DEMOCRATIC AND POPULAR REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research
University of Mohamed Khider - BISKRA
Faculty of Exact Sciences, Natural Sciences and Life
Computer Science Department

Order Number: IVA12/M2/2019

REPORT

PRESENTED TO OBTAIN THE ACADEMIC MASTER DIPLOMA IN

COMPUTER SCIENCE

OPTION: ARTIFICIAL LIFE AND IMAGE

Vehicle Registration Plate Recognition

By:

Ganibardi Mohamed Salah

Defended the 11/07/2019, in front of the jury composed of:

Belaiche Hamza	MAA	President
Mokhtari Bilal	Phd	Supervisor
Mouaki Benani Nawel	MAA	Examiner

MOHAMED KHEIDHER UNIVERSITY

Abstract

Exact Sciences and Sciences of Nature and Life Faculty
Computer Science Department

Master of Computer Graphics

Vehicle Registration Plate Recognition

by Mohamed Salah GANIBARDI

Objects recognition and reading characters are a natural and easy process for humans which can tell easily if an object is a home or a car, if a letter is a ‘C’ not a ‘G’ or is a ‘O’ not a ‘Q’, also can distinguish between a letter and a number like the letter ‘O’ and the number ‘0’, or the letter ‘I’ and the number ‘1’, what makes this process so easy is the ability of human to learn. Is this process easy for a computer? Learning is part of human nature, and the hippocampus of brain is where this happens as one keep assimilating new information on any subject, the other is trying to skill itself in. For a computer is a difference story, how we can make a computer learn?

Keywords: object recognition, optical character recognition, real-time.

Contents

List of Figures

List of Tables

Introduction

Nowadays, the majority of the developing countries suffer from the traffic congestion issues, and use a huge amount of efforts to solve such issues. Because the number of vehicles are increasing in a way that will not be controllable in the future, which makes finding a stolen car or one's that violates traffic laws hard. Vehicle Number Plate Recognition system will be a good start for resolving not only the previous two problems, but we can use it in a wide varieties of places, such as controlling gates, traffic monitoring etc. However, in order to create such system, it will be needed to construct such algorithm which will be able to identify the location of number plate of the vehicle in the frame, then extract the characters from it, and then recognize them. In this paper we are going to review our number plate recognition application based on deep learning which have several operations that can helps for resolving the problems listed before.

The memory is organized as follows. In the first chapter, we give basis notions in the field of image processing and recognition. In the second chapter, we present neural networks and deep learning architecture. In the third chapter, we give in details the used Faster R-CNN architecture for detecting and recognizing objects. In the last chapter, we gives the conception and the implementation of our used system.

Chapter 1

Background

1.1 Introduction

In our topic we are interested in Alphanumeric Characters Recognition (OCR), which is a process that allows recognizing different alphanumeric characters from a scanned image. It is a key research area of pattern recognition, and it has been widely applied in automatic number-plate recognition check, street signs, invoice reading, voice reading of pdf documents, library call number recognition, and so on.

In this chapter, we will describe in details the field of alphanumeric characters recognition, and its use for number-plate recognition.

1.2 Optical Character Recognition

1.2.1 Definition

Optical Character Recognition usually called OCR, is the ability of the computer to detect and extract a printed or handwritten text characters inside digital images, or photograph might contain a street sign or traffic sign, etc, by examining the text and translating the characters into code that can be used for data processing.

1.2.2 Optical Character Recognition use cases

Recently, Optical Character Recognition engines have been developed for various application areas including [1] :

- License plates recognizing
- Scanning printed documents into editable versions
- Process cheques without human involvement in banking
- Process paperwork in health care
- Archiving historic documents, like newspapers, magazines, or phone books into searchable formats
- Read text from digital images
- Archiving signed legal documents into an electronic versions

- Etc...

1.2.3 OCR processing steps

1.2.3.1 Preprocessing

The first and the common step in the OCR processing is called preprocessing. In this phase there is a series of operations performed to the input digital image, it convert the image into a black and white image, were the dark areas are identified as characters that need to be recognized and white areas are identified as background. Other operation as filtering and adjusting illumination can also be applied to the input image [2].

1.2.3.2 Shape matching and pattern recognition

The variety of different fonts and ways of writing a single character makes the process of OCR hard to solve. The most common way to recognize a character is to apply a machine learning algorithm on a data set of examples in order to detect patterns which allow characterizing each class with a signature which distinguish it from other classes. We can resume this on the following concepts.

1.2.3.2.1 Feature extraction

The success of OCR depends on the stability and accuracy of extracted features. The quality of the extracted features will directly affect the recognition result. According to the unified entropy theory of pattern recognition, we must ensure that the extracted features contain sufficient information to get a higher recognition rate. It consists in extracting some features from the shape of the objects in the images. These features are a kind of numerical values that can be easily used to calculate the distance between shapes. For example, possible features may be the length and the height of the object. In the literature, a variety of methods have been developed for extracting features from images, as well as characters images [Huang, Zhihu, 2010, Freeman, 1995, Dalal, Navneet,2005, Rao, N,2016].

1.2.3.2.2 Shape matching

Shape matching is the process of measuring the similarity between two given images. To do so, we calculate a distance between features extracted from images. Note that small distance between images means that they are similar and they belongs to the same class. For example, distances between different handwritten of the digits 9 must be small as possible.

Actually, the results of calculating a shape feature for an image provides one dimensional vector of values. The distance between two shapes A and B or images described each one by a vector of features is simply the Euclidean distance norm between their respective vectors v_1 and v_2.
The distance for two vector v_1 and v_2 is than given by :

$$d(A,B) = d(v_1,v_2) = \sqrt{\sum_{i=0}^{n-1}(v_i^A - v_i^B)^2}$$

Were n is the size of the vectors, and v_i^A and v_i^B are respectively the ith elements in the vectors v_1 and v_2

1.3 Machine learning

Machine Learning is a sub-area of artificial intelligence, it enables it systems to recognize patterns on the basis of existing algorithms and data sets and to develop adequate solution concepts . It is a king of algorithms which can build a model that make prediction for regression, classification tasks based only on given known data (features), and similarity measurement (distance). For example, we can find a model that predicts the price of an house based on known prices of others houses according to given features (such as area, number of pieces, etc). The required algorithms and data must be fed into the systems in advance and the respective analysis rules for the recognition of patterns in the data stock must be defined. Once these two steps have been completed, the system can perform the following tasks by Machine Learning :

- Finding, extracting and summarizing relevant data
- Making predictions based on the analysis data
- Calculating probabilities for specific results
- Adapting to certain developments autonomously
- Optimizing processes based on recognized patterns

The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis [Vapnik, Vladimir,1999] to predict an output while updating outputs as new data becomes available. In the literature, they are a lot of existing learning algorithms such as K-nearest neighbors, k-means, Support Vector Machine (SVM) [Laura Auria, 2008], neural network [Zurada, Jacek,1992], deep learning [LeCun, Yann,2015]. In this work, we will use Deep learning based algorithms.

1.4 Segmentation

1.4.1 Definition

Segmentation is one of the most important processes of image processing. Image segmentation technique is used to partition an image into meaningful parts having similar features and properties, every pixel in an image is allocated to one of a number of these features and properties.

Pixels belonging in the same region must have similar characteristics based on neighbors pixels (or close to each other), and those of different region must be different. Segmentation can be used for simplification, making image more easily analyzable, and selecting regions of interest. Thus, in our work, we use this process to select the number-plate from an image.

1.4.2 Segmentation approaches

Segmentation methods can be categorized into two types based on properties of image.

1.4.2.1 Discontinuity detection

This approach worked by partition an image, based on changes in gray-level using three types of detection [3] :

- Point detection : The detection of isolated points in an image by using a mask. we can say that a point has been detected at the location on which the mask is centered, if :

  $$|R| > T$$

  where T is the threshold and

  $$R = -(x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9) + 8 * x_5$$

  x_i is an image pixel [3].

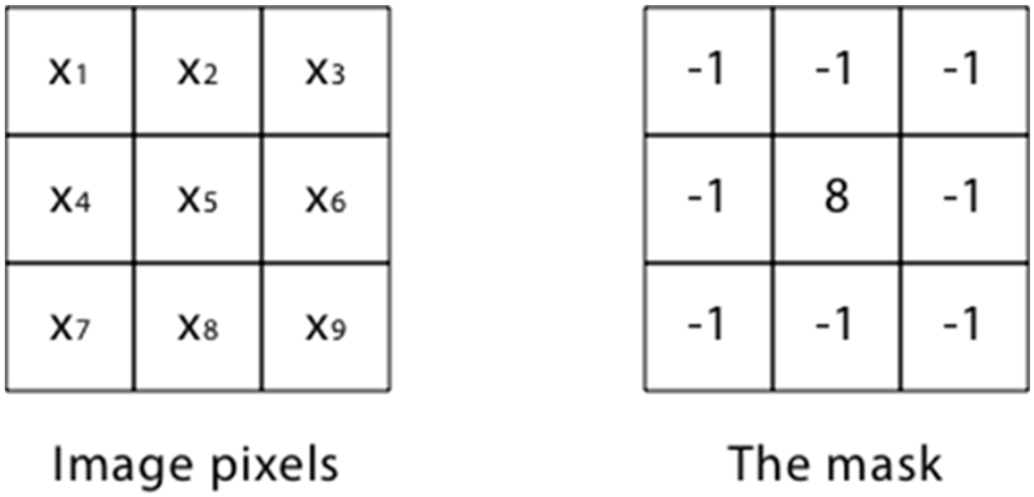

FIGURE 1.1: Point detection mask.

The gray level of an isolated point will be quite different from the gray level of its neighbors.

- Line detection : Detection of lines in an image using the following four masks R1, R2, R3 and R4 (see Figure1.2). If the first mask (see Figure1.1) were moved around the whole image, it would respond better to only one line oriented horizontally. With constant background, the maximum response would result when the line passed through the middle row of the mask. R1, R2, R3 and R4 run along the same image. Then at a certain point, we can say that the mask which has the maximum response, will make the line into its direction, i.e. if R2> R1, R3, R4 then the line has the direction 45° [3].

