

Master's thesis

Development of a Passport Identification Number Recognition System

Project

Author: Soler Mas, Pablo
Director: Jianwu, Li
Date: June-2018



Beijing Institute of Technology

Abstract

This project is a study about the practical application techniques for the development of an Automatic Passport Identification Number Recognition System.

The system proposed in this work is based on an image classifier developed by supervised learning techniques with convolutional neuronal networks. The basic image processing techniques are studied in this project due to its important role in the character recognition problems.

In this document, it is proposed a system based on a simple convolutional neural network capable of executing the text recognition and successively the segmentation, analysis and classification of the characters; thus registering the number that appears in the image.

For the development of this system it has been used the Matlab software and the Neural Network and Image Processing Toolboxes.

Index

Abstract	1
Index	2
1. Introduction.....	4
1.1. Issue	4
1.2. Target.....	4
1.3. Scope.....	4
2. Introduction to AI	6
2.1. Neural Network.....	6
2.1.1. Neural Network	6
2.1.2. Convolutional Neural Network.....	9
3. Method	14
3.1. System.....	14
3.2. Tools - Matlab	15
3.2.1. Image Processing Toolbox	15
3.2.2. Neural Network Toolbox	16
3.3. Dataset.....	16
3.4. Image Processing	18
3.4.1. Gray scale.....	19
3.4.2. Black and white – Binary	20
3.4.3. Noise	21
3.4.4. Image framing.....	21
3.4.5. Aspect ratio.....	22
3.4.6. Resize	23
3.5. Character segmentation	24
3.6. Neural Network Architecture	25
3.6.1. Image input layer.....	26
3.6.2. 1 st Convolutional Layer	26
3.6.3. 1 st Max Pooling Layer.....	27
3.6.4. 2 nd Convolutional Layer	27
3.6.5. 2 nd Max Pooling Layer.....	27

3.6.6.	3 rd Convolutional Layer.....	28
3.6.7.	Fully-connected Layer.....	28
3.7.	Network training.....	28
3.8.	Validation.....	31
3.8.1.	Neural network validation.....	31
3.8.2.	System validation.....	32
4.	Results.....	33
4.1.	Neural network validation results.....	33
4.2.	System validation results.....	33
5.	Future work.....	36
6.	Conclusions.....	37
7.	Thanks.....	38
8.	Bibliography.....	39
9.	Annex.....	40
9.1.	Training images.....	40
9.1.1.	Convolutional neuronal network training.....	40
9.1.2.	System training.....	42
9.2.	Matlab Code.....	44
9.2.1.	Training code.....	44
9.2.2.	System code.....	48

1. Introduction

The object recognition is a advanced but exciting and fast developing field, which underpins developments in cognate fields such as computer vision, image processing, neural networks and AI. Concerned with the abilities of robots and other AI implementations to recognize various thing and entities, object recognition has created a variety of computer vision product and services from the traditional area of machine inspections to more recent applications such as face recognition. Over the intervening years it has expanded considerably.

The largest companies are researching in this field in a continuous and intense manner, fact that demonstrates the potential of it. Examples of this is the collaboration of Google with the US Department of Defense offering resources for a “pilot project” to analyze drone footage using artificial intelligence or the use of this technology in Uber’s self-driving cars.

1.1.Issue

The passport is an official document used by millions of people every day in different actions and situations. The identification number of the passport is probably the most important and most requested information we can find in this document

Due to that the process of registering this number is repeated millions of times every day around the world and because of being such a repetitive process the probability of making a mistake in the process are significantly high.

On the other hand, also because of the high number of repetitions the human fatigue affects to the process causing a slowdown of it throughout the day what means a considerable waste of time.

1.2.Target

A mistake during registration process of the passport identification number could cause major problems and an increase in waste of time.

In this project it is given a tool in order to help both the agility and optimization of this action by developing a program capable of identifying and registering the identification number from an image of it.

Thus, this project can help both to reduce the time required for this repetitive action and to minimize the mistakes that could derive from human error.

1.3.Scope

The project includes the development of a program able to detect, segment and recognize the passport identification number from an image by using a convolutional

neural network. This program must include the different necessary stages from the original image reading to the output of the passport identification number.

In order to achieve this main goal it is necessary to carry out the partial objectives mentioned below:

- Creation of a dataset which will be used both for the training of the neural network and for the evaluation of this. This dataset needs to contain enough examples for every single letter of the Latin alphabet and every single number, a total of 36 different characters. During the training and evaluation process it is necessary to segment the data between training and evaluation data.
- Definition of a neural network able to recognize each of the characters it is possible to find in the dataset. It is necessary to define a Convolutional Neural Network with the optimal number and size of layers and filters for the specific use of our program.
- Development of a program able to train and evaluate the convolutional neural network successfully. In order to achieve this goal is necessary to analyze the status of the images of the dataset in order to process these images if necessary.
- Development of the final program able to obtain the passport identification number from the original image. This program must contain the next steps:
 - Pre-processing of the original image with the intention of deleting unnecessary information.
 - Detection and segmentation of the possible characters for which identification number is composed.
 - Processing of the image of each of the characters with the purpose of giving them the necessary conditions to be used by the model generated during the training of the convolutional neural network.
 - Determine which number or letter represents each of the characters using the model generated.
 - A detection and segmentation of the characters is necessary due to that the input of the network is a single character. Image processing is needed before and after segmentation.

2. Introduction to AI

Artificial Intelligence (AI) is a branch of computer science dealing with the simulation of intelligence behavior in digital computers. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience.

2.1. Neural Network

2.1.1. Neural Network

Neural networks are a model of machine learning inspired by the biological functioning of the human brain, specifically are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.[8][9][1]

These networks are composed of interconnected neurons that collaborate to generate an output based on the input data of the network. Each of these neurons is a processing unit that receives different input signals, processes this information and generates output information. Each of these neurons has a specific weight that is multiplied by the input information, the sum of these values is used in an activation function and result obtained is the output information of this neuron. On the other hand, a correction or bias factor is usually applied.[10]

Next it is visualized a scheme of the operation. It is clearly observed how the weighted summation of the input data is used as an input for the activation function. This is the process is repeated individually in each of the neurons that make up the neural network.

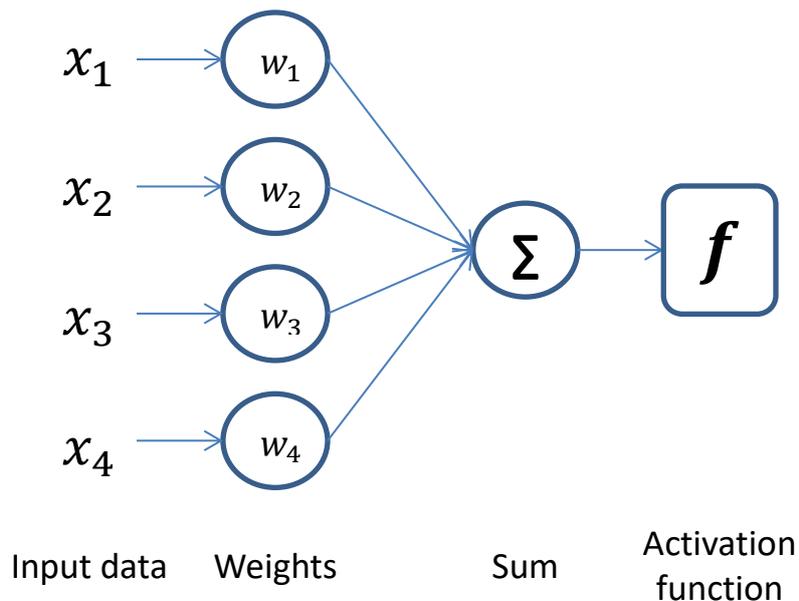


Figure 1. Activation function

$$f(\sum_{i=1}^4 x_i \cdot w_i)$$

The neural network system consists of an ordered set of intermediate layers between the input and the output where each of the layers processes the information received from the previous layer. The first layer of the neural network is called the input layer, and the last one is called the output layer. The layers in between are called hidden layers. The number of layers and the type of function determines the resolution capacity of this neural network.

Layers are a semantic group of nodes. Nodes belonging to one layer are connected to the nodes in the following and/or the previous layers. These connections are weighted edges, and they are referred to as weights.

When all nodes in the previous layer are connected to all nodes in the following layer it is called fully connected layers. Most fully connected layer also includes a bias term to account for the constants in the system.

Next it is visualized a scheme of a neural network where it is possible to observe how there is a weight for every single connection between one neuron from one layer and another neuron from the next layer.

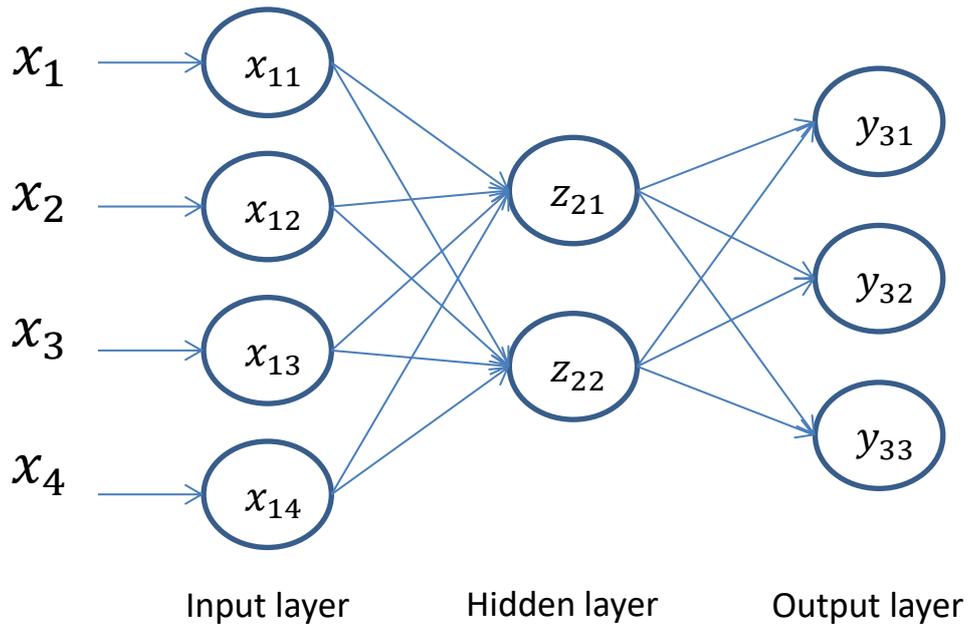


Figure 2. Neural network

2.1.1.1. Backpropagation

Backpropagation is a supervised learning method which allows to determine the optimal weights and biases of neural network. [7][8]

The training process requires a set of examples of proper network behavior, network inputs p and target outputs t . The process of training a neural network involves tuning the values of the weights and biases of the network to optimize network performance. The algorithm is formed by two main steps: feedforward and backforward. [2]

The feedforward process consists on classifying an input image based on the current weight values and comparing this output 'a' with the target output 't'.

