esprit

Se former autrement

HONORIS UNITED UNIVERSITIES

Stage d'immersion en entreprise

Specialité : Intelligence artificielle

Création d'un modèle pour la detection du modèle d'une voiture et l'extraction des chiffres à partir de la matricule

Réalisé par : Adem Boukhris

Encadrant Entreprise : Zied Rouissi

AMzIsS

report

Adem BOUKHRIS

September 2022

Contents

1 General Context

1.1 Introduction

This first chapter is dedicated for a general presentation of the internship setting. It includes a presentation of the startup and a global vision on its activities, also the general context of the project. The problematic exposition and the aims of the performed work would conclude this chapter.

1.2 Amzi

Figure 1: A typical Neural Network.

At AMZi Smart Solutions, talented professionals work on the development of smart solutions of systems for everyday life and industrials problems. Within AMZiSS, the Artificial Intelligence Innovation Group is currently responsible for the development of new initiatives that can help create safety for our clients children and make them protected online.

1.2.1 Project

Our project combines both object detection and object recognition. The first step of this project consists of the detection of the car's model and the second step consists of the detection of the tunisian license plate and the extraction of the license plate number by using OCR techniques.

1.2.2 Work objectives

Working for the creation of car detection model and license plate detection and recognition model: we start with the data collection from renowned resources, we go through the processing and the creation of detection model

for every kind of cars, then we proceed to the creation of a detection and recognition models for the license plates.

1.3 Conclusion

In this introductory chapter, we have presented the general idea of the project, starting with the introduction of the host,the problematic and the work pipeline implemented to deal with it were also presented. Therefore, the chapter which follows is devoted to the study of the state of the art in this field of artificial intelligence application and to the technical presentation of existing solutions.

2 Artificial Intelligence, and object detection

2.1 Introduction

This chapter is devoted to the presentation of artificial intelligence and the study of the state of the art in computer vision for this field of application. For this we start with a brief history of artificial intelligence, object detection and how it has evolved from the use of traditional image processing techniques to the use of Machine Learning algorithms. We will also discuss certain concepts useful for the adaptation of these techniques for the detection of cars in images and as a second step identify car's license plate in detected cars.

2.2 Artificial Intelligence

Artificial Intelligence is the ability of a computer program to learn and think. It involves using computers to do things that traditionally require human intelligence. This means creating algorithms to classify, analyze, and draw predictions from data. It also involves acting on data, learning from new data, and improving over time. Just like a tiny human child growing up into a (sometimes) smarter human adult. And like humans, AI is not perfect. Yet.

2.3 Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience

without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.
There are many different ways to get a computer to learn from data. These various ways can be categorized into three main subsections of machine learning: supervised learning, unsupervised learning and reinforcement learning:

supervised learning It can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values.

unsupervised learning used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data.

Semi-supervised learning fall somewhere in between supervised and unsupervised learning since they use both labeled and unlabeled data for training — typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy.

Reinforcement Learning is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance.

2.4 Deep Learning

Deep Learning is a subset of Machine Learning, which on the other hand is a subset of Artificial Intelligence.
Deep Learning, on the other hand, is just a type of Machine Learning, inspired by the structure of a human brain. Deep learning algorithms attempt to draw similar conclusions as humans would by continually analyzing data with a

given logical structure. To achieve this, deep learning uses a multi-layered structure of algorithms called neural networks.

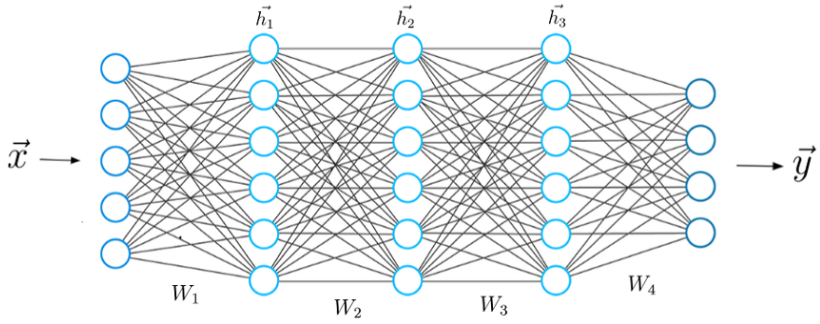

Figure 1: A typical Neural Network.

2.5 Computer vision for object detection

Computer vision algorithms for object detection tasks are among the most powerful tools in machine learning and artificial intelligence. These are decision algorithms that allow computer systems to make inferences about the real world around them, as filtered by a camera. Without object recognition, it would be almost impossible to create robots that manipulate objects, autonomous vehicles, and well-advanced video surveillance software. Understanding computer vision and object detection gives the opportunity to envision more use cases for these tools, allowing us to apply them to innovative and useful applications and systems.

2.5.1 Computer Vision

Let's start first, by defining computer vision and setting the context for our exploration of object detection: it is a field of computing that seeks to endow computers with the ability to extract and interpret characteristics of objects with high level features from images and videos.
What does the term "high-level features" mean? The interpretation of high-level traits mimics the way humans recognize objects. It's through a bottom-up information processing system. First, the edges of the object are identified and joined together. Then the details of the object are filled in, and the brain uses these recognizable patterns to determine what the object is from information seen before and labeled in the brain. High-level interpretation refers to the fact that the many small elements that make up the image come together and the object is recognized at the highest level of processing.