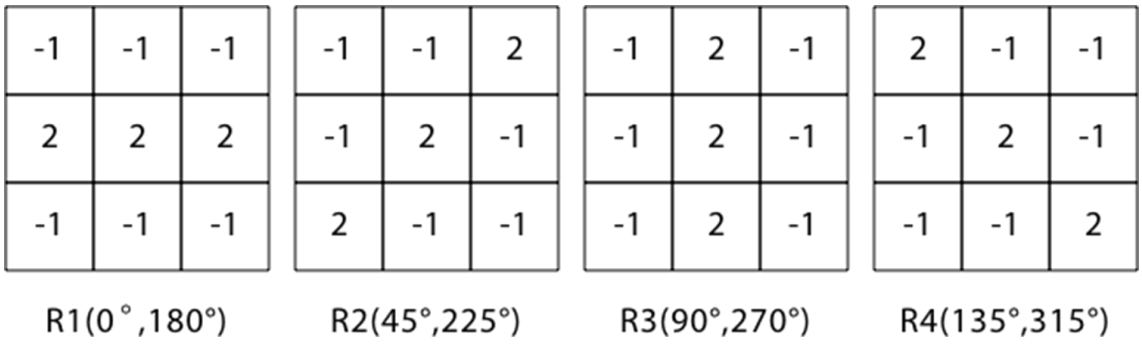

FIGURE 1.2: Line detection masks.

- Edge Detection : Edge detection methods transform an image into edge image benefits from the changes of grey tones in the image [4].

FIGURE 1.3: Edge detection example.

1.4.2.2 Similarity detection

In this methods, the image is segmented based on similarity, using several techniques, like thresholding techniques, region growing techniques and region splitting and merging, to divide the image into regions having similar set of pixels. The clustering techniques as K-means also use this methodology. The image is so divided into different clusters containing each one similar features based on a given criteria [3].

1.4.3 Segmentation for OCR

In order to achieve the Optical Character Recognition goal, we need first to use the segmentation process to extract words from the image. This is done in two different steps: line segmentation, and segmenting words.

1.4.3.1 Line segmentation

It consists of slicing a page of text or a zone of interest into its different lines, We can identify a line of text by looking for rows of white pixels with rows of black pixels in between [3] (see Figure 1.4).

FIGURE 1.4: Line segmentation example.

1.4.3.2 Segmenting words and characters

This step consist in isolating each word from another and separate the various letters of a word, by using the horizontal white space between words, which is called “interword space” to separate them, and using interletter space (horizontal white space between letters) to separate letters [3] (see Figure 1.5).

FIGURE 1.5: Segmenting words and characters.

1.5 Vehicle Plate Recognition

1.5.1 Definition

Vehicle Plate Recognition have been one of the most interesting path in the history of video surveillance, it uses Optical Character Recognition to make a computer capable of detecting and extracting text characters inside digital images contains a vehicle plate, of the purposes like traffic safety enforcement, automatic toll text collection, and vehicle parking system. Vehicle Plate Recognition can use existing road cameras or specified ones. In follows, different kind of approaches used for vehicle plate recognition.

1.6 Conclusion

In this chapter, we presented the concept of Optical Character Recognition, and its possible application for vehicle plate recognition. One of most successful strategy used in this fields is the uses of deep learning algorithms. Therefore, we will describe this kind of methods in the next chapter, and gives a detailed description of the algorithm we used.

Chapter 2

Neural Network

2.1 Introduction

Machine learning has widely used in the field of image recognition, and it allowed to achieve a spectacular results. In our work, we will use one of powerfull machine leaning techniques which is called Artificial Neural Networks. In this chapter, we will describe the basis of this concept.

2.2 Artificial Neural Network

An Artificial Neural Network is a computational model that is designed to model the way in which the brain performs a particular task or function of interest. Neural networks employ a massive interconnection of simple computing cells referred to as "neurons" inspired by the natural neurons to achieve the best performance possible [5]. Artificial Neural Network acts like the brain in two way:

- Knowledge is acquired by the network through learning process.
- Inter neuron connection strengths known as synaptic weights are used to store the knowledge.

An artificial neuron is a computational model inspired in the natural neurons. Natural neurons receive signals through synapses located on the dendrites or membrane of the neuron. When the signals received are strong enough (surpass a certain threshold), the neuron is activated and emits a signal though the axon. This signal might be sent to another synapse, and might activate other neurons.

2.2.1 Biological neural network

A neuron is a switch with information input and output. Will be activated if there are enough stimuli of other neurons hitting the input. Then, at the output, a pulse is sent to other neurons. Any single physiological action perceived, such as pain or pleasure is the output response of a collective activity due to innumerable neurons participating in the decision making control procedures in the nervous system [6].

2.2.1.1 Neural networks components

Neurons are made up of a nerve cell composed by a cell body called soma, Each soma has many dendrites and one axon. Axon signal can split hundreds of times, however,

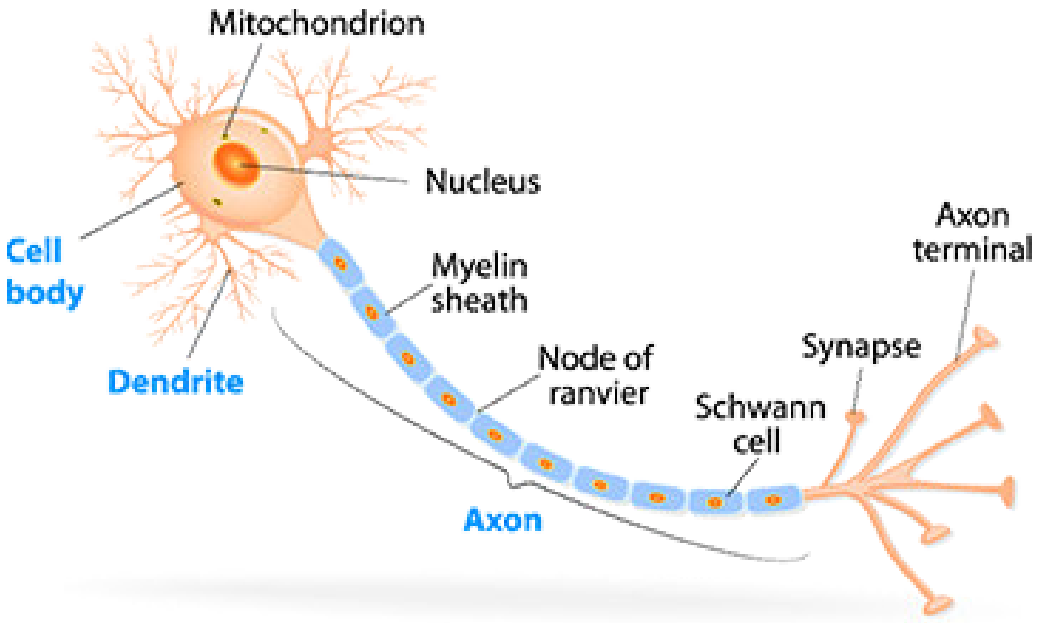

FIGURE 2.1: Biological neuron diagram [7].

Dendrites are thin structures that arise from the main cell body. Axons are nerve fibers with a special cellular extension that comes from the cell body.
Natural neurons contain hundreds of inputs. Dendrites are part of it. Synapse is characterized by effectiveness, called synaptic weight. Neuron output is formed in a following way : signals on dendrites are multiplied by corresponding synaptic weights, results are added and if they exceed threshold level on the result is applied a transfer function of neuron. Only limitation of transfer function is that it must be limited and non-decreasing. Neuron output is routed to axon, which by its branches transfers result to dendrites. In this way, output from one layer of network is transferred to the next one [6].

2.2.2 The perceptron

2.2.2.1 Definition

Perceptron is a single layer neural network and a multi-layer perceptron is called Neural Networks. Perceptron is a linear classifier (binary) and it is used in supervised learning. It helps to classify the given input data. with a simple input output relationship which shows we are summing n number of inputs multiplied with their associated weights and then sending this input to a another function with a defined threshold. Normally with perceptrons, this is a Heaviside step function with a threshold value of 0.5. This function will give a real valued single value (0 or a 1), depending on the input.

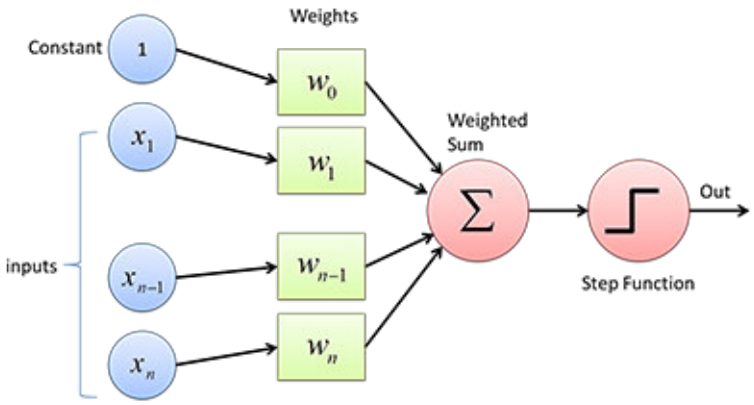

FIGURE 2.2: Perceptron.

2.2.2.2 The perceptron learning algorithm

The perceptron learning algorithm goal is to find the weights vector that can perfectly classify positive inputs and negative inputs in our data by changing them until all input records are all correctly [8].

2.2.2.3 Limitations of the perceptron

Linear models like the perceptron cannot represent some functions, can only learn to approximate the functions for linearly separable datasets. The linear classifiers that we have examined find a hyperplane that separates the positive classes from the negative classes, if no hyperplane exists that can separate the classes, the problem is not linearly separable. Where a multilayer perceptron could solve many nonlinear problems.

2.2.3 Multilayers neural network

This multi-layer network has different names : multi-layer perceptron (MLP), feed-forward neural network, artificial neural network (ANN), backprop network. It is a neural network with one input layer, one or more hidden layers, and one output layer. Each layer contains an artificial neurons or more. These artificial neuron of the multilayer perceptron is similar to its predecessor, the perceptron, but it adds flexibility in the type of activation layer we can use, wich is more generalized [5].

