Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Sistemas Computacionales

Traffic Sign Detection and Recognition with Voice Assistant

TRABAJO RECEPCIONAL que para obtener el **GRADO** de **MAESTRO EN SISTEMAS COMPUTACIONALES**

Presenta: **CARLOS SINUHÉ VÁZQUEZ ESPINOZA**

Asesor **M.C. VÍCTOR HUGO MARTÍNEZ SÁNCHEZ**

Tlaquepaque, Jalisco. Mayo de 2023.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Acronym	**Description**
AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
COCO	Common Object in Context
CPU	Computer Processor Unit
CSV	Comma Separated Value
DL	Deep Learning
FCNN	Fully-Convolutional Neural Network
FPS	Frames per Second
GPU	Graphic Processor Unit
HOG	Histogram of Oriented Gradients
HSV	Hue Saturated Values
NN	Neural Networks
OS	Operative System
RCNN	Region proposal Convolutional Neural Network
RELU	Rectified Linear Unit
RGB	Red, Green, Blue color space
ROI	Region Of Interest
SCT	Secretaria de Comunicaciones y Transportes
SVM	Support Vector Machine
TANH	Hyperbolic Tangent
VOC	Visual Object Classes

1. INTRODUCTION

There are multitude of applications for detection and recognition of images across different fields. There are some specific applications for these systems used to help people to drive for example in autonomous driving as well as other applications. There has been another focus in the use of classification models used to help drivers providing details about their surrounding while driving. In places like Guadalajara, such models are a valuable tool to reduce traffic accidents.

This document will explain the development of a detection and recognition of traffic signs model. This model has the intention of providing details about the meaning of the traffic signs. All this will happen close to real time and will be an additional information to the driver. This whole system could be used by anyone but specifically aimed to people with visual deficiencies.

With the use of a robust machine learning and the use of Deep Learning (DL), the expectative is to achieve high accuracy levels on the traffic sign detection and recognition. This system is expected to be available and affordable for most of the drivers in Guadalajara.

For the development of this model, we will use a dataset containing thousands of images of traffic signs coming from around the world. All this data must be prepared before used to have the most relevant information from it to finally use it for training of our model.

This document will be presented as a series of sections starting with the Introduction that explains the context of this research, the justification behind it, as well as the problems this is intended to solve and the objectives to be met after this research. Then we have State of Art, where different approaches used in the latest years for image detection and recognition are reviewed and compared. In next section called Theory Framework we will introduce a more in depth of the bases of the methodology and tools used.

Section 4 will focus on the obtention of a data set suitable as well as the process this will require so it can be ready to be used to successfully train the model. Section 5 will show the execution of the model previously developed and trained so we can see the results and tests done on this model. Finally, section 6 will contain the conclusions about the positive impact this project will have in the society and the impact this research has for scientific community and how this could be taken to a possible product.

1.1. Background

Nowadays, there are many Convolutional Neural Network (CNN) applications, including detection & recognition of different objects. There are also some other researches specially focused on detection and recognition of objects in images. In machine learning, there are some technics like DL that are now available because of computational resources and capacities. It’s a fact that there are different ways to predict what an object in an image is like statistic approaches as well as other based on machine learning. In Machine learning we are trying to tell the computer to execute some tasks without specifically programming them to do that, but they are indeed learning and improving with time.

Researchers like M. Swathi and K. V. Suresh [1] have demonstrated different accuracy levels when trying to positively determine the meaning of any traffic sign in front of a driver in real time specially working with traffic signs having triangular and circular shapes and limited in the way to display or show the recognized information to the driver.

Other attempts, like P. Dhar, M. Z. Abedin, T. Biswas and A. Datta [2] have shown high accuracy levels by using technics like color space modifications, determining the Region of Interest (ROI) and then using CNN but limited only to a single traffic sign shape and under controlled scenarios.

Only these couple of investigations showed previously in this section, has left a series of challenges and other areas of improvement that must be solved for the whole process of detection and recognition of images.

Therefore, there is a big area of opportunity to be filled. This could be achieved with the development of a whole system that has high accuracy, use of deep learning as well as latest methodologies for detection and image recognition with the use of machine learning and an accurate and useful voice assistant.

1.2. Justification

The development of this detection and recognition system is focused primary on people with some degree of vision impairment. There are different health conditions that cause a deficiency or loss of visual capabilities that can have a considerable impact on driving performance as mentioned by Cynthia Owsley & Gerald McGwiun Jr in Vision Impairment and Driving [3]. The main diseases can include cataract, glaucoma, diabetic retinopathy, age-related macular, etc.

Therefore, having a second visual detection and recognition system on board, could increase drivers and pedestrian's safety. These systems, plus the addition of a voice assistant to efficiently transmit the traffic signs information to the driver is meant to reduce traffic accidents.

Not only people with a visual-related health condition could be benefited by this tool, but other users that could not be paying enough attention to their surroundings while driving, new drivers in a big city that could be overwhelmed by following their map applications, being aware of the traffic etc.

This proposed detection and recognition system with voice assistant is intended to be available and affordable. Any safety feature like this, could be found in luxury brands for a high price that is out of budget for average citizens.

This project is intended to be implemented in the Guadalajara's downtown on main roads like Periférico and Lopez Mateos, where is a lot of traffic at rush hours and constant traffic all time. These main roads are suffering with the constant traffic incidents that causes traffic jams affecting a lot of drivers.

Due these streets mentioned above have some high-speed limits compared to the rest in the city, the distance between the drivers and the traffic signs in front of them gets close in a short time. This lets drivers a little time to read and understand the traffic signs before they are behind or too close.

1.3. Problem

Guadalajara, Jalisco, being the second biggest city of Mexico, has shown a constant rate of vehicle collisions. Only Jalisco, as informed by the *Instituto de Información Estadística y Geográfica de Jalisco*, was the top 1 of Mexico's states regarding fatal traffic accident with 351 casualties in 2018 [4]. This document also shows an increase from 2015 to 2018 in the number of injured people involved in vehicle accidents.

The reasons behind constant traffic accident are diverse. They can include factors like, lack of attention of drivers, some degree of visual impairment and ignorance of the precise meaning for most of the traffic signs distributed across the roads.

Other causes could be the increasing number of foreign drivers coming into the city for diverse reasons, that, while driving across the city are inevitable, under considerable stress due the amount of traffic signs that can be overwhelming and not letting the drivers to identify every one of this traffic signs and, therefore leading to possible collisions and other accidents.

This project will be mainly oriented to development of a full model for the detection and recognition of the traffic signs and posteriorly added into a mobile application for use in devices such smartphones.

Increment in vehicles but not increase in the road sizes means a high probability for traffic jams with any minor traffic accident which can affect a lot of drivers for hours, which causes a considerable number of negative effects and economic losses.

1.4. Hypothesis

Using a good image detection method, combined with a CNN for the correct traffic sign classification and recognition within Guadalajara and the rest of cities in Mexico, with the addition of a voice assistant to properly communicate the information to the driver, it is pretended to reduce the risk of traffic accidents that are affecting not only drivers but pedestrians too.

The use of a CNN complemented with the use of deep learning is expected to return accurate levels of identification for the kind of traffic signs used in the city. This will also be based in a customized data set with recent images of traffic signs seen in the streets and roads of Guadalajara.

1.5. Objective

1.5.1. General Objective:

Development of a detection and recognition system, for traffic signs, expecting to achieve a high accuracy level for the detection and classification of these traffic signs compared with other implementations made by other researchers in the past, plus, deploying the information to the driver using a voice assistant, while keeping it as friendly and understandable to the user as possible and also requiring minimum effort to be configured by the user.

1.5.2. Specific Objective:

- Getting a full data set with traffic signs that includes the used in Mexico as well as complement it with images of a little damaged traffic signs which unfortunately is common in this region.
- Cleaning and filtering of data set to extract the most relevant images as well as have a balanced number of them.
- Building a robust CNN that could be ready to consume the data set and be trained.
- Training of the model using diverse technics in order to get the best hyperparameters with a model able to get high accuracy and at same time not being too complex.
- To validate, test and compare the model against models made by other researchers.
- Implementing the model in a low-cost device or using smartphone resources using a mobile application.

1.6. Scientific or technologic share/innovation

There have been other approaches to the development of detection and recognition systems in last years as the one seen in *Traffic Sign Detection and Recognition using a CNN Ensemble* [5], where is mentioned the idea of adding a way to pass the information coming from their system to the driver.

There is no product currently available for the masses, that includes a complete image detection and recognition system and a voice assistant yet in the market with a proofed high accuracy that could be added as a module to any vehicle.

2. STATE OF ART OR THE TECHNIQUE

In later years there have been a lot of interest in the detection and recognition of images for diverse purposes, including Advance Driving Assistance System (ADAS) [1]. Within this kind of driving assistance systems, there are also some interests in traffic sign detection and recognition, and this hasn't proven having a simple and definitive solution, but a lot of different attempts and experiments has been done.

The development of a traffic and sign detection and recognition system requires a set of steps in order to achieve the intended goals with a high and acceptable accuracy, enough to have a practical application.

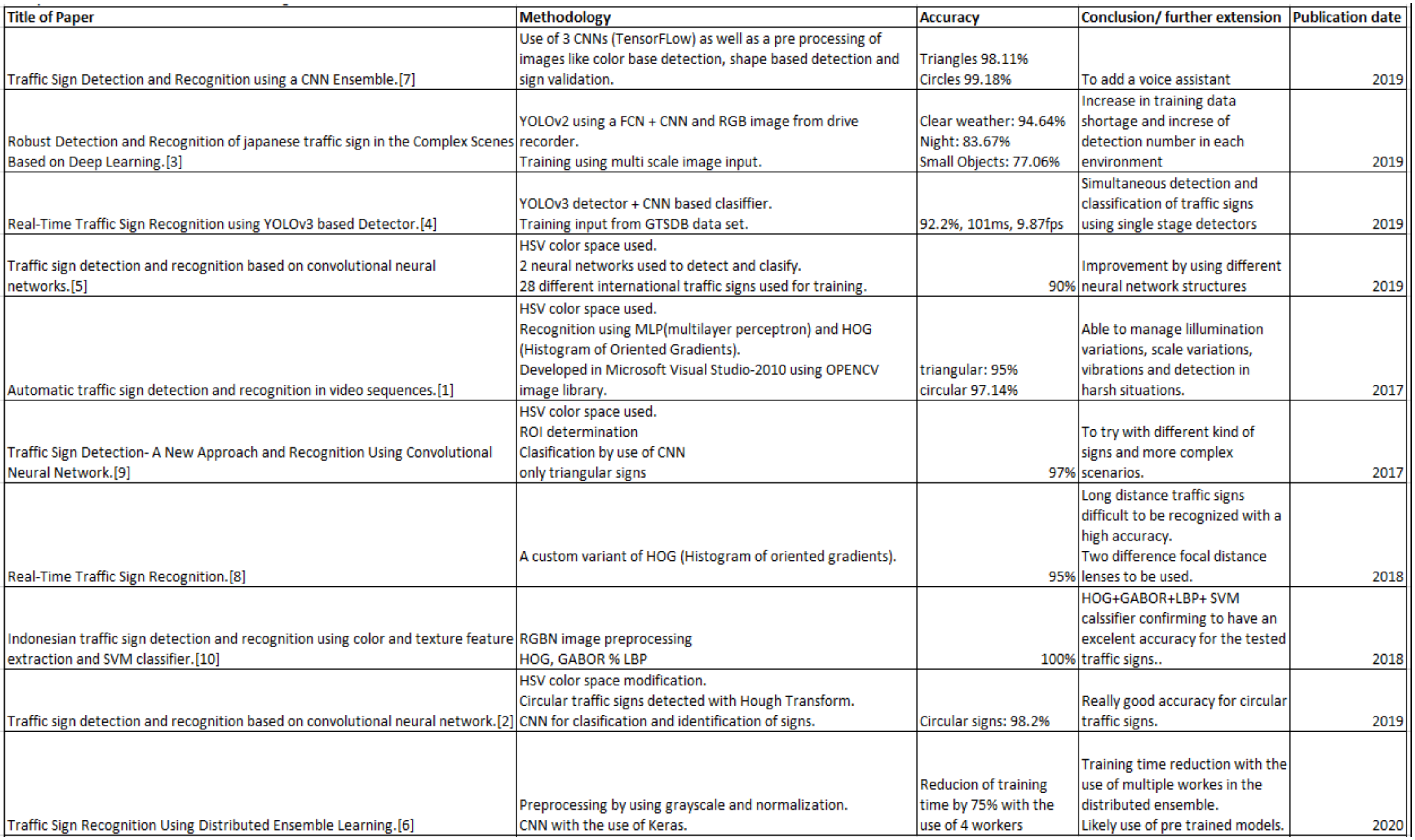
Overall, steps to develop a system like the one described before are below:

1. Preprocessing or optional processing of frames from a real-time video source.
2. Detection and defining of the ROI within the images/frames.
3. Recognition of the sign belonging to an international or local traffic sign [6].
4. Displaying or notification of the meaning of traffic signs to the driver.

In order to achieve the steps mentioned above, there are some challenges that are tried to be mitigated, including, all kind of variations like illumination as day or night, weather conditions, shape variations, damaged signs, etc. [7].

There are also different ways of development for every one of the basic steps of the system that can lead to different accuracy levels as shown in Table 1.