The function usually used for feedforwards networks is mean square error; the average squared error between the network outputs a and the target outputs t . It is defined as follows:

$$f = \frac{1}{N} \sum_{i=1}^N (t_i - a_i)^2$$

The backforward process consists on using value of the calculated error to propagate backward in order to modify all the weights of the network that have contributed to the last classification action. These weights receive a portion of the corresponding error based on its influence on this last classification. The algorithm typically used for this process is the gradient descent.

There are two different ways in which training can be implemented: incremental mode and batch mode. In incremental mode, the gradient is computed and the weights are updated after each input is applied to the network. In batch mode, all the inputs in the training set are applied to the network before the weights are updated. [2][11]

2.1.2. Convolutional Neural Network

Convolutional neural networks are a type of neural networks which are specifically designed to solve artificial vision problems such as pattern recognition. However, sometimes it can also be used in order to solve text classification or natural language processing problems.[4][8]

Just as simple neural networks, convolutional neural networks receive an input that is transformed through a series of layers of neurons. However, the entry in the convolutional neural networks is an image represented by a three-dimensional matrix that contains a numerical value that indicates the intensity value of each of the pixels contained in the image. Neural layers are organized in a three-dimensional way and the result of the different transformations made in the network results in a class or category to which the initial image belongs.

In simple neural networks, each neuron is connected to each of the neurons in the next layer, so that the activation function of each neuron uses the output of all the neurons of the previous layer. When processing images it is often preferable to analyze the structure and due to that the weights used in each of the hidden layers are not specific for each relation between an input neuron and an output neuron, but that these weights are used several times with different neurons. A clear example of this interest is the usual correlation between pixels that are located together while this correlation is weaker or non-existent between separate pixels.

There are a few distinct types of Layers. However, in this project only three main types of layer are used to build the Convolutional Neural Network architecture:

- Convolutional layers
- Pooling layers
- Normalization layers

2.1.2.1. Convolutional Layers

The Conv layer is the central building block of a Convolutional Network that does most of the computational heavy lifting. This layer's parameters consist of a set of learnable filters.

Each of the convolutional layer's filters has small spatial dimensions along with and weight but extends through the full depth of the input volume. This denotes that the neurons connect to only a local region of the input volume of the previous layer. The connections are local in space (along width and height), but always full along the entire

depth of the input volume. In addition, the neurons in this layer share the same weights and biases, which considerably reduce the number of network parameters.

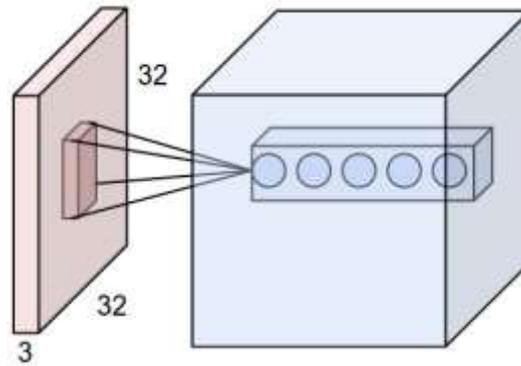


Figure 3. Convolutional layer 1

Graphically these operations can be visualized as a window that runs through the image from the left to the right and from the top to the bottom of map. In this way, the window that runs through the image as well as the weights associated with this window are the same for all the neurons.

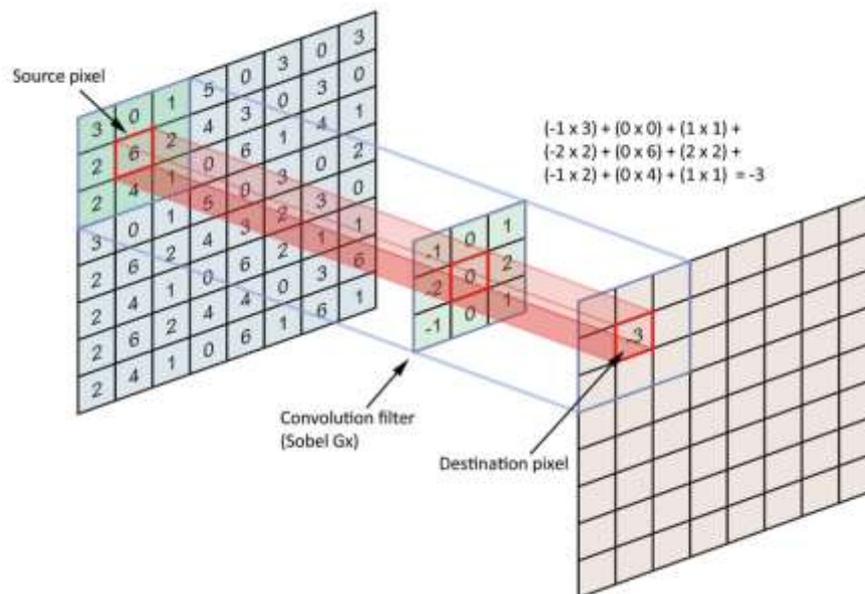


Figure 4. Convolutional layer 2

The results of this operation are various activation maps that allow to identify the presence of different specific characteristics. Both the value of the weights of the shared matrix and the value of the bias determine the characteristic that it is desired to highlight, the combination of both is called filter or kernel and its application to a region of pixels is called convolution.

In the convolutional layers it is usual that there are several filters - weight matrices - to be able to detect more than one characteristic in the image.

Depth, stride and zero-padding are the three parameters of the operation that allow to control the total volume of the output. The two values to control are the spatial size and the depth. The depth of the output volume depends directly on the number of filters and the spatial size depends on the stride and the zero-padding.

➤ **Depth**

The depth of the output volume depends on the number of filters used in the convolutional layer which is the number of neurons that connect to the same region of the input. This parameter determines the number of feature maps.

Knowing that every filter looks for a different characteristic: the greater the amount of characteristics it is wants to highlight, the higher the output volume.

➤ **Stride**

The stride is the value of the filter displacement. If the stride is 1 then the filter will move 1 pixel in each sub-operation. The larger the stride, the lower the number of parameters in the output.

➤ **Padding**

The zero-padding consists of adding zeros around the border. The zero-padding parameter indicates the size of the perimeter you want to add. This parameter allows to control the spatial size of the output: the greater the zero-padding value, the higher the output volume.

➤ **Output volume**

It is possible to compute the spatial size of the output volume as a function of the following parameters:

- n - spatial size of the input volume
- n' – spatial size of the output volume
- f - receptive field size of the Convolutional Layer neurons
- s - Stride with which they are applied
- p - amount of zero padding used on the border

$$n' = \frac{n + 2 \cdot p - f}{s} + 1$$

In order to ensure that the input volume and output volume will have the same size spatially it is generally used the following values.

$$p = \frac{f-1}{2} \quad ; \quad s = 1$$

It is very common to use zero-padding in this way.

2.1.2.2. Pooling Layers

It is common to periodically insert a Pooling layer in-between successive Convolutional layers in a Convolutional Network architecture.

Pooling Layers combine the outputs of neuron clusters at one layer into a single neuron in the next layer. The function of this process is to progressively reduce its spatial size in order to decrease the amount of parameters and computation in the network. This process removes information usually redundant and is often used in order to control the overfitting of the network.

Down-sampling makes it possible to increase the number of filters in deeper convolutional layers without increasing the required amount of computation per layer.

The Pooling Layer operates independently on every depth slice of the input and resizes it spatially. The most common type is max pooling layer, which uses the maximum value from each of cluster of neurons at the prior layer and which most common form is a pooling layer with filters of size 2x2. Another example of Pooling Layer is average pooling, which uses the average value from each of a cluster of neurons at the prior layer.

2.1.2.3. Fully-Connected Layers

The convolutional and down-sampling layers are followed by one or more fully connected layers. As its name suggest neurons in a fully connected layer have full connections to all activations in the preceding layer, as in regular Neural Networks. All the features learned by the preceding layers across the image are combined in order to identify the larger patterns and classify images. Consequently, the output size parameter in the last fully connected layer is equivalent to the amount of classes in the target data.

It is interesting to observe that the only difference between Convolutional Layers and Fully-Connected Layers is that many of the neurons in a Convolutional Layer volume share parameters so these neurons are connected only to a local region in the input. However their functionality is the same.

2.1.2.4. Batch Normalization Layer

In order to accelerate network training and decrease the sensitivity to network initialization it is usual to use Batch Normalization layers and Rectified Linear Unit layers between convolutional layers and nonlinearities.

Batch Normalization Layer first normalizes the activations of each channel by subtracting the mini-batch mean and dividing by the mini-batch standard deviation. Then, the layer shifts the input by a learnable offset β and scales it by a learnable scale factor γ .

$$x'_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma^2 + \epsilon}}$$

$$y_i = \gamma \cdot x'_i + \beta$$

2.1.2.5. Rectified Linear Unit Layer

The batch normalization layer is followed by a nonlinear activation function. The most common activation function is the rectified linear unit (ReLU).

A Rectified Linear Unit layer executes a threshold process to each element of the input, where any value below zero is set to zero.

$$\begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

2.1.2.6. Softmax Layer

After a fully connected layer is used a softmax layer in order to normalize the output of the layer. The output of this layer consists of a set of positive values, the sum of which is equal to 1 and which can be used as classification probabilities by the classification layer.

2.1.2.7. Classification Layer

The final layer used in a classification neural network is the classification layer. This layer is able to attribute one of the exclusive classes for each of the inputs using the probabilities returned by the softmax activation function. Furthermore it is also able of calculating the loss.

3. Method

Throughout this chapter we describe the method used and the process followed during the developed program. Moreover, each of these steps will be explained in detail.

3.1. System

Firstly, it is necessary to observe and understand the main scheme of the system. Understanding the global performance of the system it will be possible to deepen in each of the steps individually.

