given images.

2.2.2.1. Support Vector Machine

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. It means that, given labeled training data, the algorithm outputs an optimal hyperplane that separate the data in different classes. [11]

To obtain a good result with this method it is important to combine some terms trying to get the best possible accuracy.

Kernel

The simplest option to divide the data in classes is with a straight line but, unfortunately, it is hardly ever possible to found real data able to be separated that easy.

This is why it is needed to use non-linear applications, called Kernel.

The most known are [12] [13]:

- Polynomial: The kernel induces space of polynomial combinations of the features, up to certain degree.

$$k(x_i, x_j) = (x_i \cdot x_j + 1)^d$$

  Where d is de degree of the polynomial.

- Radial Basis Function (RBF): In this case the induced space is a space of Gaussian distributions where each point becomes probability density function of a normal distribution. This kernel has extremely flexibility but it is important to be careful with the overfitting.

$$k(x_i, x_j) = \exp(-\frac{(x_i - x_j)^2}{2\sigma^2})$$

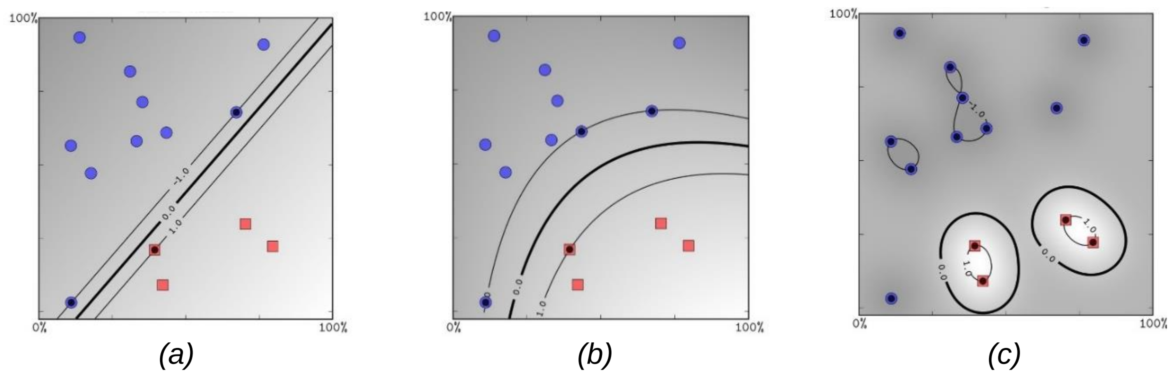


Figure 7. (a) Linear kernel, (b) Polynomial kernel d=2, (c) Gaussian kernel sigma=0.5

Margin and Regularization

A margin is a separation of line to the closest class points

There exist innumerable hyperplanes that solve the classification but the best one is the hyperplane that allow a maximum margin between the elements of each category.

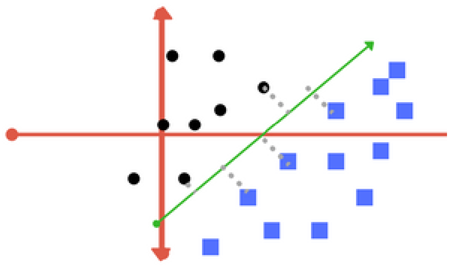


Figure 8. Good margin

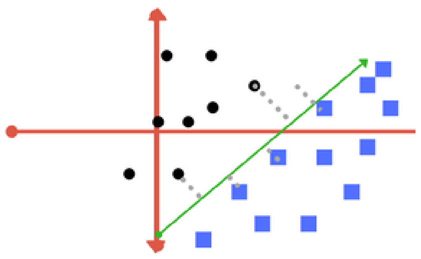


Figure 9. Bad Margin

The regularization parameter tells the SVM optimization how much it is going to avoid misclassifying each training example. [11]

For large values of this parameter, 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 regularization will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. [11]

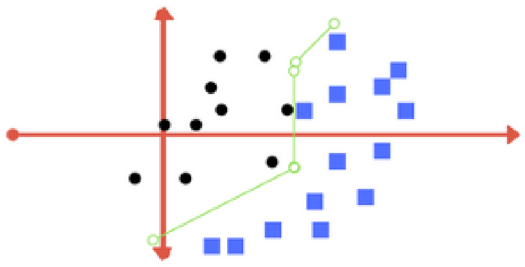


Figure 10. High Regularization

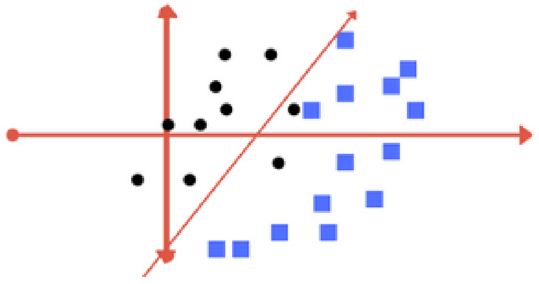


Figure 11. Low Regularization

Gamma

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning “far” and high values meaning “close”.

In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line.

High gamma means that the points close to the plausible line are going to be considered in calculation. [11]

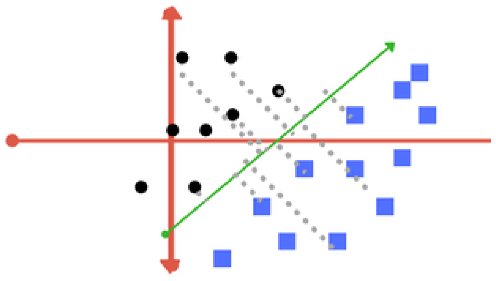


Figure 12. Low gamma

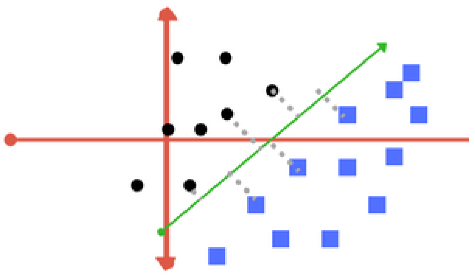


Figure 13. High gamma

2.2.2.2. Artificial Neural Networks

The Artificial Neural Networks base their structure and performance in the human neurons and their capacity to obtain results.

These models are composed by multiple processing layers to learn data representation with different abstract levels that make linear and non-linear transformations that depend on the input data generate an output similar to the expected one (label).

The learning consists in obtaining the parameters (weights and bias) of these transformations and try to obtain an output similar to the expected output. [7]

Figure 14 represents a graphic approach of a Deep Learning Neural Network.

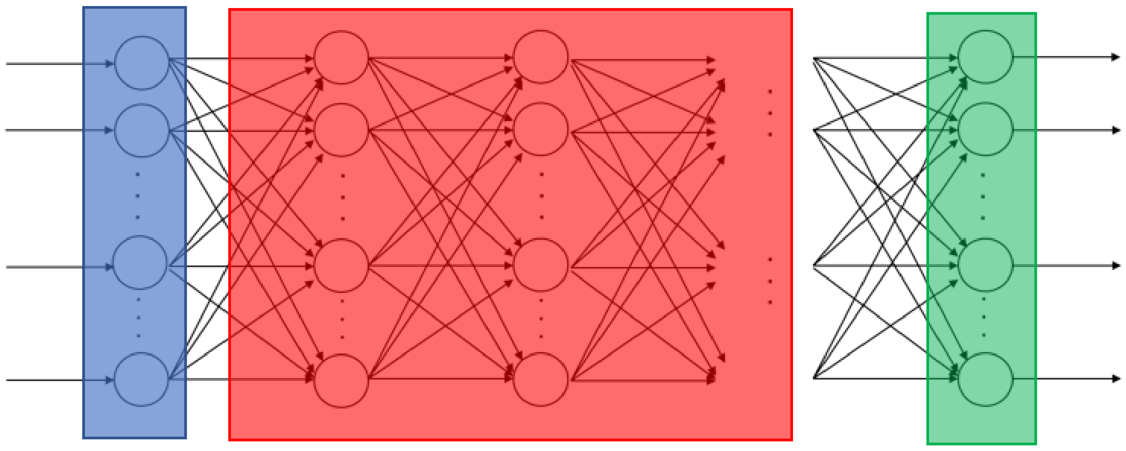


Figure 14. Neural Network

Where the blue part is the input layer that receives the input data, the green one is the output layer that returns the made prediction and the red part is called the hidden layers and it can be formed by several layers with different number of neurons.

As we will see later, neurons of different hidden layers, represented by the circles, are interconnected between them in different ways.

Nowadays, are being used neural networks with many layers each of them with different number of neurons. These neurons make a simple transformation of the data that they receive of the previous layer and pass it to the next one. The joining together of all of them allow to find complex patterns.

To understand how neural network works it is important to know what is the performance of a simple neuron. The following image shows the simplest structure of a neuron with many entries.

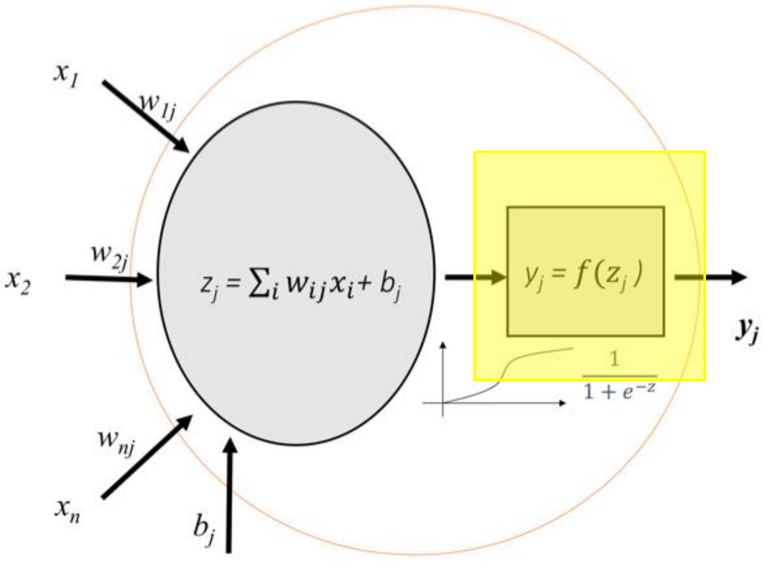


Figure 15. Structure of a neuron

Where the yellow part is an "*activation function"* that is going to be explained later.

The output of each neuron corresponds to the estimated probability of the corresponding class. This probability is obtained from the evidence collected that an image corresponds to a particular class.

It is relatively complex to define the perfect model for a neural network because different layers are useful to different purposes and each parameter matters in the final result.

2.2.2.2.1. Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a specific case of neural networks. They are often used in image processing and they have been a great advance in artificial intelligence.

These networks are also formed by an input and output layers and some hidden layer, most of them convolutional layers, that is why they are called like this (Figure 14)

The layers are formed by neurons with weights and bias that can be learned but a differential feature about these neural networks is that the inputs are supposed to be images so it is possible to codify properties to recognise specific elements.

The main idea is that each layer of the CNN is going to learn different levels of abstraction of the images. [7]

As it has been mentioned before, CNN can be formed by different types of hidden layers. [14] [7]

Convolutional Layers

It consists in filtering an image with small filters to learn local patterns of the images, being able to recognise the learned characteristic even if in other image it is located in a different position.

Other interesting characteristic of these layers is that the patterns can be learned basing on what was learned on the previous layer. This allows to the convolutional neural networks to learn complex and abstract visual concepts efficiently.

The obtained result of the filtering is a reduction of the original image as it can be seen in Figure 16.

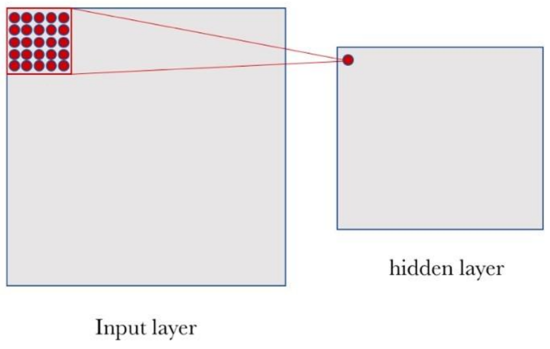


Figure 16. Convolution

It is also important to mention that it is used the same filter to all the neurons of one layer so the network parameters are greatly reduced.

The problem is that each filter is designed to recognise just a particular characteristic so to do the image recognition there are included many filters in the convolutional layer.

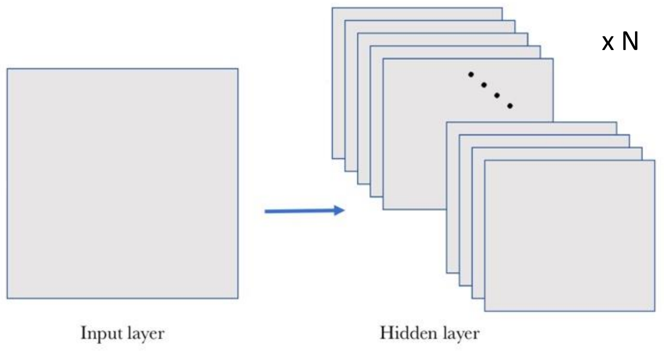


Figure 17. Convolution with N filters

Where N is the number of filters that are used in the convolutional layer.

