UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Detecting and Tracking Horses Using Deep Neural Networks

Trabajo Fin de Máster

Máster Universitario en Ingeniería Informática

Autor: Alejandro Delgado

Tutor: Miguel Rebollo Pedruelo

Tutor Externo: Dr. Huseyin Kusetogullari

2018 - 2019

Abstract

Deep Learning is a trend today. Algorithms such as neural networks enable the development of projects such as the following. The purpose of this project is the detection and tracking of horses within a controlled environment such as stables. For this purpose a dataset of 10,000 images has been created where there are dataframes of horses in different positions. With this dataset a CNN has been trained using a framework called YOLO. This software allows the detection of these horses and people in real time. To know if this framework fulfills the objective of the project, a series of tests have been carried out where the final quality of the NN can be appreciated.

Keywords: Object Tracking, Deep Learning, YOLO Framework, Machine Learning, Artificial Intelligence

Contents

List of Figures

Acronyms

AI	Artificial Intelligence.
ANN	Artificial Neural Network.
CNN	Convolutional Neural Network.
DL	Deep Learning.
GPU	Graphics Processing Unit.
ML	Machine Learning.
NN	Neural Network.
RNN	Recurrent Neural Networks.
SVM	Support Vector Machine.

1 Introduction

According to [4, p. 28], machine learning is a fundamental element within Artificial Intelligence. Machine Learning (ML) permits computational algorithms to learn from a previous input and training. One of the main algorithms used in ML to provide intelligence are NN. As it will be explained in the Analysis section and according to [4], a NN is a group of nodes that construct a computational model based on the mathematics and statistics. The purpose of these algorithms is to classify new data taking into consideration previous knowledge. These models are made of a structure composed of inputs, hidden layers and outputs, but what is Deep Learning (DL)? When a NN has a number of hidden layers greater than four or five, it becomes a deep neural network. Such deep neural networks have the capacity to learn to solve problems and the learning is called deep learning. This learning can be categorized into supervised, unsupervised and semi-supervised [5].

The world of DL is increasing more and more. This new field of computing is becoming increasingly important today. NN technology opens up a world of possibilities for classifying all types of data, whether text or images [6; 7; 8]. There are different papers where the existence of problems is solved with the application of a DL technique. In [9], using DL creates a model capable of recognizing and tracking vehicles in real time. It is a Convolutional Neural Network (CNN) that has been created using the same framework that will be used in this project. DL has been applied in different applications and there are different studies where a DL can detect different types of features such as cancer cells [10]. There are many applications in the world of medicine that this technology has been useful for advancement in fields as complicated as this type of disease. But others must also be mentioned. DL is part of artificial intelligence and it provides so many advantages to the human life such as time consuming, making life easier etc. Currently there are different devices such as Alexa [30] that allow us to make life easier. Alexa [11] is a virtual assistant which allows for example, to reproduce the music that we want, to inform us of the weather or even to control the lights of our house. All this is done using ML techniques in natural language generation and processing [12].

The possibilities offered by these algorithms are encouraging big companies such as Amazon or Google to design tools that allow anyone to solve ML problems. Google AI is a platform that allows to find ML models for any person or company. In researchers such as these [13; 14; 15] you can see some examples of these applications. Also Amazon AWS has pre-trained services for computer vision, language, recommendations, and forecasting [11]. This facilitates progress in the field of Artificial Intelligence (AI).

Once the various solutions and improvements offered by the DL are known, the view will be focused on the NNs prepared for images. There are many frameworks to create this type of NNs, for example Keras, Tensorflow. These libraries are available in Github [16; 17] and are open-source and offer a set of abstractions more intuitive and high level making it easier to develop PD models. Keras offers support for CNN and Recurrent Neural Networks (RNN). But in this project a real time detection is needed, for this reason YOLO has been chosen to carry out this CNNs. There are different papers [3; 9] where studies are carried

out on the performance and speed of this framework.

It could be said that thanks to this technology, it is possible to improve people's quality of life. But not everything is in favour of new technologies. According to [26] the subject of ethics also plays a very important role in AI. Questions like: Can a robot occupy my job? or Can a robot decide my vote in an election?, creates serious doubts as to how far one is disposed to go with technology. There are different open debates about ethics in IA [18; 19; 20]. Possibly in the future it will be necessary to reconsider whether it is good for the human race to continue advancing in the field of informatics in general and in the field of AI in particular.

Leaving to one side whether it is ethical or not, the main motivation that has made this project is that contribute new developments and improvements in the field of AI. Also researching and experimenting with one of the most powerful technologies and trends such as DL. It is also important to discuss the main element that constitutes the dataset in this project, it is horses. In this project, horses will be detected and tracked on the video frames using DL methods such as YOLO. The main advantage of the applied methodology is that it provides a real time detection and tracking under varying conditions. These conditions can be dark, rainy, sunny and others. These changing conditions will increase the complexity of the problem in this project. Although this project will not focus too much on the approach to animal behaviour but it is important to know some minimal knowledge. In [21], different terms and concepts are described that are useful to study the behaviour of horses within stables.

But, what is the aim of this project?. The main objective of this project is the detection and tracking of horses within their respective stables. This project has been carried out together with a company to cover some needs and objectives. The company Videquus (`http://videquus.se`) is focused on the monitoring and care of horses. Videquus allows to know at any moment the state of the horse. With a single click a client can connect to the internal camera of the stable and check how your animal is. Some of the possibilities offered are to send these videos to a veterinarian, so that he can give more precise diagnoses on the health of the horses. What does the company want with this work?, mainly seek the detection of horses and people inside the stable. Apart from detecting the horses, it also wants to be able to save the different movements they do. Saving these patterns opens up the possibility of reaching more advanced and interesting objectives for Videquus in other projects. Some of these objectives can include:

- Horse rolling around
- Horse with colic
- Horse getting stuck
- Horse breathing heavily or abnormal
- Horse being still
- Horse laying down and standing up

But it should be noted that the above objectives do not belong to this project and these behaviours are done manually by owners of the horses and veterinarians. In the Conclusions section, some hints are given on how to achieve them in further developments based on this model. The aim of the development shown in the later sections is the creation of a model with an CNN that is able to detect and track horses. In addition, other indirect objectives such as acquiring experience in this area of technology have also been achieved. Also mention the creation of a large data set on horses, which can be very useful for other developments.

With this description of the objective of the project, several studies can be found that have great similarity on the tracking of animals or objects in general. In [22], a detection system is proposed to monitor wildlife and detect wild animals from very disordered natural dataset. These characteristics of the images are extracted using the Deep Convolutional Neural Network (CNN). The system has been implemented to use two CNN models: VGGNet and ResNet. A standard database of trap cameras has been used to feed these models. Another document related to this the purpose of this project is [23]. In this paper is developed a mechanism for the automatic detection and recognition of individuals of different species with patrons such as tigers, zebras and jaguars. For this purpose it is used the Faster-RCNN object detection framework to detect animals in images. AlexNet is also used to extract more features from the side of the animal and train a logistic regression (or Support Vector Machine (SVM)) classifier to recognize the individuals. It should be noted that [23] provides perfect detection results on images of tigers captured by the camera. To end with related works, mention [24]. This thesis develops an object detector designed for executing trackers on less powerful hardware, for example a mobile phone. In other words, minimize computational complexity of a tracker while maintaining a respectable accuracy. This can be interesting because the same hardware that has the camera could detect the horses without the need of great computing capacity.

Finally, the next report is divided into different sections in order to explain in the most appropriate manner each of the phases that have been carried out in the project. In the first section there is an Analysis of the two main elements that constitute the development. These are the dataset and the technology used. Once the required knowledge is acquired, proceed in the Methodology section to explain the different steps and decisions that have been taken to reach the final target. With the development already working, the first results are executed and a brief analysis of them is made in the Results section. Finally, it is discussed whether the expected results are met, possible improvements and future work in the Conclusions section.

2 Analysis

The first step that has to be done in any process on data is the choice of a dataset. With this set the realization of a cleaning and normalization will depend on the state of the data. To carry this out we have to understand the entire data set perfectly, thus choose the correct model. When the clean and prepared dataset is available, the method is applied which, as explained below, is YOLO.

2.1 Problem Definition and Description

Before starting to discuss the technologies and algorithms to be used, it is important to know what the problem is. As introduced in the previous section, the main objective is detecting and tracking horses in stables. But, why does this problem or need occur?. The company Videquus has thousands of images that it wants to make the most of. In this manner, it offers more advantages to its customers than its competitors. One of these advantages is that of offering alert services to those about the current state of the horses. This requires the creation of a horse tracker and human detector. One of the technologies that allow this is the NNs. This ML method makes it possible to detect any object that is previously trained, even in real time. Therefore the objective is the creation of an NN model that detects horses in a real time environment with the highest possible accuracy. Finally it is also important that these detections should be able to save in text format such as JSON, so that you can apply other MLs techniques and extract knowledge about them.

2.2 Dataset

The data set used in this project is obtained from Videquus. This company is oriented to the care of the horses in the stables. Therefore, it has thousands of videos of horses in stables. The videos are recorded with different quality and image resolution. These captures are the dataset that will be used in the deep learning algorithm. These videos are hosted in a repository of S3 Amazon. The company Videquus has 200 folders of the cameras that have recordings. Inside each folder it is possible to find thousands of events, where each event means that there has been a sudden movement of the horse or other object inside the stable and therefore triggers an alarm that saves that movement. The selection process and the processing of these videos are explained in the Methodology section.

It is important to mention the quality available. Many of the images that are extracted from the videos do not have high quality and this negatively affects the learning of the YOLO. Figure 1 shows an example of one of the images that are part of the data set. Once an image has been extracted from each video, a data set of about 10,000 images is obtained, which will be sufficient to obtain a good performance of the deep neural network model.

Figure 1: An illustration of a horse in a frame.

Once the images are chosen with the JPG file format, the labeling of horses has been processed. The labeling process consists of selecting in each one of the images where the object to be classified is located. For this task the LabelImg application is used [25]. It is written in Python and uses Qt[1] for its graphical interface. It creates a file with the same name as the image and saves the annotations in XML or YOLO format. This last format is that it will use for this project. This tool also has hot-keys. This is a great advantage over other programs, because the labeling task is long and laborious. These features help to increase speed. As explained in [25], mainly the keys used are: "a" to go to the previous picture, "w" to show the selector box and "d" to go to the next image to label. This added to the auto save option make this application the one chosen to label these 10,000 images. In figure 2 it can be seen a capture of LabelImg in the process of labeling an image.

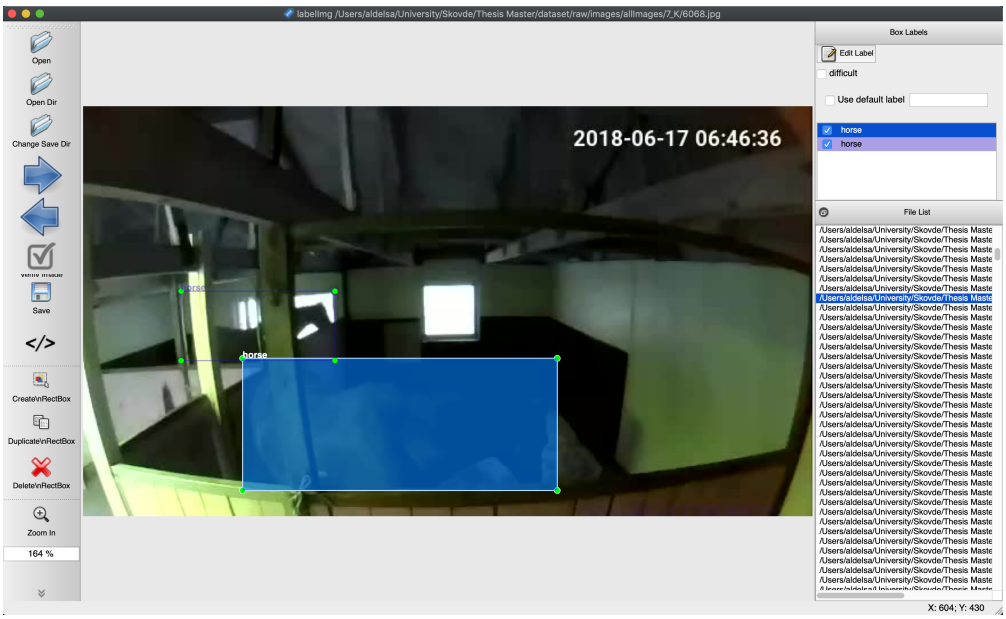

Figure 2: Using LabelImg program to label the horses in the frames.

[1]Qt is a free and open-source widget toolkit for creating graphical user interfaces.

As shown in the figure above, in the image dataframes can be found one or more horses. The labeling has been made of all the horses that could be located in an image. Also for this project people inside the stable have been tagged. As explained in the conclusions section, the detection of these people by CNN is more difficult than that of horses. This is due to the unavailability of a number of photographs of people captured by stable cameras. There are approximately 500 images with one or several people labeled, out of the 10,000 that compose the complete dataset. In contrast the rest of the images (9,500) has the existence of a horse or several. Also to mention that for the creation of this dataset any dataframe that does not contain a horse or a person has been eliminated.

The quality of the images is very important to create a model with a precision high enough to satisfy the objective of the project. In this dataset can be found different dataframes where the quality of the images is very low. In figure 3, there is an example where the brightness is very high and the position of the horse cannot be clearly appreciated. In contrast there are more images where the stable is very dark and can not be seen where the horse is. In figure 4, this is the effect that can be seen.

Figure 3: Captures stable with brightness.

Figure 4: Captures stable with darkness.

The YOLO framework has some parameters within the configuration file to improve the input images. Some of these parameters are "saturation" or "exposure". These values influence the saturation and exposure of the images. There is also another named "channels" to choose a channel of the different RGB colors that exist in an image [26]. It is possible to improve the quality of the images by changing these parameters. But also to mention that if there are images with a lot of light and other very dark ones, this change would not be entirely appropriate. For this project these values have been preserved by default.