Likewise, the goal of computer vision techniques is to give computers this ability, to evaluate an image and to recognize the parts of the image that have specific meanings.
So, rather than returning basic information about an object in an image, such as information about the different pixels that make up the image, the computer vision system can return information at a level that is meaningful to the image. He can say that an object is a car, a fruit or a person. A computer can then choose the appropriate course of action and execute other instructions depending on the classification of the object in the image. For example, if an autonomous vehicle's object detection system recognizes an object as a car, the computer can use that information to trigger the braking system to detect a collision.

2.5.2 Object detection

As mentioned, computer vision systems performing object detection like humans do, starting with the lowest processing levels and working your way up, piecing together features as you go. When digitally processing and recognizing objects, the computer performs the following steps, which are analogous to how humans recognize objects:

- Recognition of patterns
- The extraction of the characteristics of the object
- Classification of the nature of the object

Thanks to this general vision of how the object detection method works, we can deepen the techniques used in object recognition from the first algorithms to the latest approaches based on machine learning.

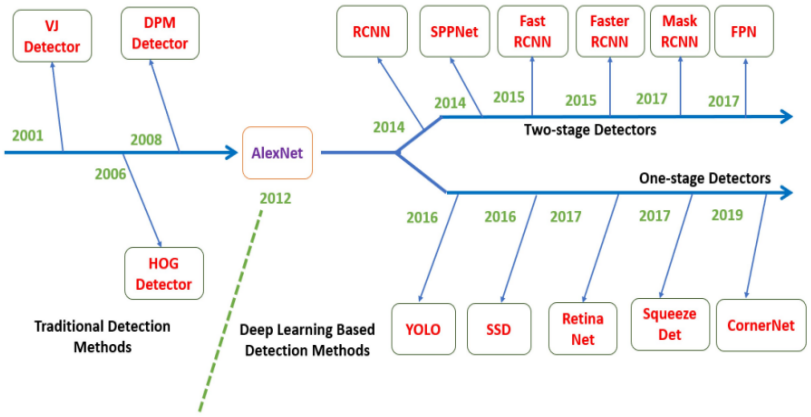

Figure 2: History of major advances in object detection and recognition.

It is broadly accepted that progress in object detection has generally spanned two historical periods: traditional object detection period (before 2014) and deep learning based detection period (after 2014). We can identify ourselves the main reason for this separation by the difference in performance between these 2 approaches. Similarly, we can notice some performance jumps between the many models based on deep learning. This tells us that even among the approaches learned about deep learning, there is a large gap technique. We deal with the most important innovations in the following.

2.5.3 Object detection Algorithms

In view of the importance that deep learning has taken in computer vision, the research community has begun to attempt to port this success to the task of object detection, which has resulted in several techniques many of which have had good success in terms of their performance. We cite three major categories that have led to the development of object detection methods:

- Sliding-window detection methods : One brute force approach to object detection is to drag windows left and 12 right, and up and down to identify objects using classification. To detect different types of objects at different scales, we use windows of various sizes and ratios.

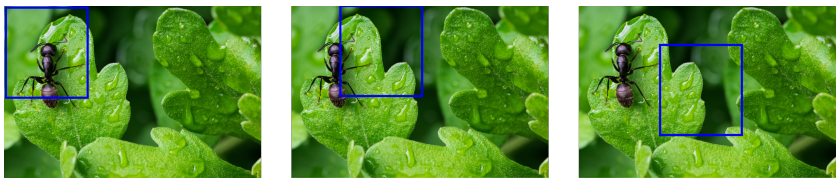

Figure 3: Sliding windows (from left to right)

We cut pieces of the image according to the windows. As shown in figure 2.3 , the patch is applied in a CNN classifier to extract the characteristics from it to later apply an SVM classifier to identify the class and another linear regressor for the box.