<u>Pooling Layers</u>

When constructing CNNs, it is common to insert pooling layers after each convolution layer to reduce the spatial size of the representation to reduce the parameter counts which reduces the computational complexity.

Basically, these layers simplify the information of the convolutional layer.

There are many ways to reduce the information, being the most known:

- Max-pooling: this method selects the maximum value of the pooling filter.
- Average-pooling: this method does the average between all the values of the pooling filter.

It is interesting to mention that with the pooling layer the image spatial relation is maintained.

As it has been said before, the convolutional layer has more than one filter so there will be the same number of pooling filters as convolutional ones because the pooling ones are applied separately to each convolutional filter.

The final result is showed in the following image (Figure 18), assuming an input image of size 28x28, 32 convolution filters of size 5x5 and pooling filters of size 2x2

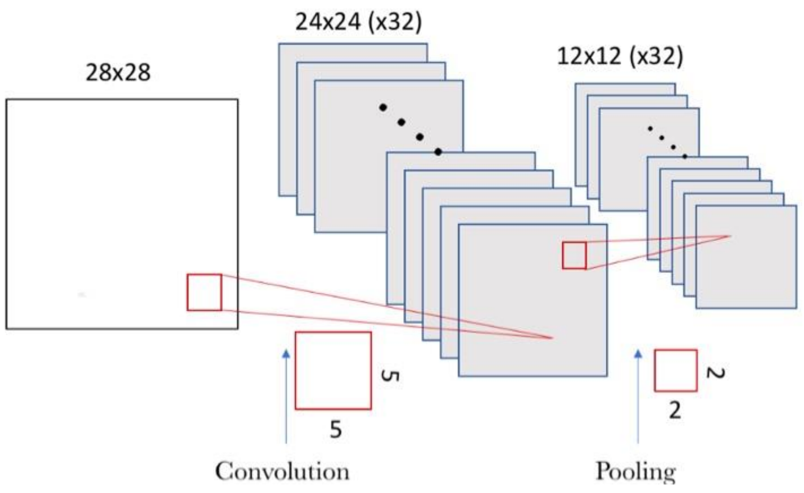


Figure 18. Convolution and pooling

Dropout Layers

These layers deactivate a percentage of neurons on a particular layer. This improves generalization because it forces the layer to learn the same concept with different neurons.

Normally they are used on fully connected layers but it is also possible to use dropout after max-pooling layers, creating some kind of image noise argumentation. [16]

Dense Layers

These layers interconnect all the neurons of one layer with all the neurons of the following layer.

They can be implemented during all the network but it is essential that an image classification model contains a Dense layer at the end of the network with the number of classes that are going to be classified.

Activation functions

To conclude with the components of a neural network it is worth mentioning the Activation functions, that are used to introduce the non-linearity in the modelling capacities of the network.

The most popular are:

Sigmoid

It returns a value between zero and one to any entry value.

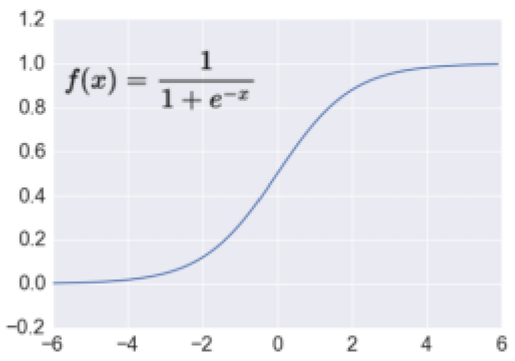


Figure 19. Sigmoid

TanH

This function represents the ratio between the hyperbolic sine and cosine.

The main difference between the sigmoid and the tanh is that the second one returns a value between one and minus one.

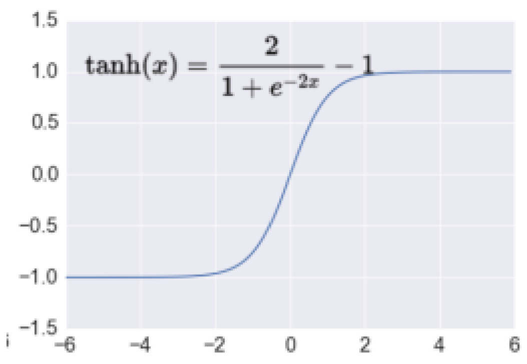


Figure 20. TanH

Rectified Linear Unit (ReLU)

This function activates a node just if it is above a certain threshold.

Normally if the entry has a value smaller than zero the output is zero and otherwise the output is going to follow a linear relation.

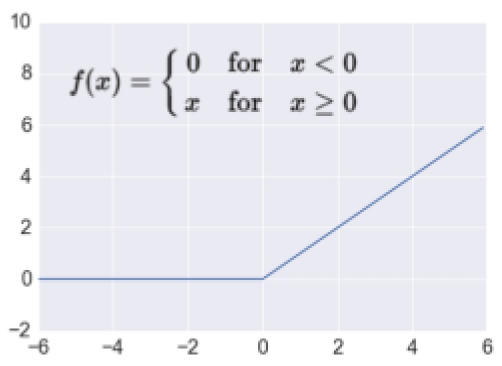


Figure 21. ReLU

<u>SoftMax</u>

It is used in the image classification because it converts the calculated evidences that an entry belongs to a certain class to a probability that the image belongs to that class. The result of the addition of all the probabilities has to be one.

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

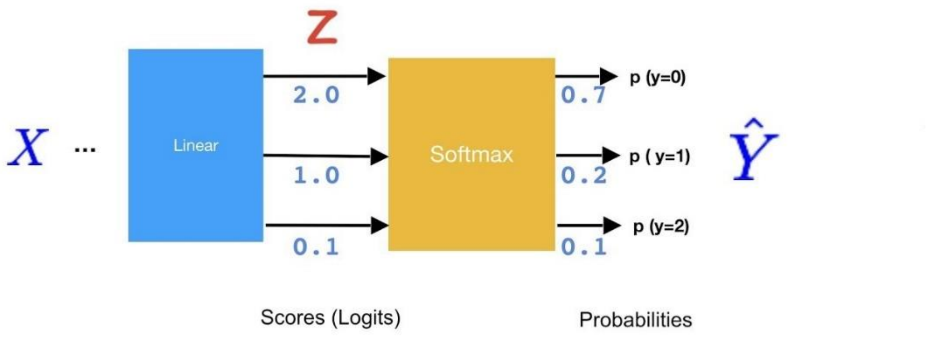


Figure 22. SoftMax

<u>2.3.2.3. Learning Process</u>

The training process of a neural network, it means, learn the values of the weights and bias, is the most genuine part of Deep Learning and it can be seen as a forward and back propagation process.

The first phase, **forwardpropagation**, occurs when the neurons apply the transformation to the information that they receive of the previous layers about the training data and then they send the values to the next layer. At the end, the network is going to have some final values and a predicted label result of the input images. After this, it is going to be used the loss function to estimate the error to compare and calculate how good or bad is the predicted result comparing with the correct label. The ideal value of the loss is zero, it means, without difference between the estimated value and the real one. That is why the model is going to adjust the parameters while the model is being trained.

Once the loss is calculated, it will be propagated backwards (**backpropagation**).

The loss information is propagated to all the previous layer neurons. Each neuron receive just a portion of the loss value and they propagate it backwards until all the neurons of the network have received a loss function. Finally, when the information is backpropagated, the network can adjust the weights and bias. For doing that, it is used a technique called **gradient descent.** This technique changes the values calculating the loss derivative and seeing how to descend the values. [7]

This process can be showed in Figure 23.

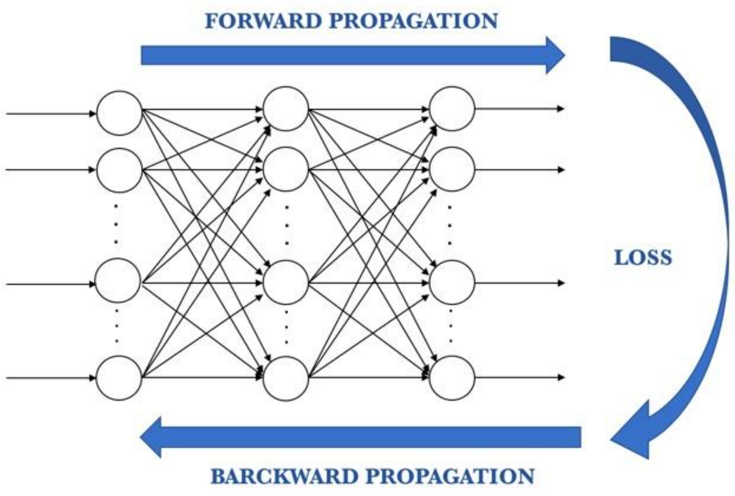


Figure 23. Learning process

Summarising, the learning algorithm consists in:

- Start with random values for the network parameters (or make use of prior knwoledge).
- Use a set of data to obtain their prediction.
- Compare these predictions with the expected result and calculate the loss function.
- Propagate the loss to all the network neurons.

- Use this information to update the information about the weights and bias.
- Repeat the process until it has been obtained a good model.

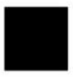
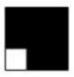
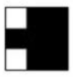
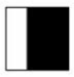
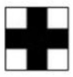
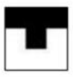
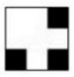
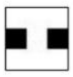
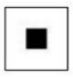
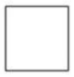
3. ENCODING AND RECONSTRUCTION

Graphic Code (GC) allows sending a large amount of information accentuating the importance of combining it with good aesthetics.

To create this new Machine Readable Code, the GC coding system involves two main processes, the encoding and decoding. The principal element of these processes is the combination between symbols and patterns which is called dictionary.

Quantum is the number of white pixels that can have a 3x3 pattern. It can be a number between 0 and 9. The maximum number of elements that are going to be used, combining alphanumeric characters with other elements (such + - * / @ # % & , . ‘ “), is going to be 75 so it is going to be used the quantum 3, 4, 5 and 6 (Table 1) due to the fact that the number of combinations (quantity of patterns) that they have is bigger than the elements that are needed so it will be possible to associate each symbol with a pattern. [1]

Table 1. Quantum values

Quantum	0	1	2	3	4	5	6	7	8	9
Distribution Black/White	9/0	8/1	7/2	6/3	5/4	4/5	3/6	2/7	1/8	0/9
Quantity of Patterns	1	9	36	84	126	126	84	36	9	1
Gray Scale Range	0-26	27-51	52-76	77-102	103-127	128-153	154-178	179-204	205-229	230-255
Quantized Color	■	■	■	▨	▨	▨	▨	□	□	□
Example of Pattern										

3.1. Encoding

The encoding process aim is to obtain an image made of pixels which contains some information using a specific grid and dictionary.

Given a base image, the first step in this process is to identify the used data cells, called candidates. Then, it is going to be chosen one cell for each message symbol and replace it with the respective dictionary pattern. The rest of the cells are going to be replaced by non-dictionary patterns to finally obtain the Graphic Code. (Figure 24) [1] [6]

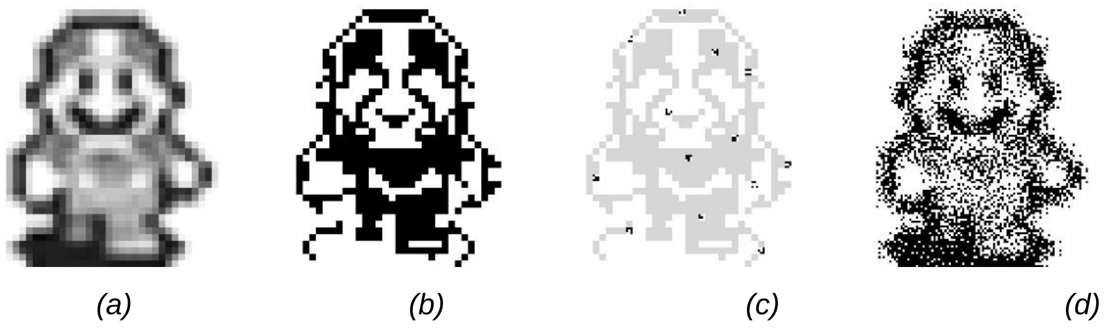


Figure 24. Encoding process. (a) Base Image, (b) Candidates, (c) Chosen cells, (d) Graphic Code

After these processes, it is going to be added a border around the obtained code which will be useful in the decoding process. This border has two elements: a black square of 3 x 3 pixels with a white central pixel and the opposite, a white square of 3 x 3 pixels with a black central pixel (Figure 25). The border is constructed alternating these two squares.

Finally, a same dimension black square is added to the top-left corner of the border, which is important in the reconstruction process to know the original position of the image. [16]

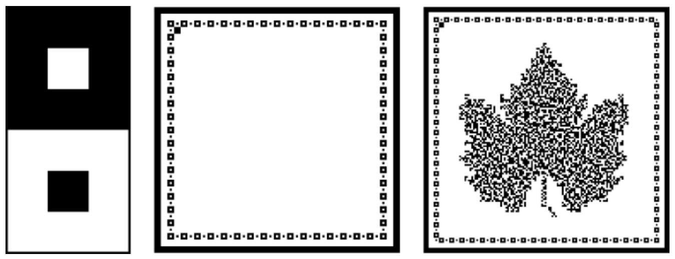


Figure 25. Detection border. (a) Black and white squares, (b) alternative square pattern, (c) complete marker containing the frame.