Table 1: Current state of the art for detection and recognition of traffic signs



Title of Paper	Methodology	Accuracy	Conclusion/ further extension	Publication date
Traffic Sign Detection and Recognition using a CNN Ensemble.[7]	Use of 3 CNNs (TensorFLow) as well as a pre processing of images like color base detection, shape based detection and sign validation.	Triangles 98.11% Circles 99.18%	To add a voice assistant	2019
Robust Detection and Recognition of japanese traffic sign in the Complex Scenes Based on Deep Learning.[3]	YOLOv2 using a FCN + CNN and RGB image from drive recorder. Training using multi scale image input.	Clear weather: 94.64% Night: 83.67% Small Objects: 77.06%	Increase in training data shortage and increse of detection number in each environment	2019
Real-Time Traffic Sign Recognition using YOLOv3 based Detector.[4]	YOLOv3 detector + CNN based clasiffier. Training input from GTSDB data set.	92.2%, 101ms, 9.87fps	Simultaneous detection and classification of traffic signs using single stage detectors	2019
Traffic sign detection and recognition based on convolutional neural networks.[5]	HSV color space used. 2 neural networks used to detect and clasify. 28 different international traffic signs used for training.	90%	Improvement by using different neural network structures	2019
Automatic traffic sign detection and recognition in video sequences.[1]	HSV color space used. Recognition using MLP(multilayer perceptron) and HOG (Histogram of Oriented Gradients). Developed in Microsoft Visual Studio-2010 using OPENCV image library.	triangular: 95% circular 97.14%	Able to manage lillumination variations, scale variations, vibrations and detection in harsh situations.	2017
Traffic Sign Detection- A New Approach and Recognition Using Convolutional Neural Network.[9]	HSV color space used. ROI determination Clasification by use of CNN only triangular signs	97%	To try with different kind of signs and more complex scenarios.	2017
Real-Time Traffic Sign Recognition.[8]	A custom variant of HOG (Histogram of oriented gradients).	95%	Long distance traffic signs difficult to be recognized with a high accuracy. Two difference focal distance lenses to be used.	2018
Indonesian traffic sign detection and recognition using color and texture feature extraction and SVM classifier.[10]	RGBN image preprocessing HOG, GABOR % LBP	100%	HOG+GABOR+LBP+ SVM calssifier confirming to have an excelent accuracy for the tested traffic signs..	2018
Traffic sign detection and recognition based on convolutional neural network.[2]	HSV color space modification. Circular traffic signs detected with Hough Transform. CNN for clasification and identification of signs.	Circular signs: 98.2%	Really good accuracy for circular traffic signs.	2019
Traffic Sign Recognition Using Distributed Ensemble Learning.[6]	Preprocessing by using grayscale and normalization. CNN with the use of Keras.	Reducion of training time by 75% with the use of 4 workers	Training time reduction with the use of multiple workes in the distributed ensemble. Likely use of pre trained models.	2020

Detection methods can vary and there yet improvements that can be done in this regard, some using Fully-Convolutional Neural Network (FCNN), or combinations of FCNN and CNN [8]. For example, preprocessing and change of color space, which can take time and reduce the number of frames per second (FPS) in practical applications.

Most of the methodologies used for images recognition and therefore the identification of the specific traffic sign, require of a training phase which needs the use of a database of thousands of images [9]. This has been identified as a crucial stage of the process, since here can be added different images in real scenarios, that greatly improve the accuracy of positive recognition and at the same time helps mitigating some of the challenges mentioned before.

In case of the use of a CNN, there are also implications in the number used as well as the structure of them, that can be reflected in better or worse accuracy and less or more time to reach the positive identification [10].

Not often mentioned, but as important as other stages in the system, is the way the identified and classified information from the traffic signs is passed to the driver. This can be done in different ways, using visual or auditive communication. Using a visual approach is not the most recommended, since it requires visual attention from driver and this attention, as short as it can be, is time not paying attention on the road ahead. An auditive solution seems to be a better option [5], as this doesn't require the driver to take its eyes out of the road and there is a fact, that most humans can handle both driving while listening at same time. Messages must be clear but, in a way, they can be felt more human like and not machine like.

As mentioned in the paragraphs before, the different approaches used by investigators across the world, are at some point falling into a series of technics, they also are facing issues that must be solved summarized as follows:

2.1. Most used detection and recognition techniques

2.1.1. Main use of CNNs for recognition of signs

Most of the models reaching high accuracy levels are using CNN as seen in the table 1. Some researchers as Vennelakanti's team [5] were using a set of 3 CNN in which they were applying 3 pairs of convolutions into the images as well as pooling the output of each convolution stage and finally having a regular fully connected NN which finally classified the traffic signs. They showed an improvement of 2.81% in the triangular shaped traffic signs when comparing the use of a single CNN and the use of 3 back-to-back CNNs reaching a final accuracy of 99.11%

The efforts done by Ying Sun's team [6] lead them to reach accuracies of up to 98.2% when using the GTSRB dataset for European traffic signs. This was done by using a color space modification from Red, Green, Blue (RGB) to Hue Saturated Values (HSV) and then Hough Transform to detect the position of the traffic sign in the whole image. After this they are using a CNN with some convolution layers and finally some dense layers to classify the traffic sign.

2.1.2. Latest uses of YOLO for quick detection and recognition of traffic signs.

As mentioned in the use of CNN by some researchers, one of the technics used broadly is the YOLO, which is used for real-time object detection. This network is based on CNN but this is different from a regular CNN in terms of efficiency oriented to detection and classification of objects in the images. As shown by Shehan P Rajendran's team [8], they use a YOLOv3 network to detect and classify a total of 43 different classes of traffic signs. With the use of this methodology, they could reach accuracies of up to 92.2%. The advantage of this technic is that all the images is processed in a single time instead of multiples times, therefore able to reach 9.87 FPS compared with the Faster Region proposal Convolutional Neural Network (Fast R-CNN) which reached 3.82 FPS.

Other case where improvements have been reached by using CNNs too, is the case of Hasegawa's team [7], where they have used an implementation of YOLOv2, which uses a CNN containing 22 convolutional processes as well as various pooling. This network was modified to be feed with multiple size images and the results compared with the YOLOv2 only has showed accuracy levels of 94.64% for a reduced set of traffic signs in clean weather conditions and considerable improvements in night and small objects.

Other examples of the use of YOLO in traffic sign recognition can be seen in table 2.

Table 2: Uses of latest YOLO versions in traffic sign detection and recognition.

Title of IEEE paper	YOLO implementation	Accuracy
On-The-Fly Traffic Sign Image Labeling	CNN + Region proposal network + YOLOv3	93.30%
YOLOv3 Algorithm with additional convolutional neural network trained for traffic sign recognition	YOLOv3 + CNN	99.28%
Real-Time Traffic Sign Recognition using YOLOv3 based Detector	YOLOv3 + CNN	92.20%
Robust Detection and Recognition of Japanese Traffic Sign in the Complex Scenes Based on Deep Learning	YOLOv2 + CNN	94.64%

2.1.3. HSV & Hough Transform

Recent approaches of detection and region of interest within the image, which in this case corresponds to the traffic signs and where they are exactly located into the images were done by transforming the color space that is used normally as the RGB into HSV which enhances some of the shapes and avoid variations of precise color that could be affected by illumination or ageing of the traffic sign at the moment of taking the input image. After this a Hough Transform must be applied to find out the edges of circular traffic signs and other specific shapes of signs. The output from this is the region where the sign is located within the image so this can be taken to feed the next classification network.

This kind of technics are showing good results as mentioned before for Ying Sun [6], research where but this also requires the use of two networks which can represent an impact into the FPS which is not mentioned but it a requirement to be able to sample and process images as fast as possible to be closer to real-time detection and recognition.

2.2. Difficulties to solve based in Recent Research

2.2.1. Real-time processing and high FPS

This is an important aspect that is not always mentioned in the recent research but really important for this particular research since when trying to take images from the street, the expectation is to process this image and detect and recognize the traffic signs as close to real-time as it is possible.

Researchers using YOLO as Hasegawa [7] and Rajendran [8] are mentioned to be able to reach up to 9.87 fps and mentioned that this technic has high processing speed and reduced false detections when compared to other CNNs.

2.2.2. Multi Scale Input

One of the issues that could be faced when sampling the frames from a real-time video source when moving across the streets is to see the traffic signs appearing in the distance as small and unrecognizable and constantly increasing their size depending on how close they are to the device capturing these frames.

Hasegawa's team [7] tried to increase the robustness of their model by adding different sizes of the same signal when training their model, specifically they used images of: 640, 672, 704, 736 and 768 pixels per side. Doing this enhancement depends in the dataset used and the resolution of the images used to train as well as the sizes of the traffic signs seen in these images.

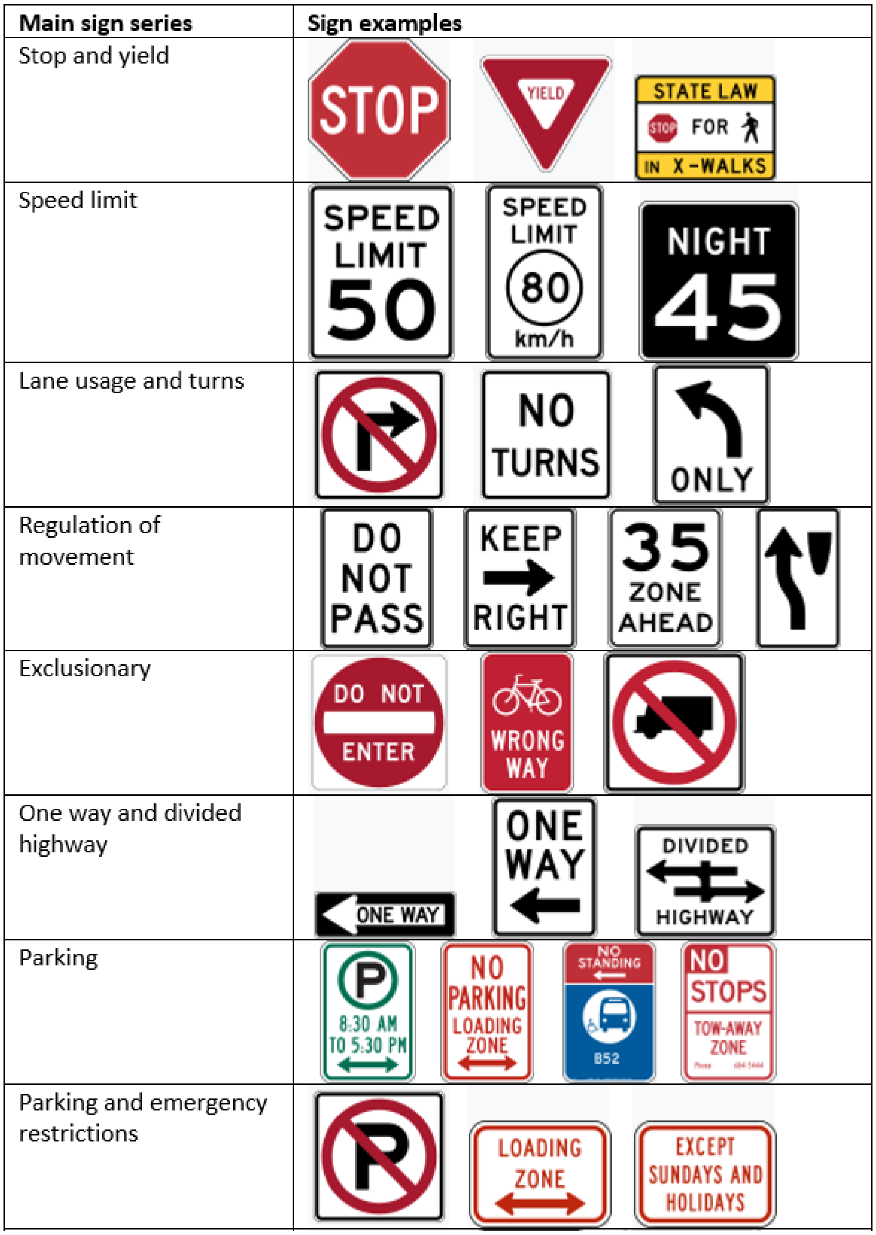
2.2.3. Multiple weather and light conditions

Weather and multiple light conditions are an issue that most of the recent researchers like Novak's team [11] were not considering and that is also impacting any solution for real-time detection and recognition since the weather as well as the sunlight or artificial lights are constantly changing in real life. Most of the models were created in controlled scenarios and ignoring his detail because of being outside of their scopes and because this depends on the images from dataset used.

To solve this, there were some technics as mentioned in this chapter, as using the HSV to minimize the light variation into the traffic signs. This could be not done in other approaches that are faster but this could be mitigated by using a really complete dataset that includes a balanced number of images in different light and weather conditions.

2.3. Different kinds of traffic signs

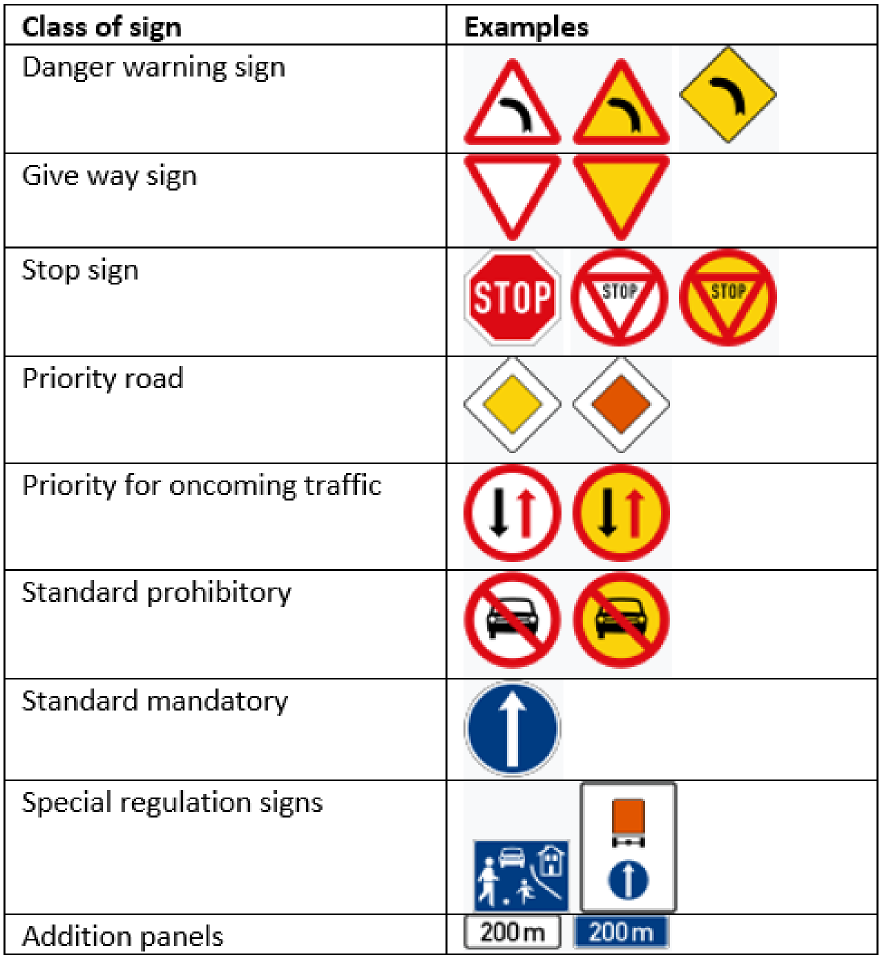
There are different kinds of traffics signs across the world. But depending on the countries they are usually using some variation of the conventions to define the details about this traffic signs. One of the documents defining traffic signs used in USA or Puerto Rico, is the Manual on Uniform Traffic Control Devices for streets and highways (MUTCD) [12] and which examples can be seen in figure 1.



