Identifying License Plate Character Segmentation Boundaries Using Convolutional Neural Networks

by

Benoit Bazard

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Examination Committee:	Dr. Matthew Dailey (Chairperson) Dr. Mongkol Ekpanyapong Dr. Vatcharaporn Esichaikul
Nationality:	French
Previous Degree:	Bachelor of Science in Information Technology TELECOM SudParis, France
Scholarship Donor:	TELECOM SudParis, France - AIT

Asian Institute of Technology
School of Engineering and Technology
Thailand
May 2017

Acknowledgments

I am grateful to both the Asian Institute of Technology and TELECOM SudParis for allowing me to work on this paper.

I am thankful to my advisor Dr. Matthew Dailey and the committe members Dr. Mongkol Ekpanyapong and Dr. Vatcharaporn Esichaikul for their support and their assistance.

I am thankful for the resources provided to me by AIT, namely the virtual machine provided by the CSIM department that I used to perform intensive computation, as well as the dataset I train my model on, which came from the Vision and Graphics Lab.

I also want to thank all my friends and my family.

Abstract

The aim of the present paper is to design a way to perform character segmentation on license plates using convolutional neural networks without explicitely performing character recognition so as to keep the segmentation and the recognition distincts.

This paper details the implementation of the algorithm and an evaluation of its global performance on a dataset of 545 license plates provided by the Vision and Graphics Lab at AIT. The system successively applies two convolutional neural networks to perform vertical segmentation and then horizontal segmentation.

The results suggest that convolutional neural networks on their own are not enough to achieve acceptable performance on a dataset of this size. A more sophisticated classifier is necessary to convert the features given by the neural networks into robust character boundaries. The main improvement needed is to eliminate non-character regions identified by the segmenter.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automatic license plate recognition is a task that has received a fair amount of focus in computer vision. The problem is to extract from still images or video sequences the sequence of characters identifying each license plate appearing in the sequence.

1.1 Background

With cameras monitoring roads, it is possible to know after the fact the movement of traffic on those roads. The possible benefits of such a system include better traffic law enforcement and intelligent transportation, which could lead to safer roads. With automatic license plate recognition, traffic characteristics can be identified almost in real time, which leads to enhanced benefits compared to having to manually check videos.

1.2 Problem Statement

The license plate recognition process can be divided into the following sequential steps:

- Picking the frame in which the license plate is the most visible
- Cropping the license plate from the frame
- Segmenting the characters on the license plate
- Recognizing the individual characters
- Putting the characters together

Each step is dependent on the success of each of the steps that come before. In particular, high accuracy on the segmentation part is necessary for the success of the whole process. An improvement for this part will thus improve the overall system. CNNs have not yet been applied to this specific problem, even though they have proven to give good results in other areas such as image classification and character recognition. In this work, I evaluate the ability of a convolutional neural network to perform segmentation of the characters in a license plate.

Figure 1.1: Sample from the dataset provided by the Vision and Graphics Lab at AIT. The first line is the registration number, and the second line is the name of the province.

1.3 Objectives

The final goal is to develop a methodology to achieve high segmentation accuracy on a given dataset comprising some 500 license plates (see Figure 1.1) using a CNN. For this end, three steps are necessary:

1. Design a tailored network for this specific task. I must pick an appropriate sequence of layers and define the layout and other hyperparameters of the neurons in each layer, including the learning rate and the loss function to minimize.
2. Implement this network with Caffe (Jia et al., 2014).
3. Train and evaluate the network.

1.4 Limitations and Scope

Several constraints apply on this project because the work must integrate with a particular workflow.

The dataset input is a set of cropped Thai license plates in JPEG format. The output used for the next step is a YAML file that specifies bounding boxes of the characters in the plate image.

The images in the dataset do not have a fixed size (they vary from 207×97 to 353×171). Illumination is not the same for each image; the number of characters is not constant. Some Thai characters are made of two separate connected components.

1.5 Thesis Outline

I organize the rest of this thesis as follows.

In Chapter 2, I provide a literature review.

In Chapter 3, I propose my methodology.

In Chapter 4, I present the results of my experiments.

Finally, in Chapter 5, I conclude and offer recommendations for further research.

Chapter 2

Literature Review

The segmentation problem in computer vision has been studied deeply. I summarize some of the major techniques that have been used for this aim and provide some insights into new methods suitable for my project.

2.1 Jaccard-Centroid Coefficient

To evaluate the accuracy of a segmentation result, we need a measure of correctness.