3.2. Decoding

The decoding model has four processes: acquisition, detection, rectification and reconstruction (Figure 26). It is necessary to implement the previously explained border to be able to do a good decoding process.

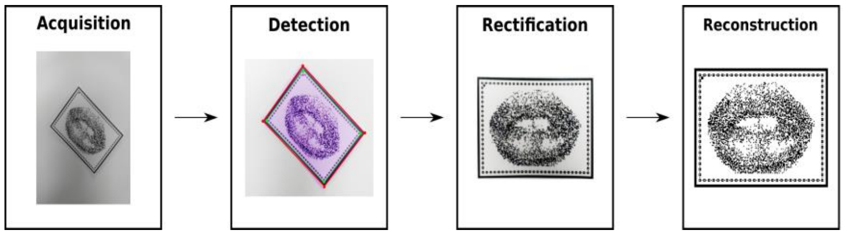


Figure 26. Decoding Processes

This method starts with the acquisition of the printed code. Then, the reconstruction border is searched to rectify the image. When the image is rectified it is made the decoding process, which consists in implementing the inverse steps that were made in the encoding process.

Acquisition and Detection

Firstly, is necessary to obtain the coded image and after this, to detect the code searching the border that is around it. To obtain it, it is necessary to detect the inner and outer quadrilaterals of the external frame (Figure 27). The choice of these rectangles is made by taking the two concentric contours that have the largest area and their respective points. [6]

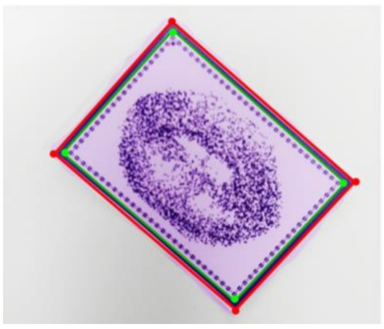


Figure 27. Inner (green) and outer (red) quadrilaterals

<u>Rectification</u>

The rectification process consists in obtaining an image where the code appears without distortion and in which it will be possible to obtain the translation and rotation between the marker and camera planes. This process is where the top-left corner black square is important because it indicates the code orientation.[6]

When the border is recognised, it is rectify using Homography, that consists in calculate a matrix from the coordinates of the border obtained in the scanning process for being able to rectify the image using the following ecuations:

$$A' = H * A$$

$$B' = H * B$$

$$C' = H * C$$

$$D' = H * D$$

Where A,B, C and D are the scanned coordinates of the border and A', B', C' and D' are the coordinates of the rectified image.

Reconstruction

Having the rectified image, the following step is to reconstruct it. This process is necessary because the initial image and the received after all the previous processes are going to be different either in size as in color. Therefore, it is necessary to identify which pixels of the obtained image belong to the original image ones.

Once they are identified it is going to be decided their color because they are going to be in greyscale instead of black and white as the original ones. To decide if these pixels are black or white it is going to be created a threshold taking into account the color of the rectification border due to the fact that it has a noticeable color difference between its squares. [1] [6]

Decoding

When the image reconstruction is finished it is possible to recover the sent message following the inverse steps that were implemented in the encoding phase.

Firstly, it is necessary to search on the grid patterns that are defined in the dictionary and when they are found, replace them by the related symbol. [1] [6] [16]

The entire process of encoding and decoding can be shown in Figure 28.

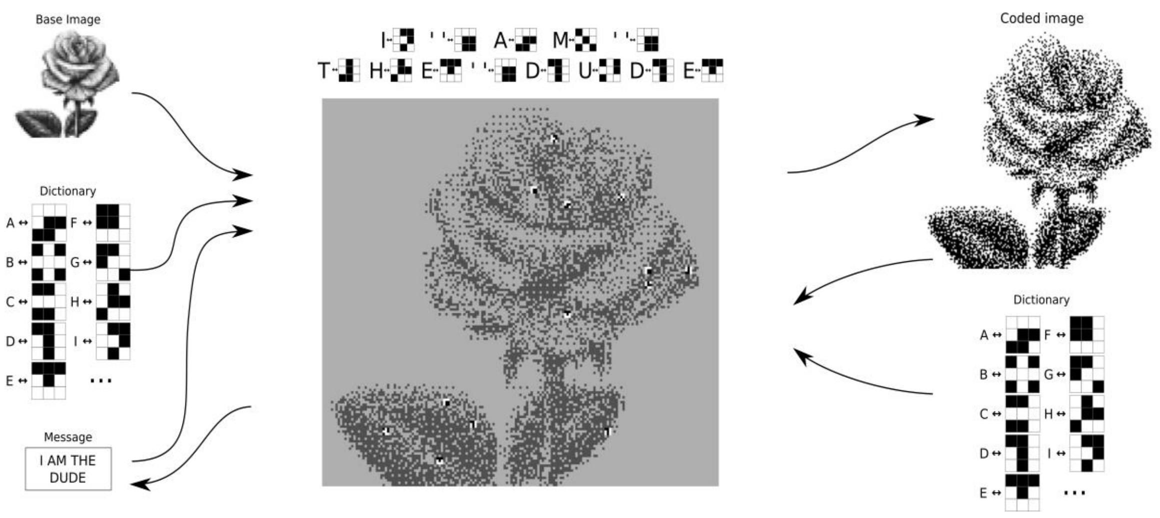


Figure 28. Encoding and decoding processes

3.3. Icon Based Graphic Code Reconstruction

All that has been mentioned in this chapter is related to Graphic Code made of black and white pixels.

But in the Graphic Code made with icons the coding processes are going to be different due to the fact that the coded images are not going to be formed by patterns but with different icons without following a base image (Figure 29). [6]

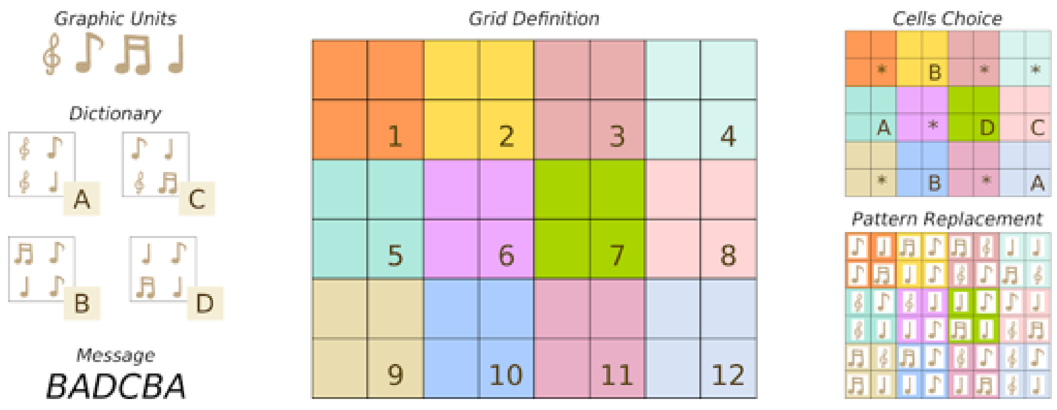


Figure 29. Process of coding icon Graphic Code.

In the Figure 29 it can be seen that in Graphic Code made of icons the previous mentioned processes are going to be slightly different. In this Graphic Code style, the dictionary is going to be created of different groups of icons to later select some cells to put the information that will be sent and fill the other with combinations of icons that do not belong to the dictionary.

The principal idea is to be able to send information combining groups of several icons with different positions.

The main advantage of using icons instead of pixels is that the position of icons does not depend on an orthonormal structure.

In Figure 30 it can be seen an example of icon Graphic Code using three types of icons. It can be seen that combining them in different ways and putting them in different positions it is possible to obtain a good appearance Graphic Code. Besides this Graphic Code is not evident it can be an advantage to avoid falsification.

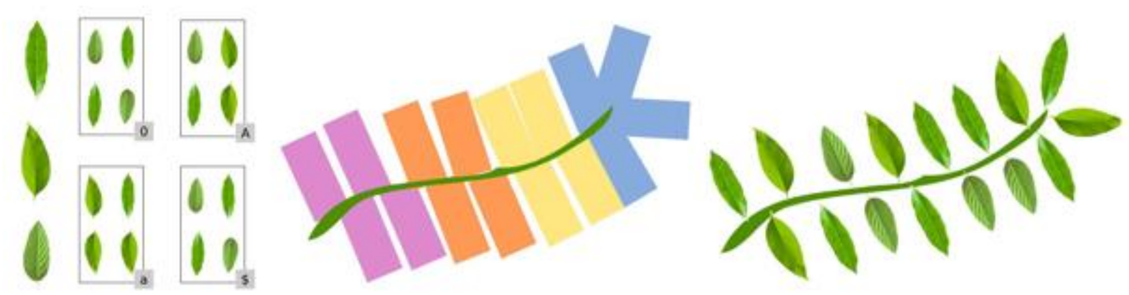


Figure 30. Icon Graphic Code

To achieve the recognition of icons, as it was said in Chapter 1, they are going to be implemented some Machine Learning methods able to recognize different kind of pictures.

Therefore, it has been done a study of two Machine Learning methods which are normally used to implement image recognition and it has been possible to compare their properties and results.

4. EXPERIMENTS AND RESULTS

The study that has been realised in this project is related with Machine Learning methods and their capacity to recognise icons. Specifically, we present implementations of Support Vector Machine and Convolutional Neural Network algorithms because they are able to recognise and divide different kind of figures in a very efficient way.

These algorithms learn patterns from the entry data and they associate them with a previous defined label. Therefore, the set of entry data is going to be crucial in the learning phase of the algorithm because it is the only thing that the machine has to decide to which group belongs each image.

4.1. Libraries and workspace

Before explaining the different processes followed to perform the images recognition, we will explain the used libraries.

Python

Firstly, it is necessary to say that the programming language which is going to be used is Python. It was chosen because of its easy installation of libraries that can be used for implementing Machine Learning methods.

NumPy

NumPy is a Python library that makes easier the use of mathematical functions to operate with vectors and matrices. [17]

Matplotlib

Matplotlib is a library to generate graphics from arrays or lists. In this case it was used to represent the results using the NumPy arrays as entry data. [18]

Modules

Two different python modules are implemented in this project: “os” and “re” modules.

The first one is used to access to the Operative System functions to manipulate directories structure (read and write documents, manipulate paths, count the number of directories or documents etc). [19]

The second one is used when there are regular expressions that use some special characters that normally in the programming language have a specific use, and it is necessary to specify that they are being used as particular expressions. [20]

TensorFlow

TensorFlow is an open-source library for machine learning created by Google with the intention of creating and training neural networks for detecting patterns similar to the human learning. [21]

Keras

Keras is an open-source library written in Python. It is specially designed to enable experimentation with deep neural networks, as CNNs. It is implemented because its good characteristics as being modular, user-friendly and extensible and it is capable of running on top TensorFlow. [22]

OpenCV

OpenCV (Open source Computer Vision) is a multiplatform library of programming functions mainly aimed at real-time computer vision. It also supports the TensorFlow framework. [23]

Scikit-learn

Scikit-learn is an open-source machine learning library for implementing it on Python. It contains some classification algorithms and, in this project, it has been used for implementing SVM algorithm. [24]

4.2. Algorithms implementation

The implementation of the two Machine Learning algorithms is going to be executed with the same group of data. This entry set is going to be formed of several icon images. In this case the chosen icons are a bone, a Christian cross, fire, a Heart, the letter A, a music note, a puzzle piece and a star (Figure 31).