Apart from these problems, images have also been detected where there are insects in front of the camera and therefore the horse is not appreciated enough. All these images made it impossible for CNN to detect horses correctly and it is more difficult to achieve good performance. For this reason, it can not be completely guaranteed that all images of the training dataset have the best conditions. Although several revisions of the dataset images have been done, it is possible that there is some noise that negatively influences the NN. The search for these images is one of the improvements that could be made in the future.

2.3 Machine Learning Methods

2.3.1 Neural Network and Convolutional Neural Networks

This section explains in detail the machine learning methods on which this project is based. Before starting specifically with You Only Look Once (YOLO), some knowledge of NNs in general will be introduced. To begin, what is a neural network?. The simplest definition of a NN, more properly referred to as an Artificial Neural Network (ANN), is provided by the inventor of one of the first neurocomputers, Dr. Robert Hecht-Nielsen. He defines a neural network as:

"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs."

In "Neural Network Primer: Part I" by Maureen Caudill, AI Expert, Feb. 1989

In other words, it is a set of layers and nodes connected to each other that divide the work to achieve a final result together. These nodes implement activation functions to carry out an outcome [4]. According to [27], in the future with NNs it will be possible to achieve objectives such as: robots vision, feelings, and prediction of their surroundings, improved stock prediction, common usage of self-driving cars or even composition of music. This can be achieved by improving existing NN architectures combined with very high computational power. This makes the NN algorithm one of most important methods in the field of artificial intelligence [4]. The following figure shows the general scheme of an artificial neural network.

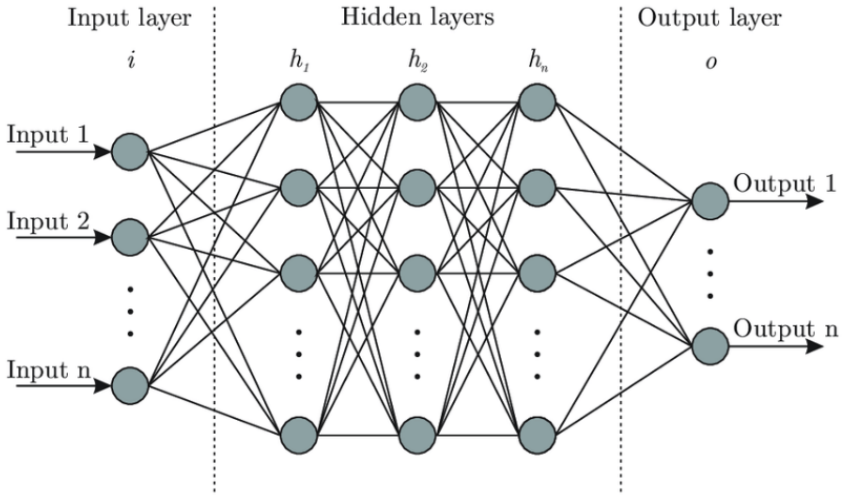

Figure 5: Architecture of NN with one input layer, 3 hidden layers and one output layer.

With this introduction to neural networks it is possible to go one step further and start discussing about CNN. As explained in [3], convolutional networks also called CovNet, describes CNN as the concept of biologically inspired hierarchical feature detectors. It can learn highly abstract features and identify objects efficiently. Basically it is a NN designed for image processing. CNNs are widely being used in various domains due to their remarkable

performance such as image classification, object detection, face detection, speech recognition, vehicle recognition, diabetic retinopathy, facial expression recognition and many more. According to [3], a general model of a CNN consists of four components: convolution layer, pooling layer, activation function, and fully connected layer [28]. The functionality of each component has been shown in the figure 6.

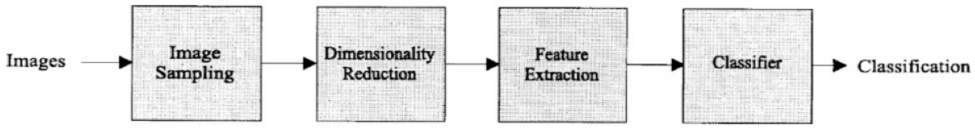

Figure 6: General process of object classification in images.

An image to classify is given to the input layer and the output is the predicted class label calculated using characteristics extracted from the image.

2.3.2 YOLO Framework

With the previous knowledge explained it is possible to start talking about YOLO's . You Only Look Once or YOLO is real-time object detection system at 45 frames per second based in a CNN. According to the official website [29], YOLO applies the model to an image in multiple locations and scales. High score areas of the image are considered detections. Applies a single NN to the entire image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted likelihood. A priori it may look like a normal object detector network but, why use this framework for this project?. The main reason is the speed at which it detects objects in real time. YOLO is extremely fast. According to [3], this means real-time video can be processed with less than 25 milliseconds of latency. In addition, YOLO achieves more average accuracy than other systems in real time. In figure 7, compared with YOLO in its last version.

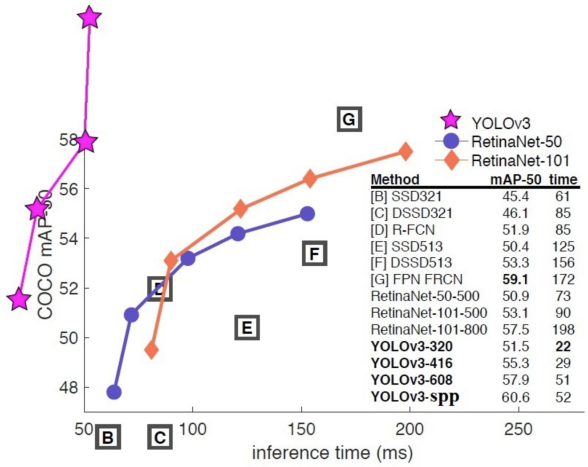

Figure 7: Performance of the YOLOv3 [1]

YOLOv3 is the fastest with a detection speed of up to 22ms. If one talks about performance it is possible to observe that it is not the one that has more precision but it is in the top 5. In order to understand the detection performance of the YOLOv3, mAP (mean Average Precision) has been used which is explained later in the Methodology section. For now just to know that the higher the mAP is, the more accuracy the object detection has. In case the priority is a higher performance over speed it can be seen that the framework SSD321 has the highest precision [30]. Also, it is necessary to mention that for the tests carried out in figure 7 a dataset named COCO has been used. As explained in [31], this dataset contains images with 91 different object types that would be easily recognizable by a 4 year old person. This dataset consists of totally 2.5 million labeled instances in 328,000 images.

One of the possible reasons that make YOLO one of the fastest networks is also due to the internal layered architecture it has. According to [3], the detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 x 1 convolutional layers reduce the features space from preceding layers.

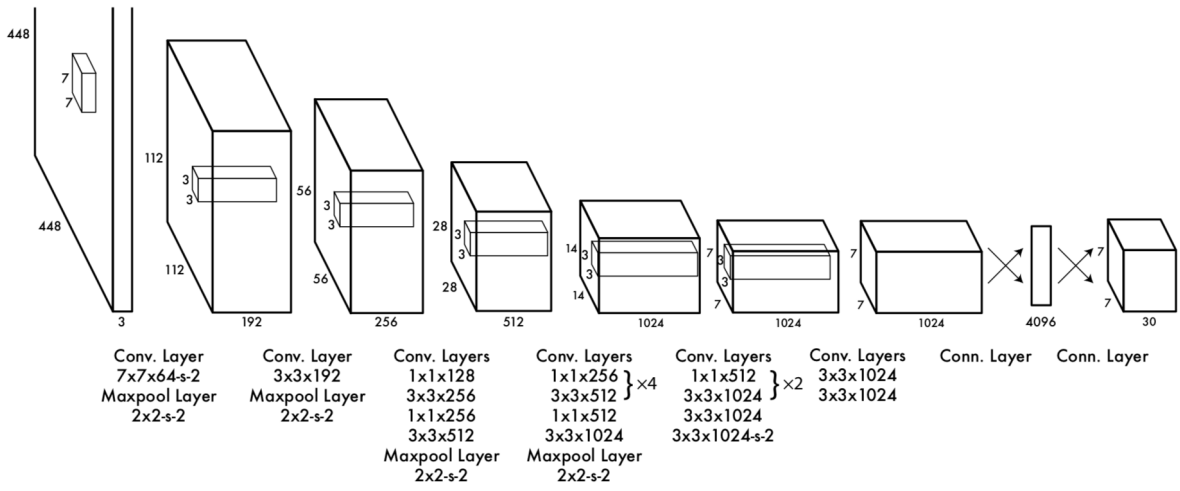

Figure 8: Illustration of YOLO architecture.

However for this project the architecture of Darknet53 has been used. This architecture has 53 more layers stacked on the standard Darknet, giving a totally convolutional underlying architecture of 106 layers. This is the reason behind the slowness of this architecture. In figure 9, an illustration can be seen.

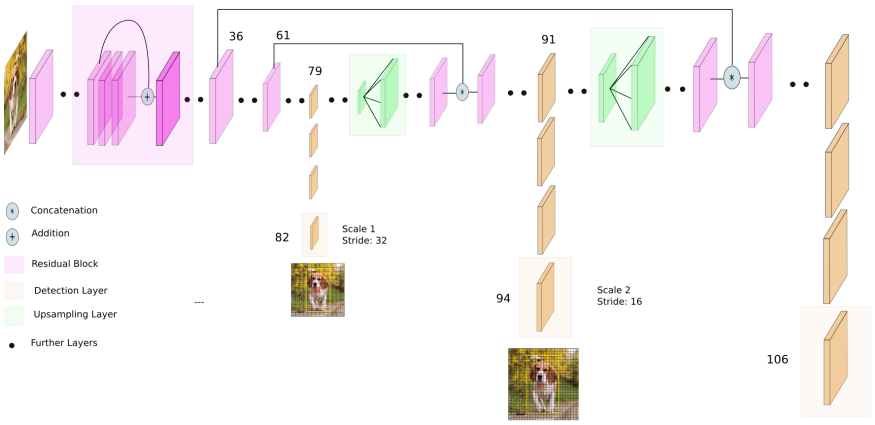

Figure 9: Illustration of Darknet53 architecture [2]

This project not only includes this multilayer architecture, there is also a version called YOLO-tiny. According to [32], this version uses 9 convolutional layers and 6 pooling layers (Figure 10). It is faster because it has less layers but possibly not as accurate as in figure 9 architecture. Having a more reduced architecture the input image resolution is also lower. This project has also been tested with this architecture. The outcomes can be seen in the Results section.

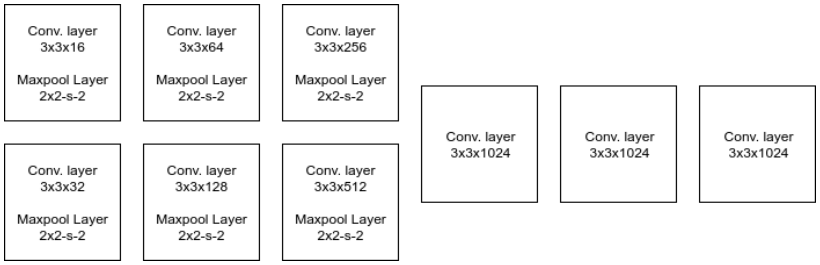

Figure 10: Illustration of YOLO-tiny architecture [3]

As it has been mentioned previously, CNNs are designed to process images. This image processing can be very heavy for computer CPUs. As described in [33], basically an image is a set of data in matrix format. CPUs are not designed for this large amount of data processing, but Graphics Processing Units (GPUs) are. For that reason a GPU becomes a fundamental component within a CNN. YOLO supports running on multiple GPUs at the same time. This improvement achieves high training and test speed. Also to say that in the case of wanting to detect objects on a file in video format, YOLO only allows to do it with a GPU and in addition the OpenCV library[2]. Besides, if the intention is to use data in image format, it is not necessary to have one. To be able to use the GPU device it is necessary the CUDA library. This library is the one that permits YOLO to communicate with the GPU hardware device.

[2]OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.

Finally, to note that there are different versions of this framework: YOLOv1, YOLOv2 and YOLOv3. As well mentioned in [1], in YOLOv3 there are a lot of little design changes to make it better. It is a little bigger than the last version, but also more accurate. For this reason it has been decided to use this version of YOLOv3, since it will possibly generate better results than its predecessors. It can not be assured because this project has not been compared with previous versions. All the YOLOv3 framework is available over the official github repository [1]. But for this project a forked version of this repository [34] has been used. Why has this unofficial repository been used? The answer to this question is simple, the repository [34] offers options that make it possible to go further in this project. One of the options that becomes indispensable is to be able to work without a graphical environment. This is essential since, as will be commented in the Methodology section, this development is based on a virtual machine from Microsoft Azure.

3 Methodology

The workflow implemented for this project is shown in figure 11.