2.2.3.1 Artificial neuron input

The artificial neuron input are based on the weights on the input connections. These input passed to the activation function or they can be ignored by a 0.0 weight on an input connection.
The net input is produced by multiplying the weights on connections by activation, the total weighted input of the artificial neuron can be represented as

$$I_i = W_i.A_i$$

where W_i present the vector of the weights leading into neuron i and A_i is the vector of activation values for the inputs to neuron i. Build on this equation by accounting the bias term that is added per layer

$$I_i = W_i.A_i + b$$

For producing the output, we have to wrap this I_i with an activation function σ

$$a_i = \sigma(I_i)$$

The output value passed to the next layer for the neuron i is the activation value a_i, the output value passed through connections to other artificial neurons as an input value. In case of the activation function is the sigmoid function :

$$f(z) = \frac{1}{1+e^{-z}}$$

The range of the output will be between 0 and 1, which is the same output as the logistic regression function.

2.2.3.2 Weights

The weights are the coefficients that use to amplify or minimize an input signal, which represent the strength of the connection between neurons, and decides how much influence the input will have on the output.

2.2.4 Biases

A bias has it's own connection weight, what makes sure that even when all the inputs are none there's gonna be an activation in the neuron.

2.2.4.1 Activation function

It's used to introduce non-linearity to neural networks, it defines the status of node if it should be activated or not based on the weighted sum. The Sigmoid activation function is one of the most used activation function, it squashes values between a range 0 to 1.

2.2.4.2 Multilayer neural network architecture

We have artificial neurons arranged into groups called layers. Building on the layer concept, we see that the multilayer neural network has the following concept: A single input layer. One or many hidden layers, fully connected. A single output layer [8]. The neurons in each layer are all fully connected to all neurons in all adjacent layers. The neurons in each layer all use the same kind of activation function. For the input layer, the input is the raw vector input. The input to neurons of the other layers is the output of the previous layer's neurons. As data flows through the network in a feed-forward fashion, it is impacted by the connection weights and the activation function kind. Let's now take a look at the specifics of each layer type.

2.2.4.3 Input layer

The input layer is responsible to receive the initial data for the neural network then duplicate it to it multiple outputs. The number of neurons in an input layer is the same number as the input feature.

2.2.4.4 Hidden layer

Hidden layer have neurons, it's the intermediate between input and output layer which apply different transformations to the input data, there are one or more layers. To encode the learned information extracted from the raw training data, hidden layers use the weight values.

2.2.4.5 Output layer

Output layer is the last layer in the network, the active nodes of the output layer combine and modify the received data from the last hidden layer data to produce the result for given inputs [8].

2.2.4.6 Connections

The connections between neurons in one layer to other neurons in other layer or the same layer. It has it weight value. The Goal of the training is to update this weight value to decrease the loss(error).

2.3 Deep Learning

Deep learning is a neural network with more than two layers. A computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers [9].

2.4 History of deep learning application

The path of deep learning was very rich, in the late 80s and early 90s, deep learning advances in modeling sequential data with recurrent neural networks, as time went on, the research community created better artificial neuron variants over the course of the late 90s. During the 2000s, researchers and industry applications began to progressively apply these advances in products like Google Translate, Amazon Echo, Self-driving cars.

2.5 Deep learning applications

Deep learning excels at identifying patterns in unstructured data, which most people know as media such as images, sound, video and text. Below is a list of sample use cases we've run across :

- TIMIT phoneme recognition (Graves et al., ICASSP 2013).
- Optical character recognition (Breuel et al., ICDAR 2013).
- Language identification (Gonzalez-Dominguez et al., Google, Interspeech 2014).
- Text-to-speech synthesis (Fan et al., Microsoft, Interspeech 2014).
- Prosody contour prediction (Fernandez et al., IBM, Interspeech 2014).
- Large vocabulary speech recognition (Sak et al., Google, Interspeech 2014).
- Medium vocabulary speech recognition (Geiger et al., Interspeech 2014). 2014).
- English-to-French translation (Sutskever et al., Google, NIPS 2014).
- Audio onset detection (Marchi et al., ICASSP 2014).
- Social signal classification (Brueckner Schulter, ICASSP 2014).
- Arabic handwriting recognition (Bluche et al., DAS 2014). 2014).
- Image caption generation (Vinyals et al., Google, 2014).
- Video-to-textual description (Donahue et al., 2014).
- Syntactic parsing for natural language processing (Vinyals et al., Google, 2014).
- Photo-real talking heads (Soong and Wang, Microsoft, 2014).

2.6 Deep learning architecture

Deep learning architecture can be divide in three layers, the input layer, multiple hidden layers and output layer, where the depth of DL architecture is defined by the number of hidden layers.
Depending on the type of hidden layers used, different non-linear functions can be learned [10].
The next two equations (2.1), (2.2) represent a simple Artificial Neural Network, where the equation (2.1) (hidden layer) represents the non-linearity observed within the data, the activation function determines the non-linearity characteristics. The equation (2.2) (output layer) produce prediction from the previous non-linearity characteristics .

$$A_1 = Activation(W_1 X + b_1) \tag{2.1}$$

$$y = W_2 A_1 + b_2 \tag{2.2}$$

2.6.1 Activation functions

The activation functions in the hidden layer, its goal is helping in mapping the non-linearity relationship between input and output. Most used activation functions in hidden layers are sigmoid and hyperbolic tangent function (e.g., tanh). There is no rule for applying specific activation functions, depending on the dataset we use, an activation function need to be evaluate [10].

2.6.2 Hidden layers

Each hidden layer complexity is defined by the number of parameters used to represent it, where the number of parameters is controlled by hyper-parameters, which are the number of hidden units and the parameter of the L2 regularisation.
The number of hidden units indicates the number of parameters (i.e. weights) in each layer and L2 regularisation reduces the magnitude of these parameters to avoid over-fitting [I. Goodfellow, Y. Bengio, A. Courville, Deep Learn. (2016).]. Tuning hyper-parameters is important to obtain a good model fit [10].

2.6.3 Output layer

Typically, the output consist of an identity activation (also referred to as linear activation) for regression problems, what makes negative predictions possible. In case where data used demands are non-negative response values, identity activation is not suitable. Instead, the rectified linear activation function can be used [10].

2.6.4 Parameters

In deep networks, we still have a parameter vector representing the connection in the network model we're trying to optimize. The biggest change in deep networks with respect to parameters is how the layers are connected in the different architectures [10].

2.6.5 Loss functions

The loss functions quantify the agreement between the predicted output (or label) and the ground truth output. The loss functions is used to determine the penalty for an incorrect classification of an input vector. The following are examples of some of the loss functions:

- Hinge loss
- Squared loss
- Logistic loss
- Negative log likelihood

2.6.6 Optimization methods

Training a model in machine learning involves finding the best set of values for the parameter vector of the model. We can think of machine learning as an optimization problem in which we minimize the loss function with respect to the parameters of our prediction function (based on our model) [11].

2.6.7 Hyperparameters

Hyperparameters are free to be chosen by the user that might affect performance. Hyperparameters fall into several categories [10] :

- Layer size
- Magnitude (momentum, learning rate)
- Regularization (dropout, drop connect, L1, L2)
- Activations (and activation function families)
- Weight initialization strategy
- Loss functions

2.7 Conclusion

In this chapter, we have presented Deep Learning which uses uses the concept of artificial neural networks that is based from the biological neural networks, what makes humans able to learn faster, with such inspiration it is possible to make a computer able to learn from images and recognize its contents.

Chapter 3

Faster Region-based Convolutional Neural Network

3.1 Introduction

The process of recognizing objects in an image pass through two steps: detection objects, and classify them. Detection consists in drawing a bounding box around the object, and the recognition consists in associating the detected object to a category. The called Convolution Neural Networks (CNN) is one of the most technique widely used to detect and recognize objects. However, this kind of methods have some drawbacks, especially because they are a fully connected layers, and need a huge number of fixed regions which are trained in a considerable time. A new kind of convolution networks such as Faster Region-based Convolution Neural Network (R-CNN) [12], Efficient and Accurate Scene Text detector (EAST), and You Only Look Once (YOLO) [13] are proposed to resolve these issues. In our work we opted to use Faster R-CNN, we are going to describe in this chapter.

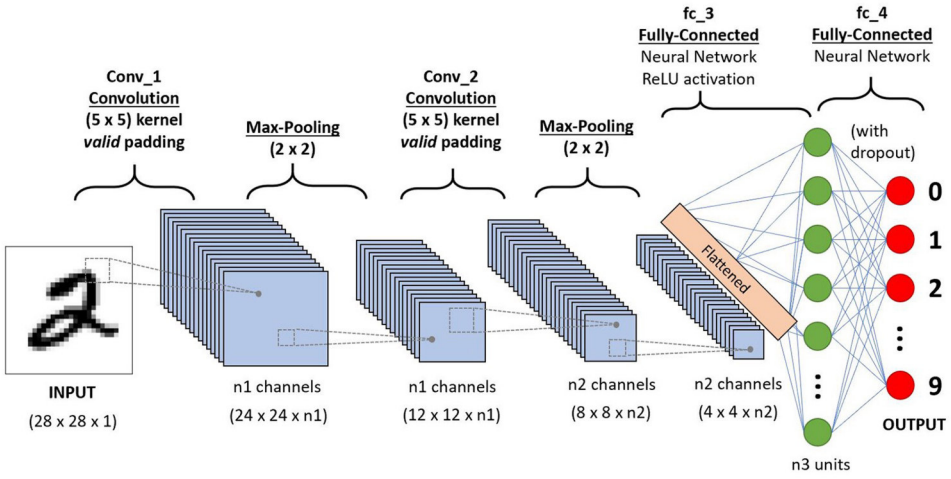

FIGURE 3.1: An example of CNN architecture

3.2 Convolution Neural network

In order to describe R-CNN, we have first to introduce the CNN architecture. CNN [14] is a deep learning network, which combines neural networks concept and convolutions. These convolutions are a hidden layers, which can be seen as a filter (also called kernel, or mask) applied to the input image.

The overall architecture of the Convolutional Neural Network (CNN) includes an input layer, multiple alternating convolution and max-pooling layers, one fully-connected layer and one classification layer [15].