A first approach would be to use the Jaccard coefficient, also known as "intersection over union," defined by

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \qquad \text{(Equation 2.1)}$$

where A and B are two rectangles, in this case the ground truth bounding box and the candidate bounding box.

One problem with this measure is that the alignment of the ground truth and candidate is not taken into account. For example, this measure gives the same results for the two examples in Figure 2.1, despite the fact that the more centered example would be better for the recognition part that comes next.

That is why Gonçalves et al. (2016) introduce the Jaccard-Centroid coefficient

$$JC(A, B) = \frac{J(A, B)}{\max(1, C \times \Delta c(A, B))}, \qquad \text{(Equation 2.2)}$$

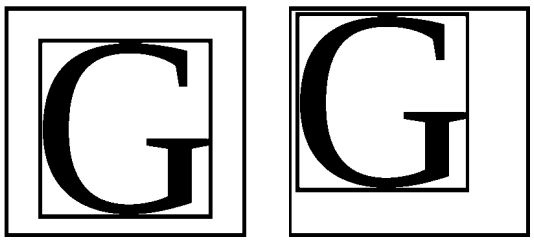

Figure 2.1: Two examples with the same Jaccard coefficient. Reprinted from Gonçalves et al. (2016).

Table 2.1: Samples of measured results of segmentation (Gonçalves et al., 2016).

Approach	Jaccard	Δc	Jaccard-Centroid
Pixel Counting	0.601	2.052	0.316
Conn. Component	0.452	1.896	0.225
Prior Knowledge-Based	0.398	10.820	0.076
Proposed Iterative Approach	0.601	1.433	0.419

where C is a constant fixed empirically at 3 in the paper. $\Delta c(A, B)$ denotes the distance between the centroids of the detected and the desired objects and is defined by

$$\Delta c(A, B) = \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}, \qquad \text{(Equation 2.3)}$$

where (A_x, A_y) and (B_x, B_y) represent the bounding boxes' centroid coordinates, respectively.

2.2 Commonly Used Techniques

With this coefficient Gonçalves et al. (2016) compares four segmentation methods on the same dataset obtaining the results shown in table 2.1.

License plate character segmentation techniques can be classified into five main categories (Gonçalves et al., 2016). More precisely, there exist techniques based on pixel connectivity, pixel projection, prior knowledge of the characters, character contours, and various combinations of these features.

2.3 Projection Analysis

One of the most common techniques for license plate character segmentation is pixel projection, as shown for example in Figure 2.2. The technique consists of computing sums over each row and column, which results in two vectors. Then, applying a simple threshold creates two classes. This technique was used for segmentation of Thai handwritten characters by Lohakan, Airphaiboon, and Sangworasil (1999) and for license plates by Juntanasub and Sureerattanan (2005).

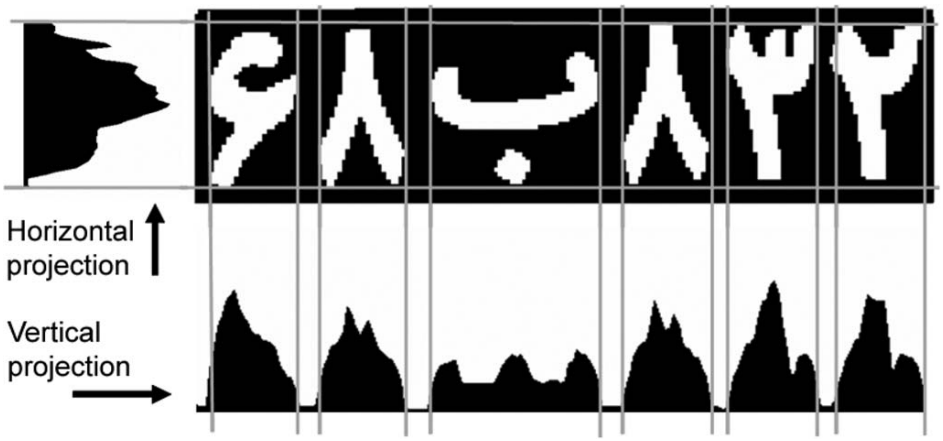

Figure 2.2: Illustration of horizontal and vertical projection. Reprinted from Anagnostopoulos et al. (2008).

2.4 Hidden Markov Models

Franc and Hlaváč (2005) used hidden Markov models for segmentation. The technique consists of comparing the input with a set of admissible segmentations and choosing the one that is more likely via maximum a posteriori estimation.

The images are normalized with nearest-neighbour rescaling so that the number of lines is 14. The number of image columns may vary around 190. There are always exactly seven characters on each license plate.