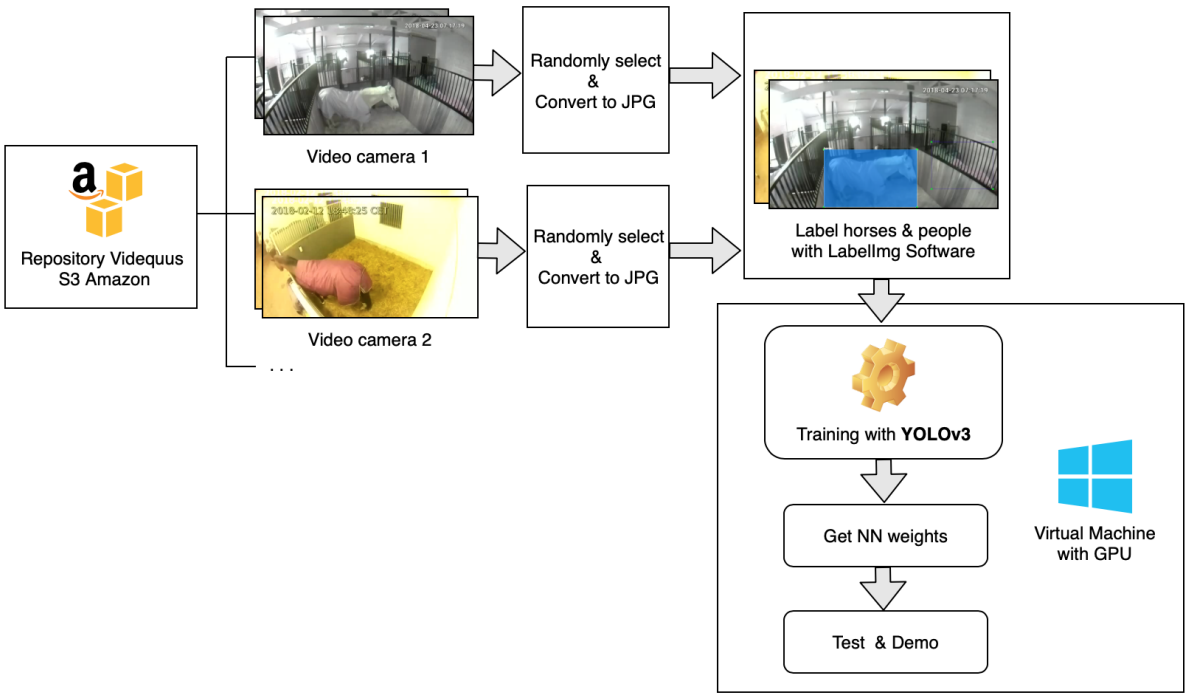

Figure 11: Project workflow.

In the figure above, it can be seen the development scheme that will be carried in this project. First, the extraction of the videos from the Amazon S3 repository. This extraction of videos is done randomly. Once the videos are downloaded, extract a dataframe from these videos and convert in image format. With the images of the horses ready, it proceeds to the labeling to create the training dataset. The second step is to train the CNN, as shown in the figure 11, this step is done in a virtual machine of Azure and using YOLOv3. Finally, when the training has finished, YOLO weights are obtained and a accuracy test is performed.

3.1 Preparing the data

All the necessary data to be able to carry out this project are in an Amazon S3 repository. The company Videquus has a structure of directories where many videos taken of the horses can be found. The first step is to go through each of the directories with cameras. Cyberduck software [35] has been used to do this, which allows to access and download the videos from this repository. Within each folder there are also folders that correspond to events that have happened. It defines an event when the horse has moved or there is someone inside in the barn. Videquus detects these events by measuring the difference in pixels between one frame and another. When there is a significant difference between one frame and another, an alarm is activated that saves the event. In order to create our dataset, videos of several

events are chosen randomly, in this way different positions of the horses are obtained. This allows to create a dataset with a great variety of positions.

For this project, videos of about 100 cameras have been used. Within these 100 cameras have been chosen approximately 100 videos of different events. The format of these videos is .ts[3], and to train the NN it is necessary in image format, such as jpg. That is why it is necessary to extract a frame from each of the videos. A shell script in A appendix has been used for this. As each video has a duration of approximately four seconds, the script goes through all the videos and extracts the frame corresponding to the second number two. It also saves this image in a new folder with a numbering. The extracted images have a resolution of 640x360, this is important when configuring the CNN. Finally after this script is executed, it is possible to find a folder with 10,000 images for training with YOLO.

It is also important to say that after the generation has been made a cleaning and review of all images. The reason is to find images without horses or images that do not intuit very well the objects that later will be labeled. This will help to train better.

Once the images are ready, the next step is to make the labeling of each of the images. This task is one of the most monotonous and longest in any ML process. But it is also important and must be done correctly. The LabelImg program has been used to label each of the 10,000 images that make up the dataset. Why this tool? As explained in section Analysis, it is an open-source software and it is available on most platforms. It also has hotkeys that allow the task of labeling to be simple and fast. This process had a total duration of approximately one week.

Figure 12: Example of labeling with LabelImg Software.

[3]TS is a standard format specified in MPEG-2 for the storage of audio, video and data.

After labeling all images with this application there will be a file in .txt format for each of the images labeled with the same name. The format for each .txt file is the following:

<object-class><x_center><y_center><width><height>

Where:

- **<object-class>**: Integer object number from 0 to (classes-1)
- **<x_center><y_center><width><height>**: float values relative to width and height of image, it can be equal from (0.0 to 1.0).

These files also need to be sent to the NN for the training process. Finally, as can be seen in figure 11, the training will be performed in a virtual machine available on the Microsoft Azure platform. This is why the dataset needs to be sent to this computer, using the "scp" command. This command allows to send any file using the terminal.

3.2 Train YOLO Deep Learning Method

Once all the images of the prepared dataset are available and as can be seen in figure 12, the next step is to train the convolutional network. This network will be built on a Microsoft Azure virtual machine with the following specifications:

- Architecture: 6 vcpus and 56 GB RAM memory
- GPU: Tesla K80 with 24 GB GDDR5
- Operating system: Ubuntu 16.04
- Location: West Europe

But, why has this virtualization platform been used? The choice is not only based on experience in this environment, but also this provider allows to use a GPU device with a student license. Apart from this platform, a creation of the same environment in Amazon Service has been tried, however it does not allow to use the GPU as easily as Azure.

When the machine is up and running, the next step is to install and configure the CUDA library. As explained in Analysis section, this library is necessary for the operation of YOLO. After the next step is the installation of YOLO. All the necessary process for its installation is available in the documentation [34]. The installation consists of cloning the repository available in Github and compile it with the desired features. In this case it is necessary to compile with CUDA and OpenCV libraries. In order to do this, the Makefile file simply has to be modified and compiled with the 'make' command.

Once the system with the framework is working, the training will be carried out with the created dataset. In this case 10,000 tagged images of horses. The command shown in figure 13, allows to run this training.

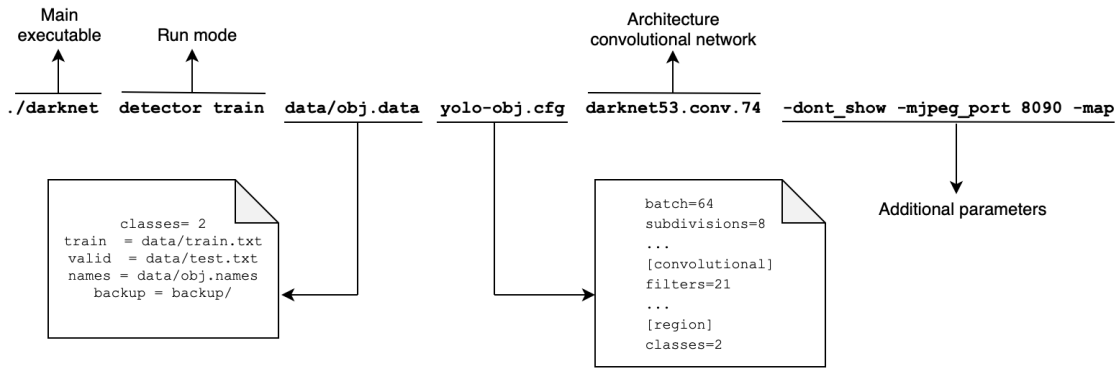

Figure 13: Execution train YOLO command.

According to [34], this command is built up of files and parameters. The first one is the program that starts the process. In this case, the mode that is executed is the "detector train". This mode is in charge of training the NN. The second parameter is a file with information about the dataset that is used in this training. In this file, the classes to be detected are located, as well as, training images, images to validate the NN, the name of the classes of objects and where they will save the weights found when training the network. The files train.txt and test.txt contain the path of all the images that are required to train or to validate. Each of the paths must be positioned on a different line, so if there are 10,000 images there will be 10,000 lines within the file. To create these files a python script has been used in B appendix. This script goes through the directory where the images are and introduces the path in the new file train.txt. In the third parameter shown in figure 13, is the file that configures the CNN. In this .cfg the parameters must be modified to adapt the model to the new dataset. It is obligatory to change the number of classes that will be detected and the filter parameter, which for this case will be 21. There are also some data such as subdivisions, which indicates in how many sections the image will be divided in the CNN. This parameter can be used to find the number that provides more precision. In the fourth parameter is located the architecture with which the network wants to be trained. In this case is darknet53.conv.74, but as seen in the results section, a smaller architecture such as YOLO-tiny can be used. Both can be downloaded from the official website [34]. Finally, the rest of the parameters offer the possibility of showing a graph via web of how the network is training. This parameter allows to calculate the mAP at the same time that the CNN is being trained and gives an idea of the accuracy of the results.

Once executed, the network starts training as many iterations as configured in the file yolo-obj.cfg. In this case, 4,000 interactions will be made. But this does not mean that it is necessary to do them all. According to [34], the training process should last until the average loss avg no longer decreases at many iterations. The duration of the training also depends on the number of subdivisions that have been configured. In figure 14 a capture of the training can be observed.

```
Region 94 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.495414, .5R: -nan, .75R: -nan,  count: 0
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.496921, .5R: -nan, .75R: -nan,  count: 0

 (next mAP calculation at 1000 iterations)
 1: 4267.098145, 4267.098145 avg loss, 0.000000 rate, 34.190319 seconds, 48 images
```

Figure 14: Screenshot YOLO executing.

Where:

- **1**: Iteration number (number of batch)
- **4267.098145 avg**: Average loss (error) - the lower, the better.

When the training process has finished, there are several files with the weights belonging to a certain number of interactions in the /backup folder. Which one to choose? In order to get the most out of the CNN, the file with the highest mAP (mean average precision) or IoU (intersect over union) should be chosen. This choice is important to avoid overfitting. This process will be seen in the next section.

3.3 Test YOLO Deep Learning Method

With the training of the CNN, the next step is to measure the performance of that network. This section explains the method that has been used to evaluate it. As explained in [36], the evaluation method to measure the performance of object recognition is mAP (mean Average Precision). This performance criterion was defined in [37] and is the one used by YOLO to evaluate.

First, the detection results are sorted by decreasing confidence. There is a "match" when they share the same label and an IoU (Figure 15) >0.5 (Intersection on Union greater than 50%). This "match" is considered a true positive if that fundamental truth object has not been used, this is important to avoid multiple detections of the same object.

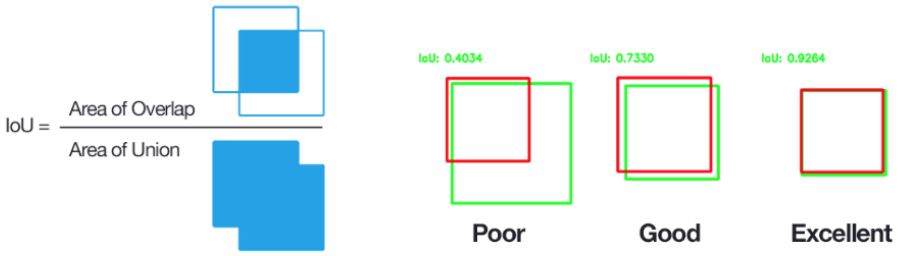

Figure 15: Formula and example of IoU (Intersect over Union).

According to [38], the IoU is given by the ratio of the area of intersection and area of union of the predicted bounding box and ground truth bounding box. Therefore the object that fits best will have a higher IoU. This parameter can be configured in the YOLO framework. By default it is 0.50, but if the detections of objects wants to be more reliability, this values

can be increased. On the other hand if the detection of more objects is wanted, this value has to be reduced. [34]. In the figure 15 obtained from [39], it can be seen how IoU works for different scenarios. To know the mAP it is necessary to calculate the accuracy and recall. These values are given by the following formulas:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

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

$$\begin{gathered} \text{TP} = \text{True Positive} \\ \text{TN} = \text{True Negative} \\ \text{FP} = \text{False Positive} \\ \text{FN} = \text{False Negative} \end{gathered}$$

Calculating the values of these formulas for the data in the example of [40] and plotting the results on a graph obtains the following result in the figure 16.

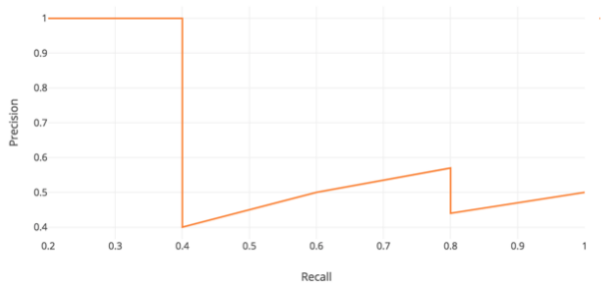

Figure 16: Graph for calculating mAP.

Finally, the AP is calculated as the area below of this curve (shown in orange in figure 16) by numerical integration. To obtain the mAP, all that is needed is to calculate the average of all the AP's.

Another value that is needed and that will later be seen in the result graphs is the Loss. This value is very often used in the related field of NNs. As shown in figure 17, NNs are trained using stochastic gradient descent. When training, the network goes searching for the smallest error, graphically moving from the curve to the lowest part of the curve [41]. Each point that is crossed in the curve goes taking the name of value Loss, until finding the final value of Loss that is a point as close as possible to the bottom of the curve. Therefore the aim of the results is to find the highest possible mAP. This will possibly coincide with the lowest Loss. To set the speed with which this minimum is found there is an additional parameter named learning rate. This value is the speed at which the error converges until the minimum is found. If this value is too high, the network may exceed the minimum of

the curve, or if the value is too low it may take too long to find it. In YOLO this parameter can be configured in the NN configuration file [34].