3.2.1 Feature extraction

Convolutional layers transform data from the input layer using the previous layer patch. each layer will compute a dot product between the region of the neurons in the input layer and the weights to which they are locally connected in the output layer [15]. Actually, Convolutional layers takes input, applies a convolution kernel, and gives us a feature map as output.

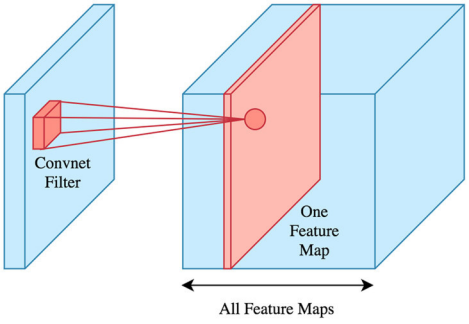

FIGURE 3.2: Feature maps [16].

The convolution operation goal is convolution neural network feature detection. Have as input a raw data, or a feature map output from another convolution.

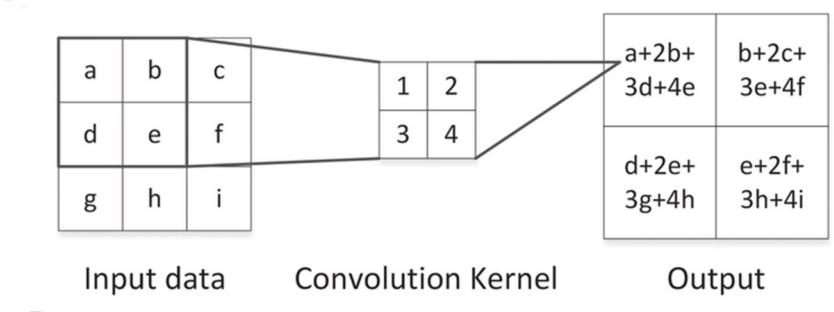

FIGURE 3.3: An example of convolution operation

The kernel slid across the input data to produce the output data. While sliding, the kernel is multiplied by the input data values within its bounds, creating a single entry in the

output feature map. In practice the output is large if the feature we're looking for is detected in the input. We commonly refer to the sets of weights in a convolutional layer as a filter (or kernel). This filter is convolved with the input and the result is a feature map (or activation map) [15].

Convolutional layers perform transformations on the input data volume that are a function of the activations in the input volume and the parameters (weights and biases of the neurons). The activation map for each filter is stacked together along the depth dimension to construct the 3D output volume. Convolutional layers have parameters for the layer and additional hyperparameters . Gradient descent is used to train the parameters in this layer such that the class scores are consistent with the labels in the training set. The major components of convolutional layers are :

- Filters
- Activation maps
- Parameter sharing
- Layer-specific hyperparameters
- Learned filters and renders
- Rectified Linear Unit (ReLU) activation functions.

3.2.2 Pooling

Pooling layers are commonly inserted between successive convolutional layers. Its goal is to reduce the spatial size of the representation. Pooling layers reduce the data representation progressively over the network and help control overfitting.
The Pooling Layer operates independently on every depth slice of the input, it uses the MAX operation to resize the input data spatially.
One of the most used pooling layer form is with a 2x2 filter size, taking the largest of four numbers in the filter area.
Pooling layers perform downsampling operations along the spatial dimension of the input data using filters (kernels).
This means that if the input is a 32x32 image, the output image would have smaller dimensions (e.g., 16x16).
The most common setup for a pooling layer is to apply 2x2 filters with a stride of 2. This will downsample each depth slice in the input volume by a factor of two on the spatial dimensions (width and height). This downsampling operation will result in 75 of the activations being discarded. Pooling layers do not have parameters for the layer but do have additional hyperparameters. This layer does not involve parameters, because It is not common to use zero-padding for pooling layers.

3.2.3 Fully connected layers

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset. See the Neural Network section of the notes for more information.

3.3 Faster Region-based Convolutional Neural Network

Faster Region-based Convolutional Neural Network or Faster R-CNN is an approved version of CNN, which is widely used for object detection. It takes an image as input and outputs bounding boxes around objects of interest with class labels. In our work, we will use it to detect registration plate in an image.

Befor talking about Faster R-CNN, we have to understand the original of it, which is the R-CNN. it main idea is to use search selective method to find the regions of interests and passe them to ConvNent, R-CNN tries to find out the areas that have possibility to be an object combining the similar pixels and textures into boxes. From search selective it uses 2000 proposed boxes. These boxes will passe to the pre-trained CNN model, where regression between the predicted bounding boxes and the ground-truth bounding boxes are computed.

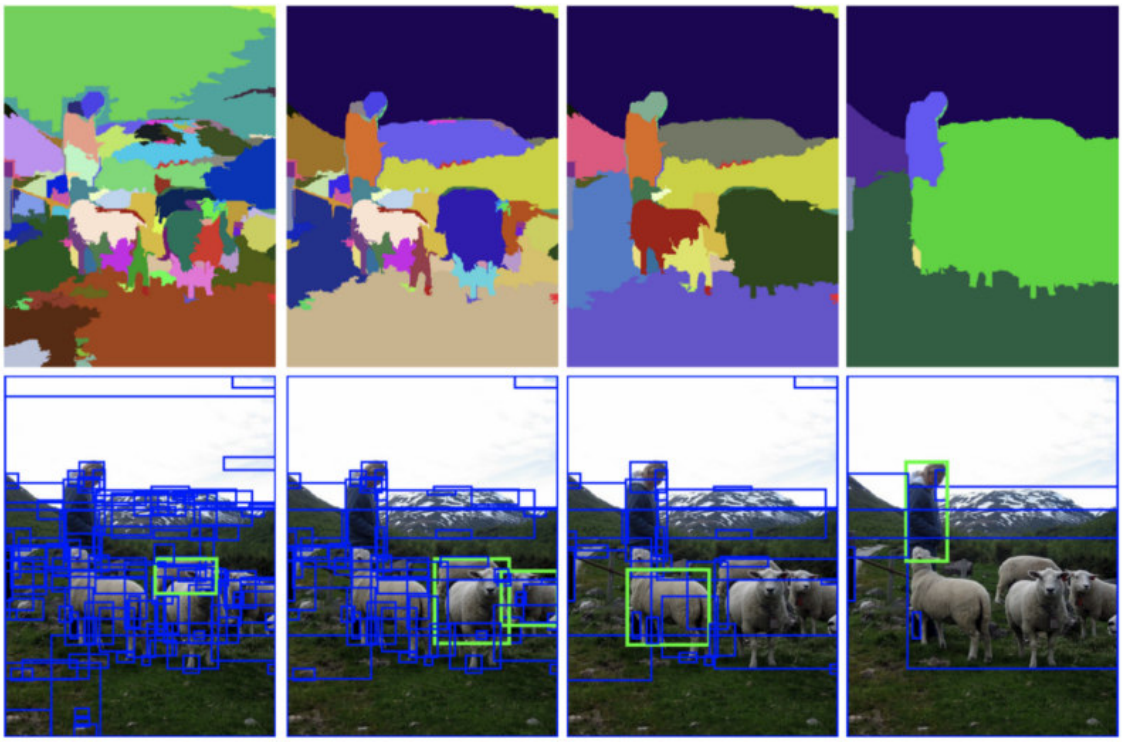

FIGURE 3.4: An example of search selective [17].

The next step is Fast R-CNN. What makes Fast R-CNN better than the original R-CNN is that Fast R-CNN passes only the original image to the pre-trained CNN, instead of

applying 2000 times CNN to proposed boxes. In this case the search selective algorithm is computed base on the output feature map, after that, it ensure the standard and pre-defined output size by using ROI pooling layer. Then it passes the valid outputs to a fully connected layer.
Two output vectors are used to predict the observed object with a softmax classifier and adapt bounding box localisations with a linear regressor.
Faster R-CNN moves forward than Fast R-CNN by replacing search selective process by Region Proposal Network (RPN), which acts as an attention network that identifies the regions the classifier should consider. It became the state of the art in object detection when it was released in 2015. It takes an image as input and runs it through 2 modules: the first uses a RPN that proposes object regions and the second is a Fast R-CNN object detector that classifies the region proposals.

3.3.1 Faster R-CNN architecture

Faster R-CNN has several moving parts, which makes it, starting from an image to obtain a list of bounding boxes, a label assigned to each bounding box and a probability for each label and bounding box [18].

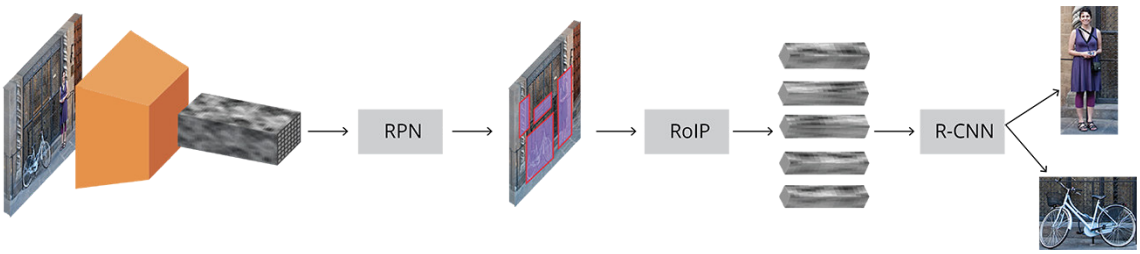

FIGURE 3.5: Complete Faster R-CNN architecture [19]

A 3 dimensions arrays (Height x Width x Depth) represents the input images. We use this as a feature extractor after are passing these arrays (tensors) through a pre-trained CNN up until an intermediate layer, ending up with a convolutional feature map [18].
The next step is Region Proposal Network (RPN). RPN is used to find up to a predefined number of regions (bounding boxes) using the features that the CNN computed.
The variable-length list of bounding boxes are generated by using anchors, where fixed sized reference bounding boxes are placed uniformly throughout the original image.
Every anchor have two problems to solve :

- Does it contain a relevant object?
- How we can adjust it to fit correctly the relevant object?