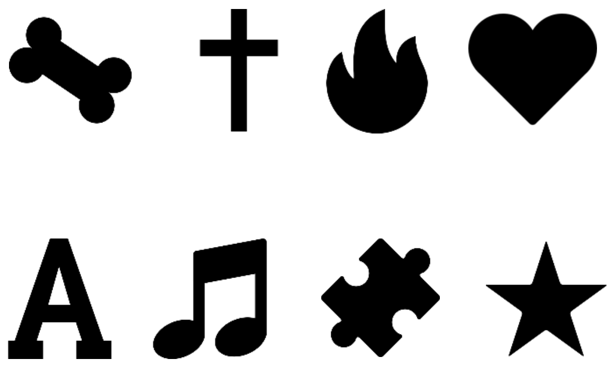
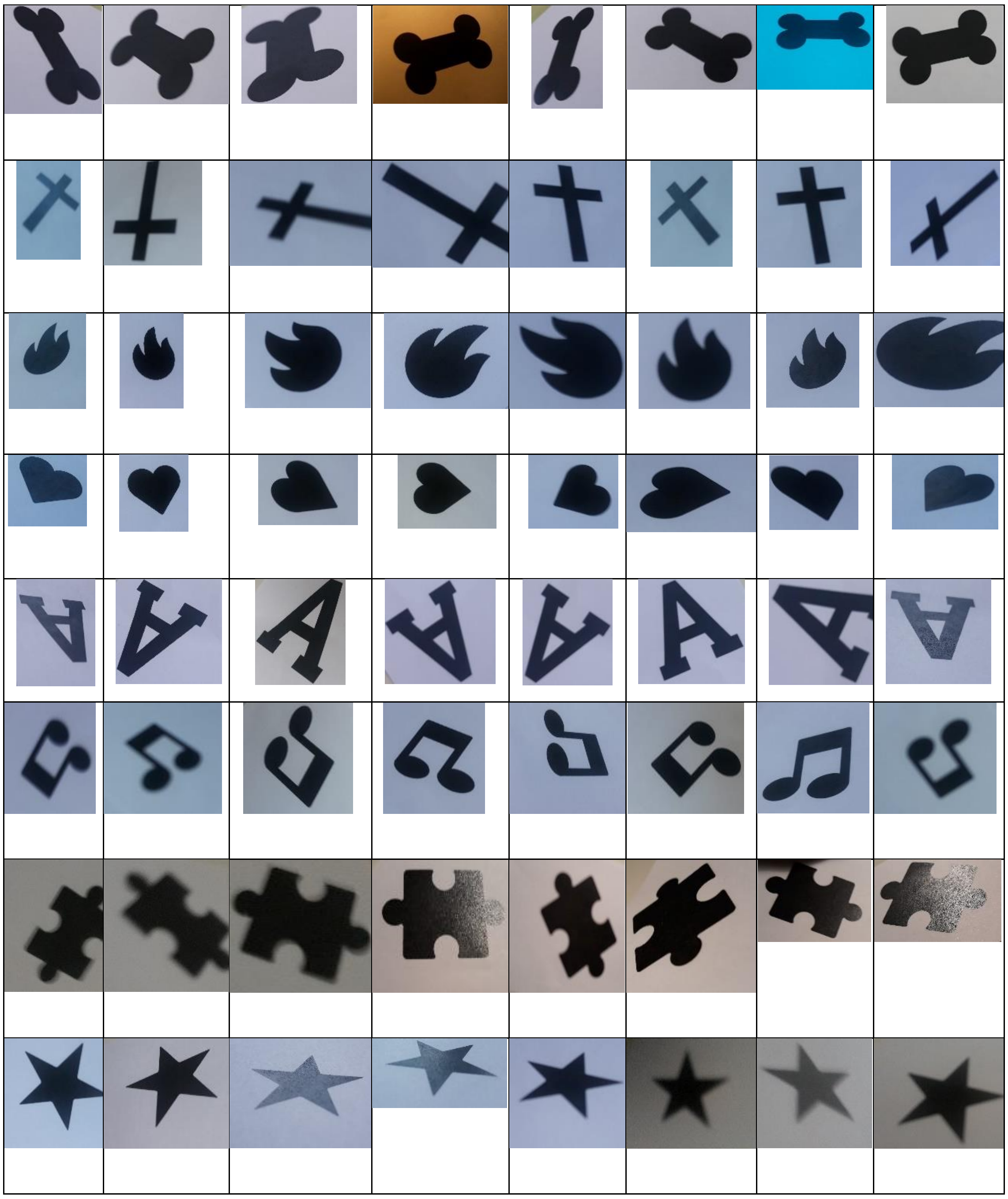


Figure 31. Icons

After the selection of the icons, it is necessary to take a lot of pictures of them and later rotate them to get more pictures (Table 32). Finally, all the images have to be resized to a lower size because being such a lot of big files, it will take much time to process them. [APPENDS I, II and III]

Once all the images have been processed, they are saved in different folders with the name of each icon. This is very important because the label that is going to be associated to each picture is the name of the folder where the icon is.

Table 2. Some images used as input data



4.2.1. SVM development

The Support Vector Machine algorithm implemented in this project has three defined parts: data extraction, SVM architecture and model training.

Data extraction

It has been created a function to load the images which returns an object with some attributes, being two of them the images and the corresponding labels. After this, all the data is divided in two groups: training and test data. [APPEND IV]

SVM architecture

As it was explained in section 2.2.2.1, there are some parameters which can entail an improvement in the accuracy of the model: kernel, margin and regularization (that is called “Cost parameter” so normally is implemented with the letter C) and gamma. [APPEND V]

In this project the implementation of these parameters is the following one:

- The chosen kernel parameter is the Radial Basis Function (RBF) with degree equals to three.
- The most appropriate values for C (margin and regularization) and gamma parameters can change depending on the number of images and icons that are going to be evaluated. Therefore, it is not recommendable to use every time the same value for these parameters. The developed solution consists of using a function implemented by scikit-learn library which select, between some possible given values, the most appropriate for each evaluated model. The possible values provided for C parameter are 1, 10, 100, 1000, 10000 and for parameter gamma 0.0001, 0.001, 0.01, 0.1, 1, 10, 100.

Model training

Finally, the model is trained doing 100 iterations to perform for each mini batch. The value of the features taken in each batch is 3. [APPEND VI]

Once the model is trained, the accuracy is calculated with the test data. The results of the implemented tests can be shown in Section 4.3.

4.2.2. CNN development

The CNN implemented in this project has the same parts as the SVM model: data extraction, CNN architecture and model training.

Data extraction

The images and labels are saved in two different arrays. To associate them correctly, the data is inserted in its related vector in the same order. Therefore, the image and its corresponding label are in the same index so they can be associated by it.

For doing this, we implemented functions from some libraries previously mentioned.

When all the data is loaded, it is divided in two sets: training and test data. The training images entail an 80 % of the input data and the test images the rest. [APPEND VII]

CNN architecture

In section 2.2.2.2 the different parts of the artificial neural networks were explained. They are composed by three different layers: input, output and hidden.

The input layer of the model that has been implemented is the loaded data and the output are the eight percentages of each possible class that the images can belong to.

The hidden layers can have many combinations. In this project the neural network has the following ones:

- Convolutional layer of 32 filters.
- Convolutional layer of 64 filters.
- Convolutional layer of 128 filters.
- Convolutional layer of 64 filters.
- Convolutional layer of 32 filters.

The filters of the convolutional layers are of size 3 x 3 pixels and the activation function used in all of them has been ReLU. After each convolutional layer it has been implemented a Max-pooling layer of size 2 x 2 pixels.

Before the output layer it has been implemented a Flatten layer which transform the data matrix to a vector, a Dense layer with 64 outputs and ReLU activation and finally a Dense layer with an output of the number of the classes, in this case eight. [APPEND VIII]

Model training

After the model is structured it is going to be compiled and trained with the training data. The number of epochs that is going to be trained is 20 with a batch size of 32. It is important not to use a high value of training epochs because, besides it will take a lot of time, there can be a problem of overfitting. [APPEND IX]

4.3. Results

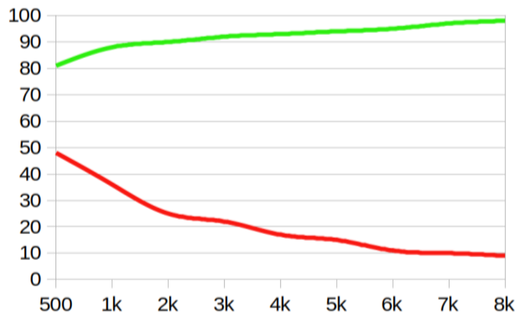
This chapter shows the results about the accuracy and loss of the different Machine Learning methods, implementing different number of images and icons.
Firstly, it is going to be mentioned the different groups of images created to do the tests:

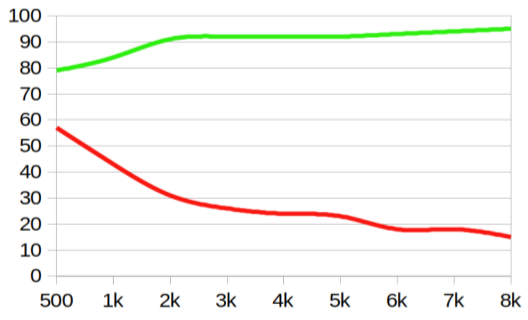
- Group of 4 icons: Heart, Star, Fire and Music.
- Group of 5 icons: Heart, Star, Fire, Music and Puzzle.
- Group of 6 icons: Heart, Star, Fire, Music, Puzzle and Bone.
- Group of 7 icons: Heart, Star, Fire, Music, Puzzle, Bone and Cross.
- Groups of 8 icons: Heart, Star, Fire, Music, Puzzle, Bone, Cross and Letter A.

Besides, the tests have been realised using different number of icons.

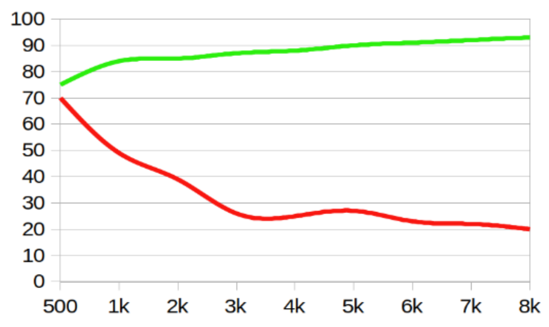
CNN

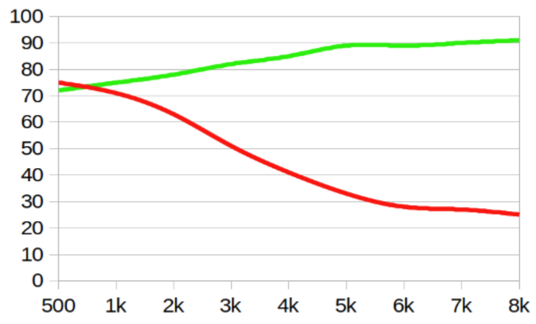
In the following figures it can be showed the different accuracy and loss curves of the different groups of icons while the number of images used increase.



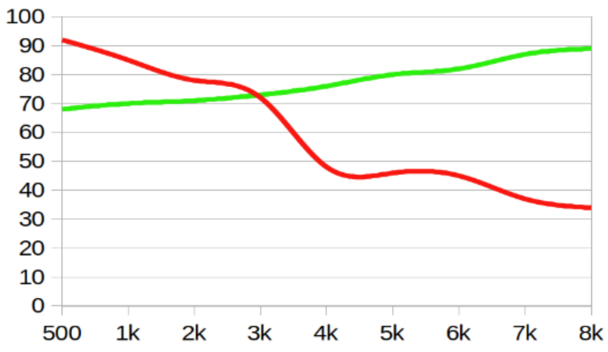


(a) *(b)*





(c) *(d)*



(e)

Figure 32. Accuracy (green) and loss (red) graphics testing different groups of icons. (a) 4 icons, (b) 5 icons, (c) 6 icons, (d) 7 icons, (e) 8 icons.

As it can be seen in Figure 31, the model accuracy grows and the loss decrease while they are added more images. This happens because, as it has been explained in chapter 2, the neurons learn different characteristics of the input data to later associate these characteristics to a specific label. Therefore, when the number of input images increases, the model will associate the identified characteristics to a label with more assurance.

Besides, the initial values of accuracy and loss are very separated with less icons, but when they are added more icons the initial values get closer. It can be seen that using 500 images with the group of 7 icons and using 500,1000, 2000 and 3000 images with the group of 8 icons the loss is higher than the accuracy so the models with these characteristics are not going to get good results.

In Figure 32 it is represented the comparison of the accuracies obtained with each group of icons while the number of images increases. It can be seen that the best accuracies obtained are with 8.000 images, in particular in the case of 4 icons. In the contrary way, the worst accuracy is obtained with 8 icons using 500 images.

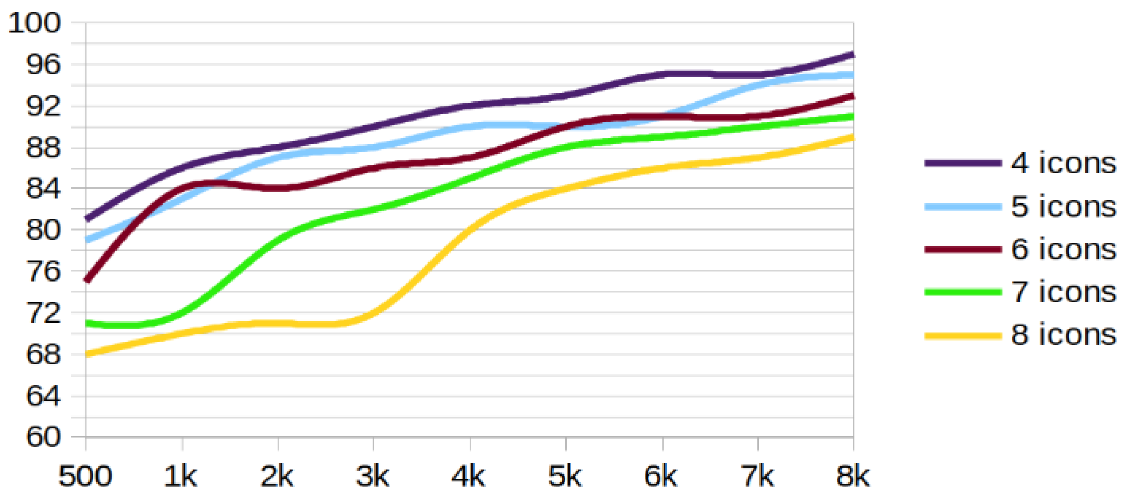


Figure 33. Accuracy comparison using different number of icons.

In the following figures it can be seen the results of the testing phase with each group of icons using 500 and 8000 images. As it has been said, the data is divided in training and test data so the data used for these results is the 20% of the 500 and 8000 images.