Main sign series	Sign examples
Stop and yield	STOP YIELD STATE LAW STOP FOR IN X-WALKS
Speed limit	SPEED LIMIT 50 SPEED LIMIT 80 km/h NIGHT 45
Lane usage and turns	NO TURNS ONLY
Regulation of movement	DO NOT PASS KEEP RIGHT 35 ZONE AHEAD
Exclusionary	DO NOT ENTER WRONG WAY
One way and divided highway	ONE WAY ONE WAY DIVIDED HIGHWAY
Parking	P 8:30 AM TO 5:30 PM NO PARKING LOADING ZONE NO STANDING B52 NO STOPS TOW-AWAY ZONE
Parking and emergency restrictions	LOADING ZONE EXCEPT SUNDAYS AND HOLIDAYS

Figure 1: Examples of the traffic and road signs used in USA and defined by MUTCD.

Other of the traffic signs convention broadly used, which includes Mexico and other 55 other countries, is the Convention on road signs and signals. Vienna, 8 November 1968. Examples from this convention can be seen in figure 2.



Class of sign	Examples
Danger warning sign	
Give way sign	
Stop sign	STOP STOP STOP
Priority road	
Priority for oncoming traffic	
Standard prohibitory	
Standard mandatory	
Special regulation signs	
Addition panels	200 m 200 m

Figure 2: Examples of traffic and road signs stablished by the Vienna's Convention on road signs and signals.

As mentioned in this chapter, Mexico is part of the Vienna convention from 1968 but it is a regulation stablished by Secretaria de Comunicaciones y Transportes (SCT) [13] for the particular road signs to be used in Mexico. The vertical traffic signs are classified mainly in Restrictive Signals, Preventive Signals, Informative Signals, Touristic and service signals, and Other Signals as a few examples of them can be observed in figure 3.

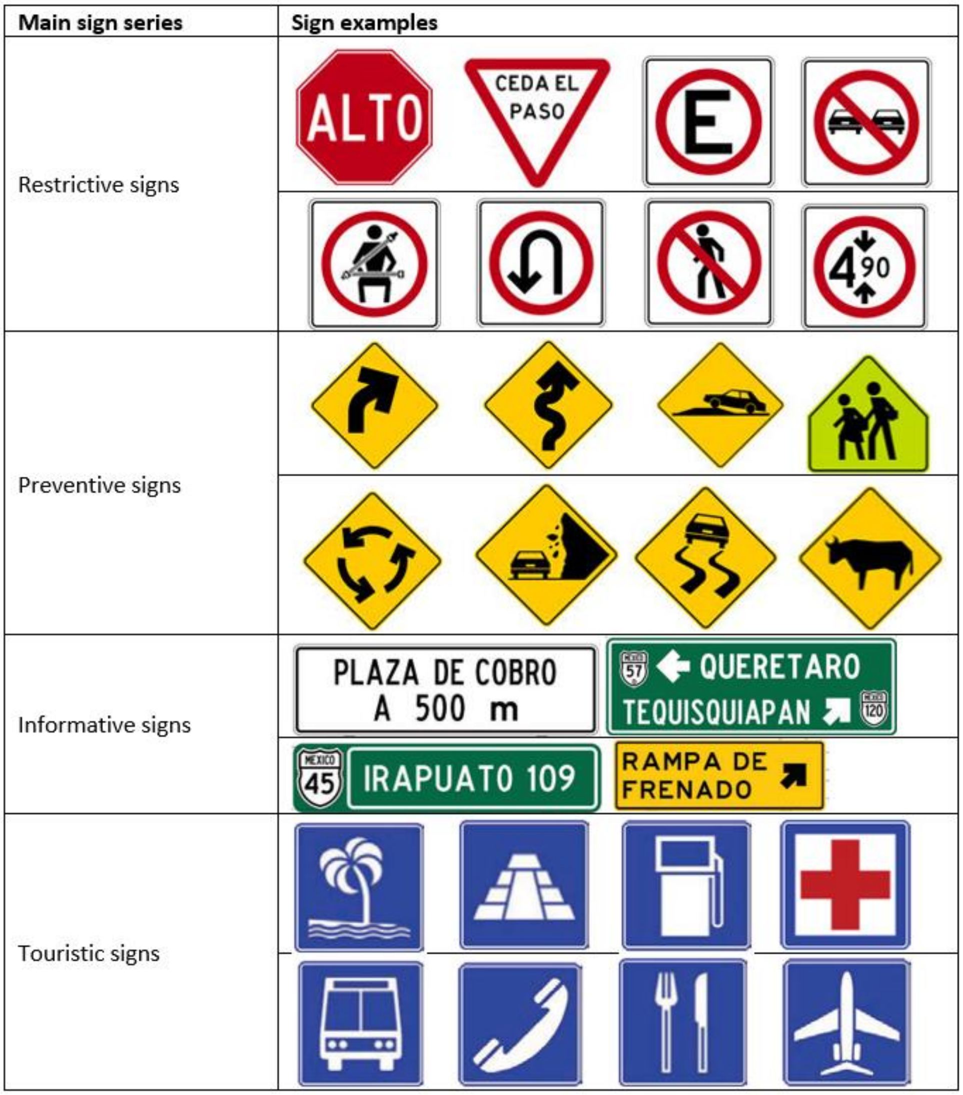


Figure 3: Examples of road and traffic signs used in Mexico and approved by SCT.

3. THEORIC/CONCEPTUAL FRAMEWORK

Machine learning comes from a discipline of Artificial Intelligence (AI), that, using algorithms allows computers to identify patterns within a massive amount of data, being able to make predictions. This learning allows to the computer, to later on, achieve some tasks without any or limited human intervention.

This term was first used in 1959 but since then it has gained relevance due to the increase in computational capacity and the boom of Big Data.

Several applications have been developed using this approach around the whole world.

3.1. Artificial Intelligence

AI combines computer science with the use of rich and robust datasets to solve different kinds of problems. It also has different sub fields like machine learning and deep learning, which are normally mentioned when talking about AI. The main intention when using AI algorithms is, to predictions and/or classification of different topics using some input data.

3.1.1. Machine Learning

Machine learning is a sub-field of the artificial intelligence and computer science which focuses on the use of amounts of data and algorithms, imitating the way the brain and human learning works, gradually improving its accuracy.

With the use of statistics and algorithms, machine learning models are able to solve classification and prediction problems. With the knowledge provided by this model, we are able to take smarter decisions within computer programs and applications up to government or company decisions.

Machine learning has today many applications which can include voice recognition, clients service, artificial vision, search engines, automatic sell/buy trading, etc.

3.1.2. Deep Learning

Deep learning is another sub-field of the artificial intelligence, and in this case is also a sub-field of machine learning. Deep learning and machine learning are usually mentioned as being similar or even the same, but they have some important differences. Deep learning automatizes the process of feature extractions of data. This of course eliminates human intervention considerably and this allows the use of bigger and complex datasets.

Deep learning has a lot of interest applications, i.e., computer vision, natural language processing, speak recognition etc.

3.2. Convolutional Neural Network (CNN)

3.2.1. Introduction

Neural networks of Artificial Neural Networks are computing systems inspired by a model of neurons in a biological brain. Each connection works similarly to a synapse from a neuron in the sense that everyone can transmit signals to other neurons [14].

The input signals are real numbers, and the output of every neuron is modeled by a non-linear function, which is the sum of its inputs (1).

$$f(xw+b)=\begin{cases}1, & xw+b\geq 0\\ 0, & otherwise\end{cases} \tag{1}$$

The simplest of them is called *Perceptron* which is composed of a single unit that takes multiple inputs and for every one of these inputs, there is a *weight* parameter that can be modified and affects directly the input value Figure 4. Then the main unit sums the inputs and then they have to go through an *activation function* like Rectified Linear Unit function, RELU (2) or Hyperbolic tangent function, TANH (3). There are also ways to induce learning in this model that is achieved by modifying or adjusting the weight parameter for the inputs for each iteration, so every time the output is closer to the expected.

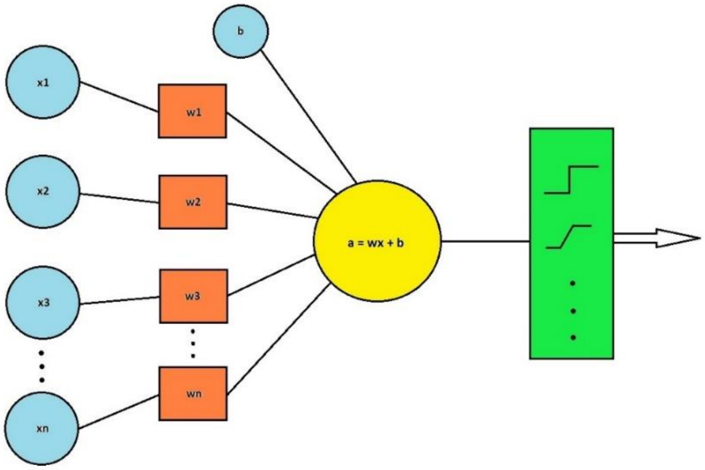


Figure 4: Basic neural network, bias, and activation functions.

Once explained the simplest model, which is a single neuron, the regular composition of a neural network consists of the use of an *input layer*, a *hidden layer* and an *output layer* as shown in Figure 5.

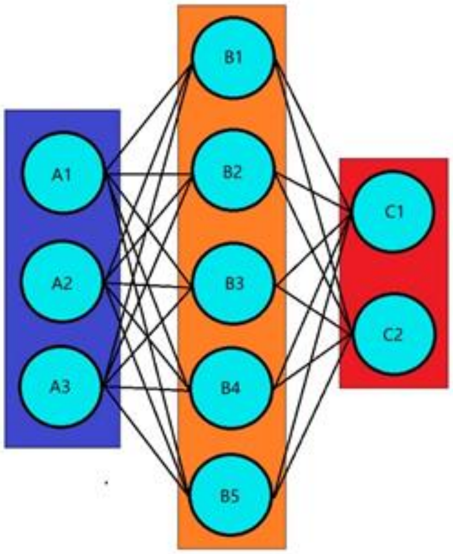


Figure 5: Neural Network's Input layer, hidden layer, and output layer from left to right.

The Input layer contains the neurons that consume the first inputs to the network and the length or number of neurons in this layer depends on the characteristics of the data used to feed the network.

The Hidden layer can contain one or more layers, depending on the needs and how many steps of processing of the data are required, normally every neuron of this layer is connected to all the neurons of the next layer.

The Output layer is the final in the model, and it is responsible for showing the final results of the processing, for example, classification of data. In this case, as shown in figure 2, having a single neuron for every class that has to be identified and providing a probability for every one of these classes. One example of this kind of activation functions used to classify is the Softmax, which can be seen in equation (4).

$$f(x) = \begin{cases} 0, & if\ x \leq 0 \\ x, & if\ x > 0 \end{cases} \tag{2}$$

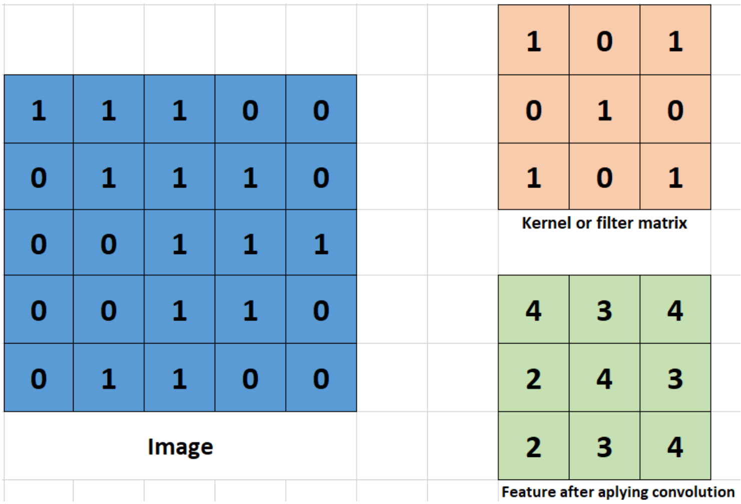
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3}$$

$$\sigma(z)_j = \frac{e^{zj}}{\sum_{k=1}^{K} e^{zk}} \quad for\ j = 1, \dots, K \tag{4}$$

3.2.2. Convolutional Network

A convolutional neural network is a deep learning algorithm that takes data, as an image, as input and has some weight and bias to assign importance to every pixel of the image and that are used as a specific characteristic of content of the image, like objects in it and these values can be used to differentiate an object from another.

This kind of network can capture temporal and spatial dependencies using one of the different filters as can be seen in figure 6.



1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

1	0	1
0	1	0
1	0	1

Kernel or filter matrix

4	3	4
2	4	3
2	3	4

Feature after aplying convolution

Figure 6: feature extraction of a feature after applying a convolution on input image

3.2.2.1. Local receptive fields and shared weights

Each neuron in a CNN consumes a defined region of the input data, so this makes the neurons to identify and learn pattern as edges or any small detail that are characteristics of an image.

This defined region of the input that a neuron takes and is exposed to is called Local Receptive Field [15].

In the case of having an image, the local receptive field would be a portion of this image, which values are taken by a certain neuron let’s say, in a convolutional layer.

There are different architectures containing CNN which required normally to be trained from start to end, but there is a way to make training simple and faster by using shared weights. This works by modeling a common section of an architecture that can be used for most of architectures which considerably reduces the times and resources required to train whatever specific architecture that reuses these shared weights [16].

3.2.2.2. Sub-sampling

A CNN usually makes use of sub sampling layers which is also called as pooling. This process is responsible to obtain a summary of a particular feature over a region of an image by using the mean or max value of this regions. This summary of characteristics of an image reduces the dimensions and therefore the complexity at the next stage and also helps to reduce overfitting.