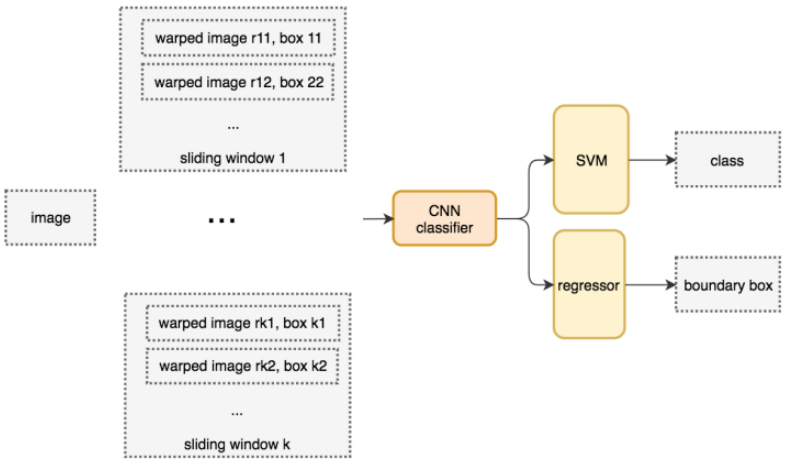

Figure 4: System flow for the sliding window detector

- One-stage detection methods : Region proposal network (RPN) detectors are accurate but not without cost. The R-CNN processes around 7 FPS (frames per second) for the PASCAL VOC 2007 [24] test which is supposed to have one of the lightest bases of computer vision competitions (in terms of resolution and number of images). As an alternative, let's take a look at sliding window algorithms again. We can drag windows on feature layers (feature maps) to detect objects. For the different types of objects, we use different shapes of windows. The fatal mistake of old algorithms is that we use windows as the final predictions.
  For this we need multiple shapes to cover most of the objects. A more efficient solution is to treat the window as a first approximation. Then we have a buzzer to simultaneously predict the class and limit box of the current sliding window.
  However, the convolution layers which represent the feature maps reduce the spatial dimension and resolution. The model will therefore only be able to effectively detect large objects. To remedy this, we perform independent object detections from multiple layers.

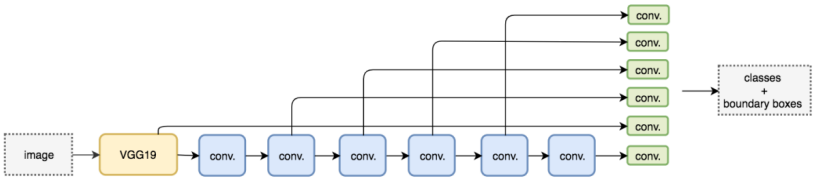

Figure 5: Simplified one-step object detection model

Since this change targeted the processing time problem that the two-

step detection models took, this was to the detriment of the performance of these models since they obtained faster results but still lacked the precision of the RPNs.

- Two-stage detection methods :one branch of object detectors is based on multi-stage models. Deriving from the work of R-CNN, one model is used to extract regions of objects, and a second model is used to classify and further refine the localization of the object. Such methods are known to be relatively slow, but very powerful, but recent progress such as sharing features improved 2 stage detectors to have similar computational cost with single-stage detectors. These works are highly dependent on previous works and mostly build on the previous pipeline as a baseline.

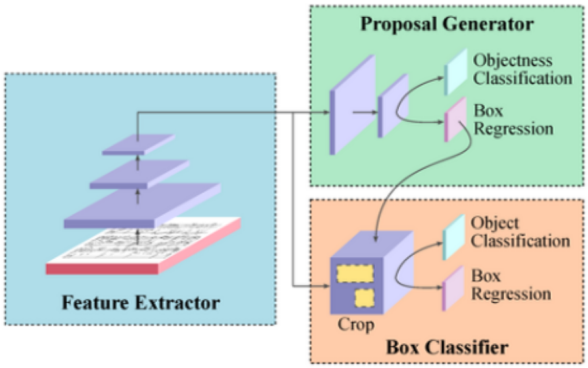

Figure 6: Basic architecture of a two stage detector

3 Implementation and results

3.1 Introduction

This chapter will then be devoted to the presentation on the data preparation techniques used to improve the performance achieved by our model, Also the training pipeline and we also present the different results of the planned experiments and we extract the necessary conclusions that will guide us in the decision-making process.

3.2 Dataset

The dataset we use in this project is the Stanford car dataset. The training set contains 8,144 images of 196 classes and the test set contains 8,041 images of 196 classes, 16,185 images in total. Each class has roughly about 50

images in training set and 50 images in test set. Each of the 196 classes is very fine-grained on the order of year, make and model of a vehicle.
Although the classes are fine-grained, each class is visually distinct from one another; for example, the dataset contains a 2012 Volkswagen Golf and a 1991 Volkswagen Golf, which are visually very distinct, relatively speaking, but it does not contain a 2011 Volkswagen Golf, which is virtually identical to the 2012 model.

Figure 7: Sample Images from Stanford Cars dataset

Each image consists of a car in the foreground against various backgrounds and viewed from various angles. The quality of each image, as described by characteristics like the focal length, lighting, and positioning of the car and camera, varies significantly from image to image – some images are professionally-taken press shots; others are relatively low-quality images collected from classifieds ads and other places on the internet.
After collecting a good amount of images, we had to match the format of the data to the yolo format which means we had to have an image and a txt file containing the object number and object coordinates on this image as folows:

<object-class><x><y><width><height>

where :

- <object-class> : integer number of object from 0 to (classes-1)
- <x><y><width><height> : Box coordinates must be in normalized format(from 0 to 1), and to do this we must divide x and width by image width, and y and height by image height.

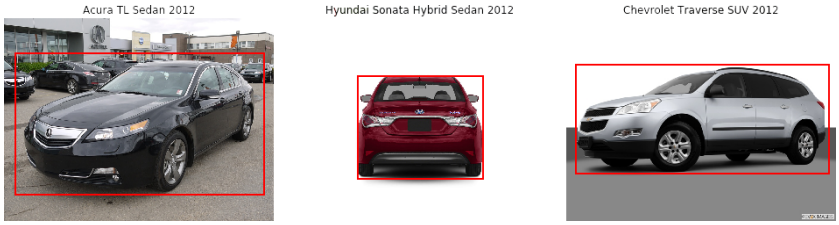

Figure 8: sample of cars images with bounding boxes

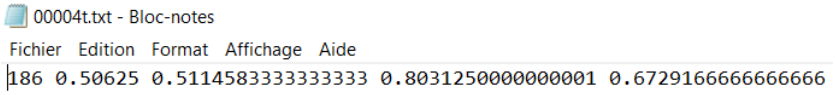

Figure 9: YOLO format (txt file)

Once we converted the annotations to Yolo format, we had a list of txt files containing with the same base file name of the images.