The parameters of the overall model are determined by training a HMM using a Parzen window density estimator with an isotropic Gaussian kernel.

A segmentation is expressed as a sequence of integer labels y_i assigned according to Equation 2.4, with y_0 assumed to be 0. The labels are the hidden states, and the observations are the column vectors.

An example segmentation is given in Figure 2.3.

The method assumes the number of characters is known and that their width is equal. Therefore, this technique cannot be applied on Thai license plates, as the number of characters and their width are unknown a priori.

$$y_i = \begin{cases} 1 & \text{if} \quad i \in \mathcal{I}, \\ y_{i-1} + 1 & \text{if} \quad y_{i-1} > 0 \quad \text{and} \quad y_{i-1} \leq w, \\ 0 & \text{otherwise.} \end{cases} \qquad \text{(Equation 2.4)}$$

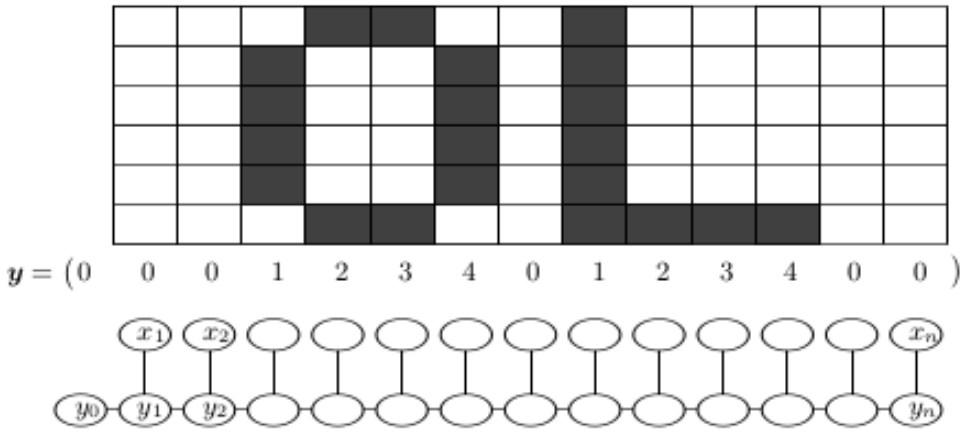

Figure 2.3: Ideal case of a license plate image aligned with a hidden Markov model. Reprinted from Franc and Hlaváč (2005).

2.5 Connected Component Analysis

This technique takes as input a binarized input (usually binarized by local or global thresholding). The two resulting classes are considered foreground pixels and background pixels. The goal is to uniquely label each connected component of the foreground pixels. Each foreground pixel is assigned a label so that connected foreground pixels share the same label. Each of the connected component is called a blob.

2.6 Without Character Segmentation

A different approach to segmentation when combined with detection is to generate regions from the image and use them as candidates for segmented subimages. That is what, for example Girshick et al. (2014), did with region CNNs (rCNNs). From each candidate region, a CNN extracts a feature vector. These candidate feature vectors are then classified with a SVM. An illustration of the rCNN is given in Figure 2.4.

This approach, even if appropriate for object detection, does not create precise enough bounding boxes to be useful in segmenting a license plate.

In a similar approach, Sermanet et al. (2013) generate candidates at all scales and all locations before classification. Kahraman, Kurt, and Gökmen (2003) use Gabor transform and connected component analysis to generate candidate plate regions. Finally, H. Li and Shen (2016) avoid character segmentation altogether by using long short term memory (LSTM).

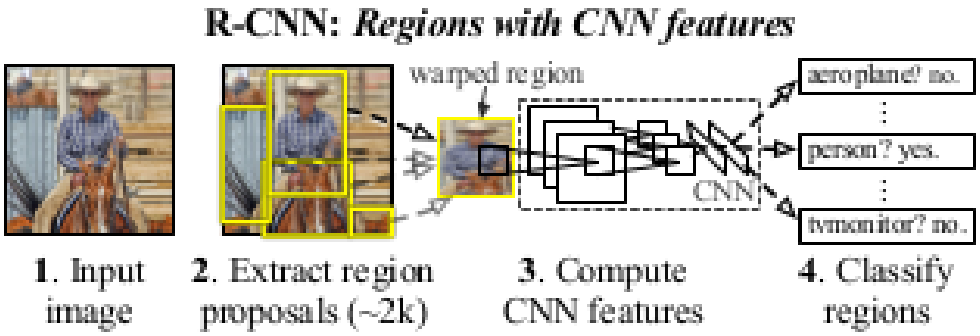

Figure 2.4: r-CNN for character segmentation. Reprinted from Girshick et al. (2014).