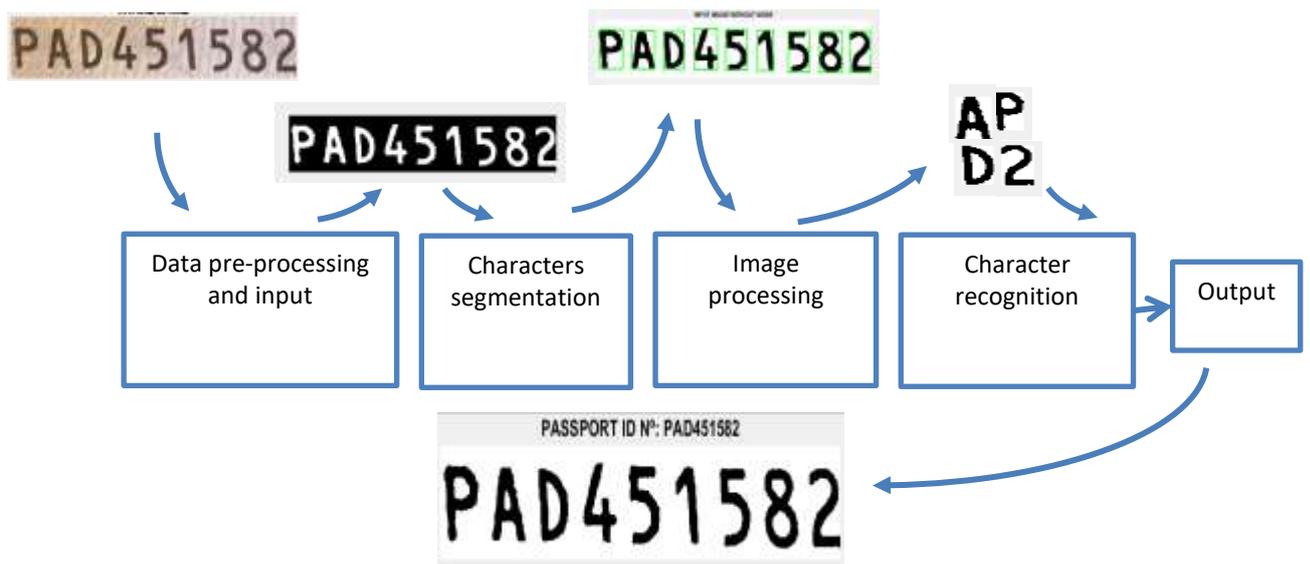


Figure 5. System scheme

The system is composed by 5 main processes mentioned below:

- The first step consist on loading the input data and process it in prepare it for the following operations.
- Following this processed data is segmented character by character due to that the neural network recognizes the characters one by one.
- Before loading the data of these characters it is necessary to process these single images in order to adapt the characteristics of the images to the requirements of the neural network. This step is also performed with the purpose of improving the accuracy of the neural network.
- Once the characters images are processed they are ready to be loaded in the neural network. At the end of this step the output will be sum of the result of every single run for every single character.
- The final step consists on reading and displaying the output that is obtained from the neural network run.

By the other hand, in order to be able to run the neural network successfully it is necessary to train it with the dataset. For this project a dataset has been created from hundreds of passport images obtaining thousands of characters images.

This is essential for the overall performance of the system and it is for this reason that prior to entering training data in the neural network these are processed in a similar way to the processes mentioned above.

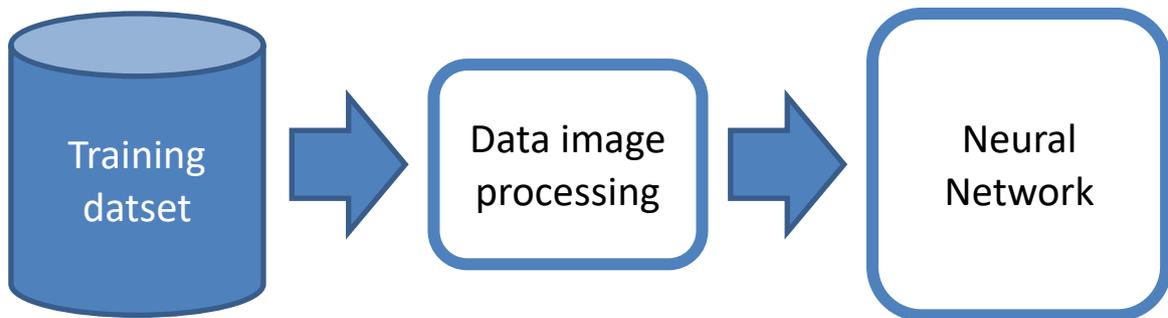


Figure 6. Network training

3.2.Tools - Matlab

The main tool used for the development of this project is the Matlab software.

Matlab is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. Designed for analyzing data, developing and implementing algorithms, creating models and interfacing with programs written in other languages Matlab combines a desktop environment tuned for iterative analysis and design processes with a programming language that expresses matrix and array mathematics directly.

Image Processing and Neural Network are the two main topics of this project. Due to that it is necessary to use the two following specific and complex toolboxes that Matlab provides.

3.2.1. Image Processing Toolbox

The Image Processing Toolbox of MathWorks provides a comprehensive environment to gain insight into your image and video data, develop algorithms, and explore implementation tradeoffs.

MathWorks guarantee that using this toolbox it is possible to execute the following operations:

- Acquire images and video from imaging hardware
- Use graphical tools to visualize and manipulate images and video
- Develop new ideas using libraries of reference-standard algorithms

- Migrate designs to embedded hardware

Throughout this project Image Processing Toolbox it is used to visualize and manipulate the images for both training and operation processes.

3.2.2. Neural Network Toolbox

The Neural Network Toolbox of Mathworks allows to create, train and simulate shallow and deep learning neural networks. This specific toolbox provides algorithms, pre-trained models and apps to create, train, visualize and simulate both shallow and deep neural networks. By using some of this functions or resources it is possible to execute several complex operations such as classification, regression, clustering or dimensionality reduction.

Deep learning networks include convolutional neural networks, directed acyclic graph (network topologies, and autoencoders for image classification, regression, and feature learning. It is possible visualize intermediate layers and activations, modify network architecture and monitor training progress.

Mathworks also provides the possibility of quickly applying learning by performing transfer learning with pre-trained deep network models for small training sets. Some of the most popular pre-trained deep network models that Mathworks provides are resNet-50, GoogLeNet, AlexNet or VGG-16. It is also possible to import models from TensorFlowKeras or Caffe.

Throughout this project Neural Networks Toolbox it is used mainly to generate and train the convolutional neural network. Consequently this toolbox it is used to define the parameters of the network and the training factors and to validate the accuracy of the network.

3.3.Dataset

The effectiveness of a digit recognition system using a convolutional neural network depends on several factors and in no case should the importance of each of these factors be under valued. It is obvious that the architecture of the convolutional neural network is an important factor as the adjustment of this network and it is worth to work on it with the maximum possible effort spending as much time and resources as necessary. However, the set of data used for training the neural network is also a factor that will determine the accuracy of our system.

Due to that it has been decided to generate a dataset using all available resources carefully in order to obtain the largest number of examples and also a high range of variation. In this case, images of passports from different nationalities have been extracted and from these images have been obtained the single characters manually and individually.

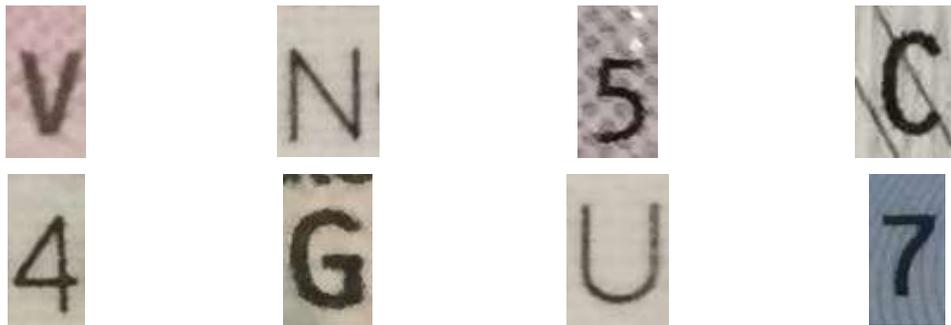


Figure 7. Dataset examples

After this process it has been obtained a dataset with 37 different characters and a total amount of 3.465 character images. Below it is shown the amount of images obtained for each of the characters:

Table 1. Dataset samples

Numbers		Letters	
0	205	a	328
1	129	b	56
2	150	c	69
3	86	d	84
4	76	e	219
5	77	f	46
6	102	g	58
7	69	h	59
8	72	i	144
9	112	j	41
		k	51
		l	105
		m	85
		n	155
		o	103
		p	151
		q	8
		r	157
		s	152
		t	75
		u	78
		v	39
		w	31
		x	31
		y	31
		z	31
		ñ	10

From this total amount of examples some of them have been used for the training process and some of them for the validation process. Due to that for some particular characters the volume of examples is not very high; during the development of this project neural network training process is considered much more important than the validating process. The validation of the convolutional neural network is not prioritized so much, since the validation process of the complete system is considered as the validating priority process. This process it is done using full images of identification passport numbers and executing each of the single steps mentioned in the section 3.1 of this report.

Due to that, only 4 examples of each of the characters are used for the convolutional neural network validation process.

3.4. Image Processing

Image processing is a decisive process for the functionality of the system. This process has several objectives and all of them are really important for the operation of the program. [5][6]

Firstly, it is convenient to modify the characteristics of the images in with the purpose of simplifying the analyze process of the interesting properties particularly interesting for the process of data recognition. It is also convenient to modify the necessary data for this project in order to reduce the weight of these images and thus obtain a faster and more efficient system.

On the other hand, the convolutional neural network defines the necessary characteristics for the input of the network. Because of that it is necessary to provide the images with the necessary properties so that they can be used as input of the convolutional neural network.

These objectives vary depending on whether we talk about images of the database or operating images. In the case of operating images, the different operations are carried out before or after the segmentation process according to the needs but with the dataset images there is not this division of operations.

Below it is shown a summary of the different executed operations and the moment in which they are carried out based on the image in which they are working.