3.3 Implementation

3.3.1 Conception

You Only Look Once (YOLO) v5[8] is adopted in the current project. As a single-stage object detector, the deep-learning architecture contains three important parts:

- YOLOv5 Backbone : It employs CSPDarknet as the backbone for feature extraction from images consisting of cross-stage partial networks.
- YOLOv5 Neck: It uses PANet to generate a feature pyramids network to perform aggregation on the features and pass it to Head for prediction.
- YOLOv5 Head: Layers that generate predictions from the anchor boxes for object detection.

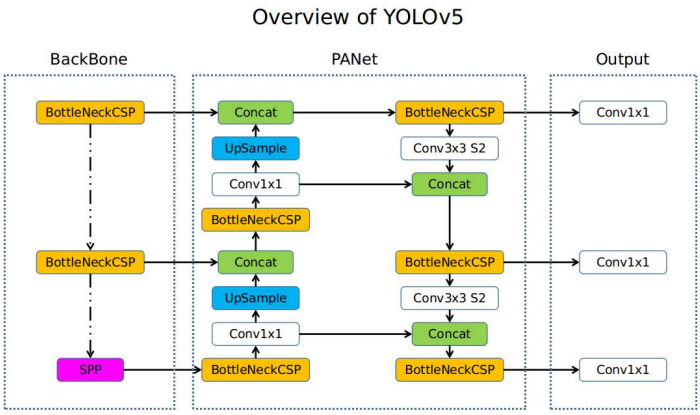

Figure 10: YOLOv5 overview

In YOLO v5, the CSP (Cross Stage Partial) Networks, which have shown significant improvement in processing time with deeper networks, are used as a backbone to extract rich in informative features from an input image. Model Neck is mainly used to generate feature pyramids, which help models to generalize well on object scaling and to identify the same object with different sizes and scales. SSP together with PANet are used as the neck to get feature pyramids.The model head is mainly used to perform the final detection part. It applies anchor boxes on features and generates final output vectors with class probabilities, objectness scores, and bounding boxes.
In YOLOv5, Leaky ReLU is used as the activation function in hidden layers and Sigmoid is used as the activation function in the final detection layer.

$$Sigmoid = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

$$Relu(z) = max(0, z) \tag{2}$$

To calculate the model loss, which is a compound loss calculated based on objectness score, class probability score, and bounding box regression score, we use binary cross-entropy with logits loss function from PyTorch.
So, for this project we used yolov5s and yolov5m for modeling. Different metrics are then used: Precision, Recall, and Mean average precision given in the equations below respectively.

$$Precision = \frac{TruePositives}{TruePositives - FalsePositives}$$

With the precision, we can determine when the cost of False positives is high.

$$Recall = \frac{TruePositives}{TruePositives - FalseNegatives}$$

The Recall calculates how many of the Actual Positives our model capture through labeling it as a True Positive.

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_n$$

$$AP = \sum_{k=0}^{k=n-1} [Recall(k) - Recall(k+1)] * Precision(k)$$

The mAP is used as a standard metric to analyze the accuracy of an object detection model. it is calculated by finding Average Precision(AP) for each class and then average over a number of classes.the Average Precision is calculated as the weighted mean of precisions at each threshold.
For the training phase, we decided to train both of yolov5s and yolov5m.
First we started training yolov5s with only 10517 images for 100 epochs and we had as result:

models	Precision	Recall	mAP@.5
yolov5s	0.845	0.927	0.93

Table 1: This is the caption for the second table.

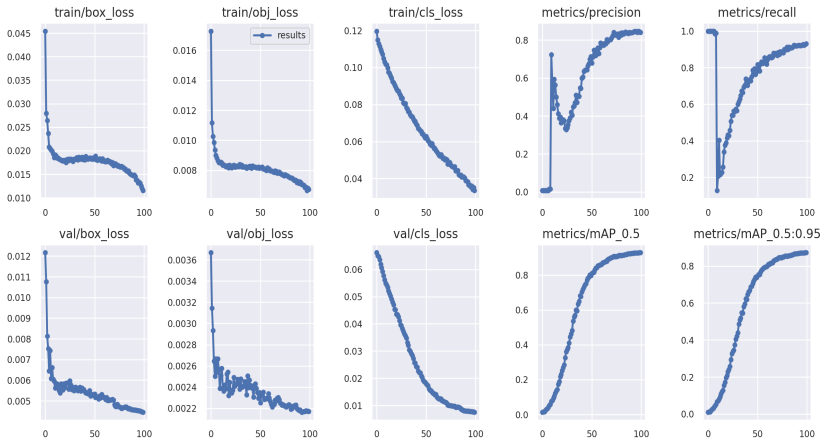

Figure 11: yolov5s training results

Then we decided to train yolov5m model with the same dataset for 150 epochs, and we noticed a clear improvement in our results:

models	Precision	Recall	mAP@.5
yolov5m	0.912	0.944	0.962

Table 2: This is the caption for the second table.