3.2.2.3. Fully connected layer or other layers

On a CNN the fully connected layers are what is known as a feed forward neural network. These are usually the final layers in the network and receive the output from the final pooling or convolutional layers. They have to be flattened, so any of the dimensions of the matrix has to be reduced to a vector and then can be consumed by the first fully connected layer [17].

3.2.2.4. CNN learning

This process is done by the whole network, first extracting the features of the input image by the use of convolution with different filters as well as pooling process applied in different layers. This has a summary of the characteristic of the input image and then is flattened and sent into the fully connected layers, where every unit or neuron used activation filters to as well as loss and cost functions to calculate first, how far is a prediction done against a validation image and then, modify the weights for each neuron accordingly trying to reduce loss and increase accuracy. This is done in several loops so this correction can be done as many times as we want.

3.2.3. CNN Architectures

3.2.3.1. VGG-16

VGG-16 is a known CNN model for detection and classification first proposed by Simonyan and A. Zisserman in a paper called: “Very Deep Convolutional Networks for Large-Scale Image Recognition”. This model was able to achieve up to 92.7% of test accuracy in ImageNet’s dataset [18]. This architecture is made of 5 sets of convolutional and max pooling layers followed by 3 fully connected layers, using for these last layers, RELU and at the final layer Softmax as activation function, this backbone can be seen in figure 7. This architecture is easy to implement due the reduced number of layers and was also used as a base for other deep learning architectures.

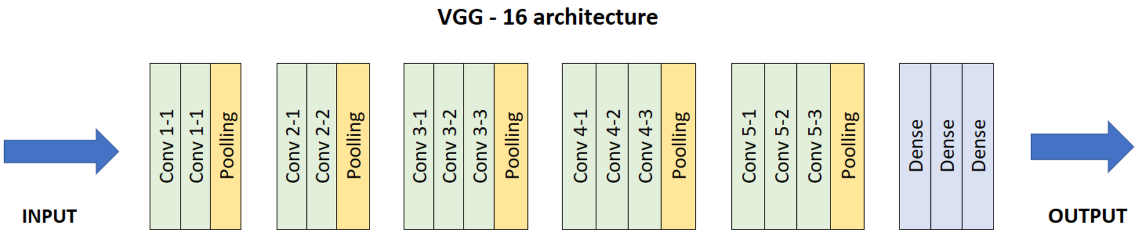


Figure 7: VGG-16 architecture layers

3.2.3.2. ResNet50

ResNet50 is a convolutional neural network that is 50 layers deep which is known for its use of residual functions. This consists of 4 stages and uses layer of convolutions and pooling operations with kernels of sizes 7x7 and 3x3. This has a pre-trained version that was trained with more than a million images from ImageNet for 1000 classes. This network can be used as backbone of other more complex architectures or implementations with extra layers at the input or at the output of this network. The pretrained version receives image input of size 224x224 pixels [19].

3.2.3.3. ResNet152

This is a deeper variant of the ResNet50 which shared things like the use of residual functions as well as having a pre-trained model available and trained with imageNet. This as well as Resnet50 was based in the VGG19 and the adding of other extra layers and was developed for imageNet and Common Objects in Context (COCO) competitions, showing good results [20].

3.2.3.4. Single Shot Detection

This approach is faster than Fast R-CNN since it eliminates the region proposal part, and this allows to achieve higher fps when working in real time images. This methodology has two parts, first extracting the feature maps and then applying convolution filters to detect the objects in the image.

3.2.3.5. YOLO

It is a real-time object detection algorithm that uses a Computer Processor Unit (CPU) or a Graphics Processor Unit (GPU) to detect images. This consist of an individual CNN to process the whole image (as its name suggests) instead of having to modify any incoming image to the system o changing its color space and it is trained using the full images.

This processes the image as different regions and provides probabilities for every region.

This algorithm can reach a high accuracy and, since it passes only once though the entire neural network, it is suitable to be used in real-time applications such as video where being able to process at a higher FPS has particular importance [21].

3.2.3.6. Region proposal Convolutional Neural Network (R-CNN)

This is a method used for object detection which is based on region proposals, which are areas where there is a high probability to contain the objects we are trying to detect and recognize within an image. The first stage is to use a selective search method to find out this region proposals, next stage is a conventional CNN, used to extract the features from the mentioned regions in the image. Then, there is a Support Vector Machine (SVM) used to classify the objected from the vectors obtained on previous CNN. Finally, a bounding box regressor can be added to find out with the localization of the objects within the two dimensions of the image, X, and Y [22].

3.2.3.7. Fast R-CNN

This is a methodology which comes directly as an improved version of R-CNN with the main intention of fix issues found in R-CNN as slow inference, hard to train due the three big stages belonging to it, and large memory requirements. These problems are addressed by extracting the features of the whole input image using the CNN, then feature maps from the region proposals are taken and resized using a pooling layer to later one being consumed by a final stage with two outputs being the SVM classifier and the bound box regressor [23].

3.3. Classification metrics

3.3.1. Accuracy

This is one of the most used metrics used to define whether the performance of a model is good or not. This is defined as the ratio of the number of correct predictions by the total number of input images or any kind of samples as can be observed in equation 5.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made} \qquad (5)$$

3.3.2. Precision

Precision is defined as the number of positive predicted results, divided by the number of positive results and the mistakenly positive results predicted by the model as can be seen in equation 6.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{6}$$

3.3.3. Recall

Recall is the number of correct positive results divided by the number of all relevant samples identified as positive as shown in equation 7.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{7}$$

3.3.4. F1-Score

F1-Score is the defined as the harmonic mean between precision and recall. This can tell us better how precise is the performance of a classifier. The way to calculate it is shown in equation 8.

$$F1 = 2 * \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{8}$$

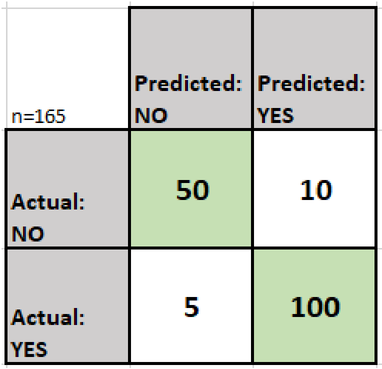
3.3.5. TP, TN, FP, FN

- True Positive: This is a case in which the model successfully made a prediction for a 1 and the real output was 1 too.
- True Negative: This is a case in which the model successfully made a prediction for a 0 and the real output was 0 too.

- False Positive: This is a case in which the model mistakenly made a prediction for a 1 but the real output must be 0.
- False Negative: This is a case in which the model mistakenly made a prediction for a 0 but the real output must be 1.

3.3.6. Confusion Matrix

This is a matrix that describes the total performance of a model. This uses the previously calculated TP, TN, FP and FN to show those values across a matrix can have variable dimension depending on the number of classes used in the model. The accuracy can be calculated by the average of the values in the main diagonal which can be seen in green in the figure 8 below [22].



n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

Figure 8: Confusion matrix and its main diagonal.

3.4. Machine and Deep learning libraries

3.4.1. Numpy and Ndarray attributes

This is a package for python language which is used for scientific computing. This allows to use data as a multidimensional array object. This object and the functions including in this package, makes mathematical operations in matrix in vectors really easy. This also allows to make different kind of operations like shape manipulation, sorting, basic algebra etc. between these multidimensional objects [23].

3.4.2. Pandas

Pandas is broadly used library for data manipulation by using DataFrame objects. This library gives us an easy way to read and write of different in-memory data structures as Comma Separated Value (CSV), text files, Excel, SQL, etc. This tool also has an easy way to index the data and make operations with it. This is able too, of handling large datasets and is optimized to be fast and efficient [24].

3.4.3. TensorFlow

TensorFlow is an open-source library for machine learning intended to be used for the development of machine learning models. This was developed by Google and open to all users on November 9th of 2015, with the intention to be used for building and training of Neural Networks in order to achieve a sort of different tasks [25].

This can be run on different CPU and GPU and is it compatible with the most commonly used Operative Systems (OS) like Windows, Linux, macOS, Android and iOS.

3.4.4. TensorBoard

TensorBoard is a TensorFlow’s tool used by engineers and researchers to visualize their machine learning experiments. Is a valuable tool used to share the implementation of a simple or very complex training procedure with other people, interested in details as hyperparameters tunning and other metrics [26].

3.4.5. Keras

Keras [27] is the high-level Application Programming Interface (API) from TensorFlow used to build and to train machine learning and deep learning models. It is supposed to be an easy tool to handle TensorFlow allowing the users to make modules and extend them in a simple way.

3.5. Annotations

3.5.1. Intro

When talking about object detection and segmentation, we would usually encounter the word annotations. These files contain the information about one or more objects of interest within images, like the labels for each of them and their position in coordinates using numeric values that describe their position with respect of the dimension of the image.

3.5.2. Coco

COCO is a commonly format used for annotation files which usually contains information about the label of the objects of interest inside of an image. This information could be the bounding boxes used for object localization. This uses JSON files, which basically contains blocks with the next 5 sections on it: Info, licenses, categories, images and annotations [28].

3.5.3. Pascal

PASCAL Visual Object Classes (VOC) as well as COCO, provides information about the images used for training, testing and validation for object detection and classification such as labels with boundary box and label with name of the class which certain object belongs. This uses XML files and the way the information is stored and organized here is slightly different than COCO [29].

4. DEVELOPMENT METHODOLOGY

For this Project different dataset were considered and every one of them had different pros and cons that will be discussed in this section. The reasons behind discarding some of this dataset until we find out the selected dataset will be shown as well as the characteristics of the selected one.

In this section we also will be covering the process of clean up and adjusting of the images to be used.

This will cover other phases as pre training and obtention of hyperparameters as well as the full training and testing of the model obtained with this dataset.

4.1. Dataset exploration

There is a variety of datasets available including the traffic sign images that can contain different number of images, from different geographical locations and with different classes and some of them could be more and less useful depending on the application.

For this document, 4 different datasets were considered before taking a decision and selecting the most compatible dataset for the solution that is proposed.

The first dataset comes from Kaggle, and it is called Traffic Signs Preprocessed [30]. This dataset contains over 86000 images with 43 classes, and it is separated in 9 pickle files which contains images normalized or in grayscale. Dataset files number 0 to 3 are RGB and the last files 4 to 8 are grayscale. The traffic sign images have a fixed size of 32x32 pixels and limited only to the traffic sign itself with no context around it. The size of the full dataset is 7GB.

Second dataset is from Electronic Research Data Archive and called German Traffic Sign Recognition Benchmark GTSRB [31]. This dataset contains over 63000 images with a .ppm extension and they contain annotations. This contains a version of the training dataset that is a Histogram of Oriented Gradients (HOG) as well as HSV histograms. The size of this dataset is approximately 3GB.

Third dataset is the Mapillary Traffic Sign Dataset [32]. This dataset contains over 100,000 images and 300 classes. This dataset contains images with different resolutions from 640x480 to 3840x2160. The images from this dataset were taken from around the globe including Latin America. The dataset includes full annotations files for each of the images as well as variations in illumination, weather conditions, times of the day and viewpoints, etc.

Fourth dataset is available from GRAM Multisensorial Analysis and Recognition Group, and it called Traffic Signs UAH Dataset [33]. This dataset contains approximately 660 images with a resolution of 2048x1360 and 1536x1024. It is separated in different classes with images having Segmentation problems, Rotation, Clusters, Shadows, Different Sizes, Deformed, etc. This dataset has been made with images of traffic signs from Spain. This has a size of 400MB with compressed images.

4.2. Dataset selection

The dataset selection process was taken by comparison and considering the needs of this research too. First dataset, being not as big as others but the first drawback with this was the resolution of the images which is only 32x32, which can only be used for classification and not for the object detection process that this document will explain later in detail. Other issue with this was the compression of this images which was pickle files that were not as easy to use in first instance as a jpeg or other common image file format. And finally, the number of classes on this as not enough and it contains images of traffic signs from Europe and not America or Latin America.

Second dataset had images where traffic signs where inside of the images, but they were not fitting other needs like lacking classes from traffic signs used in Mexico, so this was enough to discard this dataset. Other issue with this was the extension of the files which was a *.ppm* and finally the images had some color space modification like HOG.

The last dataset discarded for this research had only 660 high resolution images which were not enough for a good training. They had the advantage of having different illumination, rotation and other scenarios for the same classes, but they were just a few classes and this dataset was done in Spain, where traffic signs are way different to the traffic signs seen in Mexico's streets.

The data set to be used for this project is the Mapillary Traffic Sign Dataset which contains a large set of more than 100 thousand images with over 300 different classes as show in figure 9. This traffic sign images comes from Europe, North America, South America, Asia and Oceania. This dataset is considering images with some varieties of weathers conditions, seasons, time of day, camera, and viewpoint.

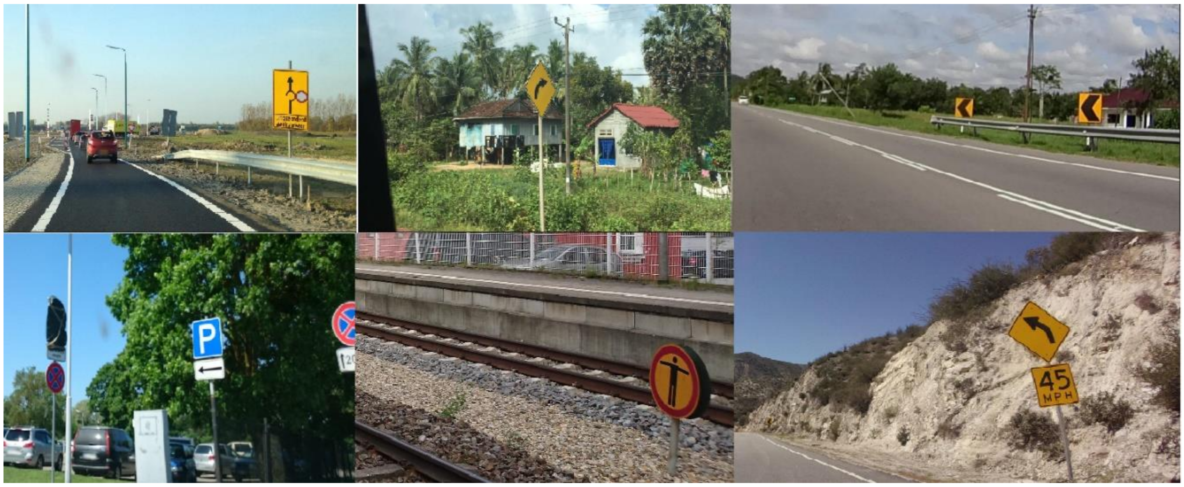


Figure 9: Images extracted from the selected dataset

From this dataset, a half has fully annotation information and other half is partially annotated which is really useful to identify the section where the one or more traffic signs are located within the image as can be seen in figure 10.