Table 2. Image processing operations

	Dataset Image	Operating Image	
		Before segmentation	After segmentation
Gray Scale	✓	✓	-
Black and White	✓	✓	-
Noise	-	✓	-
Image Framing	✓	-	-
Aspect Ratio	✓	-	✓
Resize	✓	-	✓

For the executions of these operations it is used the Image Processing Toolbox form Matlab Software.

3.4.1. Gray scale

The first step is to convert the image into a more appropriate format in order to extract the characteristics more interesting for the character recognition process. It is considered that in a recognition process the most important data to process is the outline of the character and in order to find and work with this outline data it is not necessary to work with a RGB image.[3]

In the case of the RGB image we find a combination of three intensity maps; one corresponding to each of the three colors red, green and yellow. In each of this intensity each pixel has defined a value for the intensity of this color between 0 is and 255 (2^8 -8 bits). Thereby, in a RGB image we usually find a variety of more than 16 million different colors. This factor not only increases the storage capacity required but also complicates each of the processes to be carried out.

Due to this reasoning it is considered appropriate to convert this initial image from which we start in RGB format into a grayscale image so instead of finding an intensity map of pixels for each of the three colors red, green and blue the image will be formed by one single intensity map of pixels with values from 0 to 255 (2^8).

To execute this RGB to grayscale process, it is common to make a weighted average of the intensity of each of the three colors necessarily being the sum of these weights equivalent to one. Next we can observe the weights usually used by the Matlab software:

$$X= 0.2989 * R + 0.5870 * G + 0.1140 * B$$

By carrying out this process, we will reduce the necessary storage capacity and also reduce the complexity of the operations to be carried out, thus reducing the required processing time.

In Matlab software the process is executed by the following operation:

```
Image = rgb2gray(Image)
```

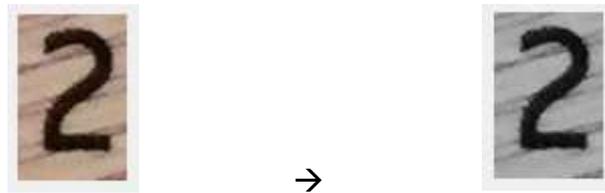


Figure 8. RGB to Gray

3.4.2. Black and white – Binary

The next necessary process consists on creating a binary image from a grayscale image. The main reason to execute this process is the same that was mentioned in the grayscale image section: this is reason is that in order to achieve the character recognition process the most important data to analyze is the outline of the image and more specifically the location of this outline pixels and not the intensity of these values.

The most common process in order to perform this procedure is to select a threshold value and modify the value of each pixel depending on if the value of this pixel is above or below the selected threshold value.

$$bn(x,y) = \begin{cases} 0 & \text{If } I(x,y) < T(x,y) \\ 255 & \text{else} \end{cases}$$

Matlab software offers the possibility of using graythresh function that computes a global threshold that can be used to convert an intensity image to a binary image with. The graythresh function uses Otsu's method, which chooses the threshold to minimize the intraclass variance of the black and white pixels. [3]

In Matlab software the process is executed by the two following operations:

```
threshold = graythresh(Image)  
Image =im2bw(Image,threshold)
```



Figure 9. Bw to Binary

3.4.3. Noise

In many occasions the images show additional information and this data is considered totally unnecessary for the main purpose of character recognition. Due to that it is executed an operation in order to clean the image from small objects.

Matlab software provide the `bwareopen` operation which removes all connected components (objects) that have fewer than defined number of pixels from a binary image, producing another binary image, `BW2`. [3]

This operation is executed the image pre-processing of full passport identification number image. Due to the dimensions of each of these images are different the defined limit value for the elimination of objects varies depending on the operation it is executed.

In Matlab software the process is executed by the following operation:

```
imagen = bwareopen(imagen,100)
```

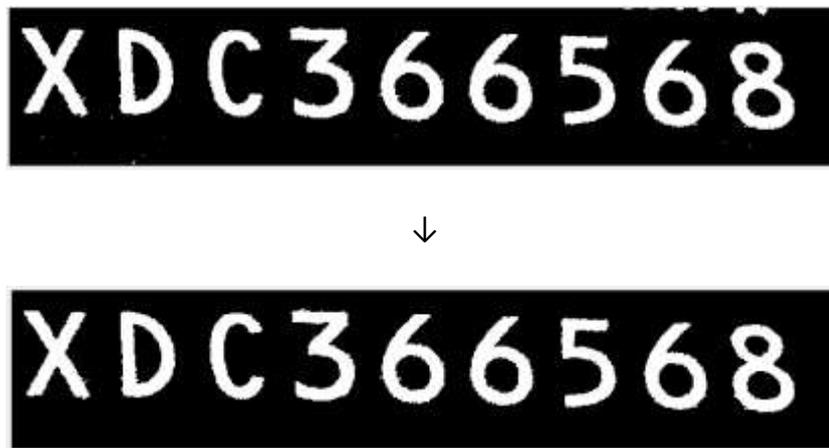


Figure 10. Noise

3.4.4. Image framing

Image framing is an operation that is executed because of similar reasons than last operation. The difference is that this process is used exclusively for the dataset images. Due to that it is certainty known that in each image of the dataset there is a one single character it is possible to look for and select object of the image with the largest area, knowing for sure that all the other objects are noise.

Matlab requires several steps to perform this process. Next, each of these necessary steps is explained:

- First of all, it is necessary to label the image so that each of the objects that are in the image is labeled. In Matlab software it is possible to perform this operation through the function `bwlabel` which labels each of the 8-connected

objects found in the image with a different value starting from 1 value and increasing progressively.

- Second, it is proceed to measure the properties of each of these found objects. Matlab allows performing this operation through the *regionprops* function. The interesting properties for the execution of the global operation are both the area of this objects and the position of them.
- The objective of the next operation is to select the object with the largest area. For this it is necessary to sort the objects according to their area in a descending way and select the first object of this list. In Matlab software it is possible to execute this operation through the *sort* function.
- Finally is executed with the purpose of cropping the image adjusting it to the object of greater area. In the past operations, the object with the largest area and its location has been measured so carrying out this operation is simple. In Matlab software it is possible to execute this operation through the *imcrop* function.

The complete process in Matlab software is executed by the following operations:

```
labeledImage = bwlabel(Image);  
measurements = regionprops(Image, 'BoundingBox',  
    'Area');  
allAreas = [measurements.Area];  
[sortedAreas, sortingIndexes] = sort(allAreas,  
    'descend');  
Bigblob = measurements(sortingIndexes(1)).BoundingBox;  
DLim = imcrop(B, [Bigblob(1) Bigblob(2)  
    Bigblob(3) Bigblob(4)]);
```

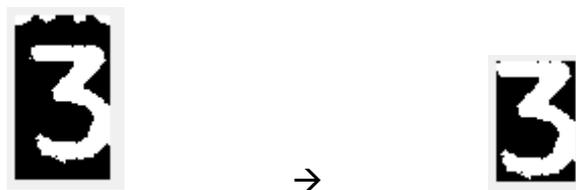


Figure 11. Image framing

3.4.5. Aspect ratio

It is important to maintain the aspect ratio because of that in this project it is executed a sum of operations in order to maintain an aspect ratio. Basically what is performed is to measure the width and the height of the image after the cropping process and add image in the necessary direction. In order to realize this operation it is necessary to use the information it is extracted from the image in the previous operation. Figuring out if

the height value is bigger or smaller than the width value it will be defined in which direction and in what quantity it is necessary to add a matrix of pixels.

To prevent the deformation of the silhouette of the character with this operation generate a characters shapes much more uniform causing also an increase in the efficiency of the system.

The complete process in Matlab software is executed by the following operations:

```
Wblob = Bigblob(3);
Lblob = Bigblob(4);
if Wblob == Lblob
else
    if Wblob > Lblob
        DLim = padarray(DLim,[Wblob-Lblob 0],'pre');
    else
        DLim = padarray(DLim,[0 Lblob-Wblob],'pre');
    end
end
end
```



Figure 12. Aspect ratio

3.4.6. Resize

After the set of operations performed previously, the size of each one of the images can vary significantly. On the other hand, the convolutional neural network has been designed for a characteristic input in which the size of the image is previously defined.

Due to that it is necessary to resize the images before using them in the neural network achieving a uniform size in all of them. In this project it has been defined a very common size in this type of programs of 28x28 pixels.

Performing this operation in Matlab software is a very simple process through the *resize* function. This operation is done as follows.

```
DLim = imresize(DLim,[28,28]);
```

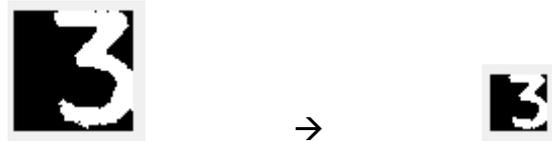


Figure 13. Resize

3.5.Character segmentation

Contours or edges are curves that join sets of contiguous pixels that have the same color or intensity. These curves allow us to locate the borders of the objects in the image, and their detection is fundamental for an artificial vision system to recognize or detect shapes in an image.

As we can see in section 3.4, three image processing operations are performed prior to the characterization of the characters. In addition to the objectives previously mentioned in this chapter, these three operations provide certain characteristics to the images so that the segmentation of the characters is done in a more optimal way. The main objective of this operation is to detect, locate and segment each of the main characters that make up the passport identification number. Because a noise filtering operation has been previously executed, each of the objects detected should be one of these characters.

Due to that, this is the last image processing operation that is performed in the complete image of the passport identification number. In this way this is an operation where the input is a complete pre-processed image of the number and the output is an adjusted image of each of the characters that make up this number.

Matlab requires different steps to perform this operation:

- First of all it is necessary to label the image with the objective of labeling each of the different objects found in the image. In Matlab software this operation is performed using the *bwlabel* function previously explained in section 3.4.3.
- The second step consists on measuring the properties of the found objects. In this case the interesting properties for the desired purpose are the quantity of objects there is in the image and their location. Matlab software allows to perform this operation with the *regionprops* function previously explained in section 3.4.4.
- Finally, it is proceed to cut out each of these found objects using the location measured previously. These images will be saved to be used and analyzed later in the neural network.
- Second, it is proceed to measure the properties of each of these found objects. Matlab allows performing this operation through the *regionprops* function. The interesting properties for the execution of the global operation are both the area of this objects and the position of them.