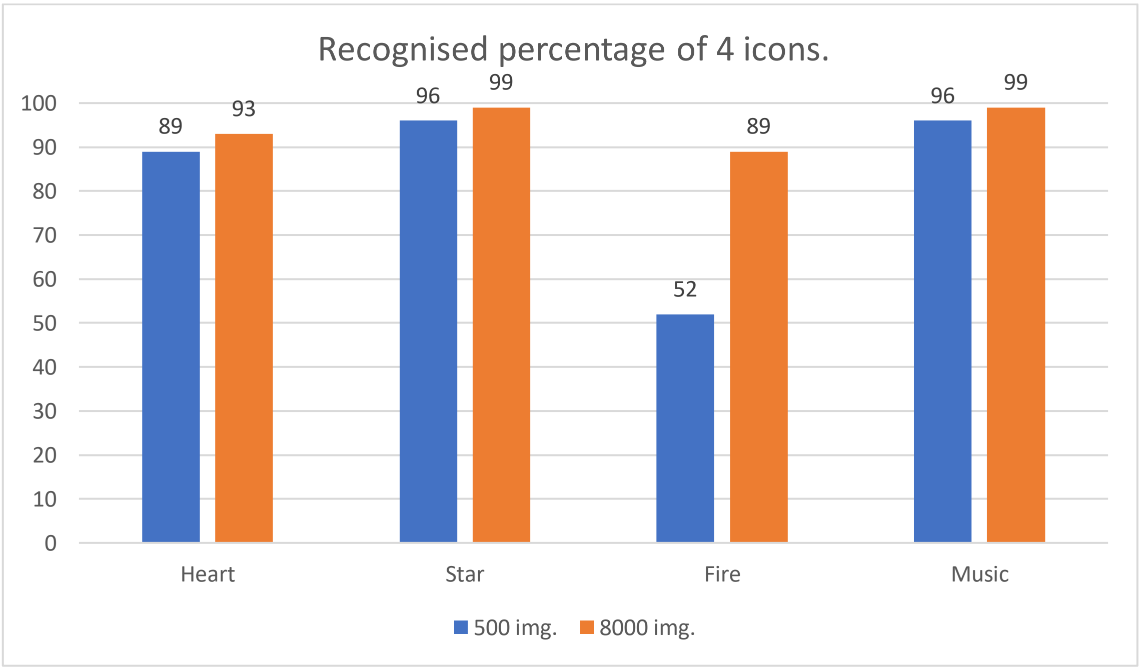


Figure 34. Percentage of recognised icons using 500 images (blue) and 8000 (orange) with the group of 4 icons in CNN model.

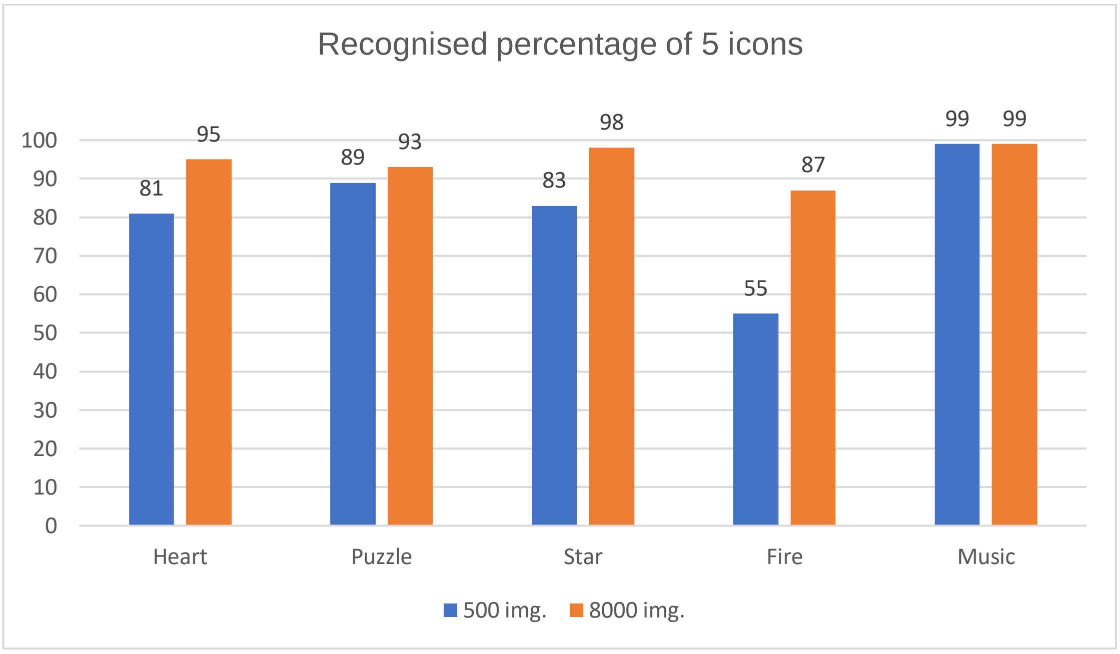


Figure 35 Percentage of recognised icons using 500 images (blue) and 8000 (orange) with the group of 5 icons in CNN model.

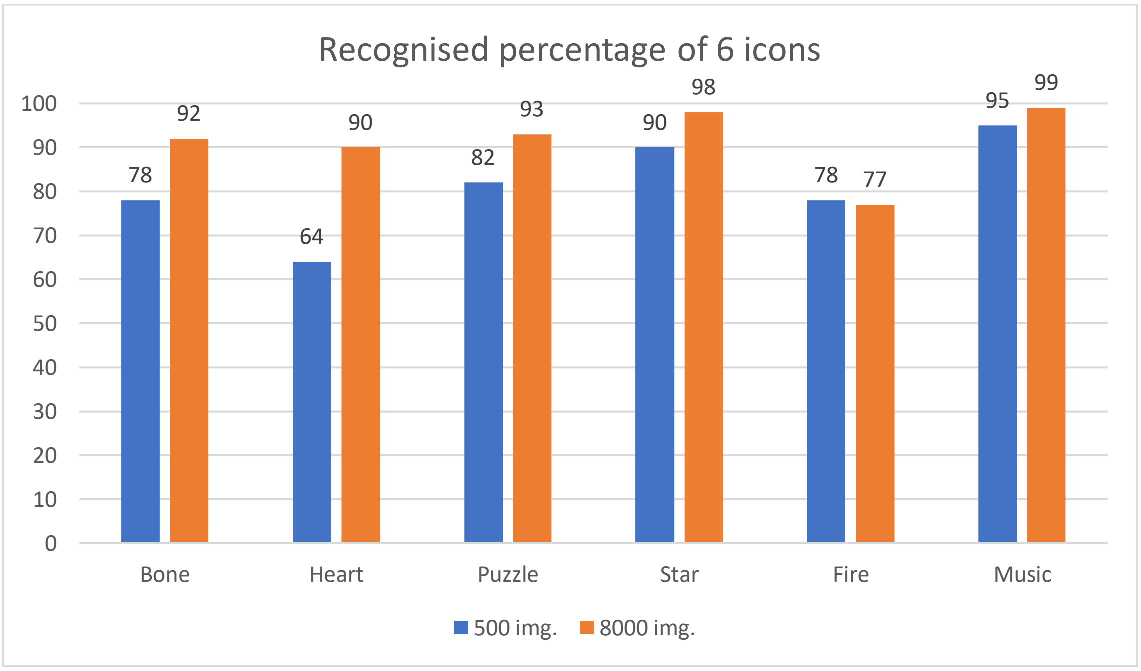


Figure 36. Percentage of recognised icons using 500 images (blue) and 8000 (orange) with the group of 6 icons in CNN model.

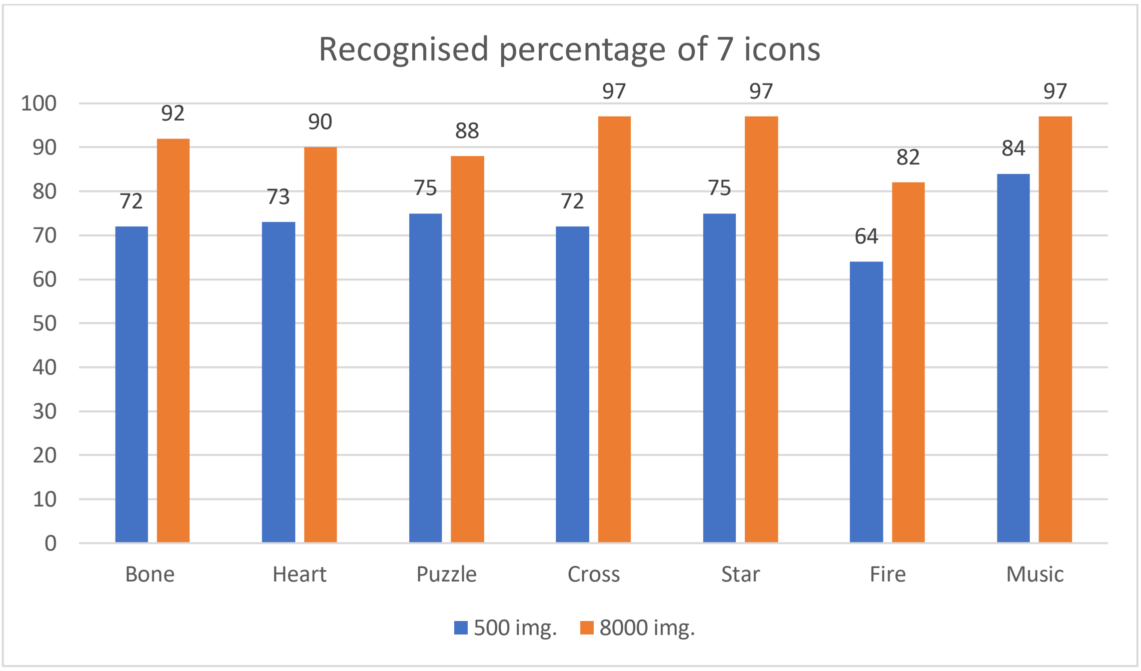


Figure 37. Percentage of recognised icons using 500 images (blue) and 8000 (orange) with the group of 6 icons in CNN model.

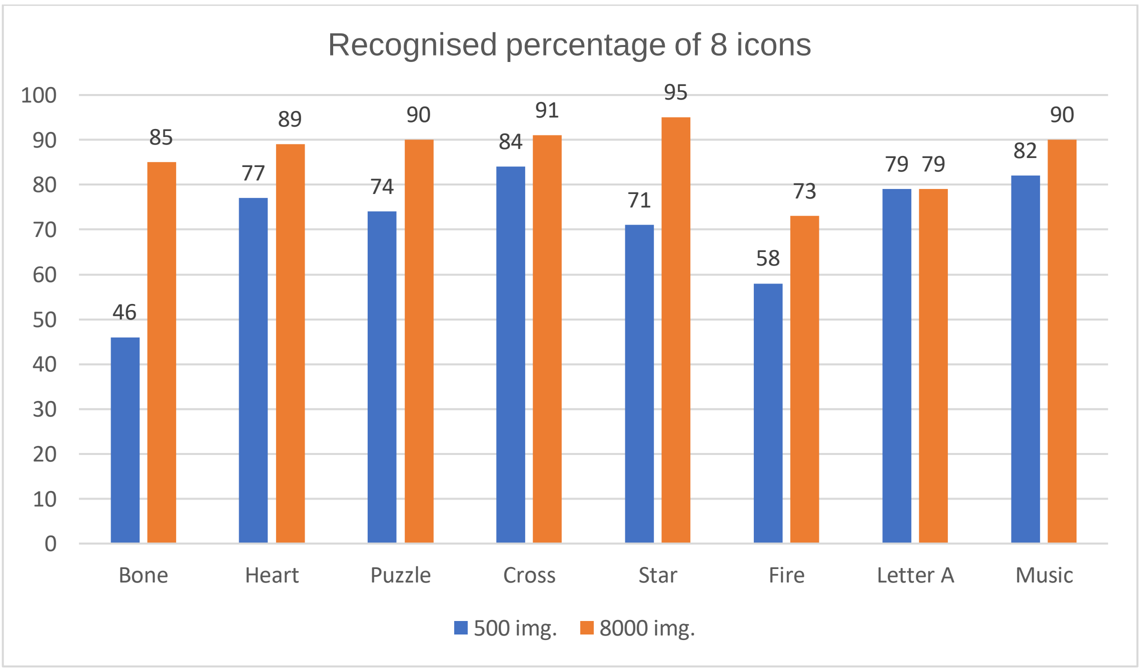


Figure 38. Percentage of recognised icons using 500 images (blue) and 8000 (orange) with the group of 6 icons in CNN model.

In the last five figures it can be seen that, generally, Fire icon has the less percentage of recognition. A curiosity about this is that the most unrecognised Fire icons were consider by the model as Heart icons. This is because CNN algorithm learns different patterns of the icon so if the icons have similar characteristics the model will have difficulties to differentiate them.

As it can be seen in the following figure Fire and Heart icons have similar characteristics.