2.7 Convolutional Neural Networks

A convolutional neural network is a neural network that performs some kind of 2D convolution at the lowest layer, with additional layers added one after another. A layer is only connected to the layer right before and the layer right after.

Each layer filters the output of the previous layer to obtain a feature map. These feature maps are then used as input for the next layer.

There are four basic layers commonly used:

- Convolutional Layer

  That is the type of layer that distinguishes convolutional neural networks from other types of neural networks (Figure 2.5): each output neuron is connected to a set of adjacent input neurons. The set of inputs is called the output neuron's receptive field. Adjacent outputs are connected to receptive fields that are slightly shifted in the input space. Receptive fields of adjacent neurons usually overlap. The filter is a matrix of weights (and a bias). Calculating the feature map means multiplying this matrix with the input for each receptive field (and offsetting the sum by the bias). This operation is therefore linear in the input units' activation. The filter used for each of the adjacent receptive fields is the same, and the adjacent neurons' weights are the same. So the feature map is a convolution of the filter and the input with a possible non-unit stride. Hence the name convolutional layer.

- Pooling Layer (subsampling)

  Since the receptive fields overlap, the number of neurons can increase with each convolutional layer. This growth implies more computing power requirements, and more difficulties training the model. It has been found that the accuracy of the network can be sustained while decreasing the number of neurons by subsampling. It is also a way to reduce overfitting. The most common way to subsample is max pooling, namely to pick the maximum value of each of the non-overlapping receptive fields.

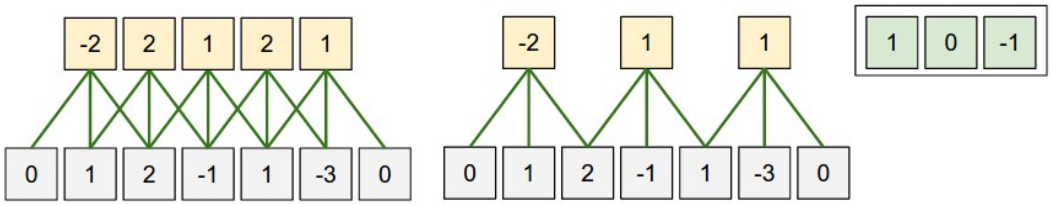

Figure 2.5: Two small convolutional layers. Reprinted from (F.-F. Li et al., 2016), the feature map is in top right, the upper layer is the output, there is overlapping of the receptive fields in the architecture on the left

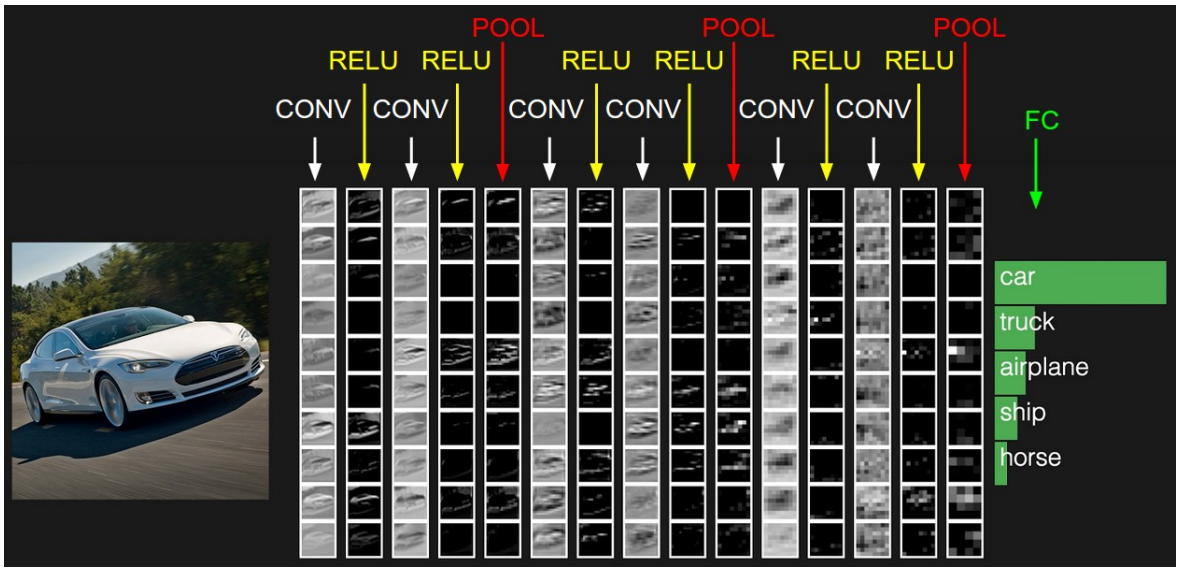

Figure 2.6: Example of a classic CNN architecture with the four basic layers and some of their feature maps. Reprinted from (F.-F. Li et al., 2016).