The complete process in Matlab software is executed by the following operations:

```
[L Ne]=bwlabel(imagen);  
propied=regionprops(L, 'BoundingBox');  
for n=1:Ne  
    [r,c] = find(L==n);  
    n1=imagen(min(r):max(r),min(c):max(c));  
    n1=imresize(n1,[28 28]);  
    imshow(~n1);  
    imwrite(n1,['C:\Users\pablo\Desktop\CNN\Ejemplo1\p  
ost\',num2str(n),'.jpg']);  
    pause(0.5)  
end
```

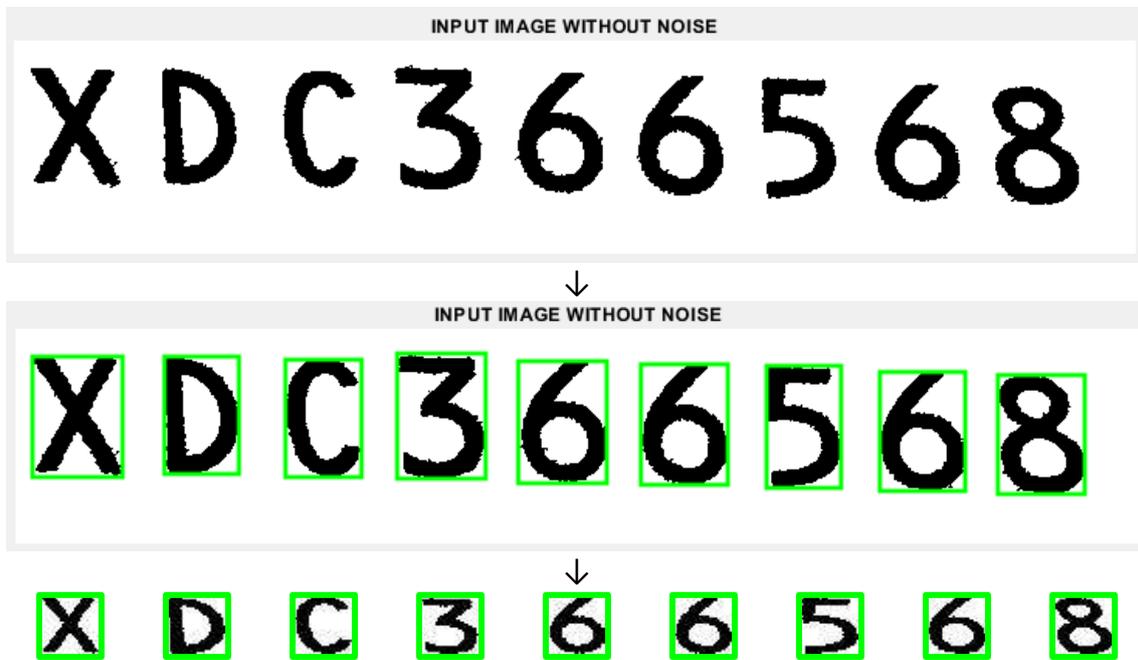


Figure 14. Image segmentation

3.6. Neural Network Architecture

As it is already mentioned in previous chapters, the neural network is one of the fundamental components of the project. This is the responsible of classifying each of the characters images extracted from the complete passport identification number image through the process of segmentation. Previously this neural network requires training through the images of the dataset generated particularly for this project.

This neural network is formed by 7 main layers which can be seen in the following figure:

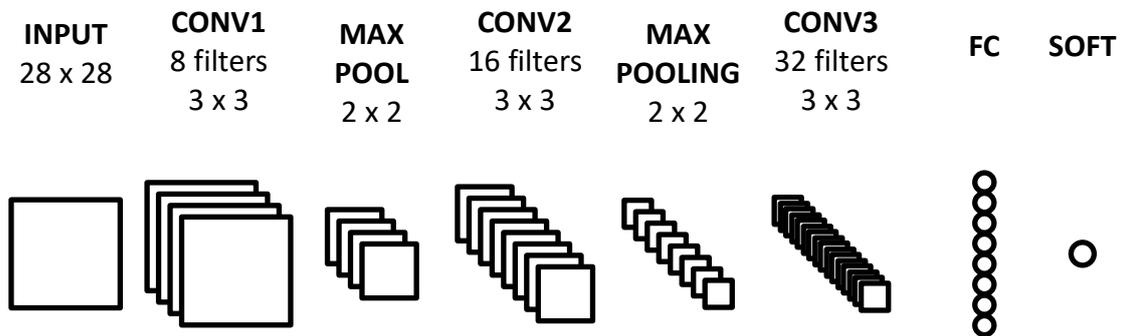


Figure 15. Network architecture

3.6.1. Image input layer

The first layer it has to be necessary an image input layer where it is specified the size for the input image. As mentioned in 3.4 section, after the image processing the size of the single character images is [28 x 28] pixels, a really common size for single character images, and because of the purpose of the network a binary image is the format requested so the channel size is [1]. Due to that it is defined a layer with a specific input size of [28 x 28 x 1]. Knowing the size of the input image it is easy to figure out that 784 neurons will be necessary for this first layer.

```
imageInputLayer([28 28 1])
```

3.6.2. 1st Convolutional Layer

This input layer is followed by the first of the three convolution layers that form this neural network. The first argument that is required is the filter size: in this case this size has a value of 3 what indicates that the filter has a size [3 x 3]. Although in this case it has not been necessary to use it, Matlab allows to define different values of height and width for different filters. Then the number of filters is also defined, where in this case is 8.

As mentioned in the previously zero-padding and stride are also important arguments that will influence in the convolutional layer functionality. For this first layer it is defined a value of 0 for the zero-padding and the stride value is not defined so it is taken a default value of 1. With these values it is ensured that the spatial output size is the same as the input size.

In order to speed up network training and decrease the sensitivity to network initialization it is used Batch Normalization layer and Rectified Linear Unit layer after every convolutional layers.

```
convolution2dLayer(3,8,'Padding',1)
batchNormalizationLayer
reluLayer
```

Knowing the amount of filters and their spatial size it is possible to figure out the amount of required neurons for this layer.

$[28 \times 28 \times 8] = 6.272$ neurons are required

3.6.3. 1st Max Pooling Layer

As mentioned in the section 3.6 convolutional layers are usually followed by Pooling layers in order to reduce the amount of parameters and computation required.

```
maxPooling2dLayer(2,'Stride',2)
```

The first required arguments are the filter size and the stride value. For the first Max Pooling layer it is specified a $[2 \times 2]$ size filter with a stride value of 2. This operation will reduce the amount of parameters to be compute by 50%.

$[14 \times 14 \times 8] = 1.568$ neurons are required

3.6.4. 2nd Convolutional Layer

In the second Convolutional Layer is defined a filter size of $[3 \times 3]$ and an amount of 16 filters. The padding value is 1 and it is used the default value for the stride 1. This ensures that the spatial output size is the same as the input size.

As in the first convolutional layer it is also used Batch Normalization layer and Rectified Linear Unit layer after every convolutional layers with the same purpose.

```
convolution2dLayer(3,16,'Padding',1)
batchNormalizationLayer
reluLayer
```

Although the size of the filter does not vary, as the amount of filters increases a 50% the number of required neurons will also increase by 50%.

$[14 \times 14 \times 16] = 3.136$ neurons are required

3.6.5. 2nd Max Pooling Layer

The second Max Pooling Layer has the same parameters as the first Max Pooling Layer with a filter size of $[2 \times 2]$ and a stride value of 2.

```
maxPooling2dLayer(2,'Stride',2)
```

By having the same values the result is also a decreasing of the amount of parameters of 50%.

$[7 \times 7 \times 16] = 784$ neurons are required

3.6.6. 3rd Convolutional Layer

In the third and last Convolutional Layer is defined a filter size of $[3 \times 3]$ and an amount of 32 filters. The padding value is 1 and it is used the default value for the stride 1. This ensures that the spatial output size is the same as the input size.

As in the previous convolutional layers it is also used Batch Normalization layer and Rectified Linear Unit layer after every convolutional layers with the same purpose.

```
convolution2dLayer(3,32,'Padding',1)
batchNormalizationLayer
reluLayer
```

Although the size of the filter does not vary, as the amount of filters increases a 50% the number of required neurons will also increase by 50%.

$[14 \times 14 \times 16] = 1.568$ neurons are required

3.6.7. Fully-connected Layer

The last step of the convolutional neural network is a Fully-connected Layer with the respective SoftMax and Classification Layers. This last step uses the data from the previous layers in order to attribute one of the exclusive classes and compute the loss.

For this layer is it only necessary to define the amount of different classes it is necessary to classify. For this project there are 10 digits and 26 characters so the total amount of classes is 36.

```
fullyConnectedLayer(36)
softmaxLayer
classificationLayer]
```

3.7. Network training

Once the image processing operations necessary for the input data and the architecture of the neural network have been defined, it is time to think about how this network will be trained.

As previously mentioned, all the operations of both image segmentation and image processing directly affect the training process. On the other hand, the training parameters selected will also straightly affect to the progress of the training procedure.

Due to that it is worthwhile to realize a brief study of the optimal training conditions for this particular convolutional neuronal network and this particular dataset created specifically for this project. The sum of these operations and the selected parameters will determine the accuracy of the neural network and will greatly affect to the accuracy of the overall system.

The first step is to define the amount of data that will be used for the training process and the amount of data that will be used for the validation process. As mentioned in section 3.3, the training dataset has been created specifically in a manual way to carry out this project. Due to this, the number of characters is not excessive and in some cases even scarce. For this reason it is decided to prioritize the training process versus the validation process and it is agreed to use only a quantity of 5 images of each single character for the data validation process and the rest of the images for the training process.