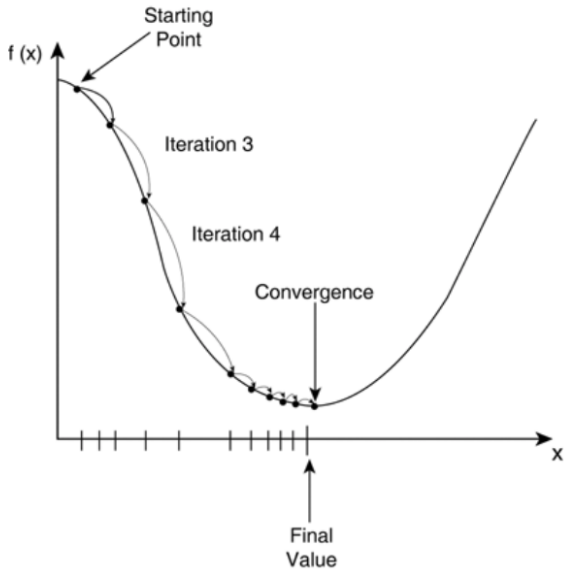

Figure 17: Calculating minimum error in stochastic gradient descent.

The forked version of YOLO being used in this project offers an option to calculate and plot the mAP in the loss window while the network is training. YOLO creates another parallel neural network that calculates the mAP every certain interactions. To do this it uses the validation data that has been indicated in the file valid.txt. It should be noted that not working with graphical environment must launch the training with a special command. This command publishes the plot by creating a small web server that can be accessed with the browser. At the end a chart like that in Results is obtained.

Once the method of evaluation of the neural network has been explained, a planning of the tests to be carried out is designed. In figure 18 a scheme of these tests can be seen.

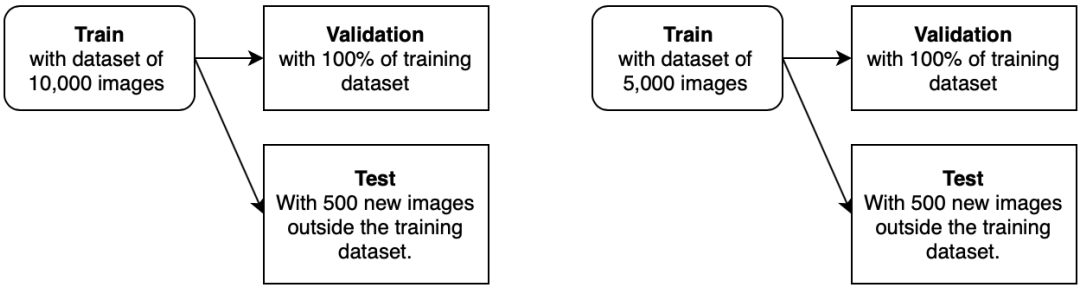

Figure 18: Test set for this project

The first training is performed with 10,000 images and 100% dataset is used to validate this training. The test of this training is performed on a new dataset of 500 images of horses that the NN has not seen before. This training can take around 36 hours to finish.

The other test is performed with half of the dataset, 5,000 images and 100% of the dataset to validate it. For each of these sets will be made for the two architectures to be compared. That is to say, three trainings will be carried out for the dataset of 5,000 images (modifying the subdivision parameter) and 3 trainings for the dataset of 10,000 images (modifying the subdivision parameter) both for the architecture of Darknet and for the architecture of YOLO-tiny. Also, for each of them a test of the 500 new images is also performed. In total, as can be seen in the figure 19, 12 training sessions have been executed.

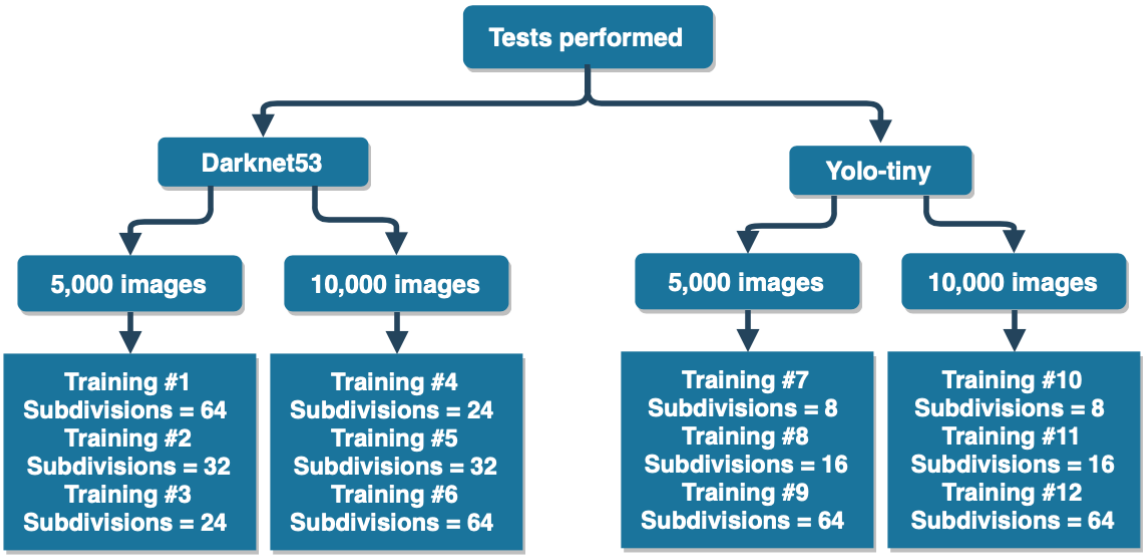

Figure 19: Testing scheme

The next aspect to consider is overfitting. According to [42], overfitting is when a method of supervised machine learning adapts too well to the dataset training. This is when there is a risk of also learning the noise from the data and memorizing certain particularities of the training dataset. This means that the predictions with the new data are not correct.

As a neural network is of the supervised learning type, it has to be alert to this problem. To address this problem and according to [34], the training must be stopped before the weights of the network adapt too well to the training data. As mentioned above, the network weights are saved in the backup folder. According to figure 20 extracted from [34], the weights for the NN of the last iterations should not be chosen because in those weights the overfitting problem increases. It should get weights from Early Stopping Point.

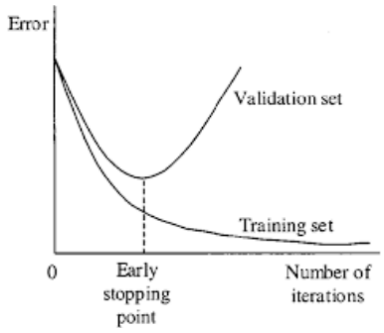

Figure 20: Point where to stop training at Yolo

On the other hand in this NN there should not be much overfitting, since according to [43] in a large dataset with a lot of variety of images the possibility of this situation decreases.

As mentioned in the Analysis section, there is a reduced version of the multilayer architecture called YOLO-tiny. The tests has also been performed using this version, because it offers more speed and the precision can be very similar. Using this architecture one can even think of running on mobile devices because computational calculation is lower [32]. The result of these tests can be seen in the following section.

In order to complete the test part and to be able to see something graphically, a demo video in .mp4 format has been created with different videos of the horses that exist in the training dataset and other videos of horses that have not been used in the training. This is intended to see the neural network work in as real an environment as possible. This framework [34], provides a command to process videos and even images in real time such as webcams. Also, to mention again that since there is no graphic environment available, it is necessary to add special parameters to save the video result and not show it. These parameters are available in the used YOLO repository. Finally, the ultimate command to use would be as follows:

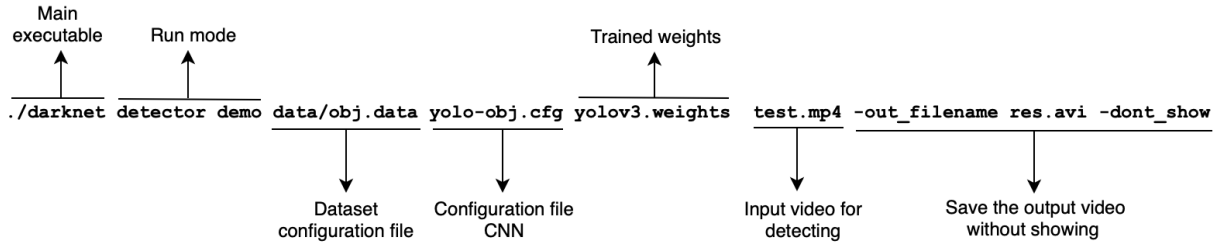

Figure 21: Execution test YOLO command

The command above is very similar to the one used in training. When it is testing the file the execution mode changes, now "detector mode" is used. Again, information about dataset and CNN configuration is needed. As it is also used in figure 21, this is indicated with the second parameter and third parameter. The fourth parameter is the weights resulting from the training. It is the most important and hardest file to create. In the fifth place is

the file in which the detection of objects is wanted, in this case people and horses. Here it can be noticed that this is a video but also YOLO accepts an image or a set of images. Finally, there are the necessary parameters to save the results in video format that returns the execution of the command.

4 Results

In this section, various tests will be carried out to provide information on the performance that CNN has acquired in its training. As mentioned in the Methodology section, the tests were performed according to the following procedure. Darknet53 architecture was used first. With this architecture three trainings have been carried out with a dataset of 5,000 images changing the parameter of subdivisions in each one of them. Second, for the same values of the parameter of subdivisions, three trainings are carried out with a dataset of 10,000 images. In these training sessions, a test has also been realized with 500 images of horses that YOLO had not seen before. The following set of tests has been done following the same criteria but using the architecture of YOLO-tiny. As can be seen in the table 2, 12 validated training sessions have been performed with the same data set and 12 other tests using new images.

Before discussing the charts, it is necessary to understand the charts resulting from the tests. On the x-axis the number of interactions that the training is executing can be observed. On the y-axis it belongs to the range of values that the Loss function is getting, which has already been explained in Methodology section. The value of Loss in each iteration is indicated by the blue line. The line that can be seen in red belongs to the mAP function that determines the accuracy of the training in each iteration. Therefore the lower the Loss error the higher the mAP value is obtained.

It should also be mentioned that some of the subdivisions used in the YOLO-tiny are not available in the architecture of Darknet53. This is because an image can not be fragmented so much in that architecture. In spite of these, other values have been searched for that can give the highest performance.

From the next page each of the tests carried out with its chart is shown. In them, it is possible to see the precision that is obtaining the NN in each interaction.

4.1 Darknet53 Layers Architecture

Training test #1

In figure 22, it can seen the graph resulting from test number 1.

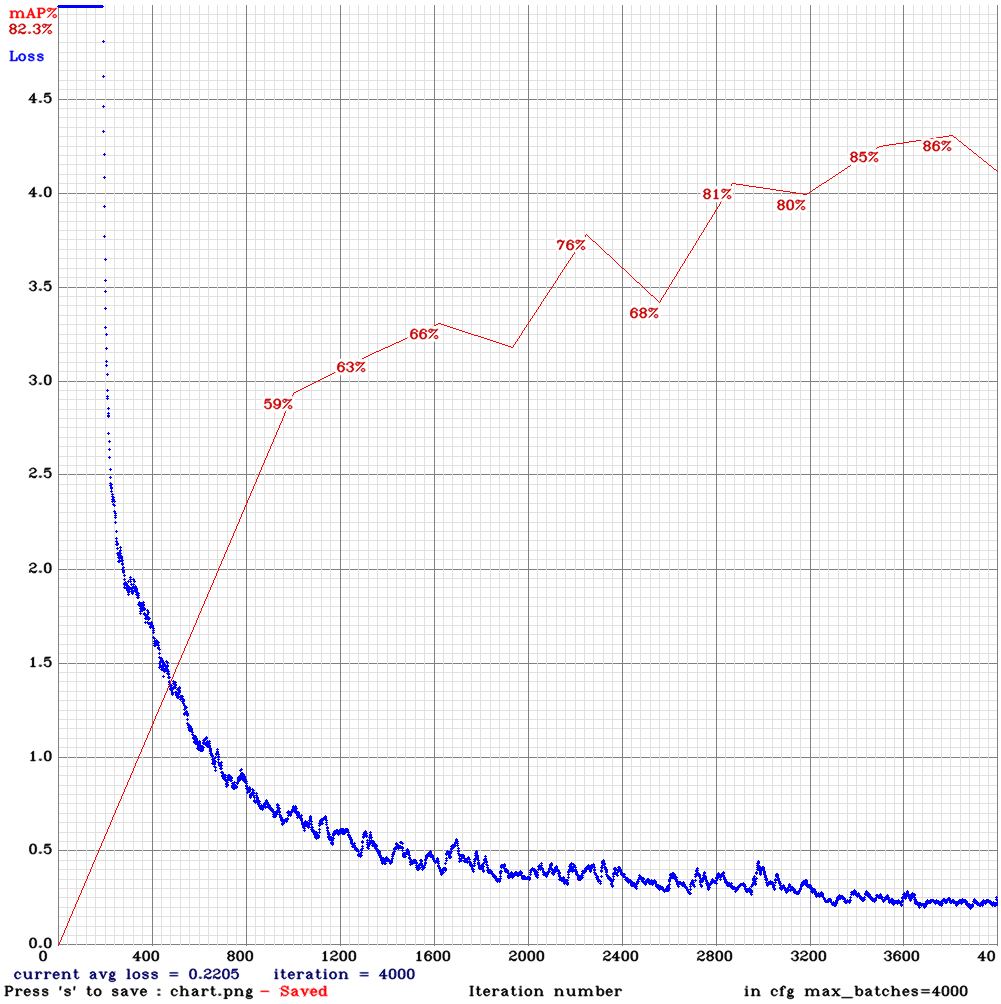

Figure 22: Chart training test number 1.

This training performed with a dataset size of 5,000. The configuration used is with the parameter subdivisions = 64 and an input resolution = 608x608. As shown in figure 22, 0.2205 of avg loss, 74% of IoU and 82% of mAP have been obtained. In the test made with 500 new images, 39% of mAP and 62% of IoU have been achieved.

This test has a very high accuracy result, but it also has to be mentioned that it takes a lot of interactions to get the highest value. This is one of the factors influencing its low performance using new images.

Training test #2

In figure 23, it can seen the graph resulting from test number 2.