The both previous models were trained with a sample of the dataset, So now we are going to present the training process with the full dataset, we trained both of yolov5s and yolov5m models with 16185 images for 150 epochs and we can see the results in the table below:

models	Precision	Recall	mAP@.5
yolov5s	0.898	0.922	0.952
yolov5m	0.907	0.953	0.964

Table 3: This is the caption for the second table.

And We can notice that yolov5m is slightly better than yolov5s.

Figure 12: sample of detected cars.

3.3.2 software and libraries used in the implementation

In this section, i am going to list out the libraries in python which are being used heavily in this project :
Pandas : is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series.

This library is built on the top of the NumPy library. Pandas is fast and it has high-performance and productivity for users. We mainly used it to read the csv files that contain all the informations about the images used in this project.

matplotlib : Matplotlib is a widely used python data visualization library. It provides many different kinds of 2D and 3D plots that are very useful for data analysis and machine learning tasks. It also provides functionalities required for loading, rescaling and displaying images.

PIL : PIL is an open-source library for image processing tasks that requires python programming language. PIL can perform tasks on an image such as reading, rescaling, saving in different image formats. It can be used for image archives, image processing and image display.

opencv : OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

Pytorch : PyTorch is a widely used free open-source machine learning library for the Python programming language used mainly for deep learning and natural language processing. It is based on the Torch library and its main features are the support of tensors on Graphical Processing Units and automatic differentiation for building and training neural networks. We mainly used it to work with yolov5 model since YOLOv5 is written in the pytorch framework.

Tensorflow : TensorFlow is an open source framework developed by Google researchers to run machine learning, deep learning and other statistical and predictive analytics workloads. Like similar platforms, it's designed to streamline the process of developing and executing advanced analytics applications for users such as data scientists, statisticians and predictive modelers.

TensorBoard : TensorBoard is the interface used to visualize the graph and other tools to understand, debug, and optimize the model. It is a tool that provides measurements and visualizations for machine learning workflow. It helps to track metrics like loss and accuracy, model graph visualization, project embedding at lower-dimensional spaces.

3.4 Conclusion

In this chapter we discussed in details the techniques used in order to detect the cars models using Yolov5. We went into the details of the data used and manipulated in order to execute this task and tested multiple models in order to finish up with with very prominent results. In the last chapter we are going to discuss the methods we used for license plate detection and recognition.

4 Optical Character Recognition

4.1 Introduction

OCR (optical character recognition) is the use of technology to distinguish printed or handwritten text characters inside digital images of physical documents, such as a scanned paper document. The basic process of OCR involves examining the text of a document and translating the characters into code that can be used for data processing. And in this project, we used OCR for licence plate recognition which is a combination of image processing, character segmentation and recognition technologies used to identify vehicles by their license plates.
So, in this chapter we present the license plate detection and the techniques that we used for license plate recognition.

4.2 License plate detection

License plate detection is commonly the first procedure in a LPR system. It aims at locating the license plate, which provides the following LPR procedure with accurate region information. Instead of processing every pixel in the input image of the system, which is very time consuming, license plate detection is a necessary process before license plate recognition.
The dataset that we use in this part of the project is the tunisian license plates dataset. it labeled using Labelimg and it contains 1000 images for cars with Tunisian license plates and XML files that contains coordinates of each license plate in the image.

Figure 13: sample of cars images with tunisian license plates.

Usually, the tunisian license plates are placed in the front and back of the vehicle. The main objective of this stage is to locate the position of the license plate after detecting the vehicle model. The traditional approaches like edge detection, morphological operation and sliding window methods seem to show good results when the license plates are in good quality. With the advancement in computer vision technology, the number of deep learning approaches were proposed. In our project, the work was mainly done with YOLO 5 (You Only Look Once).
So, before training our YOLOv5 model, we should convert the XML files that contain the license plates annotations to YOLO format, for this task we used pylabel library which can translate bounding box annotations between different formats in order to help users to prepare datasets for computer vision models including Pytorch and yolov5.

Figure 14: car with tunisian license plate.

```xml
<annotation>
    <folder>images</folder>
    <filename>145.jpg</filename>
    <path>C:\Users\images\145.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>1000</width>
        <height>747</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>LP</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>350</xmin>
            <ymin>654</ymin>
            <xmax>638</xmax>
            <ymax>729</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 15: annotation in XML file.

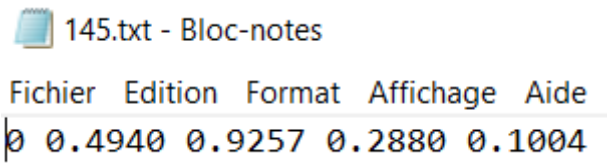

Figure 16: annotation in YOLO format.

Then we spilled our dataset in order to 70% for the training set, 20% for the test set and 10% for the validation set. Before training the model, there is a configuration file named "data.yaml" that needs to be modified.