This operation is defined in order to ensure that the images used for the validation process and the images used for the training process are randomly selected every time and the data cannot be used in both processes.

```
numValidationFiles = 5;

[imdsValidation,imdsTrain] =
splitEachLabel(imds,numValidationFiles,'randomize
');
```

The second step consists on defining the specific training parameters for the neural network. The main parameters to be defined are the following:

- SolverName - Solver to use for training the network, in the case of this project specified as 'sgdm'. This solver uses the stochastic gradient descent with momentum (SGDM) optimizer. It is selected the default moment value 0.9 which works well for most of the problems.
- MiniBatchsize - Size of the mini-batch to use for each training iteration. A mini-batch is a subset of the training set that is used to evaluate the gradient of the loss function and update the weights. It is selected the default value for this parameter 128 but due to that the total amount of dataset is reduced it is defined a MiniBatchSize value of 100.
- MaxEpochs - Maximum number of epochs to use for training. An iteration is one step taken in the gradient descent algorithm towards minimizing the loss function using a mini-batch. An epoch is the full pass of the training algorithm over the entire training set. A small study has been carried out with the objective of observe which is the minimum necessary number of epochs in order to finish the network training automatically according to the criteria

explained further on in this same section 'ValidationPatience' . To define the minimum MaxEpochs value it has been taken the maximum value found and multiplied by a safety factor of '1.2'. [12] It has been defined a MaxEpochs value of 32. Next, you can see the values obtained:

Table 3. Epochs

Nº simulation	Epoch finished
1	18
2	23
3	23
4	12
5	26

- ValidationFrequency - Frequency of network validation in number of iterations. The 'ValidationFrequency' value is the number of iterations between evaluations of validation metrics. It is defined a value of 30.
- ValidationPatience - Patience of validation stopping of network training. The 'ValidationPatience' value is the number of times that the loss on the validation set can be larger than or equal to the previously smallest loss before network training stops. It is selected the default value of 5.

```
options = trainingOptions('sgdm', ...
    'MaxEpochs',5, ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

Once both data for the training process as well as the parameters required for the process has been defined the, it is processed the training process of the neural network.

```
numValidationFiles = 5;
[imdsValidation,imdsTrain]=
splitEachLabel(imds,numValidationFiles,'randomize');
options = trainingOptions('sgdm', ...
    'MiniBatchSize',100,...
    'MaxEpochs',32, ...
```

```

'ValidationData', imdsValidation, ...
'ValidationFrequency', 30, ...
'Verbose', false, ...
'Plots', 'training-progress');
labelCount = countEachLabel(imds);
convNN = trainNetwork(imdsTrain, layers, options);
save convNN;

```

3.8.Validation

As previously mentioned, the System is composed of two distinct steps and consequently there are also two different validation processes.

3.8.1. Neural network validation

First, it exists a validation process in which is evaluated the neural network in isolation. As mentioned in the previous section 3.7 this process uses images from the dataset and therefore it is necessary to define the amount of quantity of images that will be used for this first validation process. As also mentioned, a total of 5 images of each character have been selected in order to validate the neural network accuracy. Thanks to the images are labeled this process can be executed automatically figuring out the accuracy of this neural network.

```

YPred = classify(convNN, imdsValidation);
YValidation = imdsValidation.Labels;

accuracy = sum(YPred ==
YValidation)/numel(YValidation)

```

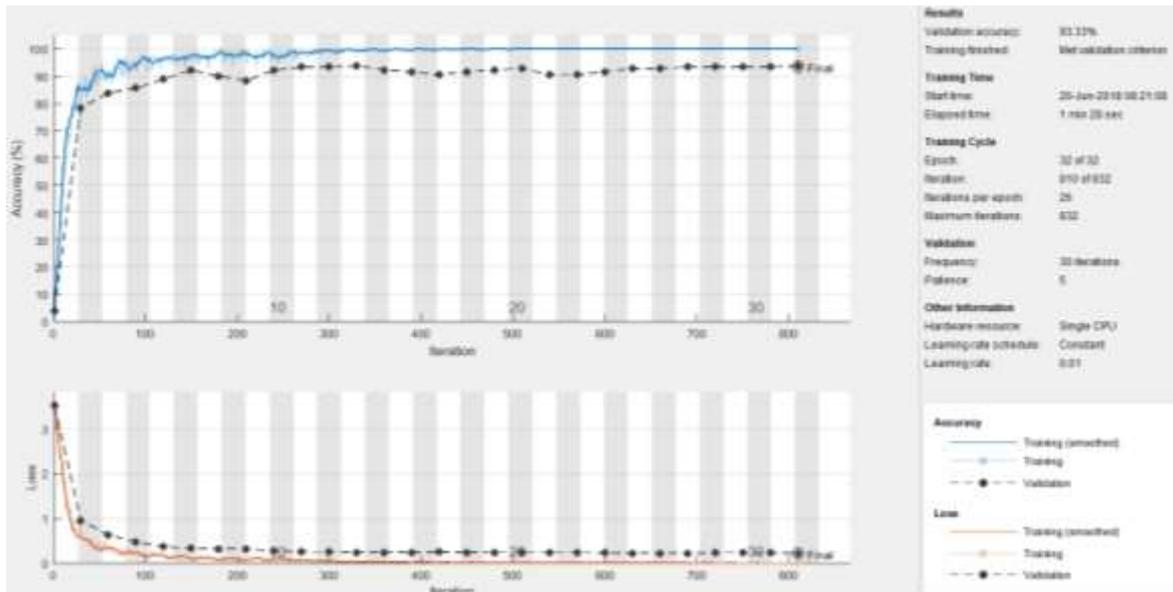


Figure 16. CNN training simulation 1

3.8.2. System validation

Secondly, it is convenient a validation of the complete system where it is verified that each one of the steps works efficiently. These stages are the image pre-processing of the initial image of the complete passport identification number, the process of segmentation, the image processing of the set of individual characters images extracted from the initial image and the analysis of each of these images. This process is done manually with a total of 12 images comparing the result obtained by the system with the real image.