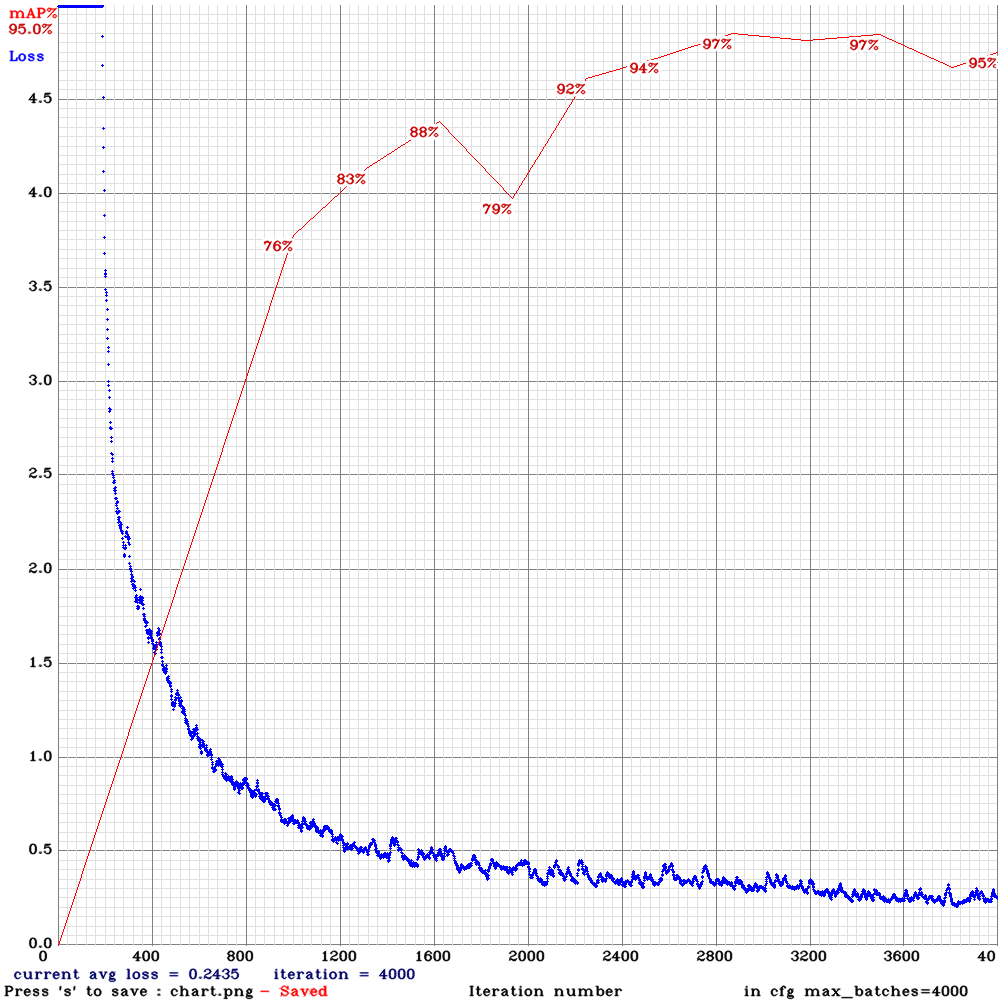

Figure 23: Chart training test number 2.

This training performed with a dataset size of 5,000. The configuration used is with the parameter subdivisions = 32 and an input resolution = 608x608. As shown in figure 23, 0.2435 of avg loss, 83% of IoU and 95% of mAP have been obtained. In the test made with 500 new images, 44% of mAP and 59% of IoU have been achieved.

If this result is analyzed it is clear that it has very good precision validating with the training dataset, 95% mAP. But if new test images are used this accuracy drops to 44%. This means that if it wants to detect in new images the performance will be lower.

Training test #3

In figure 24, it can seen the graph resulting from test number 3.

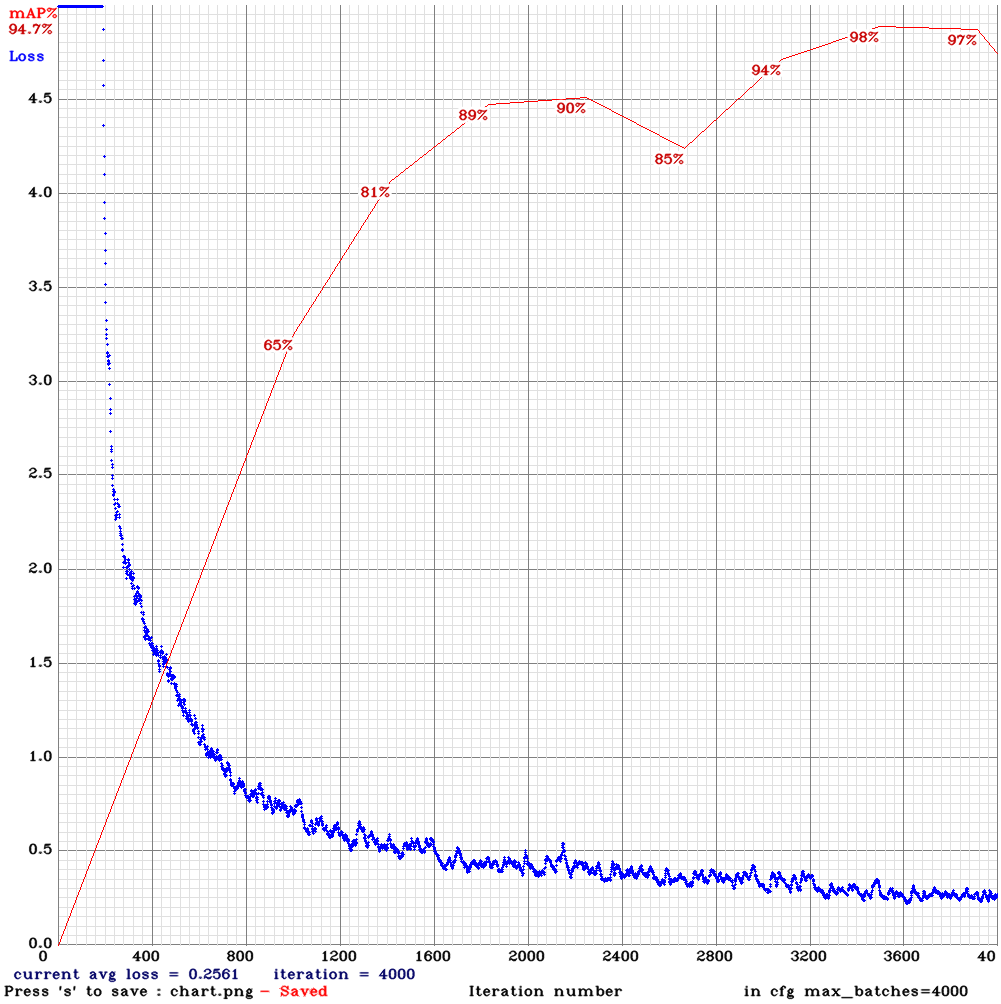

Figure 24: Chart training test number 3.

This training performed with a dataset size of 5,000. The configuration used is with the parameter subdivisions = 24 and an input resolution = 608x608. As shown in figure 24, 0.2561 of avg loss, 82% of IoU and 95% of mAP have been obtained. In the test made with 500 new images, 41% of mAP and 52% of IoU have been achieved.

This result is very similar to the one obtained in test number 2. But this training is less effective against new images, as 41% of mAP was obtained and 44% of mAP in the previous test.

Training test #4

In figure 25, it can seen the graph resulting from test number 4.

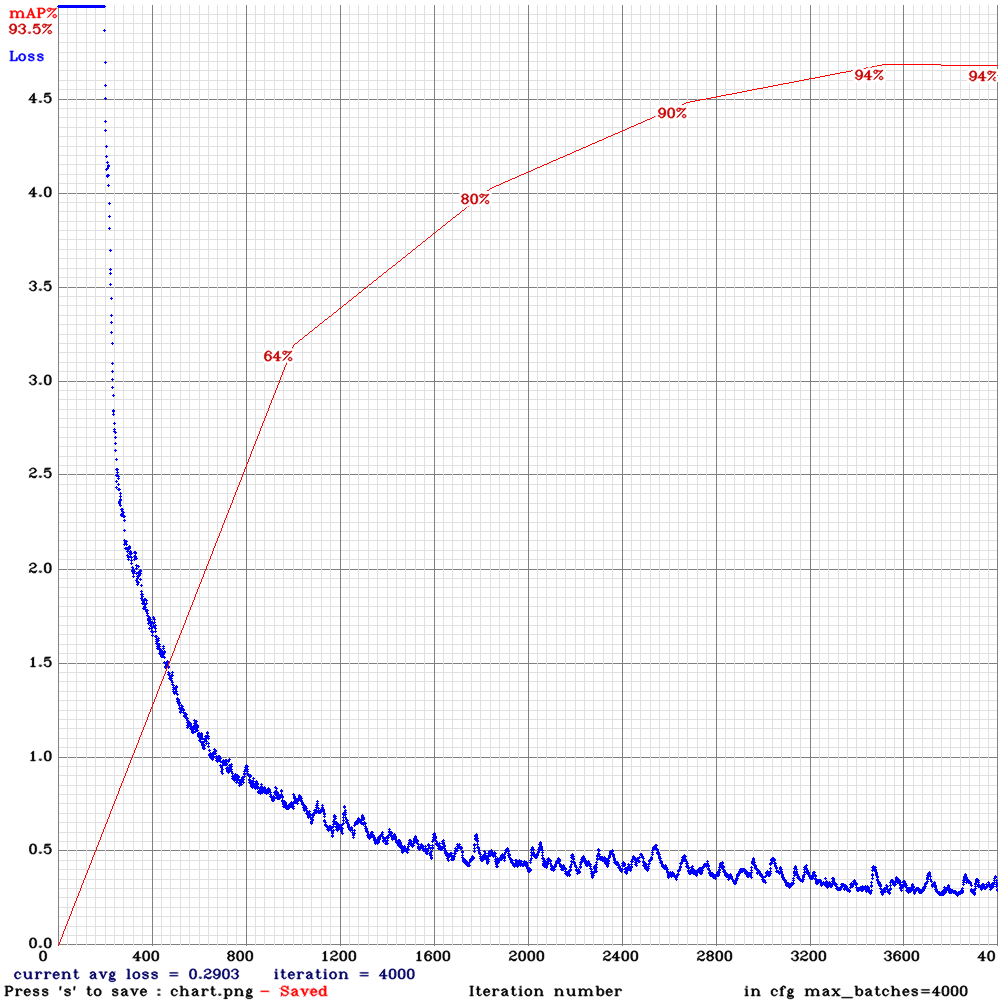

Figure 25: Chart training test number 4.

This training performed with a dataset size of 10,000. The configuration used is with the parameter subdivisions = 24 and an input resolution = 608x608. As shown in figure 25, 0.2903 of avg loss, 81% of IoU and 94% of mAP have been obtained. In the test made with 500 new images, 49% of mAP and 71% of IoU have been achieved.

This test can be one of the most interesting, as it gets a good result in terms of the datatest of 500 new images with 49% mAP. The result of the training with the complete dataset is also high (94% of mAP), but there are other tests that have a higher value than this one.

Training test #5

In figure 26, it can seen the graph resulting from test number 5.

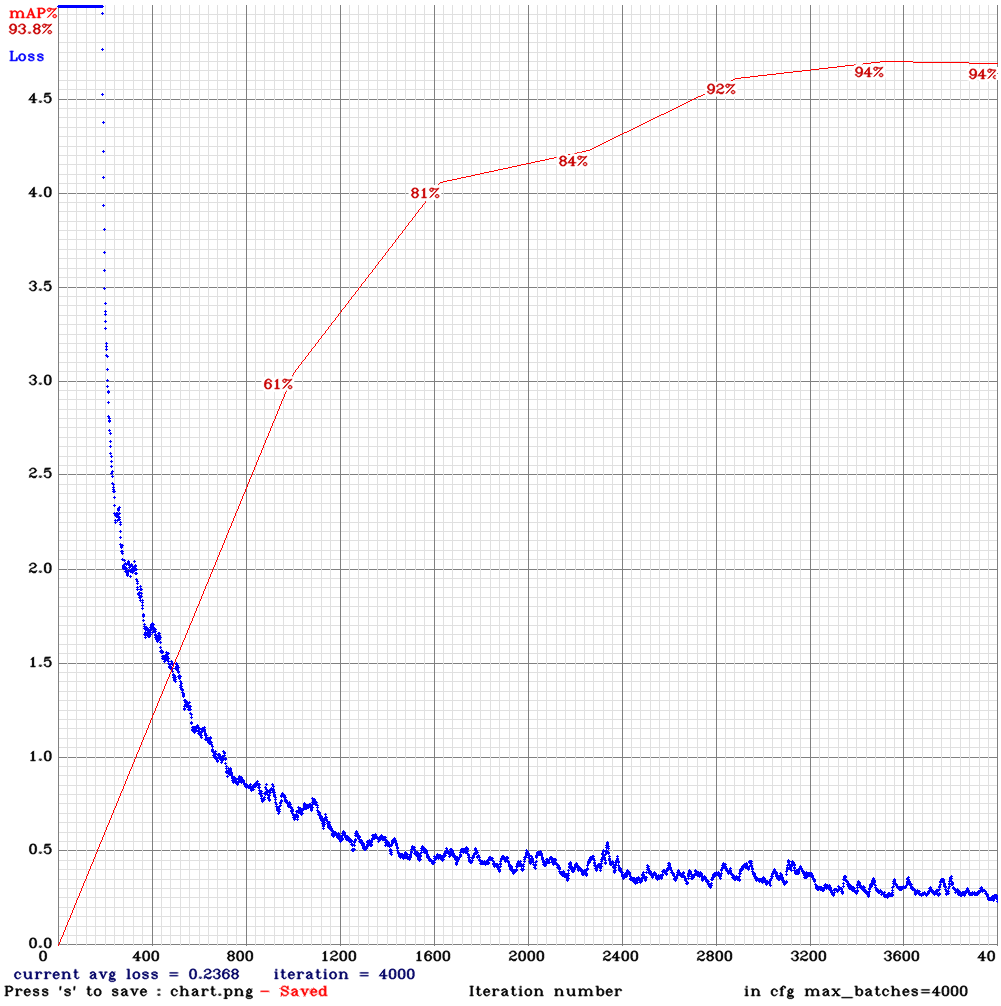

Figure 26: Chart training test number 5.