After having the list of possible relevant objects with their locations, and by using the features extracted by the CNN and the bounding boxes with relevant objects, we can apply Region of Interest (RoI) Pooling and extract those features which would correspond to the relevant objects into a new array [18].
In the end, the R-CNN module uses that array to classify the content in the bounding box, and adjusts it coordinates to fits better the object.

3.3.2 Base network

Acctualy, the first step of the Faster R-CNN is using a pre-trained CNN for classification and using the output of an intermediate layer. It's important to understand how and why it works. So, we choose the standard VGG-16 as an example [19].

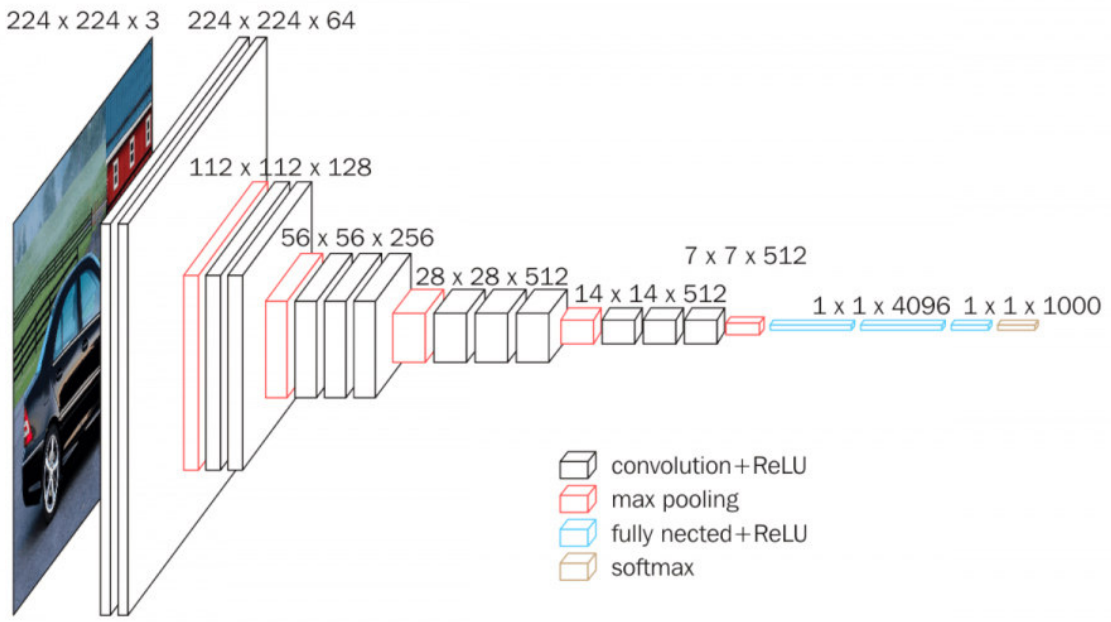

FIGURE 3.6: Base network VGG-16 architecture [19].

Each convolutional layer creates abstractions based on the output of the previous layer. The first layers are using for learn edges, and the second are using to find patterns in edges in order to activate for more complex shapes.
Like a results we have a convolutional feature map which has encoded all the information for the image while maintaining the location of the things. This convolutional feature map has spatial dimensions much smaller than the original image, but greater depth, the results of applying of the pooling between convolutional layers. The depth increases based on the number of filters the convolutional layer learns [19].

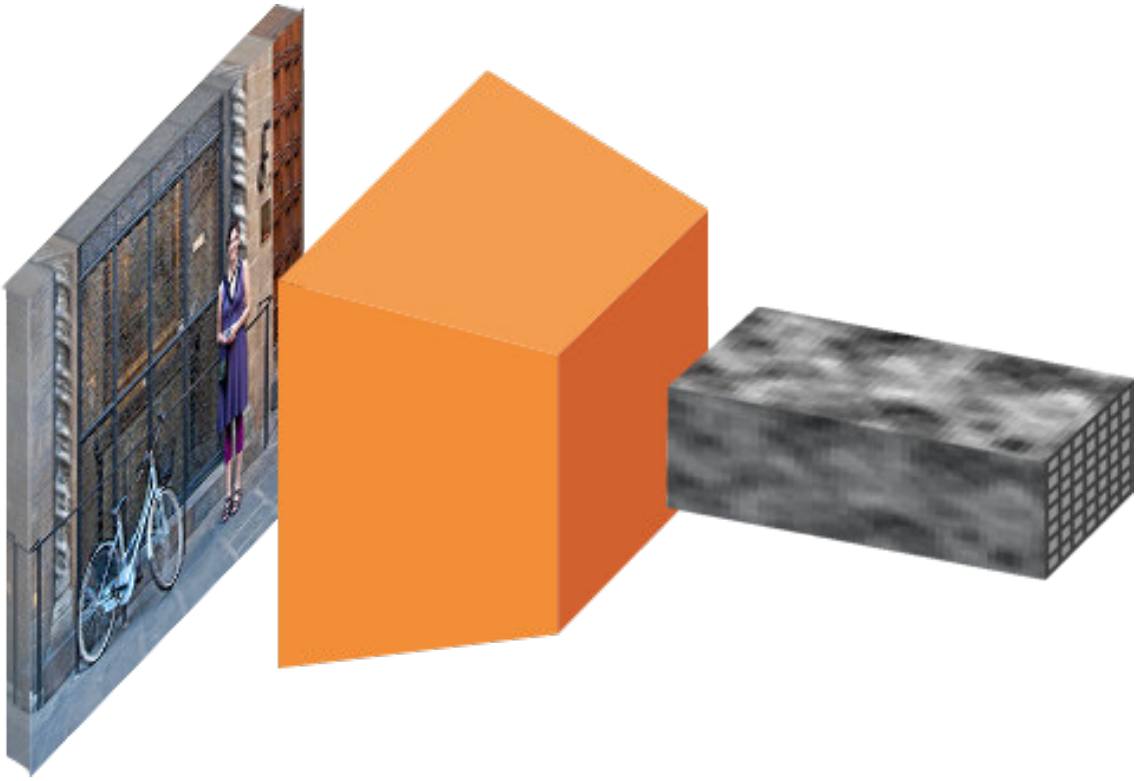

FIGURE 3.7: Image to convolutional feature map.

By today's standards VGG would not be considered very deep, ResNet architectures have mostly replaced VGG as a base network for extracting features.
ResNet is bigger than VGG, it has more capacity to actually learn what is needed. Also, ResNet uses residual connections and batch normalization wich makes it easy to train deep models.

3.3.3 Region Proposal Network

The RPN uses a convolutional layer with 512 channels and 3x3 kernel size, then, two convolutional layers using a 1x1 kernel with a number of channels depends on the number of anchors per point.

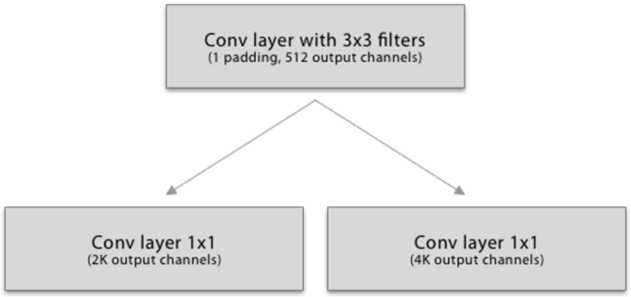

FIGURE 3.8: Convolutional implementation of an RPN architecture, where k is the number of anchors.

It produces as outputs a set of good proposals for objects based on it inputs which are the previous anchors, for each anchor it have 2 outputs :

- The probability of an anchor is an object.
- The bounding box regression.

In classification layer, we have two output predictions per anchor :

- The score of not being an object.
- The score of being an object.

For the adjustment of bounding box layer (Regression layer), we have 4 output predictions which we will apply to the anchors to get the final proposals : $\Delta_{x_{center}}, \Delta_{y_{center}}, \Delta_{width}, \Delta_{height}$.

3.3.3.1 Anchors

Anchors are fixed bounding boxes that are placed throughout the image with different sizes and ratios that are going to be used for reference when first predicting object locations, they are defined based on the convolutional feature map, the final anchors reference the original image. A set of anchors will be created for each of the points in conv width x conv height to end up with a bunch of anchors separated by r pixels. In the case of VGG, r = 16.

FIGURE 3.9: Anchor centers throught the original image [19].

3.3.3.2 Training, target and loss functions

Now, we know that RPN have two type of predictions, which are the binary classification and the bounding box regression adjustment. So for training the RPN, we need to categorize all the anchors into two types :

- Foreground, which are the anchors that overlap a ground-truth object with an Intersection over Union (IoU) bigger than 0.5.
- Background, the anchors that don't overlap any ground truth object or have less than 0.1 IoU with ground-truth objects.

After that, we sample those anchors randomly to form a mini batch of size 256.
To calculate the classification loss, all the selected anchors are needed, the classification loss is calculated using binary cross entropy. For the regression loss needs only the foreground mini batch anchors to be calculated. Now for calculating the targets for the regression, we use the foreground anchor and the closest ground truth object and calculate the correct Δ (Delta) needed to transform the anchor into the object [18].

3.3.3.3 Post processing

- Non-maximum suppression comes to solve the duplicate proposals by taking the list of proposals sorted by score, iterategs over the sorted list, then discarding

proposals with an IoU bigger than predefined threshold with a proposal that has a higher score, the commonly used value is 0.6.

- Proposal selection, After applying NMS, we keep the top N proposals sorted by score. In the paper N = 2000 is used, but it is possible to lower that number to as little as 50 and still get quite good results.

3.3.4 Region of Interest Pooling

The output of RPN is a bunch of bounding boxes with no class assigned to them. So we have classify these object proposal into categories.
Faster R-CNN reuse existing convolutional feature map in this process, by extracting fixed-sized feature maps for each proposal using region of interest pooling, wich is needed for the R-CNN to classify them into a fixed number of classes [20].