```

YPredd = classify(convNN, imds2);
for i=1:Ne
    if i==1
        resultado=char(YPredd(i));
    else
        resultado = insertAfter(resultado,(i-
1),char(YPredd(i)));
    end
    pause(0.5)
end

```



Figure 17. System training 1

4. Results

4.1. Neural network validation results

In order to carry out a more realistic validation of the results, it is decided to execute a total amount of 5 training simulations with the training parameters previously selected. After these simulations have been obtained the following results:

Table 4. CNN validation results

Nº simulation	Accuracy [%]	Epochs	Elapsed time
1	94,44	18	55s
2	93,89	23	1min 4s
3	92,78	23	1min 5s
4	93,89	12	32s
5	97,78	26	1min 13
Average	94,56	20,40	57,80s

After analyzing the results carefully is it possible to observe an average accuracy of 94,56%. Even not being an optimal value, this average accuracy value is considered as an acceptable value for a first approximation of the system. However, this value indicates that is it possible to implement several operations in order to reach a great improvement of the system.

4.2. System validation results

As mentioned in the previous section 3.8.2, it is decided to perform the validation process of the system by using 12 complete images of the passport identification number. The final results of these simulations show the following data:

Table 5. System validation results

Id number	nº errors	Nº characters	Accuracy
39459860N	0	9	100%
C7VRTLJOJ	0	9	100%
AAI974695	0	9	100%
PAD451582	0	9	100%
PAE712335	0	9	100%
PAD640868	0	9	100%
PAE1558733	0	9	100%
PAE155733	0	9	100%
PAE215764	0	9	100%
XDC366568	0	9	100%
EB3469760	0	9	100%
15CY38292	7	9	22,22%
PAA762906	5	9	44,44%
Total	12	108	88,89%

It is possible to observe that the accuracy of the System in 10 out of 12 images is 100%, what can be considered as an excellent result. However, in 2 of these images is it achieved a low accuracy of 22,2% and 44,4%.

Due to that the number of errors go from a null value to a large value, is it appropriate to suspect that the cause of this accuracy decrease derives from a problem in the image processing of the complete image. Then it is proceed to analyze in detail each of these two cases with low accuracy.



Figure 18. Low accuracy 1

In this first case, it is possible to observe clearly how the image processing is not executed in an effective way, more specifically it is the image binary process, previously commented in section 3.4.2, which is not performed correctly.

Consequently it is perceived how an analysis of this process is necessary and more specifically and analysis of the global threshold value.

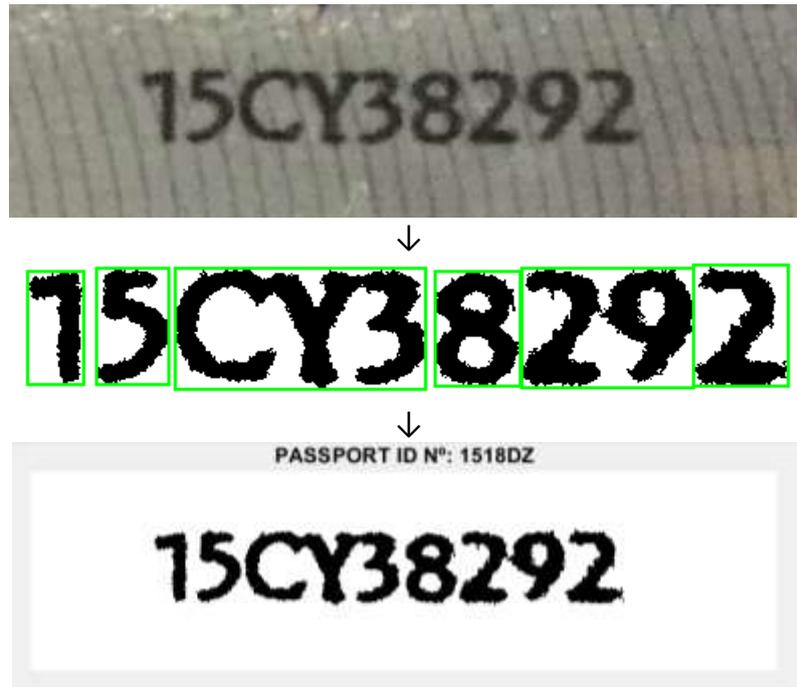


Figure 19. Low accuracy 2

In this second case it is possible to observe how from an image with 9 characters the segmentation process is only able to extract 6 groups of objects. This is due to the fact that the surfaces of the different characters are not clearly separated and therefore they are extracted as a single object. Although the problem lies in the segmentation process, the image processing of the complete image is the operation that must be analyzed and improved in order to achieve a successful extraction of the individual characters. In this case it would be necessary to implement a process that works on the contours of the images in order to clearly mark the outline of each of these objects.

5. Future work

After analyzing the results and the different problems that are generated during the neural network training process as well as the system simulations, it is planned which are the necessary operations to be implemented in the future for the optimization of the system. These measures to be implemented are the following:

- In order to improve the accuracy of the convolutional neural network it would be convenient to implement some operations which work on the outline of the image for a better recognition of the character prior to the network training process.
- With the purpose of increasing the performance of the convolutional neural network it would be opportune to carry out a study of the parameters of the system since with this process it would be possible to obtain distinctive results that would show the number and type of layers required as well as the specific parameters for each of these layers.
- In order to solve the problems in the image segmentation process it would be necessary to implement several image processing operations prior to the segmentation process. Firstly, it would be indispensable to realize a study of the image binary process and its proper threshold with the purpose of avoiding dark areas. Secondly it would be essential to implement some operations which work on the outline of the image for a better segmentation of the different objects.
- The amount of characters of the dataset created specifically for this project is generally scarce and really reduced for some particular characters. Due to that, it would be necessary to increase the volume of the dataset with the purpose of improving the quality of the network validation process and thus its accuracy.

6. Conclusions

After developing a detection, segmentation and text recognition system through the use of a convolutional neural network trained by a database created specifically for this project, the following conclusions are reached:

- Taking into account the complexity of the principal purpose of the project as well as the difficulty of the two main fields of study, neural networks and image processing; it is considered as a success the development of a system capable of operating correctly with captured in good conditions.
- Although the development of the project is considered a success, it is obvious that it is very improvable through the implementation of various operations and the realization of various optimization studies.
- The accuracy of the system with images captured in good conditions is extremely high so by implementing the suggested image processing improvements the accuracy of the system would improve substantially.
- Convolutional neural networks are a specific type of neural networks really useful to solve artificial vision problems. It is a very complex field but totally worth to research on it because of the unimaginable amount and variety of complex problems that can be solved on it.
- Matlab is a very intuitive mathematical tool. By using the Image Processing and Neural Network Toolboxes it is really powerful and allows to create, train and simulate deep learning neural networks in addition to provides a comprehensive environment to gain insight into your image and video data and develop algorithms. Mathworks also provides algorithms, pre-trained models and apps to create, train, visualize and simulate the neural networks as well as large amount of information, discussion forums, examples and tutorials in order to simplify the learning process of everything that may be necessary for the use of the software and the development of programs and systems. Definitely a very good option for the development of neural networks.

7. Thanks

I appreciate the help and support of Dr. Li Jianwu. Thanks to his experience and knowledge on the subject, I have been able to carry out this project and get into this interesting and incredible field of study.

Finally, I greatly appreciate the unconditional support of my friends and family.

8. Bibliography

- [1] MATHWORKS. Neural Network Toolbox Getting Started Guide. 2018.
[https://www.mathworks.com/help/pdf_doc/nnet/index.html]
- [2] MATHWORKS. Neural Network Toolbox User's Guide. 2018.
[https://www.mathworks.com/help/pdf_doc/nnet/index.html]
- [3] MATHWORKS. Image Processing Toolbox User's Guide 2018.
[https://www.mathworks.com/help/pdf_doc/images/index.html]
- [4] STANFORD. Stanford Convolutional Neural Networks for Visual Recognition class notes.
[<http://cs231n.github.io/>]
- [5] GUILLERMO SAPIRO. Image and Video Processing.
[<https://www.coursera.org/learn/image-processing/home/welcome>]
- [6] GONZALEZ & WOODS. Digital Image processing. 3rd edition 2011.
- [7] Neural Networks and Deep Learning free online book.
[<http://neuralnetworksanddeeplearning.com/>]
- [8] ANDREW Ng. Machine learning.
[<https://www.coursera.org/learn/machine-learning/home/welcome>]
- [9] ANDREW Ng & BONEH. STANFORD. Machine Learning.
[<http://cs229.stanford.edu/>]
- [10] DEEPLARNING4J. Introduction to Deep Neural Networks.
[<https://deeplearning4j.org/neuralnet-overview#>]
- [11] SERGEY & SZEGEDY. Batch normalization: Accelerating deep network training by reducing internal covariate shift." *preprint, arXiv:1502.03167* (2015).
- [12] SAGAR SHARMA. Epoch vs Batch Size vs Iterations
[<https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9>]

9. Annex

9.1. Training images

9.1.1. Convolutional neuronal network training

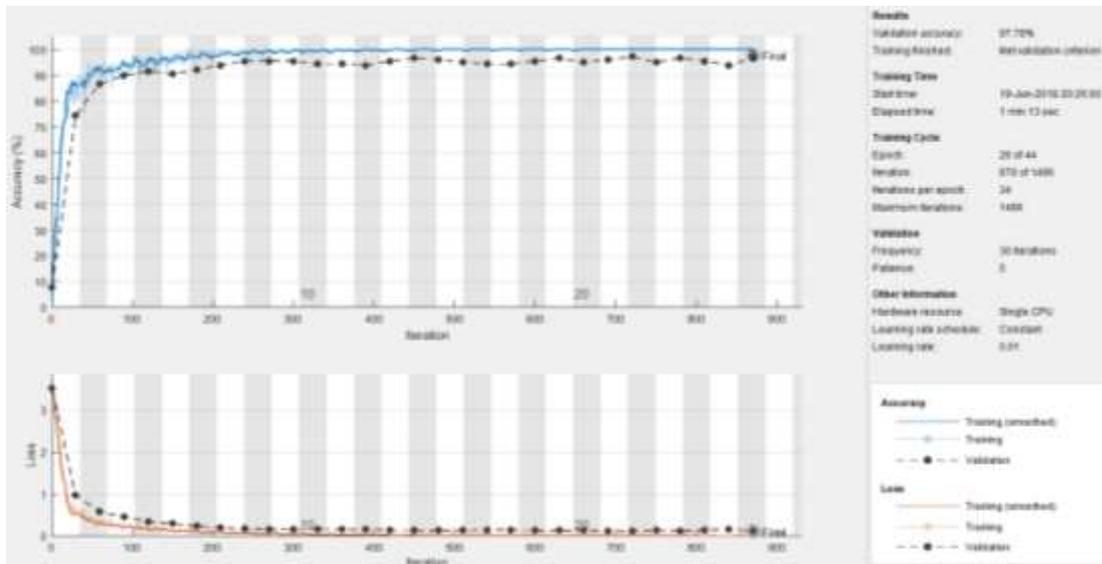


Figure 20. CNN training simulation 1

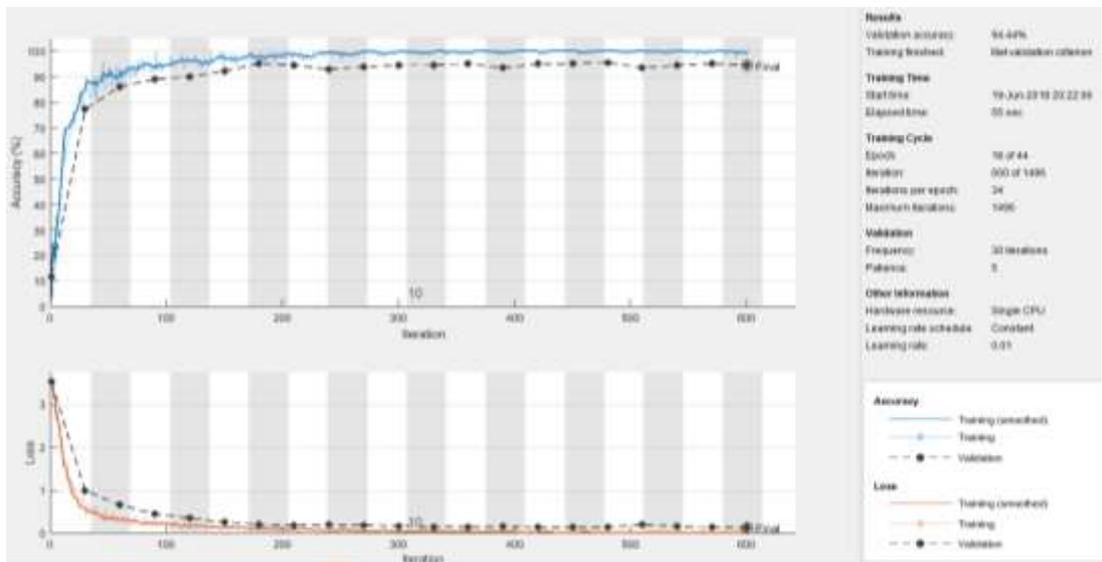


Figure 21. CNN training simulation 2

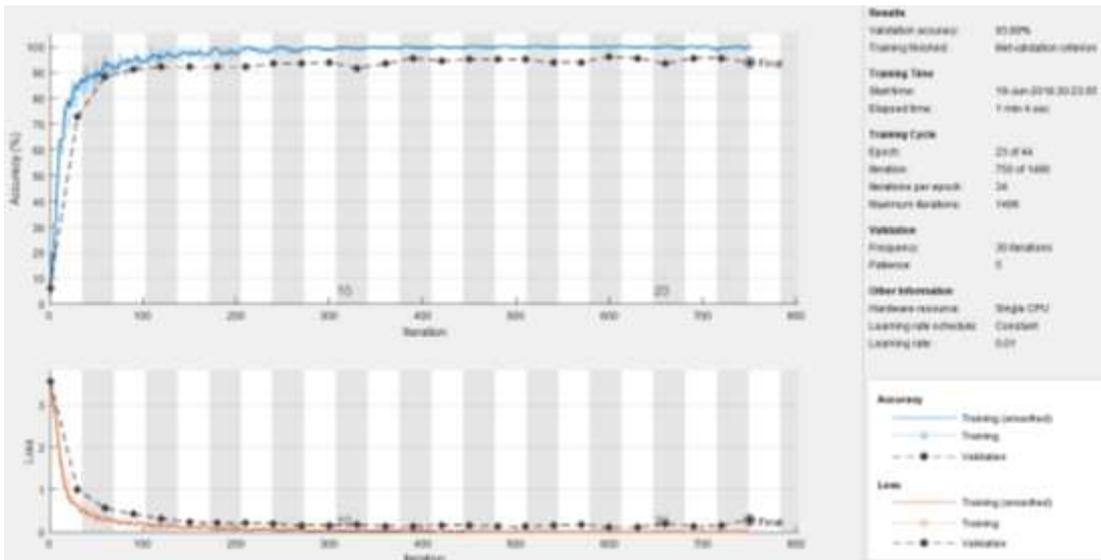


Figure 22. CNN training simulation 3

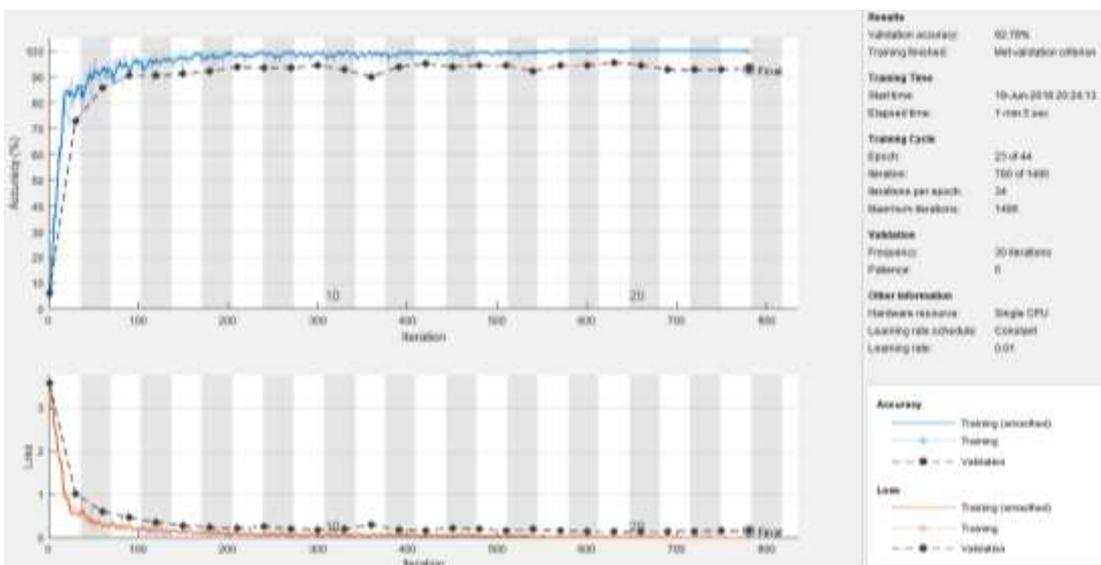


Figure 23. CNN training simulation 4

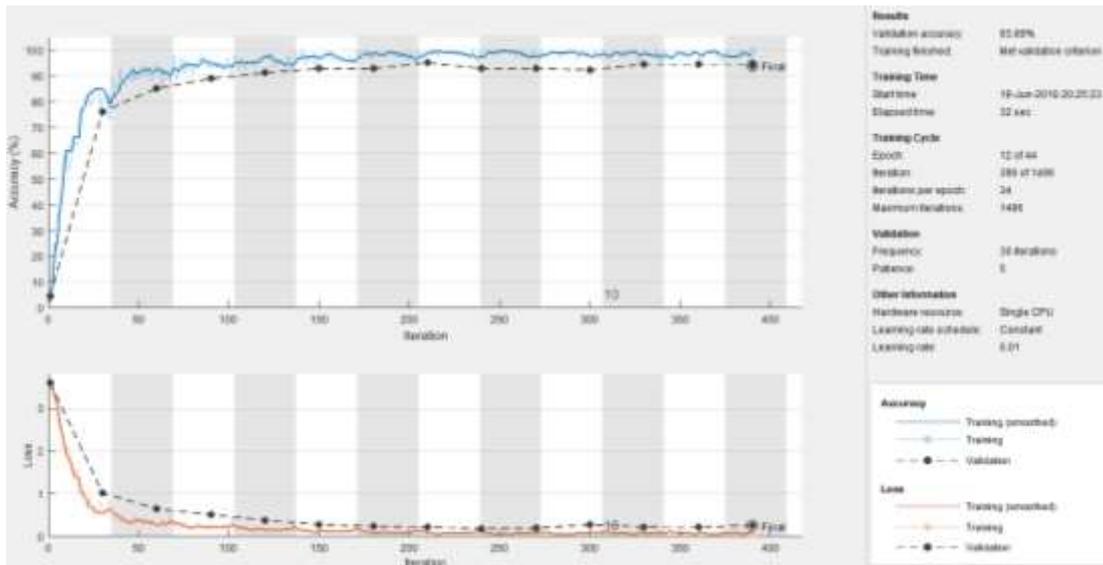


Figure 24. CNN training simulation

9.1.2. System training



Figure 25. System training 1



Figure 26. System training 2



Figure 27. System training 3



Figure 28. System training 4



Figure 29. System training 5



Figure 30. System training 6



Figure 31. System training 7



Figure 32. System training 8



Figure 33. System training 9

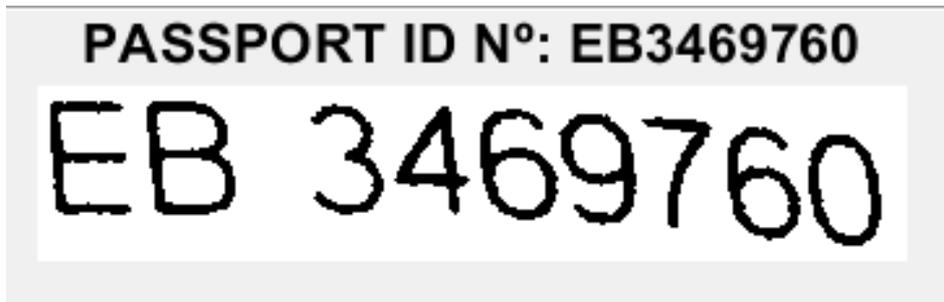


Figure 34. System training 9



Figure 35. System training 10



Figure 36. System training 11

9.2. Matlab Code

9.2.1. Training code