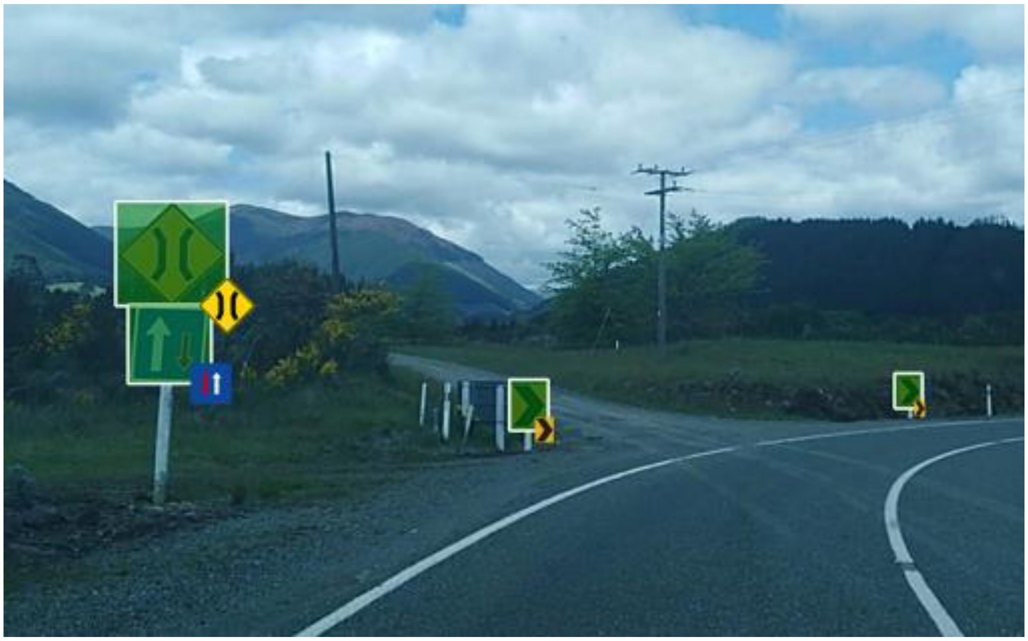


Figure 10: Example of an image from dataset with full annotations and bounding box around four traffic signs.

This dataset can be downloaded and directly from the webpage as compressed zip files which includes the images and the files for annotations that were previously classified and verified by humans for the half of this dataset. The images are large in resolution, with sizes around 3000x4000 pixels. For this whole dataset to be stored it is required 100GB in total for fully annotated and partially annotated.

Since this project will be focused on traffic signs for Guadalajara and with the possibility to be used in other cities in Mexico. This data set has to be reduced by removing the classes belonging to traffic signs not used in this region. Therefore, thousands of images from other continent will be not used for both training and testing.

4.3. Exploration of traffic signs from dataset

Since the selected traffic signs has full annotations or partial annotations of the traffic signs within the images, these were processed first to extract and have a sample of the individual traffic signs to see the distribution across all the classes as well as extracting other characteristics from them.

The process required to take every full-size image, then using the COCO annotation files to select the different X & Y coordinates to find every traffic sign in an image as well as its label for the corresponding class. Once the images and the annotation files are opened, we can have a clear idea of the kind of objects that will be identified later. A sample of the objects found can be seen in figure 11.

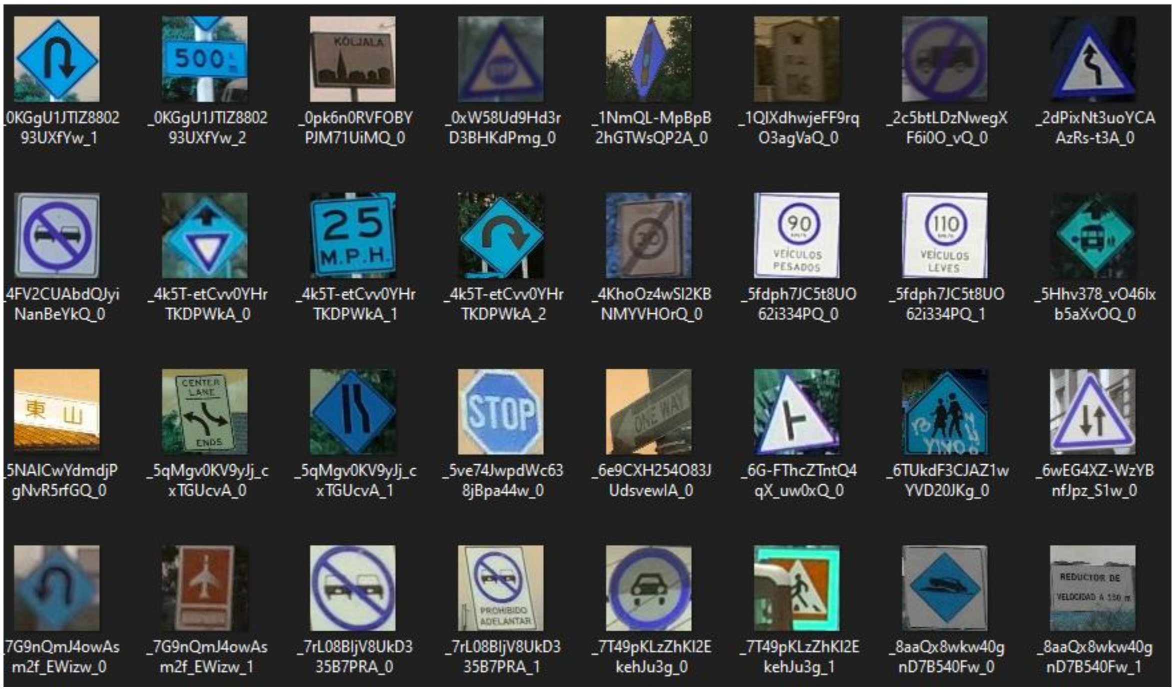
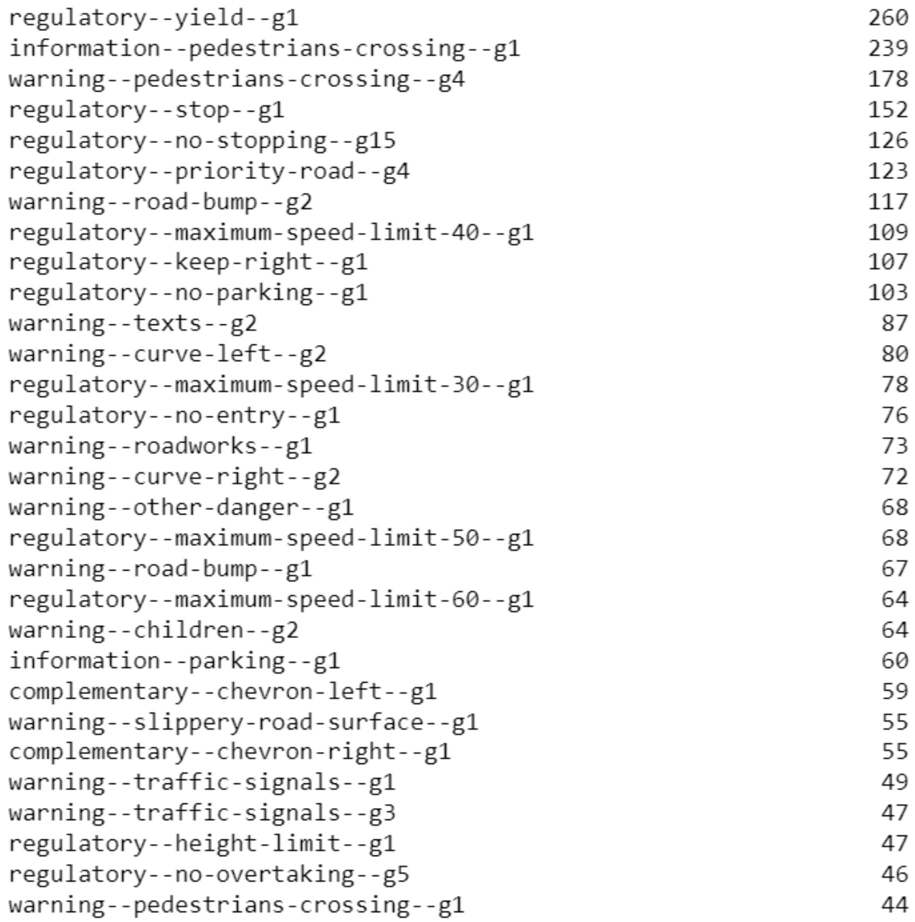

500
KOLJALA
0KGgU1JTIZ8802 93UXfYw_1
_0KGgU1JTIZ8802 93UXfYw_2
_0pk6n0RVFOBY PJM71UiMQ_0
_0xW58Ud9Hd3r D3BHKdPmg_0
_1NmQL-MpBpB 2hGTWsQP2A_0
_1QlXdhwjeFF9rq O3agVaQ_0
_2c5btLDzNwegX F6i0O_vQ_0
_2dPixNt3uoYCA AzRs-t3A_0
25 M.P.H.
90 VEÍCULOS PESADOS
110 VEÍCULOS LEVES
4FV2CUAbdQJyi NanBeYkQ_0
_4k5T-etCvv0YHr TKDPWkA_0
_4k5T-etCvv0YHr TKDPWkA_1
_4k5T-etCvv0YHr TKDPWkA_2
_4KhoOz4wSI2KB NMYVHOrQ_0
_5fdph7JC5t8UO 62i334PQ_0
_5fdph7JC5t8UO 62i334PQ_1
_5Hhv378_vO46lx b5aXvOQ_0
東 山
CENTER LANE ENDS
STOP
ONE WAY
5NAICwYdmdjP gNvR5rfGQ_0
_5qMgv0KV9yJj_c xTGUcvA_0
_5qMgv0KV9yJj_c xTGUcvA_1
_5ve74JwpdWc63 8jBpa44w_0
_6e9CXH254O83J UdsvewIA_0
_6G-FThcZTntQ4 qX_uw0xQ_0
_6TUkdF3CJAZ1w YVD20JKg_0
_6wEG4XZ-WzYB nfJpz_S1w_0
PROHIBIDO ADELANTAR
REDUCTOR DE VELOCIDAD A 150 m
7G9nQmJ4owAs m2f_EWizw_0
_7G9nQmJ4owAs m2f_EWizw_1
_7rL08BljV8UkD3 35B7PRA_0
_7rL08BljV8UkD3 35B7PRA_1
_7T49pKLzZhKI2E kehJu3g_0
_7T49pKLzZhKI2E kehJu3g_1
_8aaQx8wkw40g nD7B540Fw_0
_8aaQx8wkw40g nD7B540Fw_1

Figure 11: Samples of extracted traffic sign from full size images.

The process of sampling of every individual object in the full images described in previous paragraph had been also used to observe the distribution of the number of objects per class across all the images in the segment of the dataset containing full annotations. This useful information can be seen in figure 12.



```
regulatory--yield--g1                                  260
information--pedestrians-crossing--g1                  239
warning--pedestrians-crossing--g4                      178
regulatory--stop--g1                                   152
regulatory--no-stopping--g15                           126
regulatory--priority-road--g4                          123
warning--road-bump--g2                                 117
regulatory--maximum-speed-limit-40--g1                 109
regulatory--keep-right--g1                             107
regulatory--no-parking--g1                             103
warning--texts--g2                                      87
warning--curve-left--g2                                 80
regulatory--maximum-speed-limit-30--g1                  78
regulatory--no-entry--g1                                76
warning--roadworks--g1                                  73
warning--curve-right--g2                                72
warning--other-danger--g1                               68
regulatory--maximum-speed-limit-50--g1                  68
warning--road-bump--g1                                  67
regulatory--maximum-speed-limit-60--g1                  64
warning--children--g2                                   64
information--parking--g1                                60
complementary--chevron-left--g1                         59
warning--slippery-road-surface--g1                      55
complementary--chevron-right--g1                        55
warning--traffic-signals--g1                            49
warning--traffic-signals--g3                            47
regulatory--height-limit--g1                            47
regulatory--no-overtaking--g5                           46
warning--pedestrians-crossing--g1                       44
```

Figure 12: Distribution of number of objects per class.

It is necessary to know that this distribution of objects per class was made from images having a size of 64x64 pixels as indicated in the bounding box position values. This works well to know which classes have more data available to start creating our sample dataset.

4.4. Sample dataset generation

To start working in a solution we used a sub data set which contain just 3 classes so this can be useful to be run with different architectures, loss functions, hyperparameters, etc. This decision was also taken considering that full dataset contains over three hundred thousand images which will require too much computational resources as well as it is time consuming depending on the kind of hardware available for training.

In the other hand, working with a subset will allow us to do some fine tuning by training more times with little adjustments between each training.

This sample data set was made by selecting 3 different classes or 3 different kinds of traffic signs. The selected names for the classes were "regulatory--no-stopping--g15", "warning--pedestrians-crossing--g4" and "regulatory--stop--g1" as they are described in the dataset.

As mentioned in this chapter, our dataset contains annotations in COCO format which come in the form of individual JSON files per image in the dataset. Every one of these files contains the position as well as other descriptive data for every object in the image which can be multiple classes in same image.

To continue the preparation of our sample dataset, these COCO files were opened a modified using a python script, deleting all the other classes that were not going to be used for our training. After this step was completed, our python script was run again to end up with only images containing mostly our objects from the three selected classes. As was expected the number of objects per class was not the same for all three so we had an imbalance in our sample dataset. This was fixed by randomly dropping some percentage of the images containing more objects so the classes can have now a balance as shown in figure 13. This final sample dataset was composed of approximately one thousand objects per class and close to 2500 different images.