This training performed with a dataset size of 10,000. The configuration used is with the parameter subdivisions = 32 and an input resolution = 608x608. As shown in figure 26, 0.2368 of avg loss, 71% of IoU and 94% of mAP have been obtained. In the test made with 500 new images, 50% of mAP and 71% of IoU have been achieved.

Possibly this is one of the best results obtained. The precision of the training test is quite high, but the value of 50% of mAP with new images is interesting. Therefore this configuration is the best for a set of new images.

Training test #6

In figure 27, it can seen the graph resulting from test number 6.

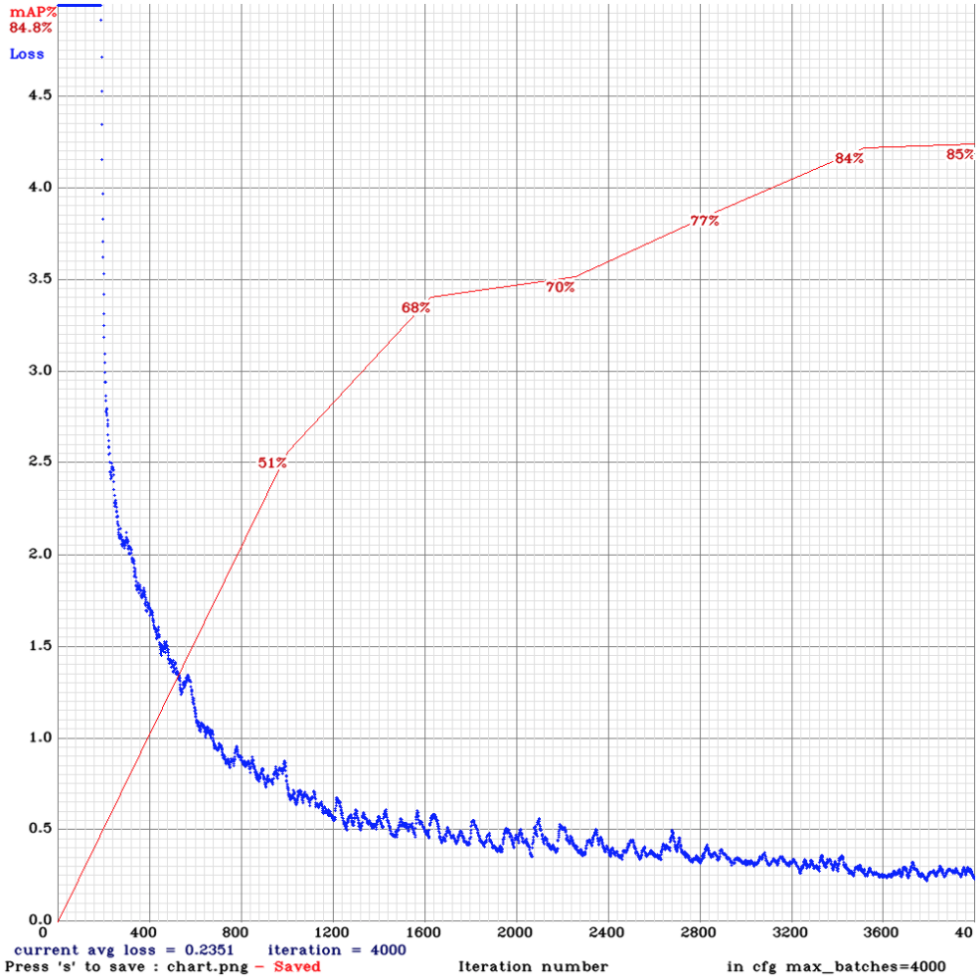

Figure 27: Chart training test number 6.

This training performed with a dataset size of 10,000. The configuration used is with the parameter subdivisions = 64 and an input resolution = 608x608. As shown in figure 27, 0.2351 of avg loss, 79% of IoU and 85% of mAP have been obtained. In the test made with 500 new images, 49% of mAP and 78% of IoU have been achieved.

This test is one of the most time-consuming to complete. The result is not as good as the training done with a dataset of 5,000. But if new images are used, this training gets better results than the smaller dataset.

4.2 YOLO-tiny Layers Architecture

Training test #7

In figure 28, it can seen the graph resulting from test number 7.

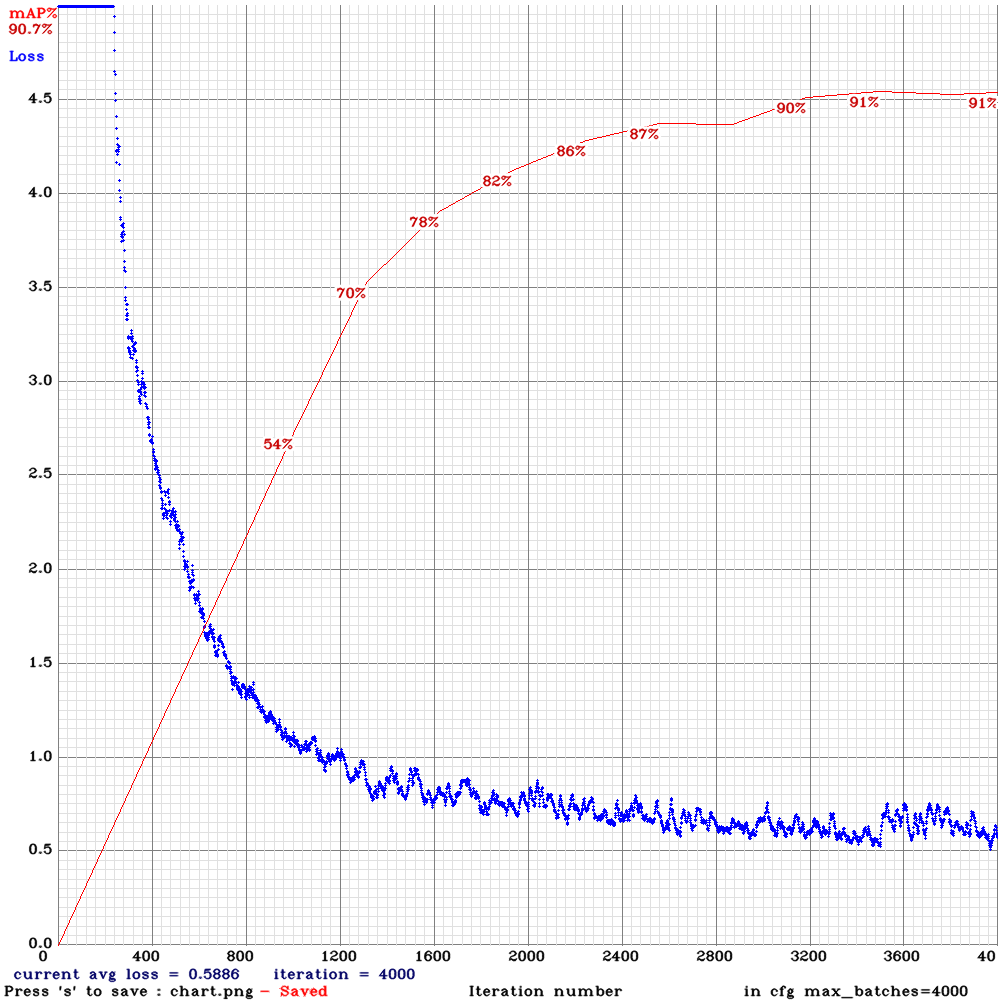

Figure 28: Chart training test number 7.

This training performed with a dataset size of 5,000. The configuration used is with the parameter subdivisions = 8 and an input resolution = 416x416. As shown in figure 28, 0.5886 of avg loss, 76% of IoU and 91% of mAP have been obtained. In the test made with 500 new images, 46% of mAP and 56% of IoU have been achieved.

This test doesn't catch too much attention. It can be noticed that if the red line is observed, it can be observed that it remains stable throughout the training. In other words, there are no pronounced peaks like in test 1 and test 2.

Training test #8

In figure 29, it can seen the graph resulting from test number 8.

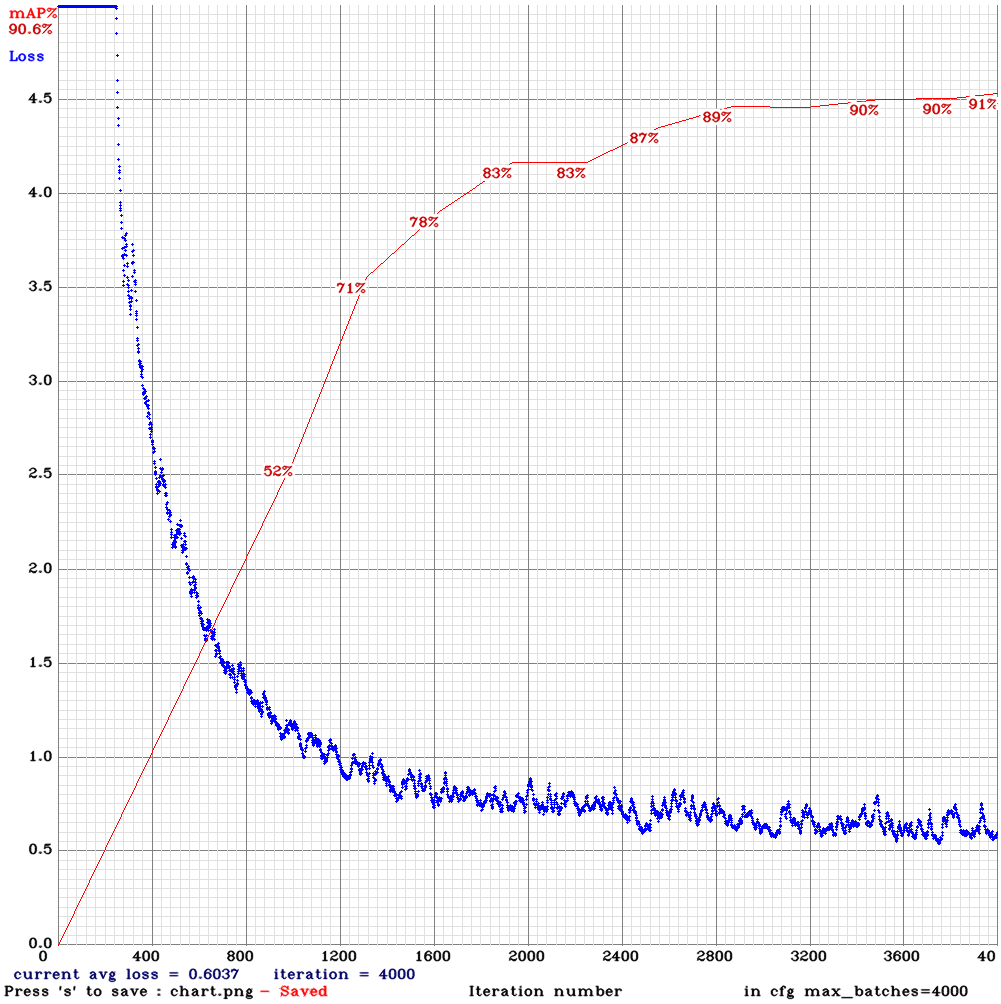

Figure 29: Chart training test number 8.

This training performed with a dataset size of 5,000. The configuration used is with the parameter subdivisions = 16 and an input resolution = 416x416. As shown in figure 29, 0.6037 of avg loss, 76% of IoU and 91% of mAP have been obtained. In the test made with 500 new images, 44% of mAP and 55% of IoU have been achieved.

The result of this test is very similar to the previous test. The only thing that can be mentioned is the stability of the accuracy.

Training test #9

In figure 30, it can seen the graph resulting from test number 9.

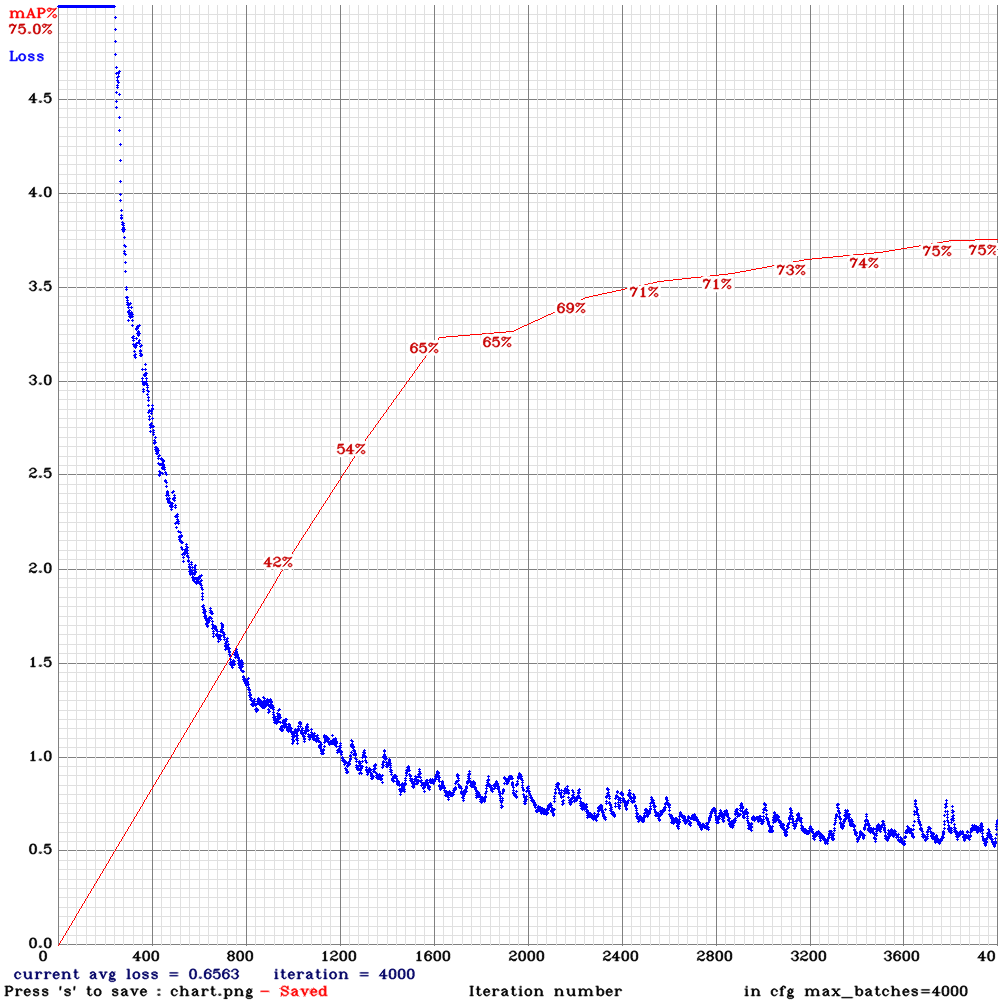

Figure 30: Chart training test number 9.