```
clear all;
```

```
close all;
```

```
%%
```

```
digitDatasetPath2 = fullfile('C:\Users\pablo\Desktop\Imagenes passport\todo22');
```

```

imds =
imageDatastore(digitDatasetPath2,'IncludeSubfolders',true,'LabelSource','foldernames
');

%%

figure(3);
imshow(preview(imds));

%%

figure(4);
perm2 = randperm(3000,20);
for i = 1:20
    subplot(4,5,i);
    imshow(imds.Files{perm2(i)});
end

%%

layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding',1)
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding',1)
    batchNormalizationLayer
    reluLayer

```

```
maxPooling2dLayer(2,'Stride',2)
```

```
convolution2dLayer(3,32,'Padding',1)
```

```
batchNormalizationLayer
```

```
reluLayer
```

```
fullyConnectedLayer(36)
```

```
softmaxLayer
```

```
classificationLayer];
```

```
%%
```

```
while hasdata(imds)
```

```
[B,info] = read(imds);
```

```
if size(B,3)==3
```

```
    B=rgb2gray(B);
```

```
    B=imresize(B,[28,28]);
```

```
    threshold = graythresh(B);
```

```
    B = ~im2bw(B,threshold);
```

```
    B = bwareaopen(B,30);
```

```
    labeledImage = bwlabel(B);
```

```
    measurements = regionprops(B, 'BoundingBox', 'Area');
```

```
    allAreas = [measurements.Area];
```

```
    [sortedAreas, sortingIndexes] = sort(allAreas, 'descend');
```

```
    Bigblob = measurements(sortingIndexes(1)).BoundingBox;
```

```
    DLim = imcrop(B,[Bigblob(1) Bigblob(2) Bigblob(3) Bigblob(4)]);
```

```
    Wblob = Bigblob(3);
```

```
    Lblob = Bigblob(4);
```

```

if Wblob == Lblob
else
    if Wblob > Lblob
        DLim = padarray(DLim,[Wblob-Lblob 0],'pre');
    else
        DLim = padarray(DLim,[0 Lblob-Wblob],'pre');
    end
end

DLim = imresize(DLim,[28,28]);
B = bwareaopen(B,30);
imwrite(DLim,info.FileName);
end
end

numValidationFiles = 5;
[imdsValidation,imdsTrain] = splitEachLabel(imds,numValidationFiles,'randomize');
options = trainingOptions('sgdm', ...
    'MiniBatchSize',100,...
    'MaxEpochs',32, ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');

%%
labelCount = countEachLabel(imds);
convNN = trainNetwork(imdsTrain,layers,options);
save convNN;

```

```
%%
```

```
YPred = classify(convNN,imdsValidation);
```

```
YValidation = imdsValidation.Labels;
```

```
accuracy = sum(YPred == YValidation)/numel(YValidation)
```

9.2.2. System code

```
clear all;
```

```
close all;
```

```
%%
```

```
imagen=imread('1.jpg');
```

```
sp=10;
```

```
delete('C:\Users\pablo\Desktop\CNN\Ejemplo1\post\*.jpg')
```

```
load convNN;
```

```
%%
```

```
figure(1)
```

```
imshow(imagen);
```

```
title('INPUT IMAGE WITH NOISE')
```

```
%%
```

```
if size(imagen,3)==3
```

```
    imagen=rgb2gray(imagen);
```

```
end
```

```
%%
```

```
threshold = graythresh(imagen);
```

```
imagen =~im2bw(imagen,threshold);
```

```
figure(11);
```

```
imshow(imagen);
```

```

%%
imagen = bwareaopen(imagen,150);
pause(1)
figure(111);
imshow(imagen);
%%
figure(2)
imshow(~imagen);
title('INPUT IMAGE WITHOUT NOISE')
%%
[L Ne]=bwlabel(imagen);
%%
propied=regionprops(L,'BoundingBox');
%%
hold on
for n=1:size(propied,1)
    rectangle('Position',propied(n).BoundingBox,'EdgeColor','g','LineWidth',2)
end
hold off
pause (1)
%%
figure
for n=1:Ne
    [r,c] = find(L==n);
    n1=imagen(min(r):max(r),min(c):max(c));
    n1=imresize(n1,[28 28]);
    imshow(~n1);
    imwrite(n1,['C:\Users\pablo\Desktop\CNN\Ejemplo1\post\',num2str(n),'.jpg']);
    pause(0.5)
end

```

```
end
```

```
digitDatasetPath2 = fullfile('C:\Users\pablo\Desktop\CNN\Ejemplo1\post\');
```

```
imds2 =
```

```
imageDatastore(digitDatasetPath2,'IncludeSubfolders',true,'LabelSource','foldernames  
'');
```

```
YPredd = classify(convNN,imds2);
```

```
for i=1:Ne
```

```
    if i==1
```

```
        resultado=char(YPredd(i));
```

```
    else
```

```
        resultado = insertAfter(resultado,(i-1),char(YPredd(i)));
```

```
    end
```

```
    pause(0.5)
```

```
end
```

```
%%
```

```
figure(99)
```

```
imshow(~imagen);
```

```
title(['PASSPORT ID Nº: ',resultado],'FontSize',16);
```