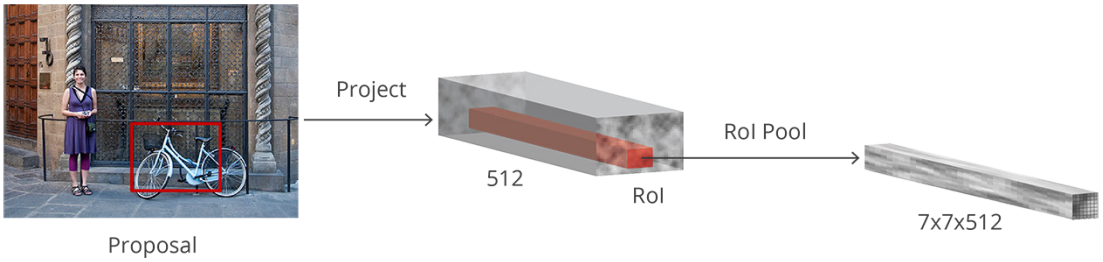

FIGURE 3.10: Region of Interest Pooling [19].

3.3.5 Region-based Convolutional Neural Network

The final step of Faster R-CNN's pipeline is Region-based convolutional neural network (R-CNN)
In this step we need to use the extracted features from the previous layer for classification. R-CNN tries to mimic the final stages of classification CNNs where a fully-connected layer is used to output a score for each possible object class.
R-CNN has two different goals:

Classify proposals into one of the classes, plus a background class (for removing bad proposals).Better adjust the bounding box for the proposal according to the predicted class.

R-CNN takes the feature map for each proposal, flattens it and uses two fully-connected layers of size 4096 with ReLU activation.
Then, R-CNN uses two different fully-connected layers for each different object :

- Classify The fist layer with N+1 units, where N is the total number of classes and 1 is for the background class.
- The second layer with 4N units for each of the N possible classes, which are regression prediction, $\Delta_{center_x}, \Delta_{center_y}, \Delta_{width}, \Delta_{height}$.

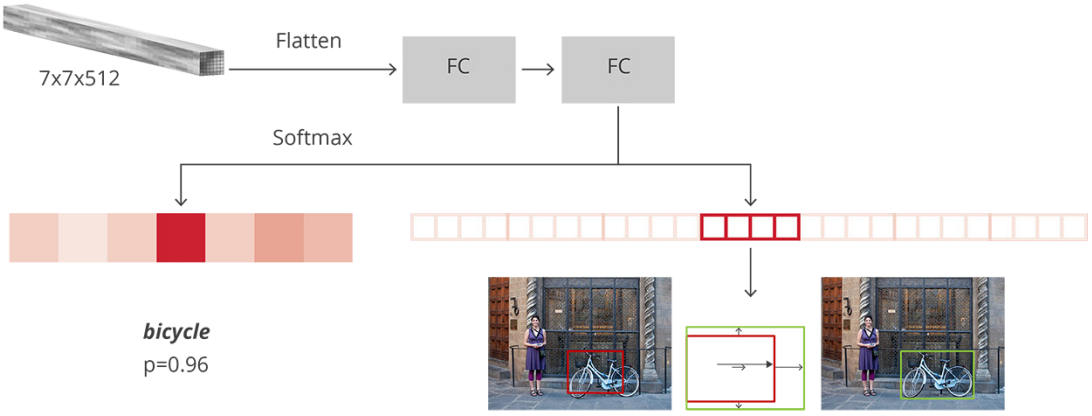

FIGURE 3.11: R-CNN architecture [21].

3.4 Conclusion

In this chapter we described Faster R-CNN, which it proved it capability to solve complex computer vision problems, which make is a canonical model for deep learning-based object detection. Faster R-CNN is state of the art object detection networks depend on region proposal algorithms to hypothesize object locations. In this work, we introduce a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals.

Chapter 4

Implementations

4.1 Introduction

This chapter gives the general design of our application. We gives first the description and the objectives of the project, and than the used materials and softwares. Finally, we present the achieved results and discussion.

4.2 Project description and objectives

Our objective of this project is to implement a vehicle registration plate recognition system, which has accurate results in character recognition for a single image, in real-time, and searching for existed license plate using cameras.

4.3 Concept

The proposed algorithm has been implemented for recognition of registration plate characters after the processing of captured image and plate detection.

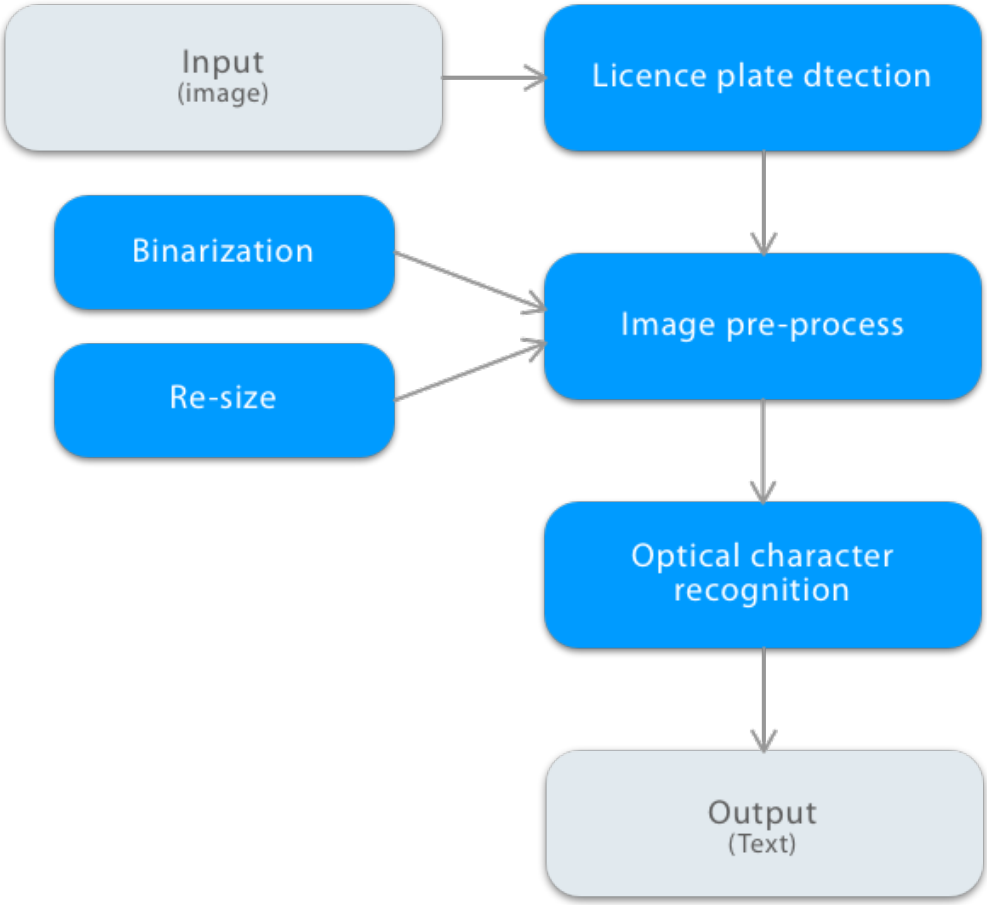

FIGURE 4.1: General design.

4.3.1 Licence plate detection

This step is for locating the number plate in the captured image. Once the number plate is located in the image, we save a new image contains only the number plate. The goal of this step is to make sure that our system will not recognize any characters in the input image, but only those of the licence plate.

As we said earlier, we use Faster R-CNN for licence plate detection. Faster R-CNN takes an image as input and runs it through 2 modules : the first module uses a RPN that proposes object regions and the second is a Fast R-CNN object detector that classifies the region proposals

After getting the output feature map from the pre-trained model which is VGG-16, our input image has 800x600x3 dimensions (Width,Height,RGB), the output feature map would be 50x37x256 dimensions.

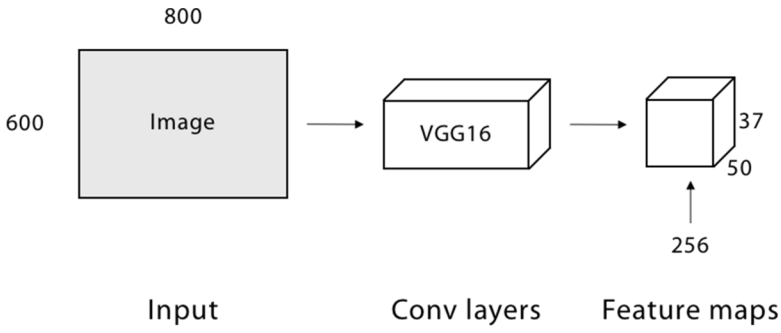

FIGURE 4.2: First step of FasterR-CNN.

Each point in 50x37 is considered as an anchor. We need to define specific ratios and sizes for each anchor (1:1, 1:2, 2:1) for three ratios and 128^2, 256^2, 512^2 for three sizes in the original image.

After that, the RPN is connected to a Conv layer with 3x3 filters, with 1 padding and 512 output channels. The output is connected to two 1x1 convolutional layer for determine if the box is an object or not and for the box-regression

In our case, each anchor has 9 boxes (3x3), in total, we have 16650 boxes (50x37x9) in the original image. We choose only 256 of these boxes as a mini batch, a 128 of them are positive which represents foregrounds and the other 128 are negative which represents backgrounds. To avoid overlapping (duplicate proposals), we applied non-maximum suppression. Now, we are done with RPN.

In the second stage of Faster R-CNN we will use a ROI pooling for the proposed regions. The output is 7x7x512. Then, we flatten this layer with some fully connected layers. The final step is a softmax function for classification and linear regression to fix the boxes location.

Finally we will use the bounding box coordinates (the coordinates of the box that bound the object detected) to crop the input image if an object is detected. The output is in image that contains only a licence plate.

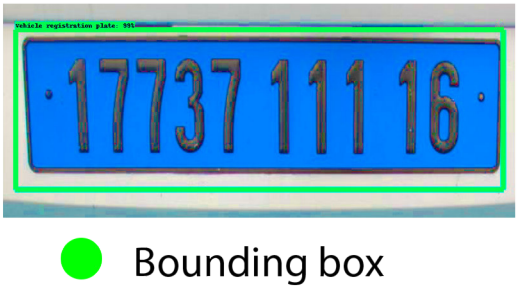

FIGURE 4.3: An example of bounding box

4.3.1.1 Pre-process image

For reason to have more accurate results some image process proposed

4.3.1.1.1 Binarization

High quality binarized image give us more accuracy in character recognition as compared original image because noise is present in the original image, so we used a binirization method to convert a licence plate image from grey scale (0 to 256 gray levels) in to black and white image (0 or 1) (see Figure 4.4).