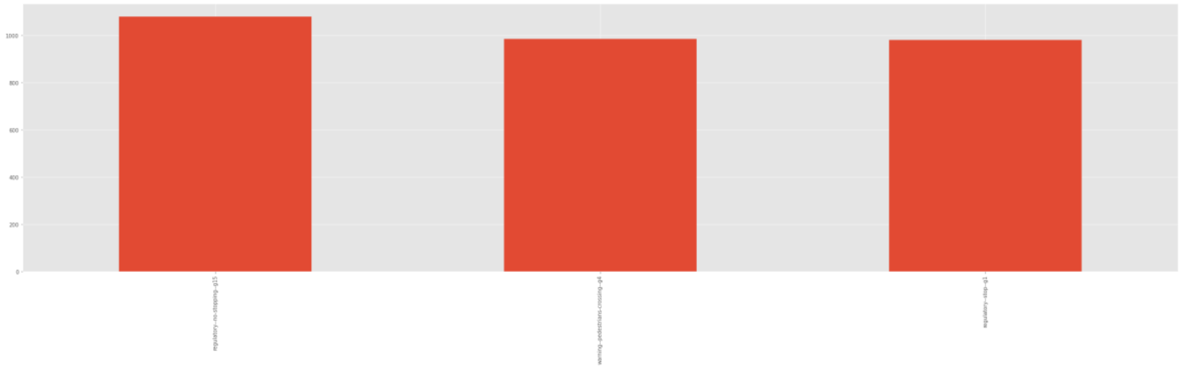


Figure 13: Balanced classes of the sample dataset.

The next requirement for our training process was to build a *.CSV* file which contains the most important data for training our model, this data is the path of the image, X1 coordinate, Y1 coordinate, X2 coordinate, Y2 coordinate, and finally the name of the class for this specific object in a single row of this file. There was a second little file that only contains the different classes names we will be using to train. These two processes were made using one task in the same python script used for the modification of the COCO files as well as the balance of classes.

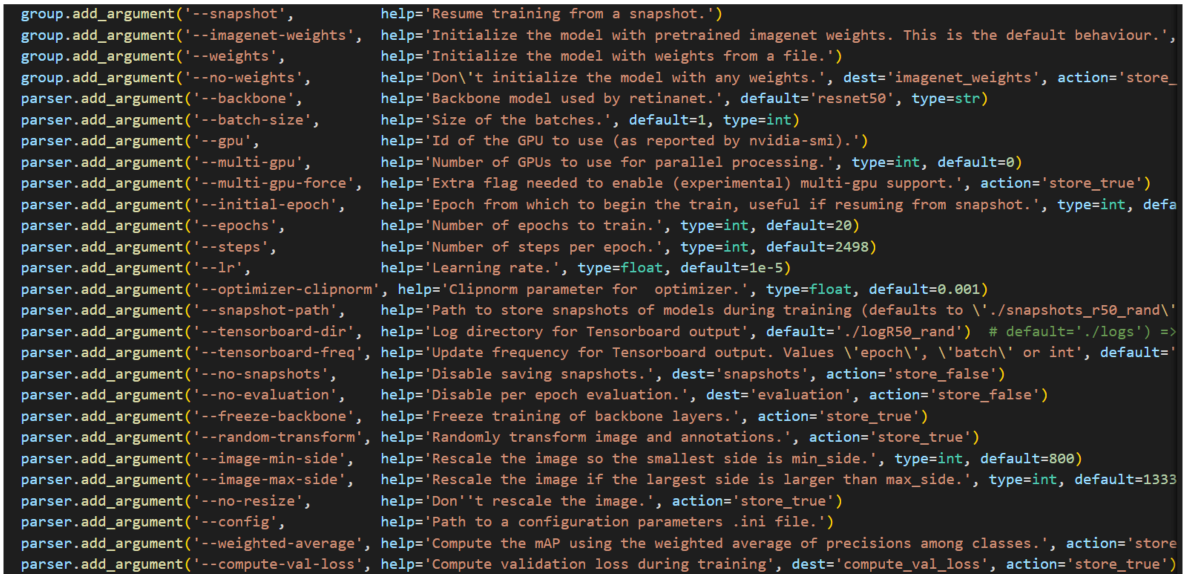
Once our training file, called final_file_train.csv, we can then proceed to the training process which will be performed using a series of python scripts that offer many different parameters to be configured but, in this case, using most of the default values for every one of them which are adjusted for image detection and recognition. This uses Keras and Tensor flow as the framework chosen to make the final training.

The hardware used for this series of training of our sample data set was a laptop with a dedicated NVIDIA GeForce RTX 2060 graphic card and Ryzen 9 CPU.

4.5. Model creation from sample dataset

We have different options in terms of the available neural network architectures, as we mentioned earlier in this chapter there was a series of script in python which offered options like Resnet50, Resnet152, RetinaNet, Densenet, VGG, etc. In this experiment we will use Resnet50 that is set as default backbone architecture within the application used.

The application or solution used for training of our model was taken from an open repository which is called keras_retinanet [34], this contains a series of python files with the classes, functions, libraries, and other utilities that finally use Keras and Tensorflow. This python application allows us to easily configure a diverse number of options immediately impacting the training process of our model. An example of the many arguments allowed to be used can be seen in figure 14.

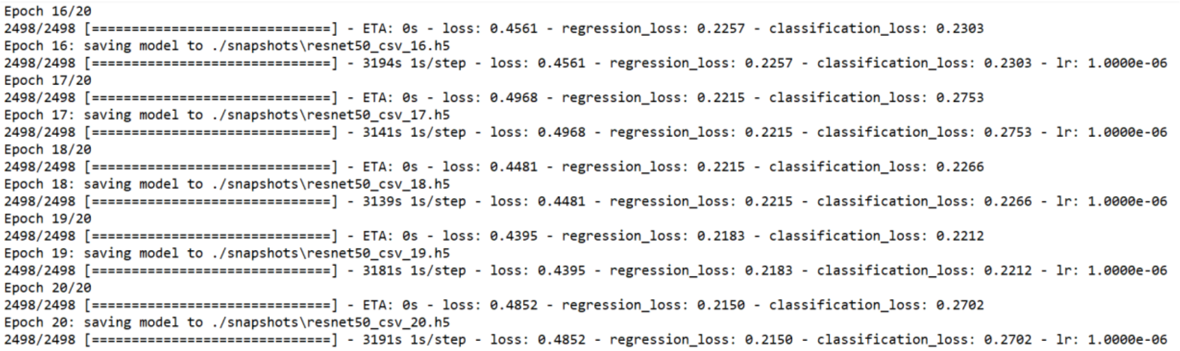


```
group.add_argument('--snapshot',          help='Resume training from a snapshot.')
group.add_argument('--imagenet-weights',  help='Initialize the model with pretrained imagenet weights. This is the default behaviour.',
group.add_argument('--weights',           help='Initialize the model with weights from a file.')
group.add_argument('--no-weights',        help='Don\'t initialize the model with any weights.', dest='imagenet_weights', action='store_
parser.add_argument('--backbone',         help='Backbone model used by retinanet.', default='resnet50', type=str)
parser.add_argument('--batch-size',       help='Size of the batches.', default=1, type=int)
parser.add_argument('--gpu',              help='Id of the GPU to use (as reported by nvidia-smi).')
parser.add_argument('--multi-gpu',        help='Number of GPUs to use for parallel processing.', type=int, default=0)
parser.add_argument('--multi-gpu-force',  help='Extra flag needed to enable (experimental) multi-gpu support.', action='store_true')
parser.add_argument('--initial-epoch',    help='Epoch from which to begin the train, useful if resuming from snapshot.', type=int, defa
parser.add_argument('--epochs',           help='Number of epochs to train.', type=int, default=20)
parser.add_argument('--steps',            help='Number of steps per epoch.', type=int, default=2498)
parser.add_argument('--lr',               help='Learning rate.', type=float, default=1e-5)
parser.add_argument('--optimizer-clipnorm', help='Clipnorm parameter for  optimizer.', type=float, default=0.001)
parser.add_argument('--snapshot-path',    help='Path to store snapshots of models during training (defaults to \'./snapshots_r50_rand\'
parser.add_argument('--tensorboard-dir',  help='Log directory for Tensorboard output', default='./logR50_rand')  # default='./logs') =>
parser.add_argument('--tensorboard-freq', help='Update frequency for Tensorboard output. Values \'epoch\', \'batch\' or int', default='
parser.add_argument('--no-snapshots',     help='Disable saving snapshots.', dest='snapshots', action='store_false')
parser.add_argument('--no-evaluation',    help='Disable per epoch evaluation.', dest='evaluation', action='store_false')
parser.add_argument('--freeze-backbone',  help='Freeze training of backbone layers.', action='store_true')
parser.add_argument('--random-transform', help='Randomly transform image and annotations.', action='store_true')
parser.add_argument('--image-min-side',   help='Rescale the image so the smallest side is min_side.', type=int, default=800)
parser.add_argument('--image-max-side',   help='Rescale the image if the largest side is larger than max_side.', type=int, default=1333
parser.add_argument('--no-resize',        help='Don''t rescale the image.', action='store_true')
parser.add_argument('--config',           help='Path to a configuration parameters .ini file.')
parser.add_argument('--weighted-average', help='Compute the mAP using the weighted average of precisions among classes.', action='store
parser.add_argument('--compute-val-loss', help='Compute validation loss during training', dest='compute_val_loss', action='store_true')
```

Figure 14: Example of available training parameters in keras_retinanet application.

Some of the parameters were running 2490 steps per epoch, 20 epochs and a starting learning rate of 0.00005.

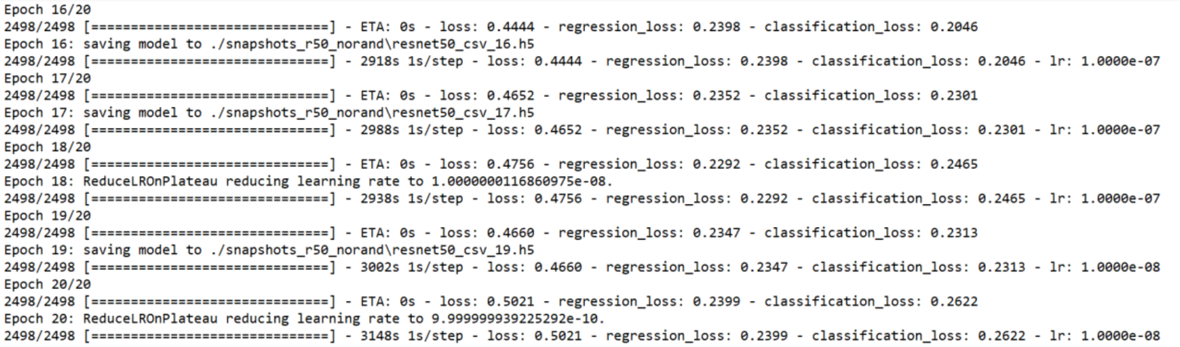
In terms of hyperparameters tunning there is an automatic adjustment of the learning rate done when there is a plateau detected, coming from the values of regression and classification loss not changing significantly for a series of epochs. This reduction in the learning rate had a positive effect of the reduction of the parameters mentioned as can be observed in the last five epochs of the training of our sample dataset in figure 15.



```
Epoch 16/20
2498/2498 [==============================] - ETA: 0s - loss: 0.4561 - regression_loss: 0.2257 - classification_loss: 0.2303
Epoch 16: saving model to ./snapshots\resnet50_csv_16.h5
2498/2498 [==============================] - 3194s 1s/step - loss: 0.4561 - regression_loss: 0.2257 - classification_loss: 0.2303 - lr: 1.0000e-06
Epoch 17/20
2498/2498 [==============================] - ETA: 0s - loss: 0.4968 - regression_loss: 0.2215 - classification_loss: 0.2753
Epoch 17: saving model to ./snapshots\resnet50_csv_17.h5
2498/2498 [==============================] - 3141s 1s/step - loss: 0.4968 - regression_loss: 0.2215 - classification_loss: 0.2753 - lr: 1.0000e-06
Epoch 18/20
2498/2498 [==============================] - ETA: 0s - loss: 0.4481 - regression_loss: 0.2215 - classification_loss: 0.2266
Epoch 18: saving model to ./snapshots\resnet50_csv_18.h5
2498/2498 [==============================] - 3139s 1s/step - loss: 0.4481 - regression_loss: 0.2215 - classification_loss: 0.2266 - lr: 1.0000e-06
Epoch 19/20
2498/2498 [==============================] - ETA: 0s - loss: 0.4395 - regression_loss: 0.2183 - classification_loss: 0.2212
Epoch 19: saving model to ./snapshots\resnet50_csv_19.h5
2498/2498 [==============================] - 3181s 1s/step - loss: 0.4395 - regression_loss: 0.2183 - classification_loss: 0.2212 - lr: 1.0000e-06
Epoch 20/20
2498/2498 [==============================] - ETA: 0s - loss: 0.4852 - regression_loss: 0.2150 - classification_loss: 0.2702
Epoch 20: saving model to ./snapshots\resnet50_csv_20.h5
2498/2498 [==============================] - 3191s 1s/step - loss: 0.4852 - regression_loss: 0.2150 - classification_loss: 0.2702 - lr: 1.0000e-06
```

Figure 15: Last five epochs of training of sample dataset.