```yaml
plotly > data.yaml
train: /content/Final_dataset/images/train
val: /content/Final_dataset/images/val
test: /content/Final_dataset/images/test

is_coco: False

nc: 1
names: ['LP']
```

Figure 17: Configuration file.

The results after training the yolov5s model on tunisian license plates dataset

for 150 epochs is displayed in the Figure 9. Precision on the validation set is 0.965, the recall is 1 and mAP@.5 is 0994 which are great results.

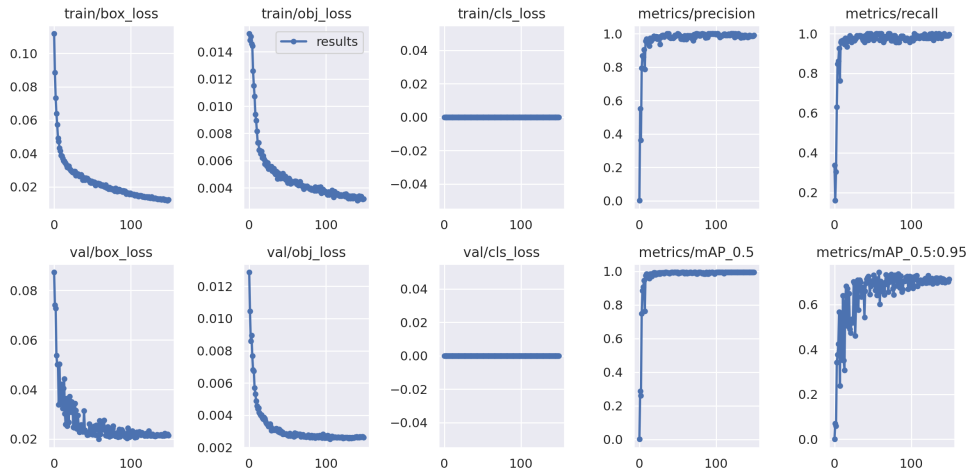

Figure 18: yolov5s training results.

Figure 19: sample of detected license plated.

After extracting the license plate from the images, we are ready for the next step.

4.3 License plate recognition

for this step, we decided to use 2 approaches:

- Using PaddleOCR
- Using a convolutional neural network

4.3.1 PaddleOCR

PaddleOCR is an ocr framework or toolkit which provides multilingual practical OCR tools that help the users to apply and train different models in a few lines of code. it offers a series of high-quality pretrained models. This contains three types of models to make OCR highly accurate and close to the commercial products. It provides Text detection, Text direction classifier and Text recognition.
Here is what the processing pipeline of PaddleOCR looks like.

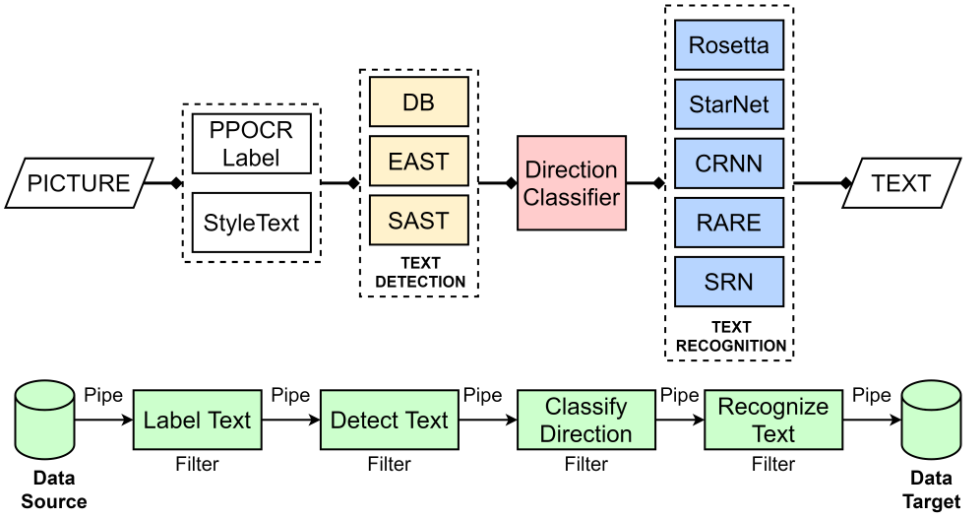

Figure 20: paddleOCR architecture.

PaddleOCR supports more than 80 languages so we won't face the problem of text recongition since the tunisian license plates contain arabic words.
In this section, we will implement PaddleOCR's PP-OCRv3. This model can be implemented in just a few lines of code and that too in a matter of milliseconds. First off, we installed the required toolkits and dependencies for the OCR implementation, we chose the language which we want to recognise then we initialized the model and this is the output of the model.

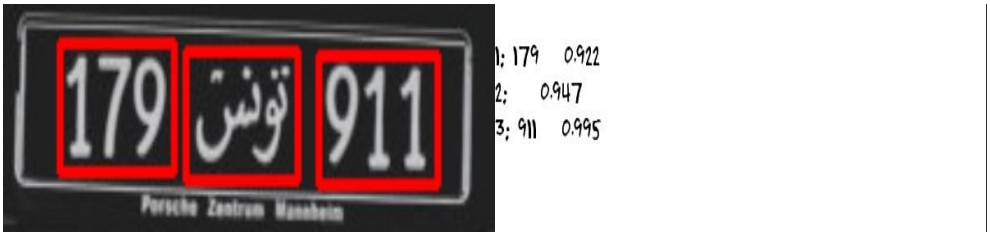

Figure 21: output of PaddleOCR (1) .