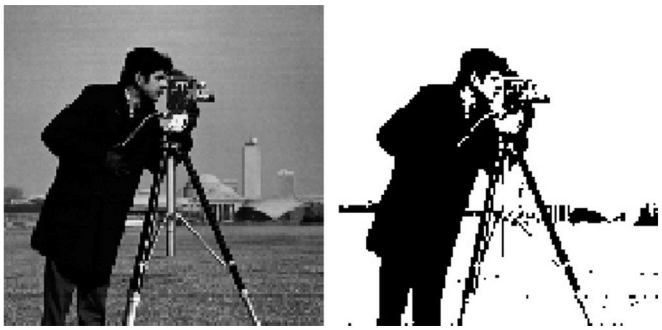

FIGURE 4.4: An example of binarization

The method of binarization we choose is Otsu's thresholding method. Otsu Method used for automatic binarization level decision, based on the shape of the histogram, involving iterating through all the possible threshold values, and calculating a measure of spread for the pixel levels each side of the threshold [22].
Otsu's thresholding method divide in 4 steps :

- In the first step, we separate the pixels into two clusters (q1,q2) according to the threshold.

$$q1(t) = \sum_{i=1}^{t} p(i)$$

$$q2(t) = \sum_{i=t+1}^{I} p(i)$$

where p represents the image histogram.

- In the second step, we will find the mean of each cluster.

$$m1(t) = \sum_{j=1}^{t} \frac{ip(i)}{q1(t)}$$

$$m2(t) = \sum_{j=t+1}^{I} \frac{ip(i)}{q2(t)}$$

- In the third we will calculate the individual class variance.

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - m1(t)]^2 \frac{p(i)}{q1(t)}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{I} [i - m2(t)]^2 \frac{p(i)}{q2(t)}$$

- In the fourth step we will square the difference between the means.

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) = q1(t)[1 - q1(t)][m1(t) - m2(t)]^2$$

This expression can safely be maximized and the solution is t that is maximizing $\sigma_b^2(t)$.

4.3.1.1.2 Re-size

Some licence plate have too many characters, what make space between characters too small, which make the characters recognition process difficult. To solve this problem we add some width to the image that contain the registration plate by changing fx from 1 to 1.5, where fx is the width of the image, in goal to give more space between characters and making characters recognition more accurate (see Figure 4.5).

Original image Re-sized image

FIGURE 4.5: An example of image re-size.

4.3.1.2 Optical character recognition

In this step, we will extract the text form the previous image we worked on using Tesseract OCR (see Figure 4.6).

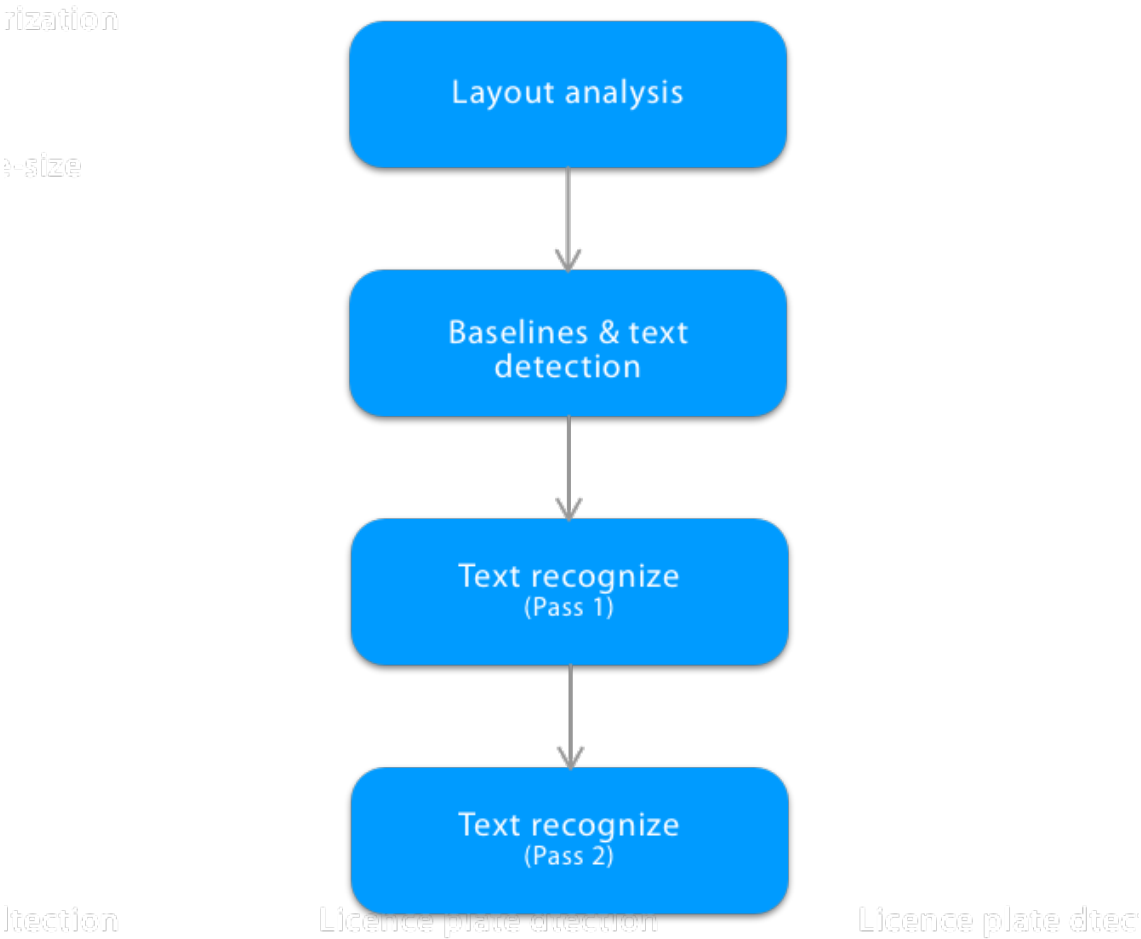

FIGURE 4.6: Architecture of our OCR engine.

4.3.1.2.1 Layout analysis

In this first step, we will divide the input image (the output of the previous step) into two types of areas, areas which contains text, and areas without text, using the following steps [2] :

- Vertical lines detection, using the morphological processing from Leptonica. where these elements are removed from the input image before passing the cleaned image to connected component analysis.
- The candidate tab-stop connected components that look like they may be at the edge of a text region are found and then grouped into tab-stop lines.
- Scanning connected components from left to right and top to bottom.

4.3.1.2.2 Baseline and text detection

Line Finding We used a line finding algorithm from Tesseract. The key parts of this algorithm are blob filtering and line construction. It has the following steps :

- Removing the drop caps vertically touching characters by the application of a percentile height filter.
- The median height approximates the text size in the region, so it is safe to filter out blobs that are smaller than some fraction of the median height, being most likely punctuation, diacritical marks and noise.
- The filtered blobs are more likely to fit a model of non-overlapping, parallel, but sloping lines. Sorting and processing the blobs by x-coordinate makes it possible to assign blobs to a unique text line, while tracking the slope across the page, with greatly reduced danger of assigning to an incorrect text line in the presence of skew. Once the filtered blobs have been assigned to lines, a least median of squares fit [4] is used to estimate the baselines, and the filtered-out blobs are fitted back into the appropriate lines.
- In the last step of the line creation process, the merges blobs that overlap by at least half horizontally, putting diacritical marks together with the correct base and correctly associating parts of some broken characters.

Baseline Fitting : After Finding the lines, we use a quadratic spline to fit the baseline by partitioning the blobs into groups with a reasonably continuous displacement for the original straight baseline (see Figure 4.7).

FIGURE 4.7: An example of baseline detection

Fixed pitch detection and chopping : To determine whether the text lines are fixed pitch, Tesseract test them. Where it finds fixed pitch text, Tesseract chops the text into characters using the pitch, and disables the chopper and associator for the word recognition step (see Figure 4.8).

FIGURE 4.8: An example of fixed pitch

4.3.1.2.3 Classification

The classification proceeds as a two-step process :

- First step involves a class pruner that creates a shortlist of character classes that the unknown might match.
- The classes shortlisted in step one are taken further to the next step, where the actual similarity is calculated from the feature bit vectors. Each prototype character class is represented by a logical sum-of-product expression with each term called a configuration.

4.3.1.2.4 Training

We used a pre-trained data, which the classifier was trained in total on more than 60k samples, using different fonts and attributes (normal, italic, bold and bold italic), for the size we use the same size for all characters.

4.4 The graphical user interface

The application allows the users to choose between 3 operations. Searching for vehicle, streaming road cameras or extracting licence plate from an image.

We provide a simple interface for the user, starting from the home page wich allow user to navigate between the application operations (see Figure 4.9).

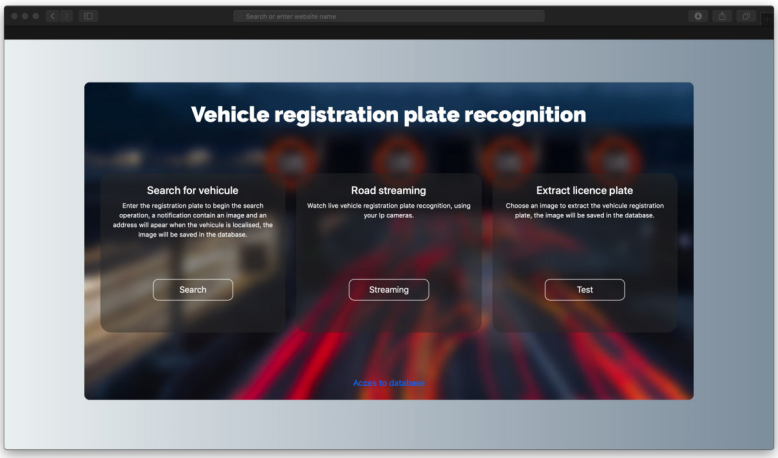

FIGURE 4.9: Home page.