- Non Linearity (ReLu)

  This layer just maps negative values to 0. It introduces a non linearity in the model and can increase the accuracy of the model without increasing the computing power required much.

- Fully-Connected Layer

  This layer is the same as typical layers in general neural networks. All neurons in one layer are connected to all the neurons in the previous layer. Fully-connected layers are usually at the end of a convolutional neural network. The last of these layer often uses a softmax function to normalize the outputs, which gives a neuron for each class with value between 0 and 1 indicating the certainty of the input belonging to that class.

2.8 Summary

From this literature review, we can see that various techniques have been used to tackle the problem of character segmentation for license plates.

Nevertheless, to my knowledge, there has not been an attempt to use convolutional neural networks to tackle the specific problem of character segmentation for license plates. Neural networks have been used in a more holistic manner by combining segmentation and recognition.

The methodoly proposed next tries to perform segmentation alone using CNNs.

Chapter 3

Methodology

The details of how I developed the proposed model are explained here.

3.1 Global Overview

As a reminder, a global system overview can be seen in Figure 3.1. My methodology concerns only the character extraction part. How the model works for a license plate is illustrated in Figure 3.2.

3.2 Caffe

I use Caffe to design the neural networks for segmentation. Caffe (Jia et al., 2014) is a deep learning framework useful for designing, training, and evaluating deep learning models.

3.3 Basic Preprocessing

This project needs as input a license plate dataset and labels of the ground truth character segmentation in order to train the two neural networks. The dataset and the labels needs to be converted to a format usable by Caffe, I use LMDB (Lightning Memory-Mapped Database Manager).

3.4 The Architecture

The architecture consists of two distinct convolutional neural networks (see Figure 3.2), one for the text segmentation and one for the character segmentation.

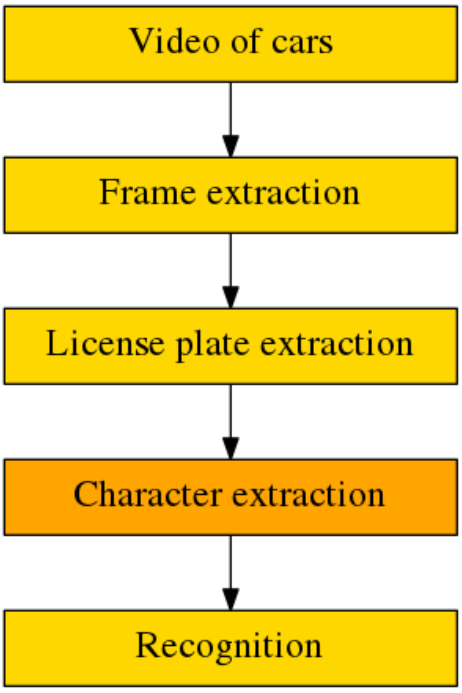

Figure 3.1: Global overview of the system. My focus is the character extraction step.