```
[2022/09/07 14:41:49] ppocr DEBUG: dt_boxes num : 3, elapse : 4.670226097106934
[2022/09/07 14:41:49] ppocr DEBUG: cls num  : 3, elapse : 0.017966032028198242
[2022/09/07 14:41:49] ppocr DEBUG: rec_res num  : 3, elapse : 0.020617008209228516
[[[24.0, 9.0], [73.0, 9.0], [73.0, 40.0], [24.0, 40.0]], ('179', 0.9224894046783447)]
[[[78.0, 11.0], [126.0, 11.0], [126.0, 42.0], [78.0, 42.0]], ('0.9466626644134521 ,'تونس)]
[[[135.0, 12.0], [186.0, 12.0], [186.0, 43.0], [135.0, 43.0]], ('911', 0.9946349263191223)]
```

Figure 22: output of PaddleOCR (2).

4.3.2 CNN

This chapter will be devoted to the presentation of the dataset, the training process of a classification model, then the extraction of digits from a license plate by using the CNN model.
First let's talk about the dataset, the dataset we used in this part of the project is contains 10606 images of digits, this is a classification dataset that can be used for computer vision tasks,it contains 10 classes from 0 to 9.

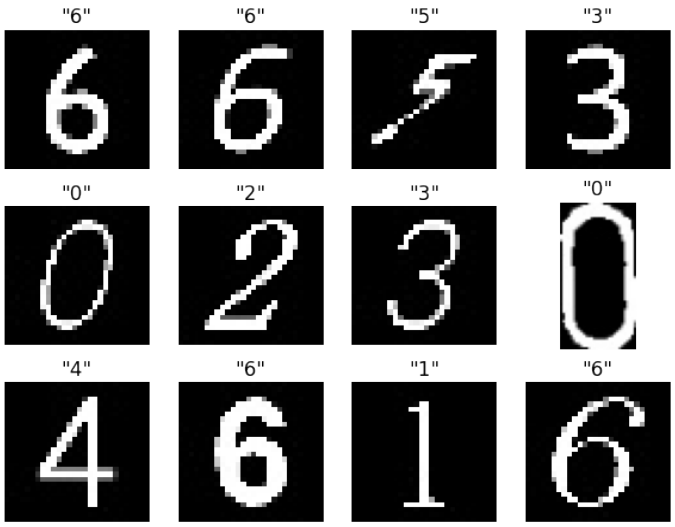

Figure 23: sample images of digits (0-9).

Then we used the data augmentation technique which is a technique of applying different transformations to original images which results in multiple transformed copies of the same image. Each copy, however, is different from the other in certain aspects depending on the augmentation techniques you apply like shifting, rotating, flipping, etc.

Figure 24: sample of images generated by using data augmentation.

After the data preparation phase, we started training our model. In this part of the project, we used MobileNetv2 which is a convolutional neural network that is 53 layers deep.
We created our model with pre-trained MobileNetV2 architecture from imagenet and we started training the model.we will use 'categorical crossentropy' as loss function, 'optimizer' as optimization function.
The results after training the model on digits for 150 epochs is displayed in the Figure 9. the model achieved an accuracy of 99.06