Jumping up to the search for vehicle page, in which the user can type the licence plate number to begin the searching operation (see Figure 4.10).

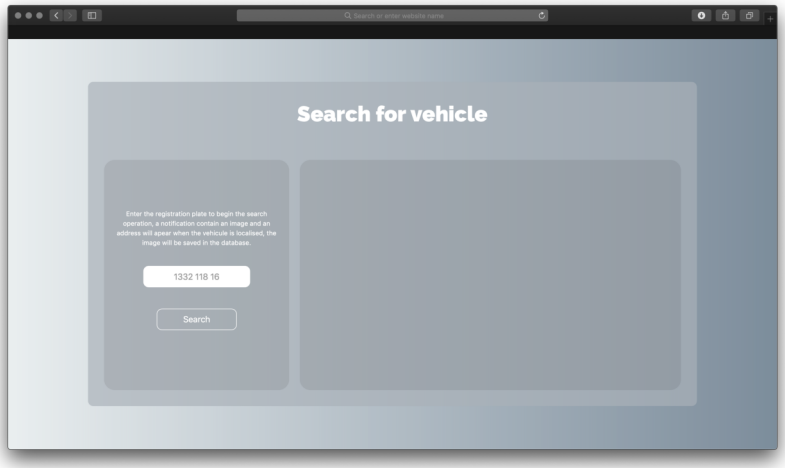

FIGURE 4.10: Search for vehicle page.

The next figure (see Figure 4.11) represent test page, which give the user the possibility of testing the applications on chosen images.

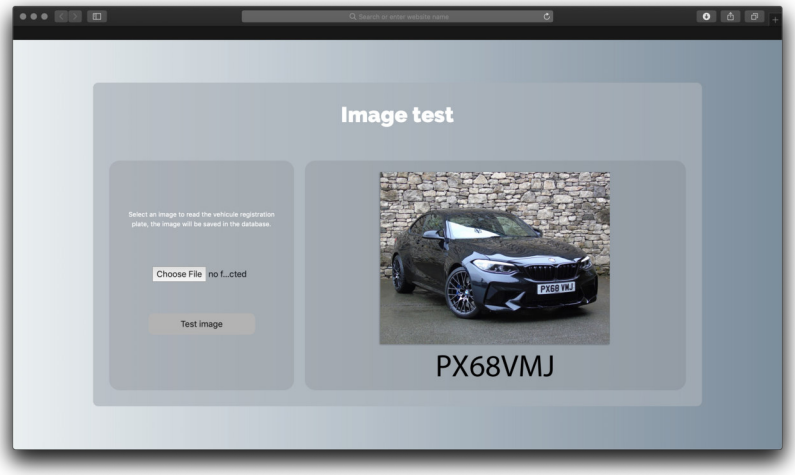

FIGURE 4.11: Extracting licence plate from an image.

The forth page (see Figure 4.12) represent road streaming page, which shows all connected cameras and allow the user to choose a one these camera to start a real time number plate recognition using that camera.

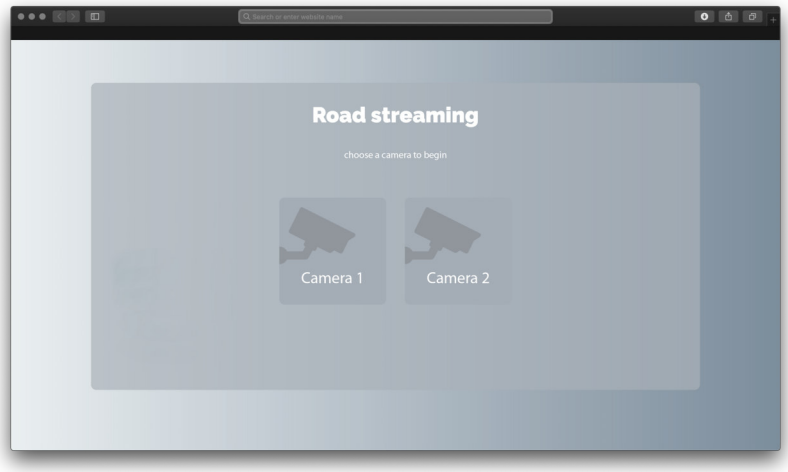

FIGURE 4.12: Road streaming page.

4.4.1 Used Softwares and Materials

4.4.1.1 Materials

MacBook Pro (Retina, 13-inch, mid 2014).
Processor : 2.6 GHz Intel Core i5.
Memory : 8 GB 1600 MHz DDR3.
Graphics : Intel Iris 1536 MB.

4.4.1.2 Softwares

Operating System : macOS Mojave version 10.14.2.
Programming language : Python.
API : Tensorflow, Keras, Tesseract, Leptonica.

4.5 Limitations

- Our application have three types of use. One of them is real time licence plate recognition, which use a real time object detection and optical characters recognition, due to hardware limitation, we will have a low frame per second.
- Licence plate have a lot of types, due to countries or states regulations. Several models have additional text, which makes our application not capable of separating between the licence plate number and the extra text.

4.6 Conclusion

In this chapter, we did an implementation of an Vehicle Registration Plate Recognition application, using the original Faster R-CNN with our custom dataset for the detection of registration plate. For characters recognition we use a pre-trained data from Tesseract. After few tests, we found that the image pre-process (Otsu's thresholding method and the image re-size) make a huge deference in results.

Conclusion

In this project, we have presented the foundation of an Vehicle Registration Plate Recognition application using deep learning approaches in two part, for licence plate detection and for characters recognition. We use also in this project two image process, which are image binarization and image re-sizing, these process helps a lot in characters recognition, what makes our application more accurate. Our proposed solution works in general cases, where the distance from camera to the vehicle is reasonable and weather conditions are good. For the future work, we propose to use a Graphics Processor Unit (GPU) instead of only the Central Processing Unit (CPU), what makes our application run smoothly and give us more frames per second. As well as the image pre-process we can add a de-skew algorithm which helps in case where the number plate are skewed.

References

[1] Prashant Manoharrao Kakde Raamesh Gowri Raghavan and Sukant Khurana. *Applications of OCR You Haven't Thought Of*. Accessed: 2019-04-13. 2018. URL: https://medium.com/swlh/applications-of-ocr-you-havent-thought-of-69a6a559874b.

[2] Arindam Chaudhuri et al. "Optical character recognition systems". In: *Optical Character Recognition Systems for Different Languages with Soft Computing*. Springer, 2017, pp. 9–41.

[3] Salem Saleh Al-Amri, NV Kalyankar, and SD Khamitkar. "Image segmentation by using edge detection". In: *International journal on computer science and engineering* 2.3 (2010), pp. 804–807.

[4] Poonam Dhankhar and Neha Sahu. "A review and research of edge detection techniques for image segmentation". In: *International Journal of Computer Science and Mobile Computing* 2.7 (2013), pp. 86–92.

[5] Jason Yosinski et al. "Understanding neural networks through deep visualization". In: *arXiv preprint arXiv:1506.06579* (2015).

[6] Josh Patterson and Adam Gibson. *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.", 2017.

[7] *Introduction to Neural Networks*. Accessed: 2019-04-02. 2016. URL: https://www.neuraldump.net/2016/03/introduction-to-neural-networks/.

[8] Imad A Basheer and Maha Hajmeer. "Artificial neural networks: fundamentals, computing, design, and application". In: *Journal of microbiological methods* 43.1 (2000), pp. 3–31.

[9] Md Zahangir Alom et al. "A State-of-the-Art Survey on Deep Learning Theory and Architectures". In: *Electronics* 8.3 (2019), p. 292.

[10] Sundaravelpandian Singaravel, Johan Suykens, and Philipp Geyer. "Deep-learning neural-network architectures and methods: Using component-based models in building-design energy prediction". In: *Advanced Engineering Informatics* 38 (2018), pp. 81–90.

[11] Christian Szegedy, Dumitru Erhan, and Alexander Toshkov Toshev. *Object detection using deep neural networks*. US Patent 9,275,308. 2016.

[12] Ross Girshick Jian Sun Shaoqing Ren Kaiming He. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Accessed: 2019-05-30. 2015. URL: https://arxiv.org/abs/1506.01497.

[13] Ross Girshick Ali Farhadi Joseph Redmon Santosh Divvala. *You Only Look Once: Unified, Real-Time Object Detection*. Accessed: 2019-06-05. 2016. URL: https://arxiv.org/abs/1506.02640.

[14] Geoffrey E. Hinton Alex Krizhevsky Ilya Sutskever. "ImageNet Classification with Deep Convolutional Neural Networks". In: 1.1 (2012).

[15] Rikiya Yamashita et al. "Convolutional neural networks: an overview". In: *Insights into imaging* 9.4 (2018), pp. 611–629.

[16] Arden Dertat. *Applied Deep Learning - Part 4: Convolutional Neural Networks*. Accessed: 2019-06-18. 2017. URL: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2.

[17] Jasper RR Uijlings et al. "Selective search for object recognition". In: *International journal of computer vision* 104.2 (2013), pp. 154–171.

[18] Jan Hosang et al. "What makes for effective detection proposals?" In: *IEEE transactions on pattern analysis and machine intelligence* 38.4 (2015), pp. 814–830.

[19] AKA Vierja Javier Rey. *Faster R-CNN: Down the rabbit hole of modern object detection*. Accessed: 2019-06-03. 2018. URL: https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/.

[20] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[21] Yinghan Xu. *Faster R-CNN (object detection) implemented by Keras for custom data from Google's Open Images Dataset V4*. Accessed: 2019-05-28. 2018. URL: https://towardsdatascience.com/faster-r-cnn-object-detection-implemented-by-keras-for-custom-data-from-googles-open-images-125f62b9141a.

[22] BATHINDA Puneet GRDIET. "Binarization Techniques used for Grey Scale Images". In: *International Journal of Computer Applications (0975 – 8887)* 71.1 (2013), pp. 3–11.

Dedication

i must express my very profound gratitude to my mother, my wife, my brothers and sisters, my best friends and specially, my deceased father *for providing me with unfailing support and continuous encouragement throughout my years of study.*
This accomplishment would not have been possible without them.
Thank you. . .