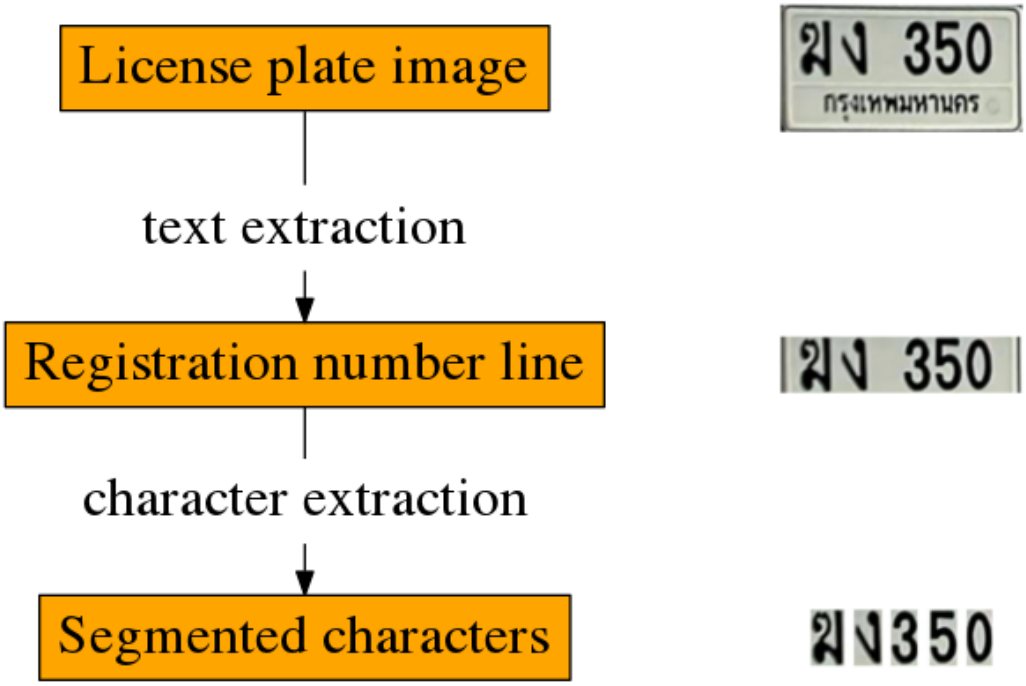

Figure 3.2: Data flow of the segmentation process.

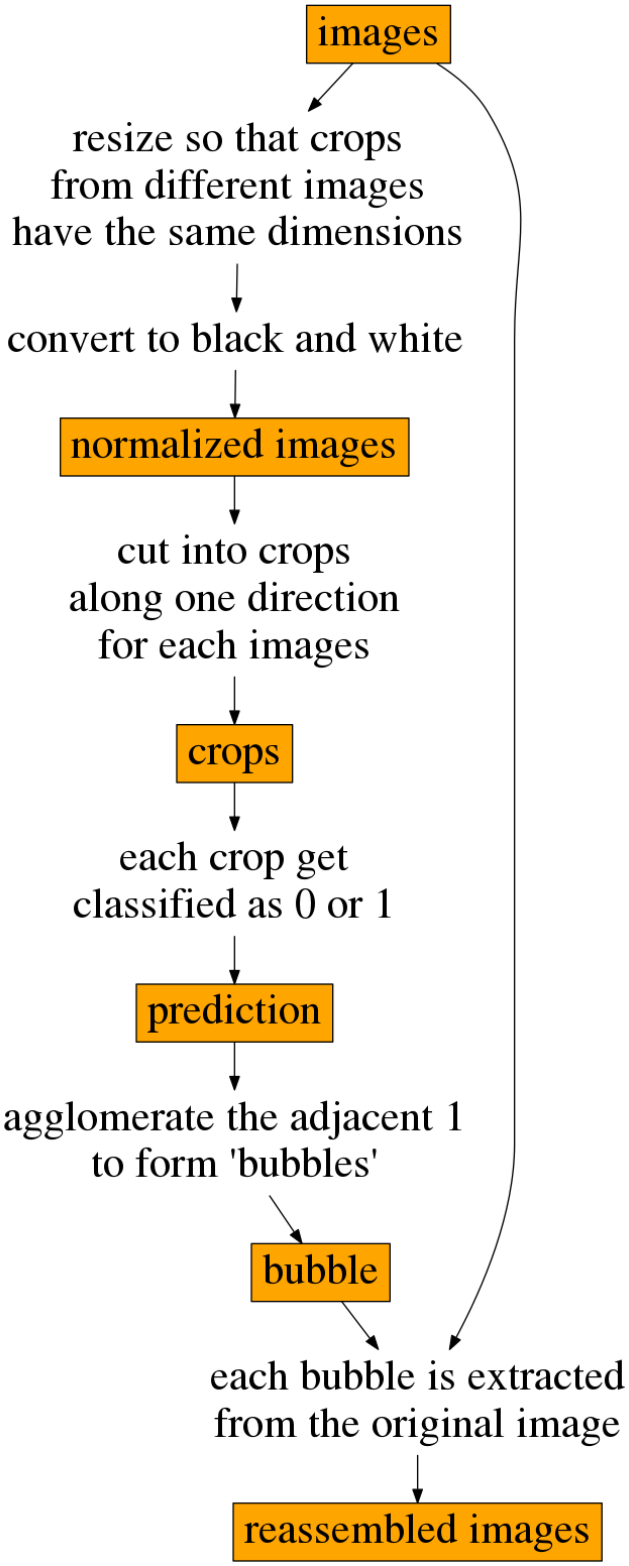

Figure 3.3: Shuttering and curtaining follow the same general process.

3.4.1 The fundamental process

These two parts are very similar and illustrated in Figure 3.3. Each part works on one dimension and only uses the labels from that dimension (the shuttering part only cares about the y coordinates of the labels, and the curtaining part only uses the x coordinates of the labels).