This training performed with a dataset size of 5,000. The configuration used is with the parameter subdivisions = 64 and an input resolution = 416x416. As shown in figure 30, 0.6563 of avg loss, 72% of IoU and 75% of mAP have been obtained. In the test made with 500 new images, 43% of mAP and 56% of IoU have been achieved.

A pronounced fall in the mAP value can be seen here. This is due to the high value in the subdivision parameter.

Training test #10

In figure 31, it can seen the graph resulting from test number 10.

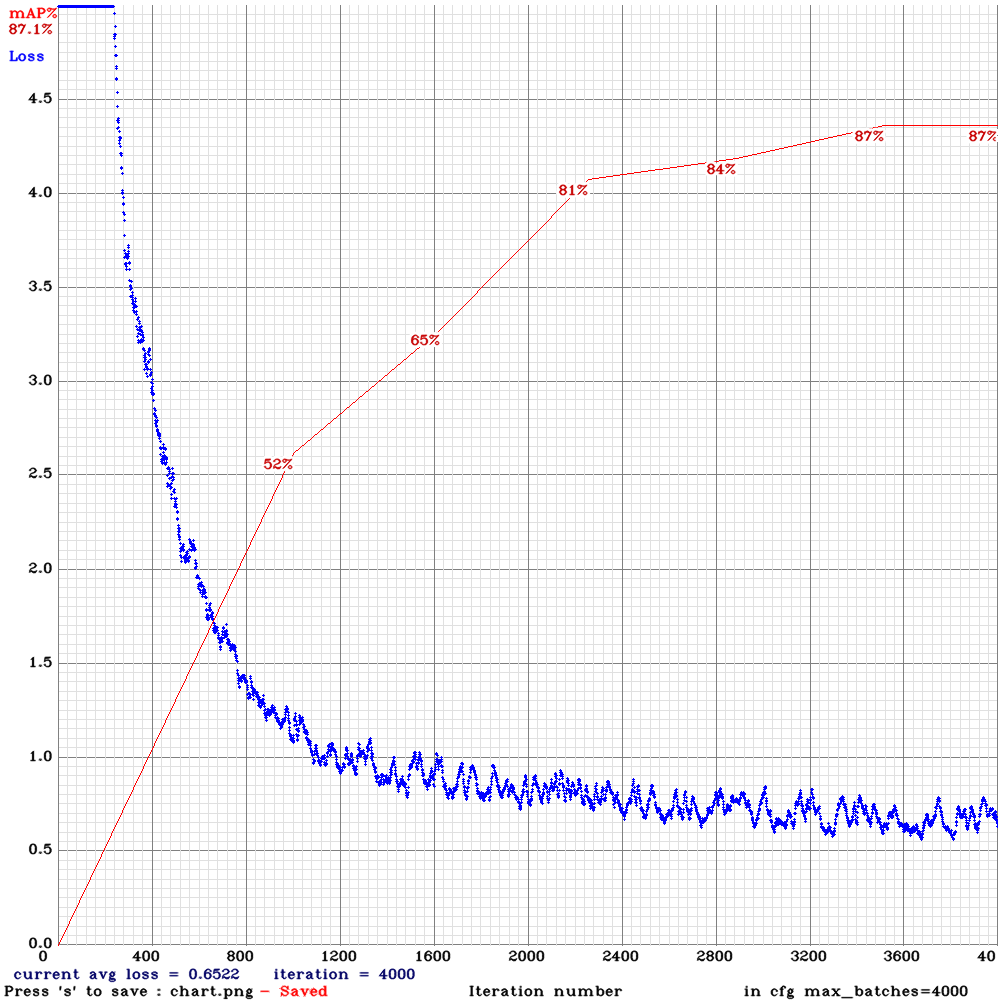

Figure 31: Chart training test number 10.

This training performed with a dataset size of 10,000. The configuration used is with the parameter subdivisions = 8 and an input resolution = 416x416. As shown in figure 31, 0.6522 of avg loss, 75% of IoU and 87% of mAP have been obtained. In the test made with 500 new images, 49% of mAP and 73% of IoU have been achieved.

In this chart it is possible to appreciate also a good precision in the training, but it is highlighted the good performance that it has with new images. This is due to the fact that the dataset in this test is 10,000 images. For this reason the overfitting is lower and the detection is better adapted to new data.

Training test #11

In figure 32, it can seen the graph resulting from test number 11.

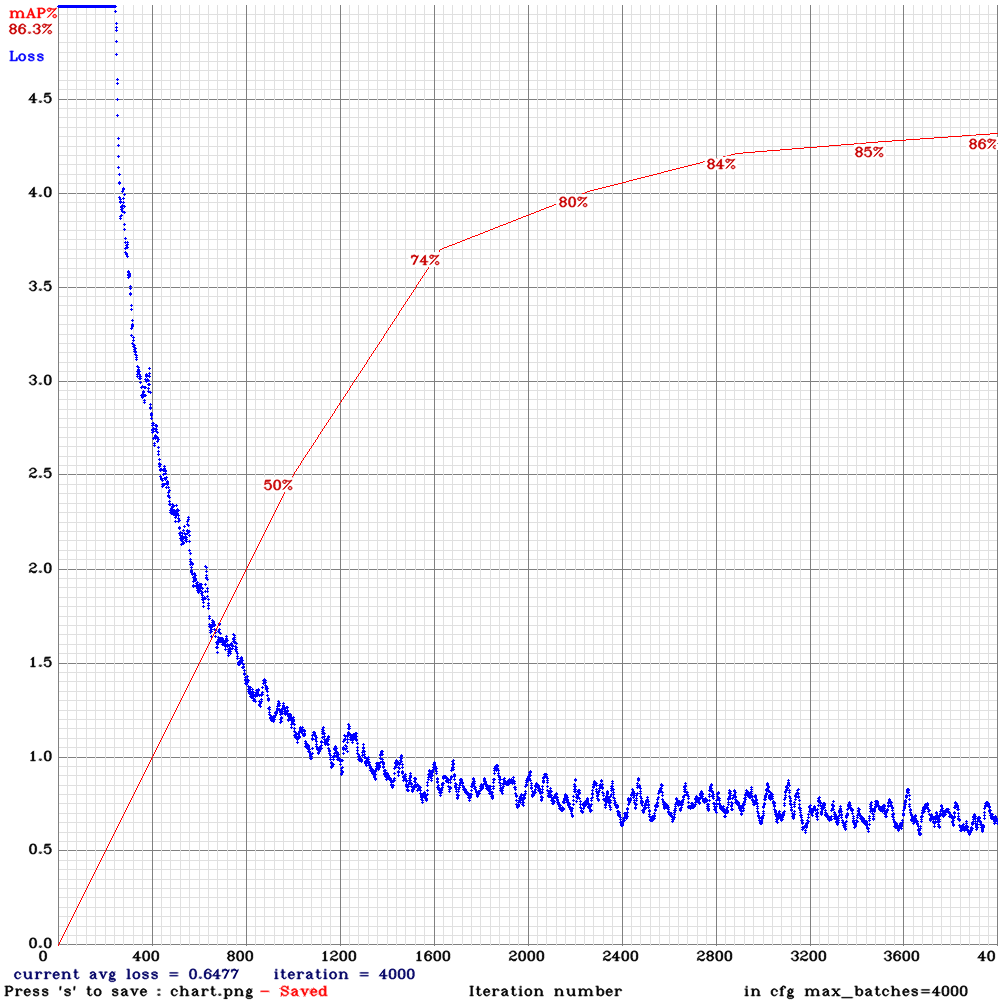

Figure 32: Chart training test number 11.

This training performed with a dataset size of 10,000. The configuration used is with the parameter subdivisions = 16 and an input resolution = 416x416. As shown in figure 32, 0.6477 of avg loss, 75% of IoU and 86% of mAP have been obtained. In the test made with 500 new images, 49% of mAP and 75% of IoU have been achieved.

The results obtained in this test are very similar to the previous one. In this graph it is also possible to see a high mAP in the training, and with respect to the detection of new images, the precision is one of the highest of the 12 tests performed.

Training test #12

In figure 33, it can seen the graph resulting from test number 12.

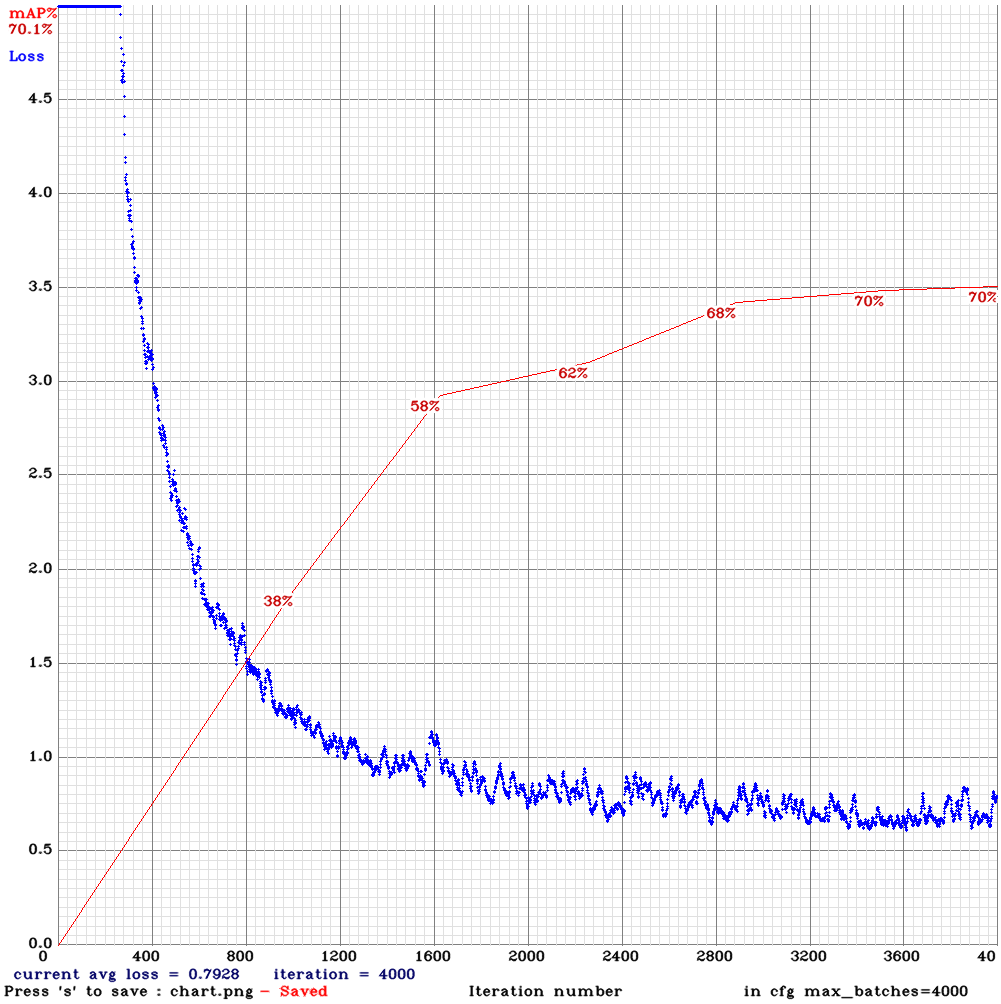

Figure 33: Chart training test number 12.

This training performed with a dataset size of 10,000. The configuration used is with the parameter subdivisions = 64 and an input resolution = 416x416. As shown in figure 33, 0.7928 of avg loss, 74% of IoU and 70% of mAP have been obtained. In the test made with 500 new images, 46% of mAP and 73% of IoU have been achieved.

In the last of the tests carried out it can be seen that it is not one of the best performers. Regarding the precision in the training, this test has one of the lowest of all the tests. This is due to the high number in the subdivisions. Also with respect to new images the value obtained is not one of the highest.