Besides the first training using Resnet-50 architecture with our sample dataset, we also used the provided option to enable data augmentation by using a random transform generator function to extend the sample dataset of images by adding new images on the fly for the batches of images we were using to train. This function works by applying a series of transformation of every image which can include rotation, translation, shear, scaling, and flips on X and Y coordinate and all of these changes consider the corresponding effect on the bounding box coordinates which are automatically adjusted into the new transformed image. Unfortunately, the data augmentation seems to have a negative effect on the training as can be seen on last epochs classification and regression loss in figure 16.



```
Epoch 16/20
2498/2498 [==============================] - ETA: 0s - loss: 0.4444 - regression_loss: 0.2398 - classification_loss: 0.2046
Epoch 16: saving model to ./snapshots_r50_norand\resnet50_csv_16.h5
2498/2498 [==============================] - 2918s 1s/step - loss: 0.4444 - regression_loss: 0.2398 - classification_loss: 0.2046 - lr: 1.0000e-07
Epoch 17/20
2498/2498 [==============================] - ETA: 0s - loss: 0.4652 - regression_loss: 0.2352 - classification_loss: 0.2301
Epoch 17: saving model to ./snapshots_r50_norand\resnet50_csv_17.h5
2498/2498 [==============================] - 2988s 1s/step - loss: 0.4652 - regression_loss: 0.2352 - classification_loss: 0.2301 - lr: 1.0000e-07
Epoch 18/20
2498/2498 [==============================] - ETA: 0s - loss: 0.4756 - regression_loss: 0.2292 - classification_loss: 0.2465
Epoch 18: ReduceLROnPlateau reducing learning rate to 1.0000000116860975e-08.
2498/2498 [==============================] - 2938s 1s/step - loss: 0.4756 - regression_loss: 0.2292 - classification_loss: 0.2465 - lr: 1.0000e-07
Epoch 19/20
2498/2498 [==============================] - ETA: 0s - loss: 0.4660 - regression_loss: 0.2347 - classification_loss: 0.2313
Epoch 19: saving model to ./snapshots_r50_norand\resnet50_csv_19.h5
2498/2498 [==============================] - 3002s 1s/step - loss: 0.4660 - regression_loss: 0.2347 - classification_loss: 0.2313 - lr: 1.0000e-08
Epoch 20/20
2498/2498 [==============================] - ETA: 0s - loss: 0.5021 - regression_loss: 0.2399 - classification_loss: 0.2622
Epoch 20: ReduceLROnPlateau reducing learning rate to 9.999999939225292e-10.
2498/2498 [==============================] - 3148s 1s/step - loss: 0.5021 - regression_loss: 0.2399 - classification_loss: 0.2622 - lr: 1.0000e-08
```

Figure 16: Last five epochs of training of sample dataset with data augmentation.

The training process was not limited to only the first run as mentioned before, but there were a series of runs with different architectures, selecting finally Resnet-50 and Resnet-152. These two architectures were used with and without data augmentation to compare some results and looking for the best results. The use of data augmentation, as we could expect from the bigger number of images to be processed, has impact on the computational resources and therefore in the time taken for every epoch to be complete. A comparison of the time consumed by the machine used for these experiments can be seen in table 3.

Table 3: Time per epoch (in minutes) for training of sample dataset.

ARCHITECTURE	TIME (MINUTES) PER EPOCH
RESNET-50	21
RESNET-50 + IMAGE TRANSFORMATION	30
RESNET-152	41
RESNET-152 + IMAGE TRANSFORMATION	50

The processing time for every image and therefore, every epoch of the training process is noticeable different when using only the CPU vs the graphics card. The enablement of the graphic card to be available for Tensorflow was a key to be able to make more experiments as well as trying data augmentation.

In terms of the performance of our training process, the results of the classification loss were considerably low and therefore good results are to be expected when doing some tests of the final model. The model with the best results in this sample dataset training was using the Resnet50 architecture, with the mentioned reduction in classification loss, as observed in the figure 17.

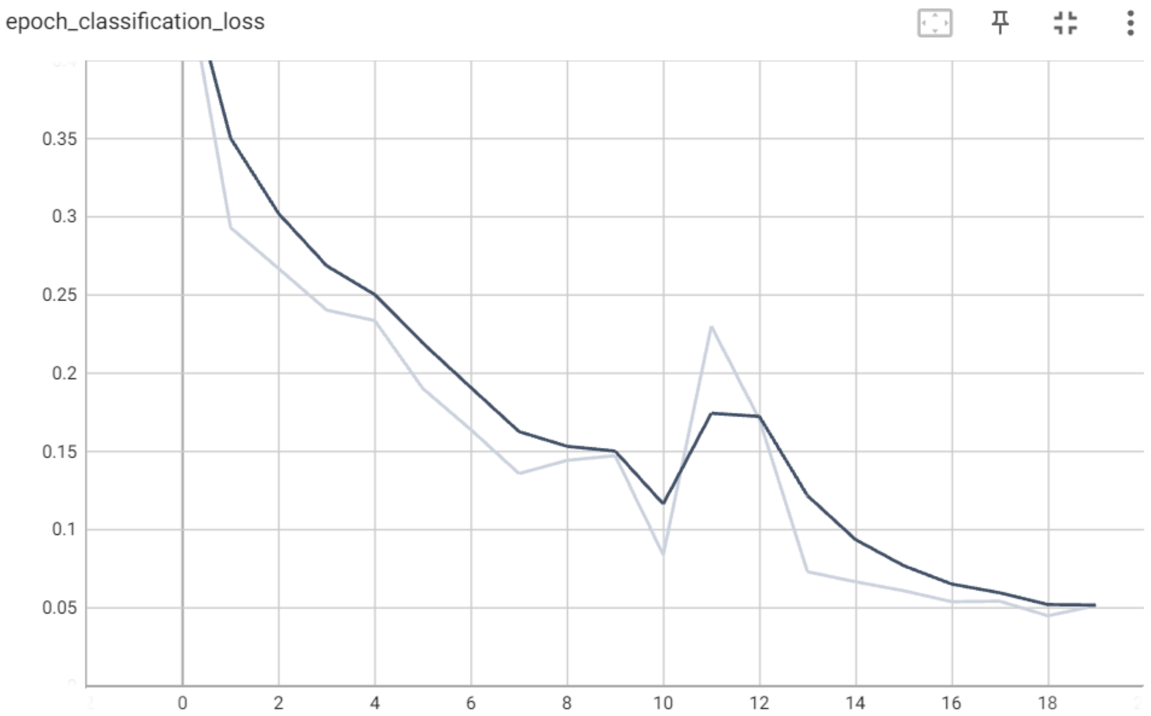


Figure 17: Classification loss value in the vertical axis across the 20 epochs in the horizontal axis.

4.6. Predictions from trained model

After this training, a file containing the trained model was generated and this was used for few predictions of local images coming from dataset used for training as well as images coming from internet which were never seen by our model in the training process. One example of the predictions done over one image containing a pedestrian crossing traffic sign can be seen in figure 18. This show us how the model does a really good job identifying both the position and the kind of object it was classifying it with an accuracy of 100% as a pedestrian crossing even with some blur in the image.

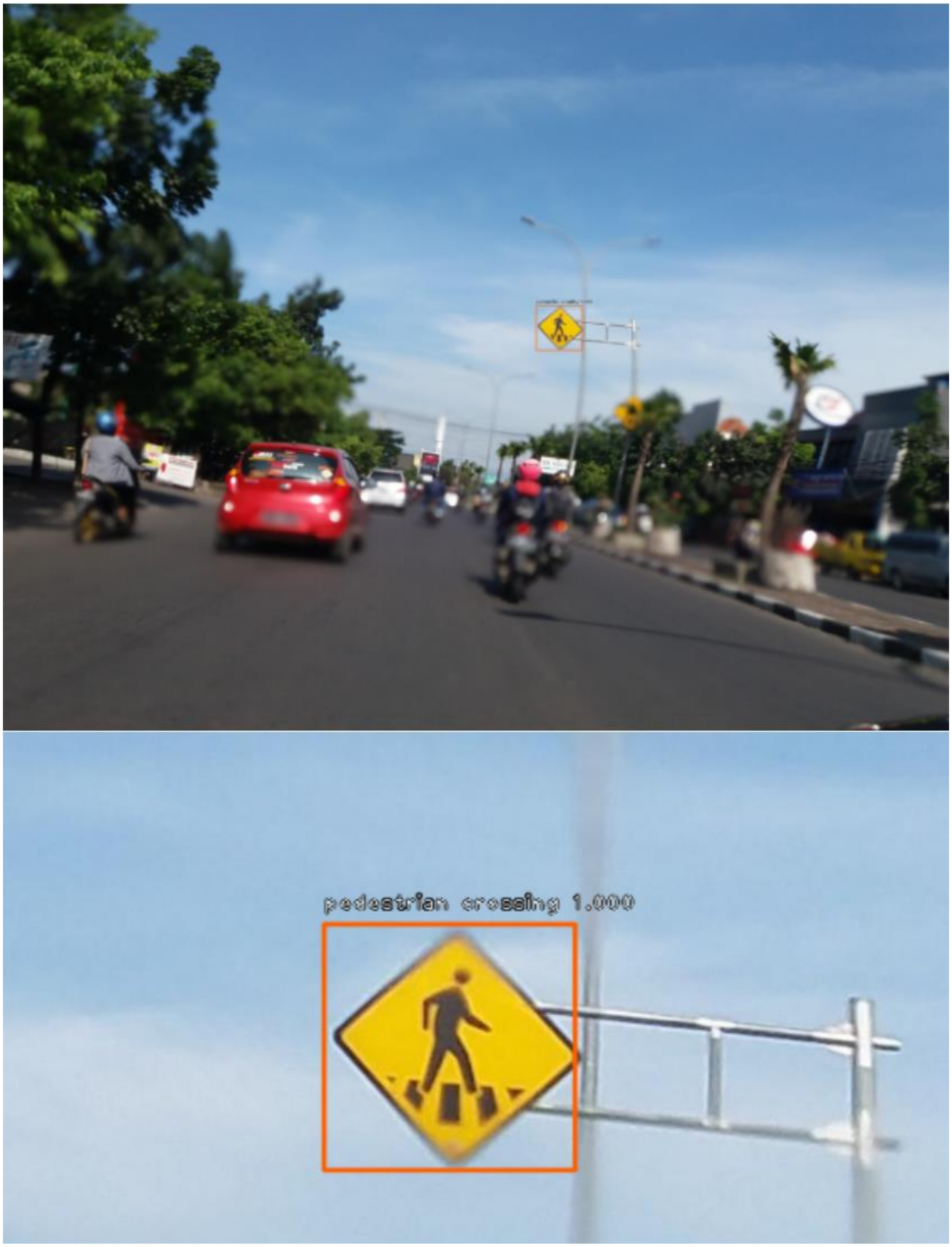


Figure 18: Detection and recognition of a pedestrian crossing within an image.

Another accurate prediction of the position and classification of *don't stop* signs could be seen in figure 19. This image belongs to the same sample dataset used to train this model and it confirms how accurate the model could be even for multiple objects in the same image. To add a bit more of complexity these traffic signs are small in proportion with the dimension of the whole image.

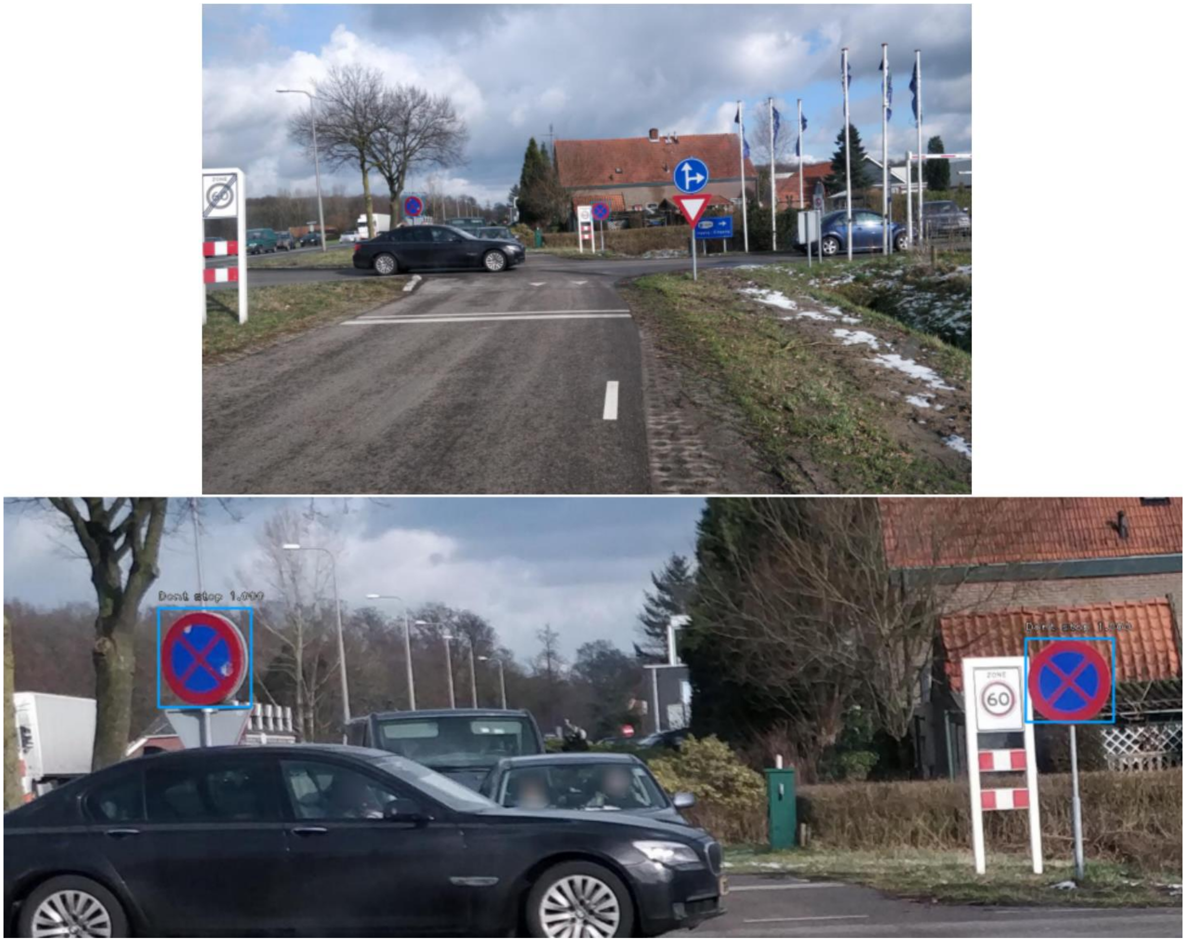


Figure 19: Detection and recognition of two Do not stop traffic signs within an image.

Finally, we have a prediction done for an image containing two Stop traffic signs which are on a side of the road and are labeled in Spanish as can be seen in figure 20. This model was able to handle both English and Spanish Stop traffic signs which another positive point for the results obtained with the Resnet-50 architecture.