3.4.1.1 Normalization

From the raw dataset, each image is scaled on one dimension so that the crops extracted afterward have the same dimensions. The images are scaled with nearest-neighbor interpolation. The images are then converted from three-channel RGB to one-channel greyscale using the ITU-R 601-2 luma transform.

$$L = R * 299/1000 + G * 587/1000 + B * 114/1000$$

3.4.1.2 Cropping

The normalized images are then cut into n overlapping crops (n is the dimension that was not scaled, so scaling does not change the number of crops). Each adjacent crop is shifted by one pixel in one direction (stride of one). The crops span all pixels in the other direction.

It is actually the central row/column that is classified. The other pixels are only context to help the neural network. The context being symmetric, the number of pixels in the direction that is not scaled is odd.

To also have a context in the border of the images, copy padding is used. The border row/column is replicated as many times as necessary (see Figure 3.6 and Figure 3.7).

3.4.1.3 Prediction

Each crop is then classified by a neural network. The neural network should output 1 if the crop is part of the text/character or 0 otherwise.

In more detail, the output softmax layer provides a vector of numbers between 0 and 1 (which can be considered as the probability that the central row/column is part of text/character). The Caffe models of both networks are illustrated in Figure 3.11 and Figure 3.10. There is then a threshold that converts all the numbers to 0s and 1s if they are bigger or smaller than the threshold of 0.5 (see Figure 3.9).

3.4.1.4 Bubbling

At this stage, for each original image, we have a list of 0s and 1s that corresponds to a segmentation in one direction. Bubbling consist of grouping adjacent 1s into "bubbles."

More concretely, for each original image, a list of couples of crop numbers is created. Each couple of crop numbers delimits a line of text for the shuttering part (and then is a couple of y coordinates) or a single character for the curtaining part (and then is a couple of x coordinates). Each crop number is inclusive; if the bubble is one pixel large, the two crop numbers are the same.

3.4.1.5 Reassemblage

Each bubble is then extracted from the original image as a separate image.

3.5 The training process

In order to train a neural network, the dataset must be split into a training set, a validation set, and a test set. The two neural networks share these sets.

The text CNN is trained on labels corresponding to y coordinates directly on the original dataset (the training set).

The character CNN is trained on "perfect shutters." The labels are used to perfectly extract text and create perfect shutter images. The character CNN is then trained with only the labels corresponding to x coordinates (the training set).

When the two networks have been trained, the images from the dataset undergo text extraction and then character extraction on the validation set without taking the labels into account.

This process is illustrated in Figure 3.8.

As a result, from an image of a license plate, we get the images of each of the characters separately.

3.6 Assumptions

The system is somewhat biased. The dataset must meet some constraints for the system to be useful. The boundaries are always either completely horizontal or completely vertical. The bounding boxes do not overlap.

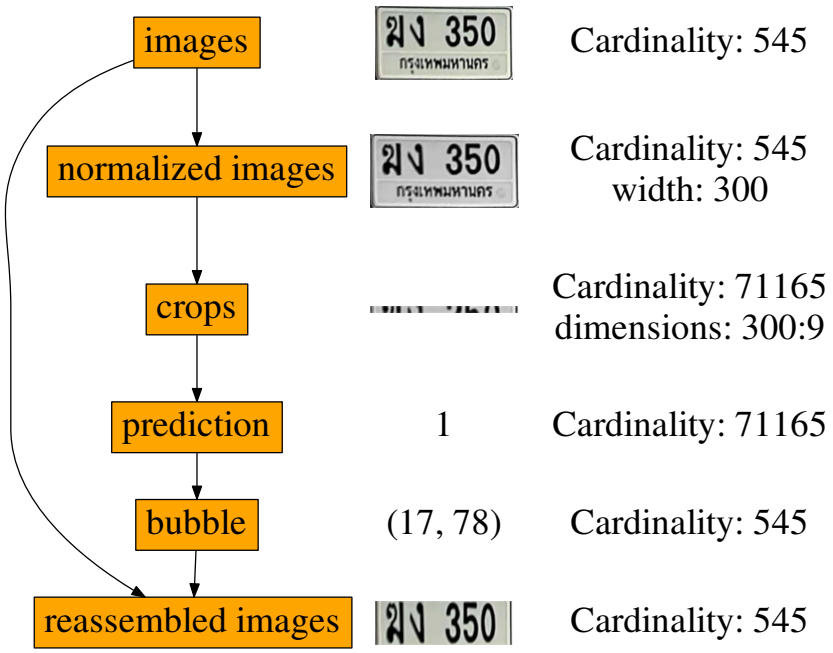

Figure 3.4: Text extraction process.