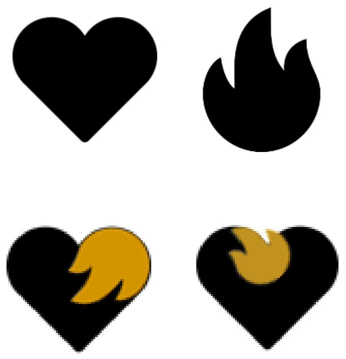


Figure 39. Similarities between Fire and Heart icons

Finally, they were done some accuracy tests with groups of 4 different icons in each group:

- Group 1: Heart, Star, Fire, Music
- Group 2: Cross, Heart, LetterA, Puzzle
- Group 3: Star, Bone, Letter A, Music

The following figure shows the different accuracy of each model.

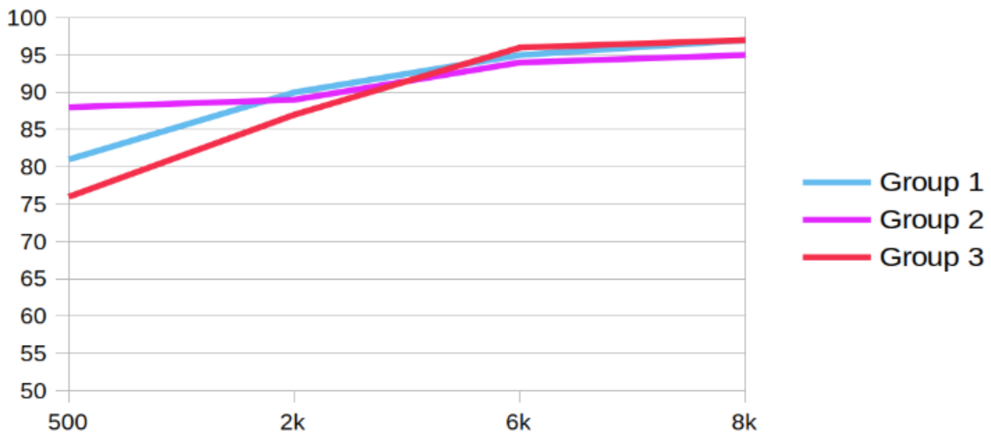


Figure 40. Accuracy differences between different groups of icons.

In this test the obtained accuracies are different for each group of icons even though they are used the same number of training images thus it is demonstrated that for obtaining a good accuracy it is also important to choose icons that, evaluating them together with a model, it is possible to differentiate them in the most effective way.

SVM

The following figure shows the accuracy of the SVM models using 4 and 8 icons while the input data is increasing. It can be seen that the accuracy with 4 icons is better than with 8 icons.

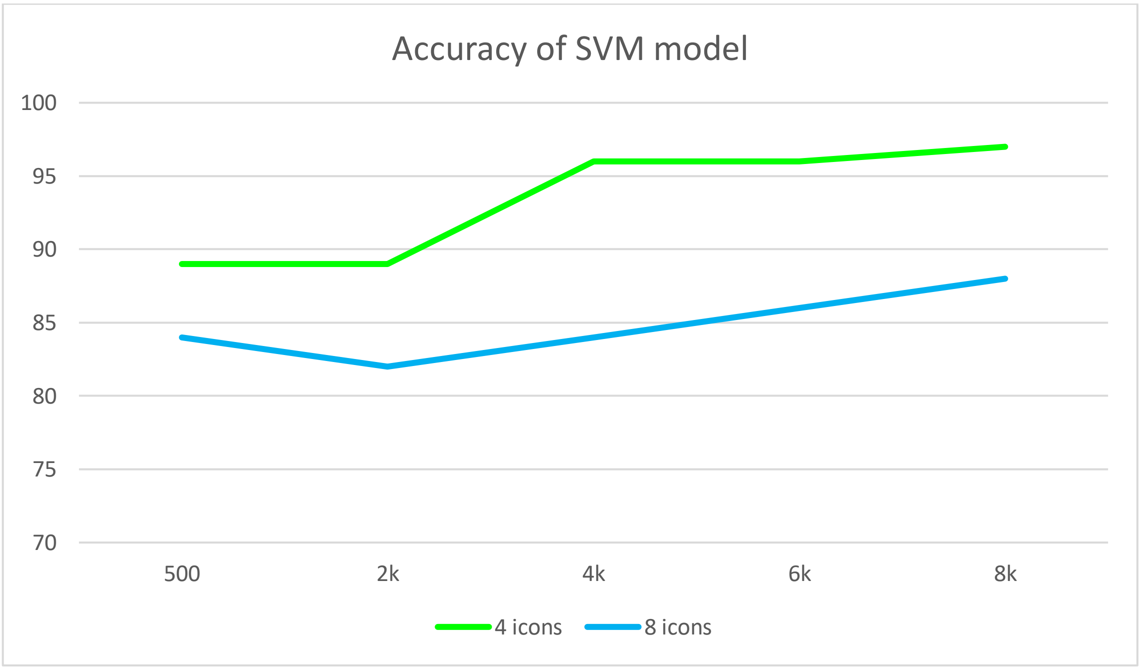


Figure 41. Accuracy of the SVM model using 4 and 8 icons while the input data increases.

As it has been said before, the models are going to calculate the best values for C and Gamma parameters depending on the input data. The following table shows the chosen values for these parameters.

Table 3. Values of C and Gamma parameters

	4 icons		8 icons	
	C	**Gamma**	**C**	**Gamma**
500 images	10	1	100	1
2000 images	10	1	100	1
4000 images	10000	1	100	10
6000 images	100	10	100	10
8000 images	100	10	1000	10

The following figures show the results of the test phase done after the training, where the input data is a vector with random images of the icons. In spite of being few images, the obtained results are really good because hardly all of the predicted labels match with the original ones.

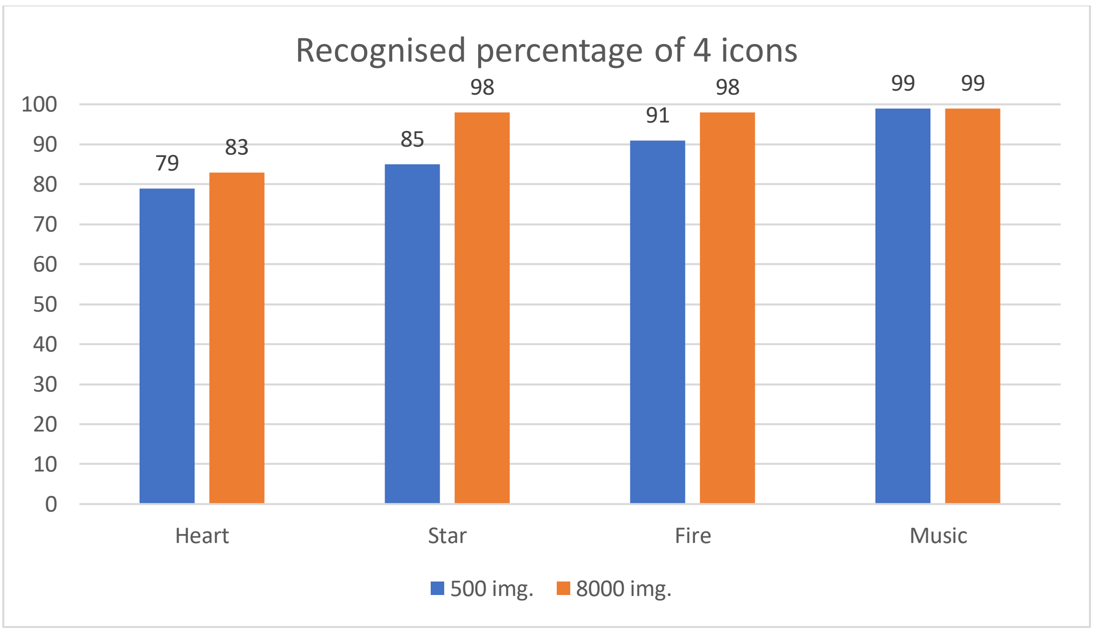


Figure 42. Percentage of recognised icons using 500 images (blue) and 8000 (orange) with the group of 4 icons in SVM model.

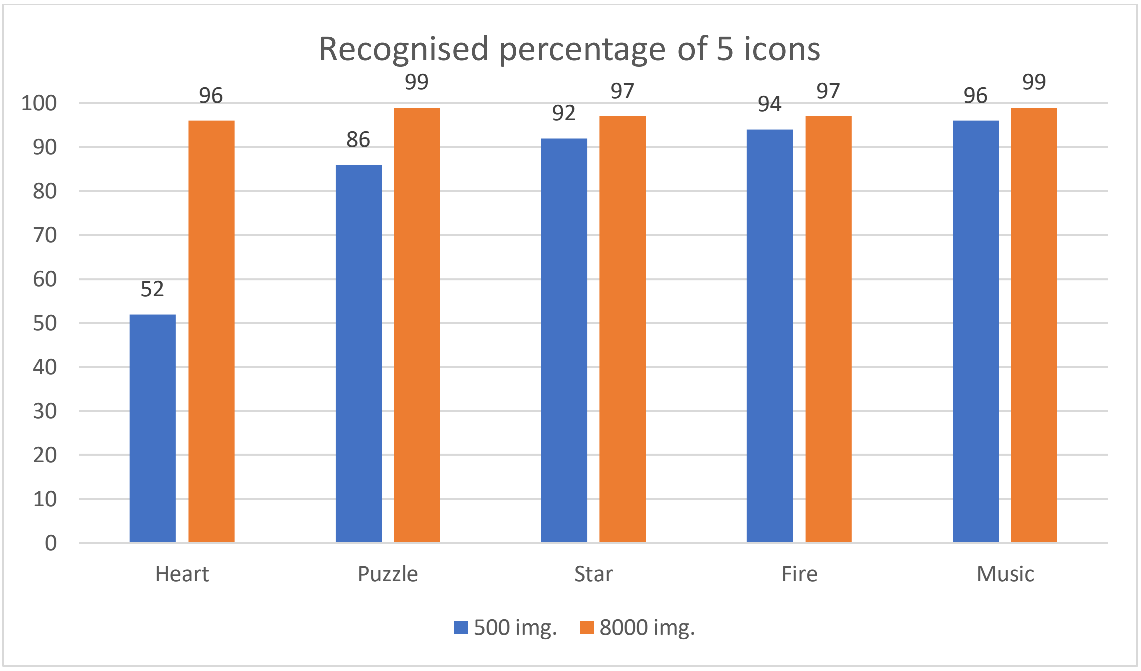


Figure 43. Percentage of recognised icons using 500 images (blue) and 8000 (orange) with the group of 5 icons in SVM model.

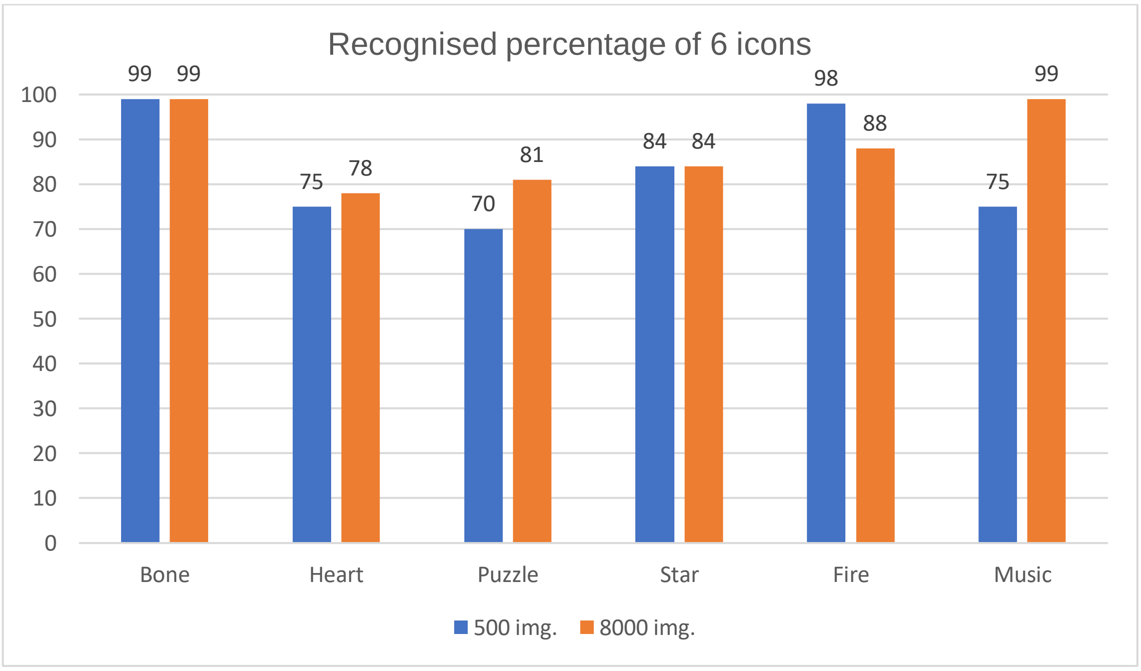


Figure 44. Percentage of recognised icons using 500 images (blue) and 8000 (orange) with the group of 6 icons in SVM model.