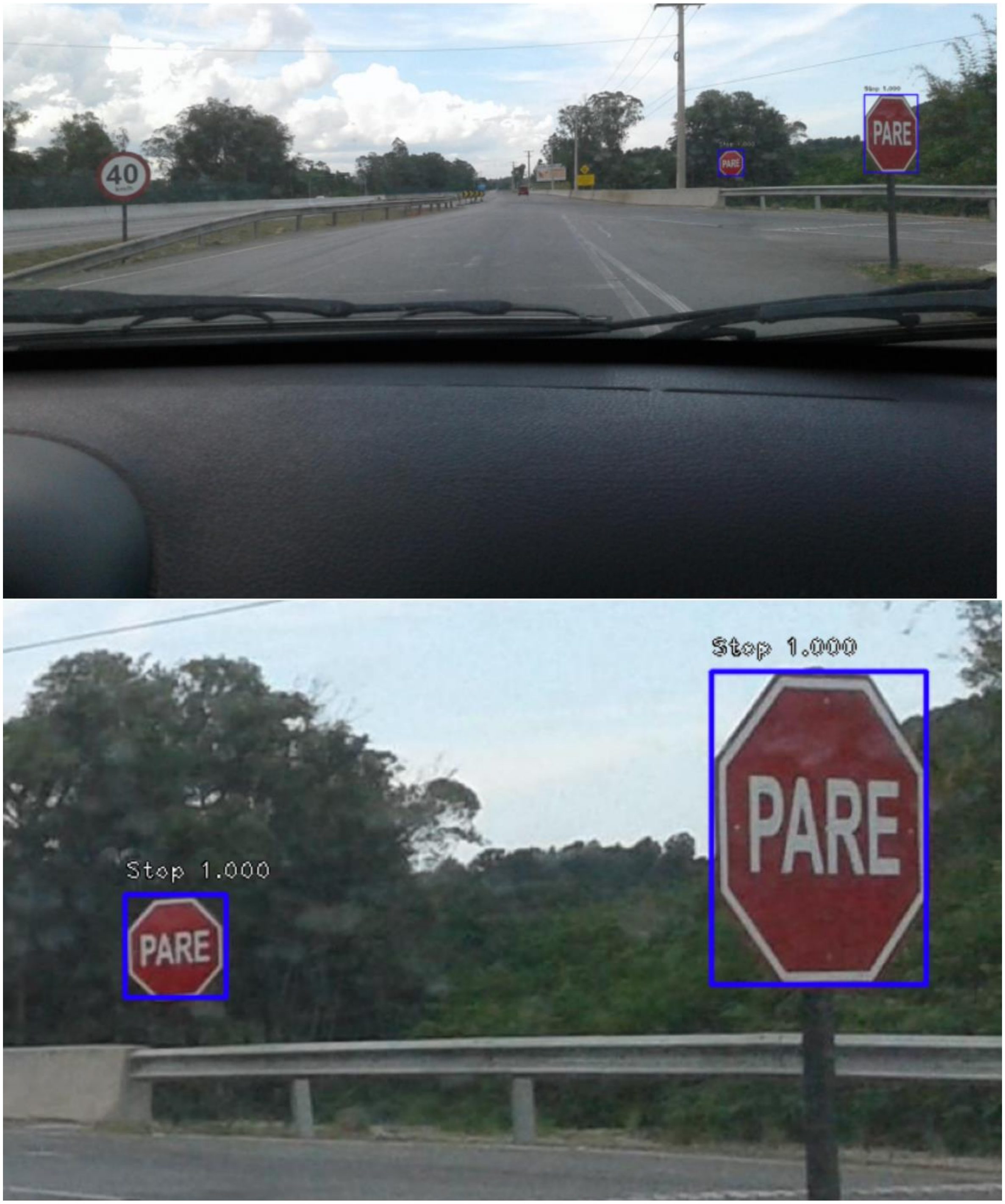


Figure 20: Detection and recognition of two Stop traffic signs within an image.

5. RESULTS AND DISCUSSION

During this chapter, we will analyze the results against the main objective of this investigation which was to develop a model capable to detect and recognize traffic sign letters with the use of a CNN architecture. As we could see in chapter 4, the results of the predictions made after getting our model from a Resnet-50 architecture were really promising and capable and able to achieve a high accuracy level in both localization of the object within the provided image and the correct classification of the traffic sign reaching a confidence value of 100% for objects in belonging to the three classes chosen on our sample data set and tested using images from this.

This experiment showed us too that regardless the really low classification loss and regression loss values, the resulting model showed an important level of overfitting that makes our trained model to behave exceptionally well with detecting and recognizing the traffic signs within images used in the training process but missing most of the images from internet or other sources containing the same traffic signs objects as from our three selected classes in the sample dataset.

This kind of problems could be mitigated using a richer dataset which includes many times the images used so; our model can be prepared for a wider range of scenarios. Another option to improve the performance of our model is the use of data augmentation that include a series of transformations made on copies of the images used in our sample dataset as rotations, changes in color, changes in shape as well as keeping the corresponding changes in the annotation’s coordinates. Both processes can make better dataset which has less possibilities to fall in overfitting issues.

This represent a big challenge since an extra effort has to be made just to increase the dataset, involving tasks such as: taking pictures of the traffic signs of interest in many places with different lighting conditions and other weather conditions, manually locating the traffic signs within every image and defining their X and Y coordinates as well as manually classifying the objects and creating an annotations file for every image with this data

6. CONCLUSIONS

- Resnet50 and Resnet152 architectures showed to perform really well for our sample dataset being able to detect a wide range of sizes and multiple number of objects in both detection and recognitions of the objects within the images within our sample dataset.
- Resnet50 particularly showed a good balance between size of the architecture and the number of operations applied to every image and the good performance as well as the reduced training time and lower resource consumption compared with Resnet152.
- The Mapillary dataset contains more than 100 thousand images and more than 300 classes, but, after training our model, we could observe that there are not enough images for every class to obtain a model that is strong enough to handle all different real-life scenarios.
- Enabling the dedicated graphic card to be used in the training process significantly reduces the time required for training process.
- Using a whole project like keras_retinanet that does an automatic adjustment in learning rates and that handles different options like different architectures epochs, dataset transformation etc. makes the training process definitely easier.
- Working with high resolution images increases the computational power requirement but it has shown to have a positive impact when detecting and identifying traffic signs that appear to be far away in the frame, this because of this small object in image having enough pixels, therefore, the CNN can extract a good amount of information from them yet.

7. FUTURE WORK

- The dataset must be increased vastly with all different scenarios, sizes and other modifications for the targeted traffic signs.
- Training and creating the model using new incoming architectures that are different that Resnet like transformer architecture.
- To do a training for a full dataset instead of a sample dataset which then could be used for a final product or application.
- To implement the trained model into a little device like an Arduino with a camera module to test the detection and recognition of the traffic signs into every one of the sampled frames trying to get a high FPS count as possible.
- To implement the voice assistant using one of the available libraries that are well trained to recognize text sentences and play them as audio which currently offer different language options.

8. REFERENCES

[1] M. S. a. K. V. Suresh, «Automatic traffic sign detection and recognition in video sequences,,» de *nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, 2017.

[2] M. Z. A. T. B. a. A. D. P. Dhar, «Indonesian traffic sign detection and recognition using color and texture feature extraction and SVM classifier.,» de *2018 International Conference on Information and Communications Technology (ICOIACT)*, Yogyakarta, 2018.

[3] C. O. a. G. M. Jr., «sciencedirect,» 1999. [En línea]. Available: https://www.sciencedirect.com/science/article/pii/S0039625799000351?via%3Dihub.

[4] IIEG, «iieg.gob.mx,» 2019. [En línea]. Available: https://iieg.gob.mx/ns/wp-content/uploads/2019/07/Ficha-Informativa_Accidentes-de-tr%C3%A1nsito-por-clase-de-accidente-2010-2018.pdf.

[5] S. S. R. R. D. S. D. M. a. P. H. A. Vennelakanti, «Traffic Sign Detection and Recognition using a CNN Ensemble,» de *IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, 2019.

[6] P. G. a. D. L. Y. Sun, «Traffic Sign Detection and Recognition Based on Convolutional Neural Network,» Hangzhou, China, 2019.

[7] Y. I. a. Y. C. R. Hasegawa, «Robust Detection and Recognition of Japanese Traffic Sign in the Complex Scenes Based on Deep Learning,» de *IEEE 8th Global Conference on Consumer Electronics (GCCE)*, Osaka, Japan, 2019.

[8] L. S. R. P. a. S. V. S. P. Rajendran, «Real-Time Traffic Sign Recognition using YOLOv3 based Detector,» de *10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Kanpur, India, 2019.

[9] M. T. Islam, «Traffic sign detection and recognition based on convolutional neural networks,» de *International Conference on Advances in Computing, Communication and Control (ICAC3)*, Mumbai, India, 2019.

[10] S. G. P. a. M. Panda, «Traffic Sign Recognition Using Distributed Ensemble Learning,» de *Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, 2020.

[11] V. I. a. B. P. B. Novak, «Yolov3 algorithm with additional convolutional neural network trained for traffic sign recognition,» *2020 Zooming Innovation in Consumer Technologies Conference* , 2020.

[12] Federal Highway Administration, «Manual on Uniform Traffic Control Devices (MUTCD).,» 2021. [En línea]. Available: https://mutcd.fhwa.dot.gov/. [Último acceso: 2021].

[13] S. d. C. y. Transportes, «Manual de señalizacion vial y dispositivos de seguridad,» 2014. [En línea]. Available: Secreatria de comunicaciones y transportes, Manual de señalizacion vial y dispositivos de seguridad, 2014. [Online]. Available: http://www.sct.gob.mx/fileadmin/DireccionesGrales/DGST/Manuales/NUEVO-SENALAMIENTO/manualSenalamientoVialDispositivosSeguridad.

[14] I. Corporativa, «www.iberdrola.com,» [En línea]. Available: https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico. [Último acceso: 28 Nov 2020].

[15] R. Alake, «Understand local receptive fields in convolutional neural networks,» 03 Nov 2021. [En línea]. Available: https://towardsdatascience.com/understand-local-receptive-fields-in-convolutional-neural-networks-f26d700be16c#:~:text=The%20local%20receptive%20field%20is,neural%20network%20for%20character%20recognition. [Último acceso: 11 Nov 2021].

[16] R. Sagar, «What is weight sharing in deep learning and why is it important,» 24 Jul 2020. [En línea]. Available: https://analyticsindiamag.com/weight-sharing-deep-learning/. [Último acceso: 11 Nov 2021].

[17] «Multi-Layer Neural Network,» [En línea]. Available: http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/. [Último acceso: 11 Nov 2021].

[18] M. Hassan, «VGG16 - convolutional network for classification and detection,» 24 Feb 2021. [En línea]. Available: https://neurohive.io/en/popular-networks/vgg16/. [Último acceso: 18 Nov 2021].

[19] «ResNet-50 convolutional neural network - MATLAB,» [En línea]. Available: https://www.mathworks.com/help/deeplearning/ref/resnet50.html#:~:text=ResNet%2D50%20is%20a%20convolutional,%2C%20pencil%2C%20and%20many%20animals. [Último acceso: 11 Nov 2021].

[20] «ResNet-152 trained on imagenet competition data,» [En línea]. Available: https://resources.wolframcloud.com/NeuralNetRepository/resources/ResNet-152-Trained-on-ImageNet-Competition-Data/. [Último acceso: 11 Nov 2021].

[21] O.-. O. D. Science, «Overview of the YOLO Object Detection Algorithm,» 2018. [En línea]. Available: https://medium.com/@ODSC/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0.

[22] S. Elfouly, «R-CNN (Object Detection),» Medium, 19 November 2020. [En línea]. Available: https://medium.com/@selfouly/r-cnn-3a9beddfd55a. [Último acceso: 10 November 2021].

[23] S. Elfouly, «Introduction: Fast R-CNN (Object Detection),» 2019. [En línea]. Available: https://medium.com/@selfouly/part-2-fast-r-cnn-object-detection-7303e1988464. [Último acceso: 28 Nov 2020].

[24] A. Mishra, «Metrics to evaluate your machine learning algorithm,» 28 May 2020. [En línea]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234. [Último acceso: 11 Nov 2021].

[25] «What is numpy?,» [En línea]. Available: https://numpy.org/doc/stable/user/whatisnumpy.html. [Último acceso: 11 Nov 2021].

[26] «About pandas,» [En línea]. Available: https://pandas.pydata.org/about/. [Último acceso: 11 Nov 2021].

[27] T. Flow, «Tensor Flow, Una plataforma de extremo a extremo de código abierto para el aprendizaje automático,» [En línea]. Available: https://www.tensorflow.org/?hl=es-419. [Último acceso: 28 Nov 2020].

[28] G. Oshri, «Introducing TensorBoard.dev: A new way to share your ML experiment results,» [En línea]. Available: https://blog.tensorflow.org/2019/12/introducing-tensorboarddev-new-way-to.html?hl=es-419. [Último acceso: 11 Nov 2021].

[29] Keras, «About keras,» [En línea]. Available: https://keras.io/about/. [Último acceso: 28 Nov 2020].

[30] T. A. I. Team, «Understanding Pascal VOC and coco annotations for object detection,» 05 Jun 2020. [En línea]. Available: https://towardsai.net/p/machine-learning/understanding-pascal-voc-and-coco-annotations-for-object-detection. [Último acceso: 11 Nov 2021].

[31] R. Khandelwal, «COCO and Pascal VOC data format for Object detection,» 2019. [En línea]. Available: https://towardsdatascience.com/coco-data-format-for-object-detection-a4c5eaf518c5.

[32] V. Sichkar, «Traffic Signs Preprocessed,» 31 Aug 2019. [En línea]. Available: https://www.kaggle.com/valentynsichkar/traffic-signs-preprocessed.

[33] C. I. J. S. a. M. S. J. Stallkamp, «German Traffic Sign Recognition Benchmark GTSRB,» 10 05 2019. [En línea]. Available: https://sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/published-archive.html.

[34] S. M. AB, «Mapillary Traffic Sign Dataset,» [En línea]. Available: https://www.mapillary.com/dataset/trafficsign. [Último acceso: 17 Apr 2021].

[35] GRAM, «Traffic Signs UAH Dataset,» [En línea]. Available: https://gram.web.uah.es/research.html. [Último acceso: 17 Apr 2021].

[36] Fizyr, «Keras-retinanet/keras_retinanet at Main · fizyr/keras-retinanet,» 28 July 2020. [En línea]. Available: https://github.com/fizyr/keras-retinanet/tree/main/keras_retinanet. [Último acceso: 15 October 2022].