RESULTS TABLE							
Darknet53 Architecture							
5,000 images for training							
#	**Subdivisions**	**Input resolution**	**Type**	**Size dataset**	**Avg Loss**	**IoU**	**mAP**
1	64	608x608	Train	5,000	0.2205	0.7378	0.8227
		608x608	Test	500	X	0.6186	0.3914
2	32	608x608	Train	5,000	0.2435	0.8264	**0.9497**
		608x608	Test	500	X	0.5927	0.4393
3	24	608x608	Test	5,000	0.2561	0.8160	0.9474
		608x608	Train	500	X	0.5156	0.4051
10,000 images for training							
4	24	608x608	Train	10,000	0.2903	0.8096	0.9354
		608x608	Test	500	X	0.7119	0.4892
5	32	608x608	Train	10,000	0.2368	0.8091	0.9376
		608x608	Test	500	X	0.7115	**0.4957**
6	64	608x608	Train	10,000	0.2351	0.7899	0.8477
		608x608	Test	500	X	0.7791	0.4859
Yolov3-tiny Architecture							
5,000 images for training							
#	**Subdivisions**	**Input resolution**	**Type**	**Size dataset**	**Avg Loss**	**IoU**	**mAP**
7	8	416x416	Train	5,000	0.5886	0.7625	0.9074
		416x416	Test	500	X	0.5590	0.4578
8	16	416x416	Train	5,000	0.6037	0.7577	0.9058
		416x416	Test	500	X	0.5544	0.4402
9	64	416x416	Train	5,000	0.6563	0.7203	0.7502
		416x416	Test	500	X	0.5608	0.4285
10,000 images for training							
10	8	416x416	Train	10,000	0.6522	0.7548	0.8714
		416x416	Test	500	X	0.7292	0.4867
11	16	416x416	Train	10,000	0.6477	0.7458	0.8629
		416x416	Test	10,000	X	0.7454	0.4872
12	64	416x416	Train	10,000	0.7928	0.7442	0.7009
		416x416	Test	500	X	0.7338	0.4639

Table 2: Results tests

4.3 Discussion

About the results obtained in the tests it can be said that most of them reach a good level of accuracy. Among them, test 2 has almost 95% in the validation of training and test 5 reaches almost 50% in the detection of new images. Most tests with 5,000 images achieve high values in training validation. This indicates the existence of overfitting in them. But in the trainings carried out with 10,000 images the precision in the validation is a little lower, on the contrary better results are obtained with the introduction of new images that the network has not seen before.

The next question is, which of the two architectures to choose? As can be seen in the table of results, the difference in accuracy of some cases is not very high. Although a better mAP can be appreciated with Darknet53 architecture. But it should be said that with the YOLO-tiny architecture the results are getting faster. This can be an important detonator in some cases. In addition, it can be observed that in most tests the best results have been obtained using a small subdivision value. The red coloured row in the table 2 indicates the training with the highest accuracy. The yellow coloured row indicates the test with the highest accuracy.

Finally, the YOLO framework offers two possibilities to return detections. One is graphical. That means that it returns a file in video format or image format, where there is a rectangle that envelops the whole detected object with the detection percentage. In the figure 34 it is possible to see a screenshot of a demo video.

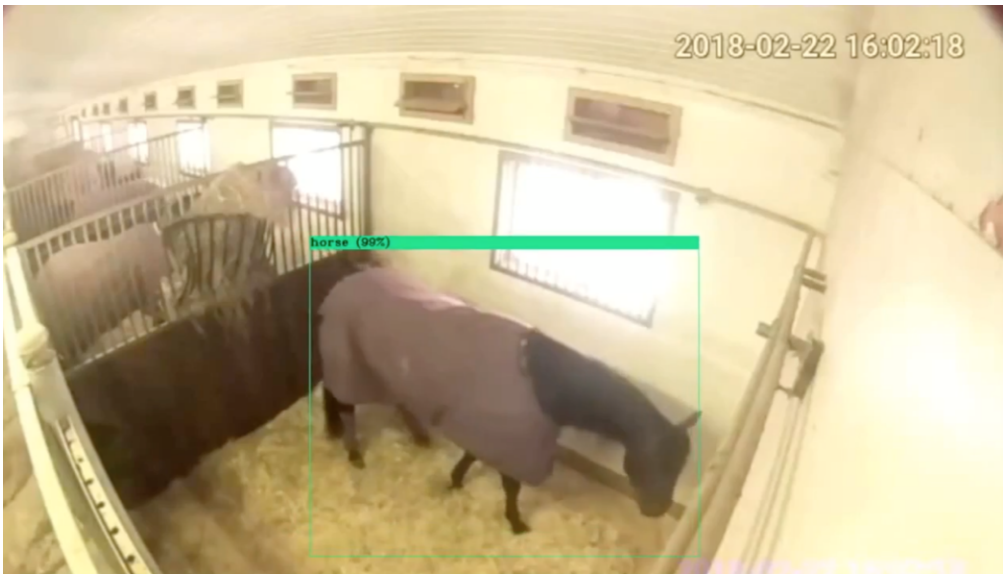

Figure 34: Graphical result of the detection of a horse.

The other option that allows YOLO is to convert the detection results into a JSON file. This option provides opportunities for further analysis. For example to apply some other ML method that of knowledge about the trajectory that the horses do in the stable. With this it is possible to compare tracks that have made sick horses with the horse being analyzed.

In figure 35, it is possible to see a capture of the generated JSON file.


```
{
 "frame_id":326,
 "filename":"data/obj/1166.jpg",
 "objects": [
  {"class_id":1, "name":"horse", "relative_coordinates":{"center_x":0.612535, "center_y":0.611233,
 ]
},
{
 "frame_id":327,
 "filename":"data/obj/1167.jpg",
 "objects": [
  {"class_id":1, "name":"horse", "relative_coordinates":{"center_x":0.546199, "center_y":0.126799,
 ]
},
```


Figure 35: Result detection in JSON format

5 Conclusions

In order to complete this project, the conclusions that have been found will be presented. As can be seen in table 2, the results obtained with this framework are very satisfactory. In some of the tests very high values of precision have been reached. Therefore, it can be said that the objective of this project has been successfully achieved. It is necessary to mention that this dataset is trained with videos from 100 cameras. Probably in the future the company will add more cameras and therefore the predictions in these new cameras are not so effective. This can be solved with re-training the NN but including images of the new cameras in the dataset. This would preserve the performance of the NN.

5.1 Problems Solved

The problems encountered in carrying out this work should also be mentioned. One of them was the installation and configuration of the GPU and CUDA in virtual machine of Microsoft Azure. This was very laborious because of the compatibility of the versions between CUDA library and YOLO framework. Another difficulty was the task of downloading and labeling the 10,000 images that compose the dataset. This task was very laborious and repetitive. The last difficulties encountered was the amount of time that YOLO needs to train the NN. Some of the training sessions spent more than 30 hours in order to train the NN.

5.2 Future Work

The first point to consider is the possibility of improving the NN for the detection of horses. To do this it is possible to add more images to the dataset and images with better quality and resolution. It can also be an improvement to clean the dataset of possible images that do not have a decent quality. It is possible to include more images with people. In the dataset created the presence of people was a little poor and therefore the NN does not detect people with as much precision as it does with horses. Apart from improving the quality of the dataset it can find some improvement by modifying some configuration parameters in YOLO files. These parameters can be found in [34].

One of the most important possible future works is to be able to discover more knowledge about the horses detected with the NN. Using other ML techniques and algorithms it could be predicted when a horse is going to be sick or if it is already sick. Also to detect when the horse is standing or lying down. This would be very interesting for the company Videquus but also for other companies in the sector. In addition, this project can not only be applied to the field of horses but it can be used in any other animal or even peolple. Therefore even a dog owner could detect when his own pet is ill.

Finally, the possibility of publishing the dataset created in this project. Since it is a very large dataset and can open the possibility for future projects. In this manner be able to contribute to new value for the society.

References

[1] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[2] A. Kathuria, "What's new in yolo v3?," Apr 2018.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[4] D. J. Hemanth and V. V. Estrela, *Deep learning for image processing applications*. IOS Press, 2017.

[5] A. Yavariabdi and H. Kusetogullari, "Change detection in multispectral landsat images using multiobjective evolutionary algorithm," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, pp. 414–418, 2017.

[6] K. Leetaru, "Today's deep learning "ai" is machine learning not magic," Nov 2018.

[7] T. Celik and H. Kusetogullari, "Solar-powered automated road surveillance system for speed violation detection," *IEEE Transactions on Industrial Electronics*, vol. 57, pp. 3216–3227, 2010.

[8] F. G. Yasar and H. Kusetogullari, "Underwater human body detection using computer vision algorithms," *26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2018.

[9] Y. He and L. Li, "A novel multi-source vehicle detection algorithm based on deep learning," *2018 14th IEEE International Conference on Signal Processing (ICSP)*, 2018.

[10] B. Wu, B. Fan, Q. Xiao, T. Kausar, and W. Wang, "Multi-scale deep neural network for mitosis detection in breast cancer histological images," 2017.

[11] "Amazon alexa - build for amazon echo devices." https://developer.amazon.com/alexa.

[12] B. Marr, "Machine learning in practice: How does amazon's alexa really work?," Oct 2018.

[13] J. McClean, S. Boixo, V. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature Communications*, vol. 9, p. 4812, 2018.

[14] C. J. Shallue and A. Vanderburg, "Identifying exoplanets with deep learning: A five-planet resonant chain around kepler-80 and an eighth planet around kepler-90," *The Astronomical Journal*, vol. 155, p. 94, 2018.

[15] J. Krause, V. Gulshan, E. Rahimy, P. Karth, K. Widner, G. Corrado, L. Peng, and D. Webster, "Grader variability and the importance of reference standards for evaluating machine learning models for diabetic retinopathy," *Ophthalmology*, 2018.

[16] F. Chollet *et al.*, "Keras." https://keras.io, 2015.

[17] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[18] N. Bostrom and E. Yudkowsky, "The ethics of artificial intelligence," *The Cambridge Handbook of Artificial Intelligence*, p. 316334.

[19] P. Boddington, "Introduction: Artificial intelligence and ethics," *Artificial Intelligence: Foundations, Theory, and Algorithms Towards a Code of Ethics for Artificial Intelligence*, p. 15, 2017.

[20] N. Sharkey, "The ethical frontiers of robotics," Dec 2008.

[21] J. J. Cooper and G. J. Mason, "The identification of abnormal behaviour and behavioural problems in stabled horses and their relationship to horse welfare: a comparative review," *Equine Veterinary Journal*, vol. 30, no. S27, p. 59, 2010.

[22] G. K. Verma and P. Gupta, "Wild animal detection from highly cluttered images using deep convolutional neural network," *International Journal of Computational Intelligence and Applications*, vol. 17, no. 04, p. 1850021, 2018.

[23] G. S. Cheema and S. Anand, "Automatic detection and recognition of individuals in patterned species," *Machine Learning and Knowledge Discovery in Databases Lecture Notes in Computer Science*, p. 2738, 2017.

[24] T. Verhulsdonck, "One shot object detection," 2017.

[25] Tzutalin, "Labelimg." Git code: https://github.com/tzutalin/labelImg, 2015.

[26] M. L. Guru, "Undrestanding convolutional layers in convolutional neural networks (cnns)," 2018.

[27] S. Lakra, T. Prasad, and G. Ramakrishna, "The future of neural networks," 02 2012.

[28] A. C. H. Kusetogullari and H. Grahn, "Object recognition using shape growth pattern," *Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis*, pp. 47–52, 2017.

[29] J. Redmon, "Darknet: Open source neural networks in c." http://pjreddie.com/darknet/, 2013–2016.

[30] Tsang, "Review: Dssd-deconvolutional single shot detector (object detection)," Dec 2018.

[31] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick, "Microsoft coco: Common objects in context," *Computer Vision ECCV 2014 Lecture Notes in Computer Science*, p. 740755, 2014.

[32] M. Hollemans, "Real-time object detection with yolo," May 2017.

[33] D. Shiffman, *Learning processing: a beginners guide to programming images, animation, and interaction*. Elsevier/Morgan Kaufmann, 2015.

[34] AlexeyAB, "Alexeyab/darknet github repository." Github on https://github.com/AlexeyAB/darknet, May 2019.

[35] "Cyberduck." https://cyberduck.io/.

[36] J. Cartucho, "Cartucho/map." Github on https://github.com/Cartucho/mAP, May 2019.

[37] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, p. 303338, 2009.

[38] R. J. Tan, "Breaking down mean average precision (map)," Mar 2019.

[39] A. Rosebrock, "Intersection over union (iou) for object detection," Jun 2016.

[40] J. Hui, "map (mean average precision) for object detection," Mar 2018.

[41] J. Brownlee, "Loss and loss functions for training deep learning neural networks," January 2019.

[42] T. Dietterich, "Overfitting and undercomputing in machine learning," *ACM Computing Surveys*, vol. 27, no. 3, p. 326327, 1995.

[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, p. 8490, 2017.

Appendices

A Shell Script

```
#!/bin/bash
# Extract image from video .ts and insert into folders
# @author Alejandro Delgado

# Path where there are the videos
pathSource="./videos/all/"

# Path to save the new images
pathTo="./images/all/"

# Counter the names of new images. Example: 1.jpg, 2.jpg, etc.
count=1

# Name of each new folder with images
currFolder=1

# Number of files in each folder
numFile=1000

# To create the first folder "1_K/""
mkdir "$pathTo$currFolder"_K

# Navigate through the folder where all the videos are located
for i in $(ls $pathSource)
do
    # Extract the frame in the 2 second and save in format image
    ffmpeg -i $pathSource$i -ss 00:00:02.000 -vframes 1
    "$pathTo$currFolder"_K/$count.jpg

    # Each numFile, new folder is created
    if [ `expr $count % $numFile` -eq 0 ]
    then
            let currFolder++
            mkdir "$pathTo$currFolder"_K
    fi
        let count++
done
echo "Done" #End
```

B Python Script

```
1  import sys
2  import os
3  from os import listdir
4  from os.path import isfile, join
5
6  nameNewFile= sys.argv[2]
7  root = sys.argv[1]
8  txt = open(nameNewFile, 'w')
9
10 for path, subdirs, files in os.walk(root):
11      for name in files:
12         txt.write(os.path.join(path, name) + '\n')
13
14 print "Done"
```