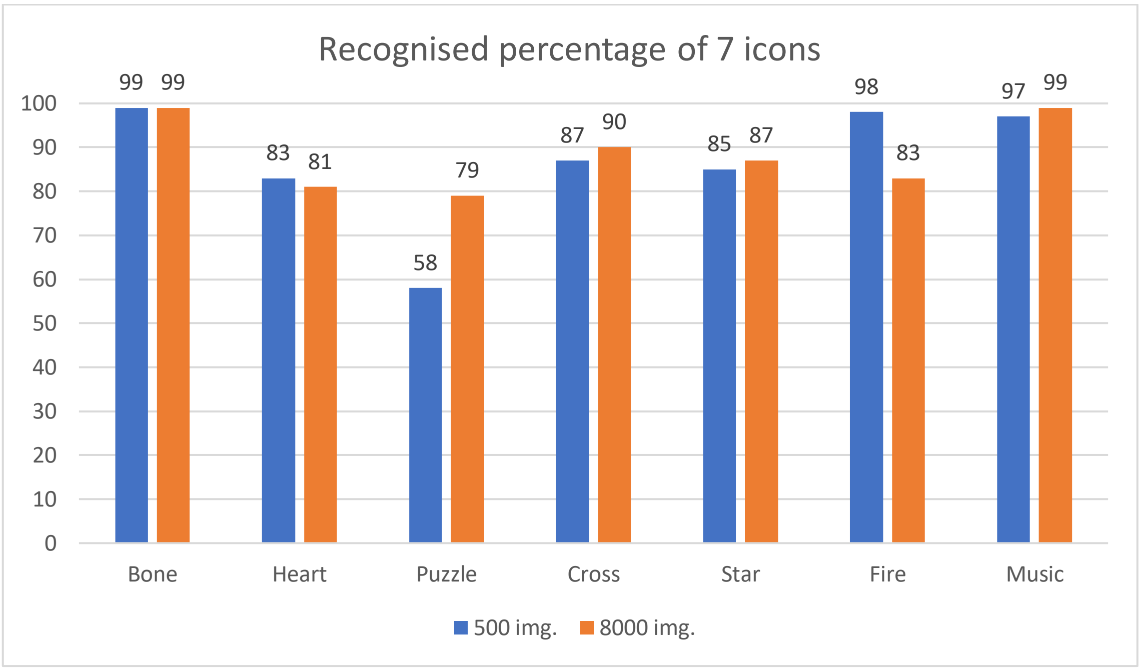


Figure 45. Percentage of recognised icons using 500 images (blue) and 8000 (orange) with the group of 7 icons in SVM model.

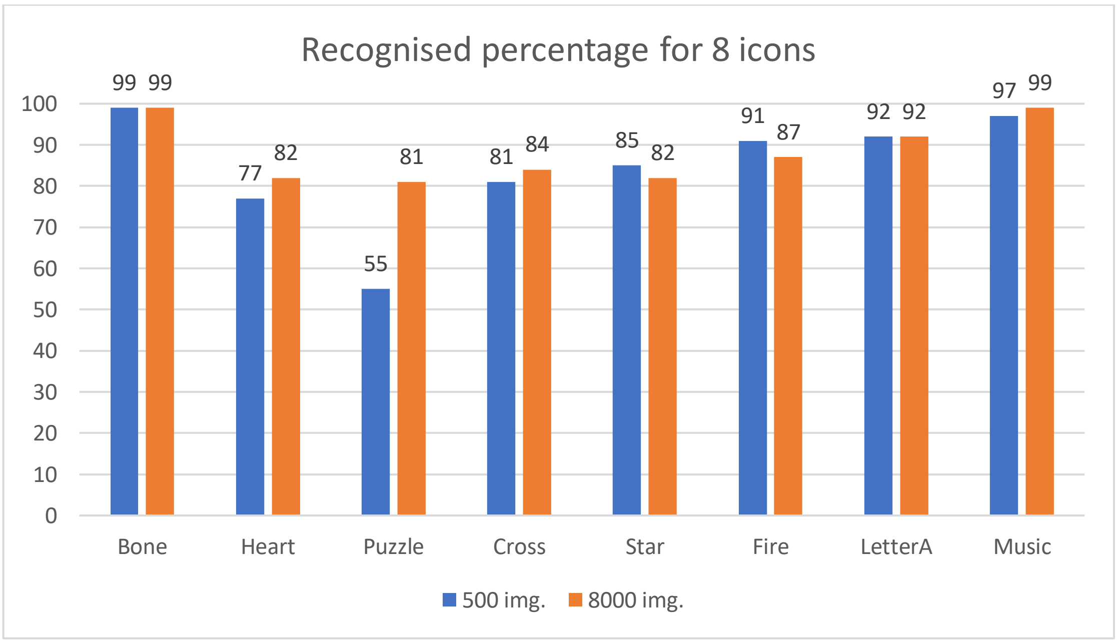


Figure 46. Percentage of recognised icons using 500 images (blue) and 8000 (orange) with the group of 8 icons in SVM model.

SVM algorithms are characterized for their ability to separate few classes. In the results obtained in the training phase of SVM models it can be seen that with 4 icons the recognition is not bad but when they are added more icons the percentage starts to decrease. In the last two figures it can be seen that the model is able to separate 2 of the icons with a very high percentage of recognition but the rest of the icons do not have such a good percentage.

4.4. Comparison

After doing the different tests with the two algorithms changing their characteristics, they can be obtained some differences between them.

The first main difference between these two methods is the time that they take for executing. CNN algorithm takes less than 30 minutes to compile the worst case (8 icons with 8.000 images of each one) while SVM algorithm takes at least 6 hours to compile in the worst case. This can be because the image values used in SVM are normally a selection of some of them, but in this case, they are used all the values.

Other difference about the results is that with CNN models it is needed a big number of images to achieve good percentage of recognition. As it can be seen in the result figures, testing the models with 500 images, the CNN ones hardly ever achieve the 90% of recognition. Nevertheless, SVM models get good results with less images but when they are added more icons the percentage start to be worse.

5. CONCLUSION

The study of this project has consisted of comparing two Machine Learning methods (SVM and CNN) and seeing the results implementing them in icon recognition.

About the CNN model it has been demonstrated that the model achieved better accuracy as more images are included, having the same number of icons. But if the number of icons change and the number of images is the same the accuracy is worse.

Besides, the accuracy depends totally in the input data so it is going to change depending on the chosen icons and the combination of them.

Finally, the CNN algorithm is really fast but the problem is that they are needed a lot of images to be able to get a good accuracy.

About the SVM algorithm, the first problem is that it does not load a lot of images so it is not possible to get an accuracy as good as in CNN model. Besides, it is very slow compiling.

Despite these disadvantages, this model does not need a lot of images to have a good result but it never reaches more than 90% of accuracy. Besides, adding more icons with the same number of images does not entail worse accuracy.

5.1. Future implementation

The study of this project has been realised to be able to implement these techniques in Graphic Code. The aim is to create some different icons and code information on them for later do the decoding using Machine Learning algorithms.

REFERENCES

[1] L. Cruz, B. Patrão, N. Gonçalves, “Large Scale Information Marker Coding for Augmented Reality using Graphic Code”, 2018.

[2] Y. Adrián, “Código de Barras- Qué es y Definición”, 2019. [Online]. Available: https://conceptodefinicion.de/codigo-de-barra/

[3] G. Alsina González , “Definición de Código de barras”, 2016. [Online]. Available: https://www.definicionabc.com/tecnologia/codigo-de-barras.php

[4] “¿Qué es un código QR?”, [Online]. Available: https://www.unitag.io/es/qrcode/what-is-a-qrcode

[5] “¿Qué son los códigos QR y cómo funcionan?”, [Online]. Available: https://computerhoy.com/noticias/internet/que-son-codigos-qr-como-funcionan-14973

[6] L. Cruz, B. Patrão, N. Gonçalves, “Graphic Code: Creation, Detection and Recognition”, 2018.

[7] J. Torres, “Deep Learning- Introducción práctica con Keras”, [Online]. Available: https://torres.ai/deep-learning-inteligencia-artificial-keras/

[8] M. Rouse, “Aprendizaje automático (machine learning)”, 2017 https://searchdatacenter.techtarget.com/es/definicion/Aprendizaje-automatico-machine-learning

[9] “What Is Machine Learning? | How It Works, Tecchniques & Applications”, [Online]. Available: https://www.mathworks.com/discovery/machine-learning.html

[10] “Conceptos fundamaentales en Machine Learning: función de pérdida y optimización”, [Online]. Available: https://planetachatbot.com/conceptos-fundamentales-en-machine-learning-funci%C3%B3n-de-perdida-y-optimizaci%C3%B3n-e30c25404622

[11] S. Patel, “Chapter 2: SVM (Support Vector Machine)-Theory”, 2017, [Online]. Available: https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

[12] “Kernel Functions- Introduction to SVM Kernel & Examples”, [Online]. Available: https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

[13] C. Alonso Gonzalez, “SVM: Máquinas de Vectores Soporte”, [Online]. Available: https://www.infor.uva.es/~calonso/MUI-TIC/MineriaDatos/SVM.pdf

[14] O. Gazi Yalçin, “Image Classification in 10 Minutes with MNIST Dataset”, 2018, [Online]. Available: https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d

[15] L. Araujo Santos, “Dropout Layer-Artificial Inteligence”, [Online]. Available: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/dropout_layer.html

[16] L. Cruz, B. Patrão, N. Gonçalves, “An Augmented Reality Application Using Graphic Code Markers”, 2018.

[17] “What is NumPy?”, [Online]. Available: https://docs.scipy.org/doc/numpy-1.13.0/user/whatisnumpy.html

[18] “Matplotlib: Python plotting—Matplotlib 3.1.1 documentation”, [Online]. Available: https://matplotlib.org/

[19] “OS Module in Python with Examples”, [Online]. Available: https://www.geeksforgeeks.org/os-module-python-examples/

[20] “Python Regular Expressions”, [Online]. Available: https://www.tutorialspoint.com/python/python_reg_expressions

[21] “TensorFlow”, [Online]. Available: https://es.wikipedia.org/wiki/TensorFlow

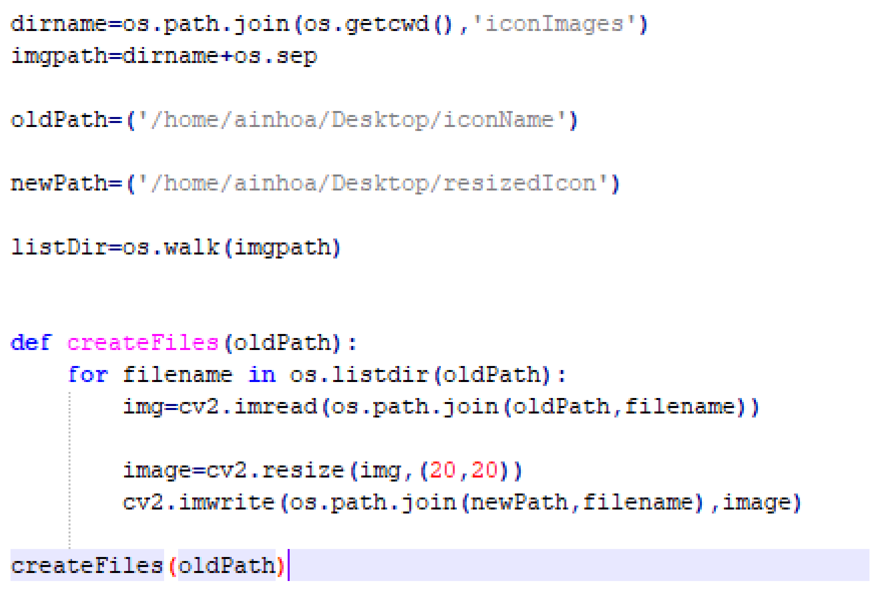
[22] “What is Keras? The deep neural network API explained”, [Online]. Available: https://www.infoworld.com/article/3336192/what-is-keras-the-deep-neural-network-api-explained.html

[23] “OpenCV- About”, [Online]. Available: https://opencv.org/about/

[24] “What is Scikit-Learn?”, [Online]. Available: https://www.codecademy.com/articles/scikit-learn

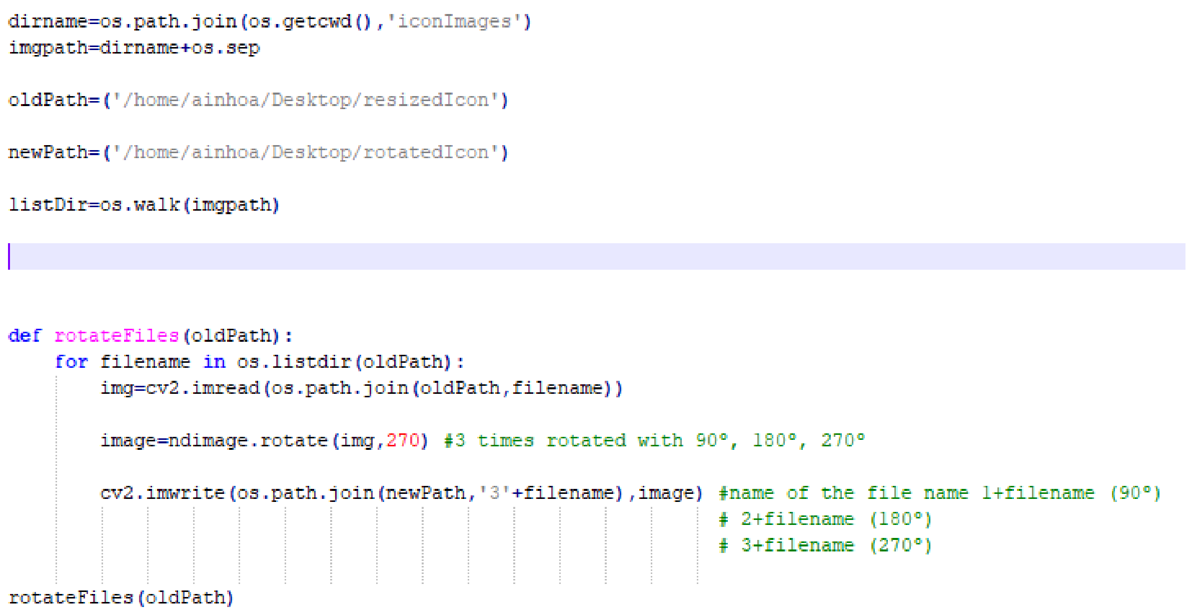
APPENDICES

I. Code for resize icons



```
dirname=os.path.join(os.getcwd(),'iconImages')
imgpath=dirname+os.sep

oldPath=('/home/ainhoa/Desktop/iconName')

newPath=('/home/ainhoa/Desktop/resizedIcon')

listDir=os.walk(imgpath)


def createFiles(oldPath):
    for filename in os.listdir(oldPath):
        img=cv2.imread(os.path.join(oldPath,filename))

        image=cv2.resize(img,(20,20))
        cv2.imwrite(os.path.join(newPath,filename),image)

createFiles(oldPath)
```