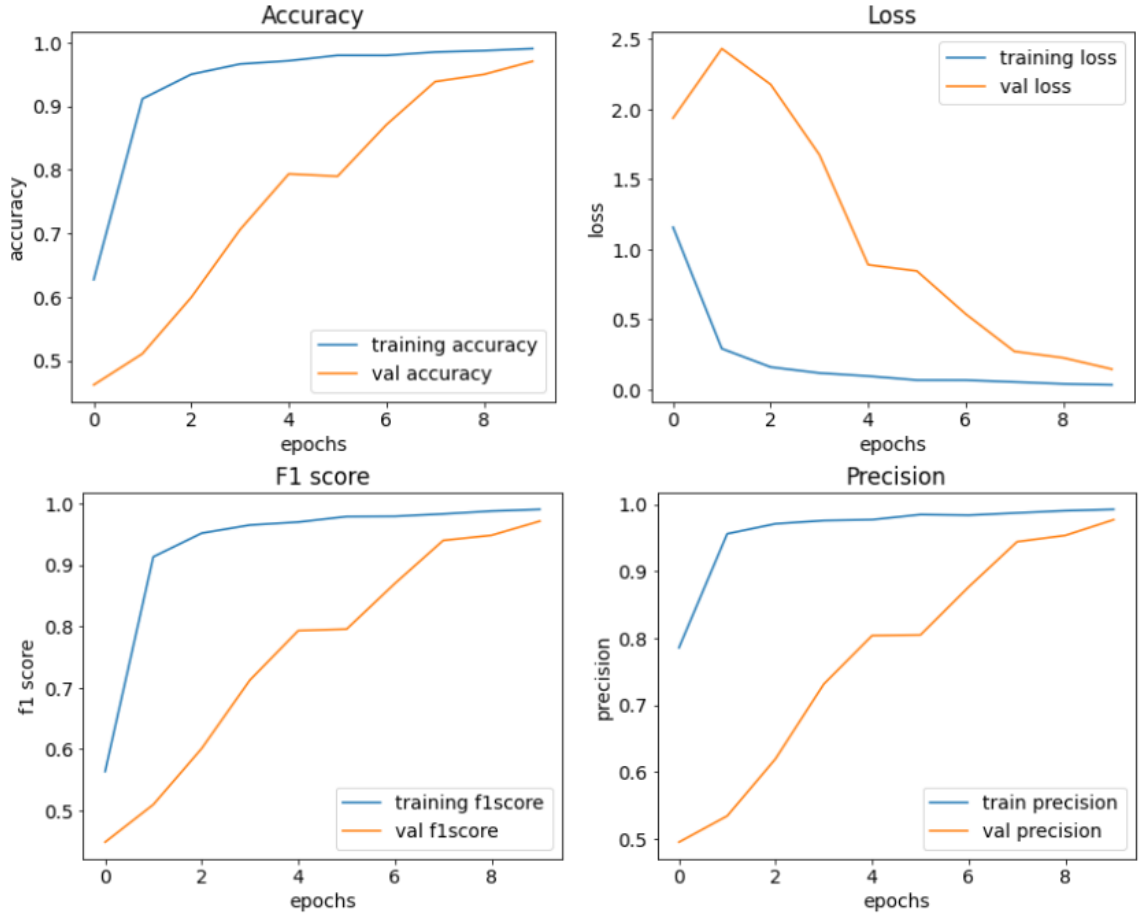

Figure 25: results of the training process.

So our model works well, next we need to prepare the license plate images.

first we normalized the license plate image pixel values by converting the image to float array and dividing it by 255,then we seek to convert our grayscale image to a binary image so we used the thresholding technique which consists of selecting a threshold value T, and then setting all pixel intensities less than T to 0, and all pixel values greater than T to 255 in order create a binary representation of the image. then we used the erosion and dilation techniques in order to remove noise and isolate the individual elements in the image.

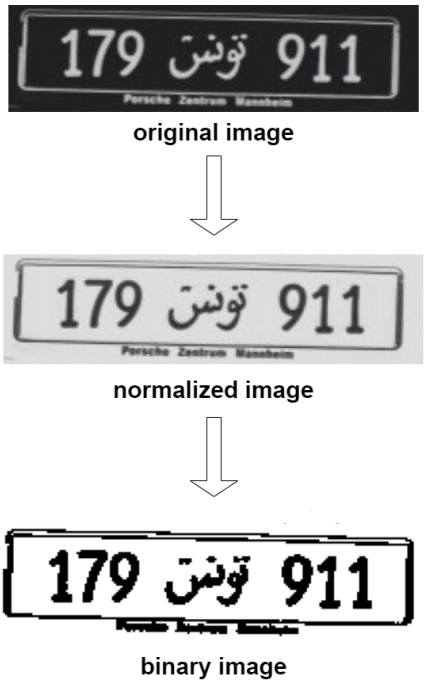

Figure 26: License plate pre-processing steps.

then we scrolled through the image to detect the contours in the image and return the coordinates of rectangle enclosing it and we don't want to detect all the contours so we checked the dimensions of every contour to filter out the digits by contour's size and finally we detected only digits in the image.

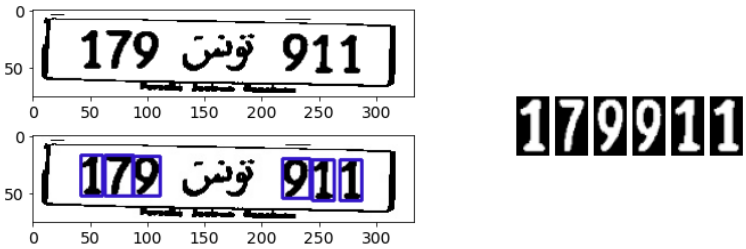

Figure 27: License plate segmentation.

Finally, we used our model to predict the digits in the image.

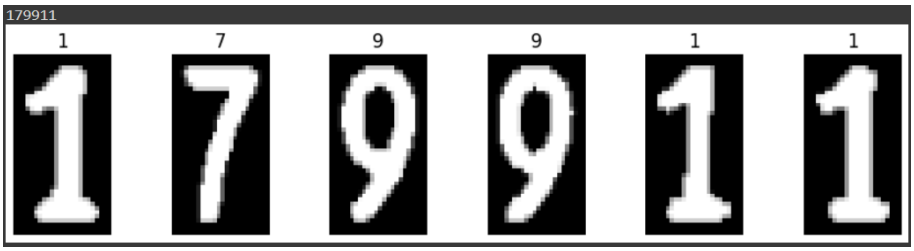

Figure 28: Model output.

4.4 Results

Finally, we can see that both of our approaches worked well, we had good results concerning PaddleOCR and our CNN model.

4.5 conclusion

In this chapter we discussed how this work was done, how we located and extracted the tunisian license plate from the images by using yolov5, and then how we extracted numbers from located license plates by using two approaches : PaddleOCR and MobileNetV2.

Conclusions drawn from all the previous parts clearly show that yolov5 performs well in detecting car's model,

4.6 Perspectives and alternatives

For the future of this project, we can use detectron2 models(Faster R-CNN, Fast R-CNN, Mask R-CNN, R-CNN) and we can correct the license plate alignment.