II. Code for rotate icons



```
dirname=os.path.join(os.getcwd(),'iconImages')
imgpath=dirname+os.sep

oldPath=('/home/ainhoa/Desktop/resizedIcon')

newPath=('/home/ainhoa/Desktop/rotatedIcon')

listDir=os.walk(imgpath)



def rotateFiles(oldPath):
    for filename in os.listdir(oldPath):
        img=cv2.imread(os.path.join(oldPath,filename))

        image=ndimage.rotate(img,270)  #3 times rotated with 90°, 180°, 270°

        cv2.imwrite(os.path.join(newPath,'3'+filename),image)  #name of the file name 1+filename (90°)
                                                               # 2+filename (180°)
                                                               # 3+filename (270°)

rotateFiles(oldPath)
```

III. Code for flip icons

```
dirname=os.path.join(os.getcwd(),'iconImages')
imgpath=dirname+os.sep

oldPath=('/home/ainhoa/Desktop/rotatedIcon')

newPath=('/home/ainhoa/Desktop/flipIcon')

listDir=os.walk(imgpath)



def flipFiles(oldPath):
    for filename in os.listdir(oldPath):
        img=cv2.imread(os.path.join(oldPath,filename))

        image=cv2.flip(img,0) # values: 0 for flippinf vertically, 1 for flipping horizontally,
                              #  -1 for flipping horizontally andvertically at the same time

        cv2.imwrite(os.path.join(newPath,'v'+filename),image) # names: v (vertically),
                                                              # h (horizontally), hv (both)

flipFiles(oldPath)
```

IV. SVM data load

```
def load_image_files(container_path,dimension=(20,20)):

    image_dir=Path(container_path)
    folders=[directory for directory in image_dir.iterdir() if directory.is_dir()]
    categories=[fo.name for fo in folders]

    descr= " A image clasiffication dataset"
    images=[]
    flat_data=[]
    target=[]
    for i, direc in enumerate(folders):
        for file in direc.iterdir():
            img=skimage.io.imread(file)
            img_resized=resize(img,dimension,anti_aliasing=True,mode='reflect')
            flat_data.append(img_resized.flatten())
            images.append(img_resized)
            target.append(i)
    flat_data=np.array(flat_data)
    target=np.array(target)
    images=np.array(images)

    return Bunch(data=flat_data,target=target,target_names=categories,images=images,DESCR=descr)


image_dataset=load_image_files("./icons/6kimages/8img/")


x_train,x_test,y_train,y_test=train_test_split(image_dataset.data,image_dataset.target,test_size=0.3,random_state=109)
```

V. SVM architecture

```
rng= RandomState(0)


pca=decomposition.MiniBatchSparsePCA(n_components=150,alpha=0.8,n_iter=50,batch_size=3,random_state=rng)
svc=svm.SVC(kernel='rbf',class_weight='balanced')
model=make_pipeline(pca,svc)



param_grid={'svc__C':[1,10,100,1000,10000], 'svc__gamma':[0.0001,0.001,0.01,1,10,100,1000]}

grid=GridSearchCV(model,param_grid)
```

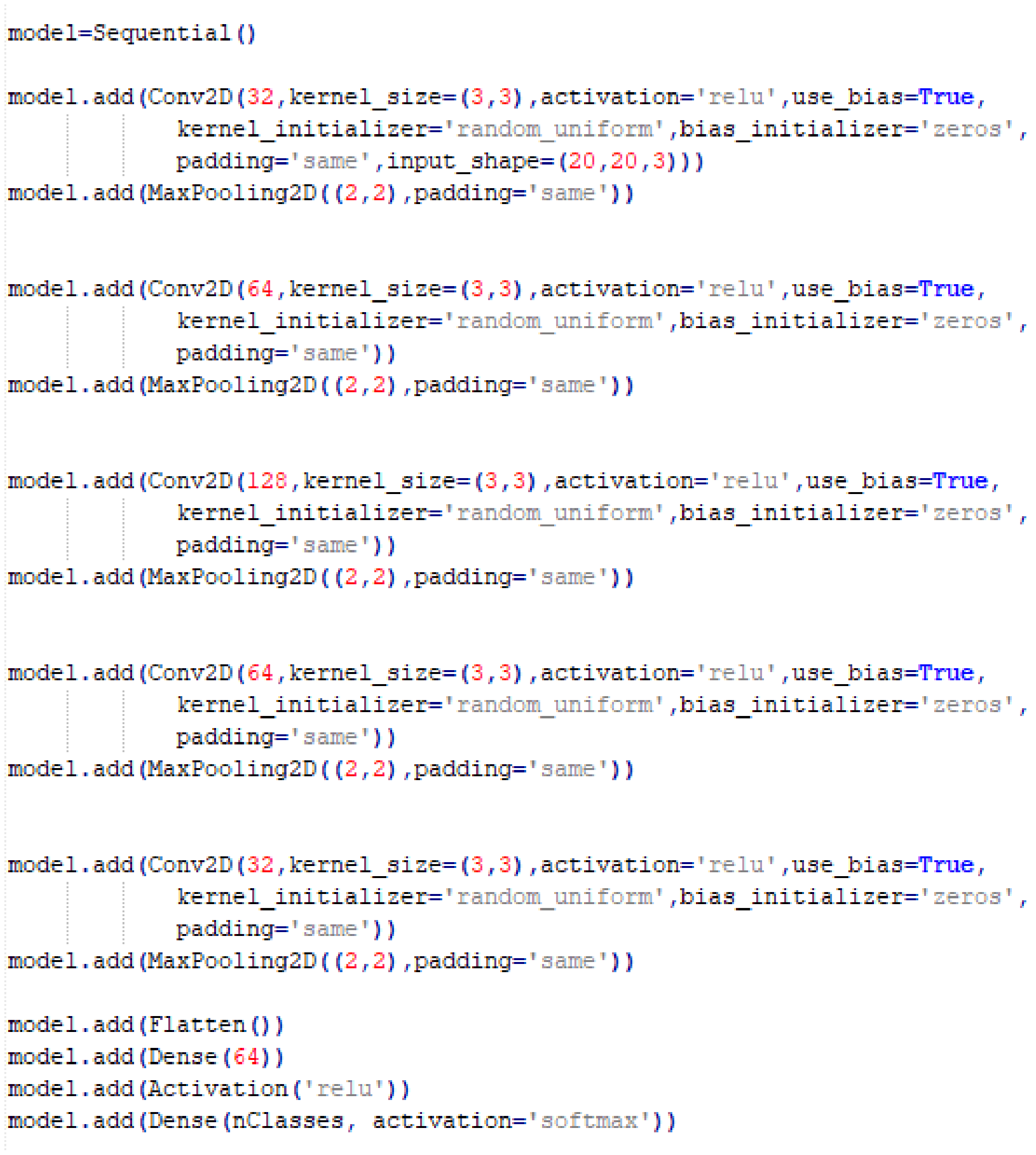
VI. SVM training and testing

```
grid.fit(x_train,y_train)


print(grid.best_params_)



model=grid.best_estimator_


yfit=model.predict(x_test)

print('Accuracy: ',accuracy_score(y_test,yfit))



dirname=os.path.join(os.getcwd(),'./icons/6kimages/8img')
imgpath=dirname+os.sep
nameClasses=(os.listdir(imgpath))


mat=confusion_matrix(y_test,yfit)

sns.heatmap(mat.T, square=True,annot=True,fmt='d',cbar=False,xticklabels=nameClasses,yticklabels=nameClasses)
plt.xlabel('True label6k8')
plt.ylabel('Predicted label6k8');

plt.show()
```

VII. CNN data load

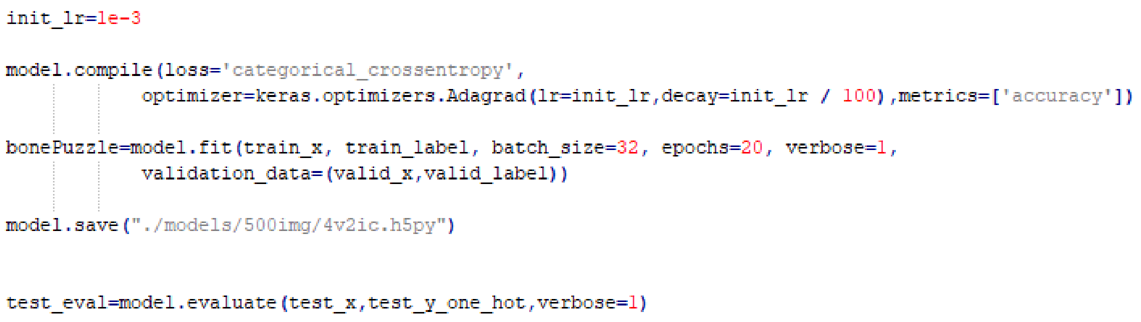
```python
dirname=os.path.join(os.getcwd(),'./icons/500images/4img')
imgpath=dirname+os.sep
listDir=os.walk(imgpath)

images=[]
directories=[]
dircount=[]
prevRoot=''
cant=0
labels=[]
indice=0

for root, dirnames, filenames in (os.walk(imgpath)):


    for filename in filenames:
        filepath=os.path.join(root,filename)
        image=plt.imread(filepath)
        images.append(image)
        cant=cant+1

        b="Reading..." +str(cant)
        print (b,end="\r")

        if((prevRoot!=root) and cant==len(filenames)):

            print(root,cant)

            prevRoot=root
            directories.append(root)
            dircount.append(cant)
            cant=0

for cantidad in dircount:
    for i in range (cantidad):
        labels.append(indice)
    indice=indice+1

y=np.array(labels)
x=np.array(images, dtype=np.uint8)

classes=np.unique(y)
nClasses=len(classes)


train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.2)

train_x=train_x.astype('float32')
test_x=test_x.astype('float')
train_x=train_x / 255.
test_x=test_x / 255.

train_y_one_hot= to_categorical(train_y)
test_y_one_hot=to_categorical(test_y)

train_x,valid_x,train_label,valid_label=train_test_split(train_x,train_y_one_hot,test_size=0.2,random_state=13)
```

VIII. CNN architecture



```
model=Sequential()

model.add(Conv2D(32,kernel_size=(3,3),activation='relu',use_bias=True,
            kernel_initializer='random_uniform',bias_initializer='zeros',
            padding='same',input_shape=(20,20,3)))
model.add(MaxPooling2D((2,2),padding='same'))


model.add(Conv2D(64,kernel_size=(3,3),activation='relu',use_bias=True,
            kernel_initializer='random_uniform',bias_initializer='zeros',
            padding='same'))
model.add(MaxPooling2D((2,2),padding='same'))


model.add(Conv2D(128,kernel_size=(3,3),activation='relu',use_bias=True,
            kernel_initializer='random_uniform',bias_initializer='zeros',
            padding='same'))
model.add(MaxPooling2D((2,2),padding='same'))


model.add(Conv2D(64,kernel_size=(3,3),activation='relu',use_bias=True,
            kernel_initializer='random_uniform',bias_initializer='zeros',
            padding='same'))
model.add(MaxPooling2D((2,2),padding='same'))


model.add(Conv2D(32,kernel_size=(3,3),activation='relu',use_bias=True,
            kernel_initializer='random_uniform',bias_initializer='zeros',
            padding='same'))
model.add(MaxPooling2D((2,2),padding='same'))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dense(nClasses, activation='softmax'))
```

IX. CNN training and testing



```
init_lr=1e-3

model.compile(loss='categorical_crossentropy',
            optimizer=keras.optimizers.Adagrad(lr=init_lr,decay=init_lr / 100),metrics=['accuracy'])

bonePuzzle=model.fit(train_x, train_label, batch_size=32, epochs=20, verbose=1,
            validation_data=(valid_x,valid_label))

model.save("./models/500img/4v2ic.h5py")


test_eval=model.evaluate(test_x,test_y_one_hot,verbose=1)
```