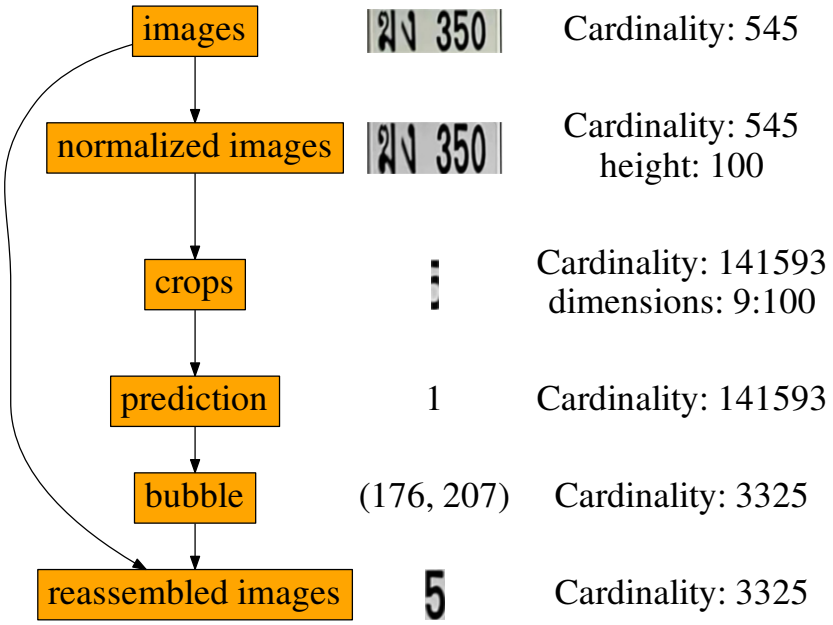

Figure 3.5: Character extraction process.

Figure 3.6: A shutter: the result of splitting of a license plate before text segmentation. In practice, there are many more smaller crops.

Figure 3.7: A curtain: the result of splitting the registration number line before character segmentation. In practice, there are many more smaller crops.

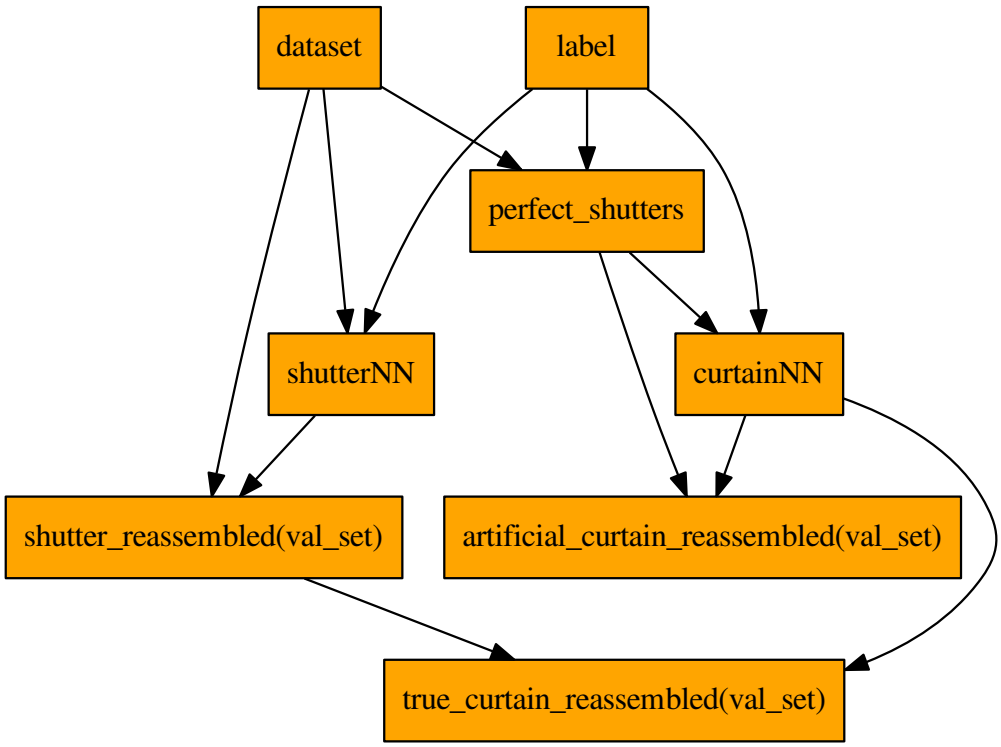

Figure 3.8: Data flow of the learning process.

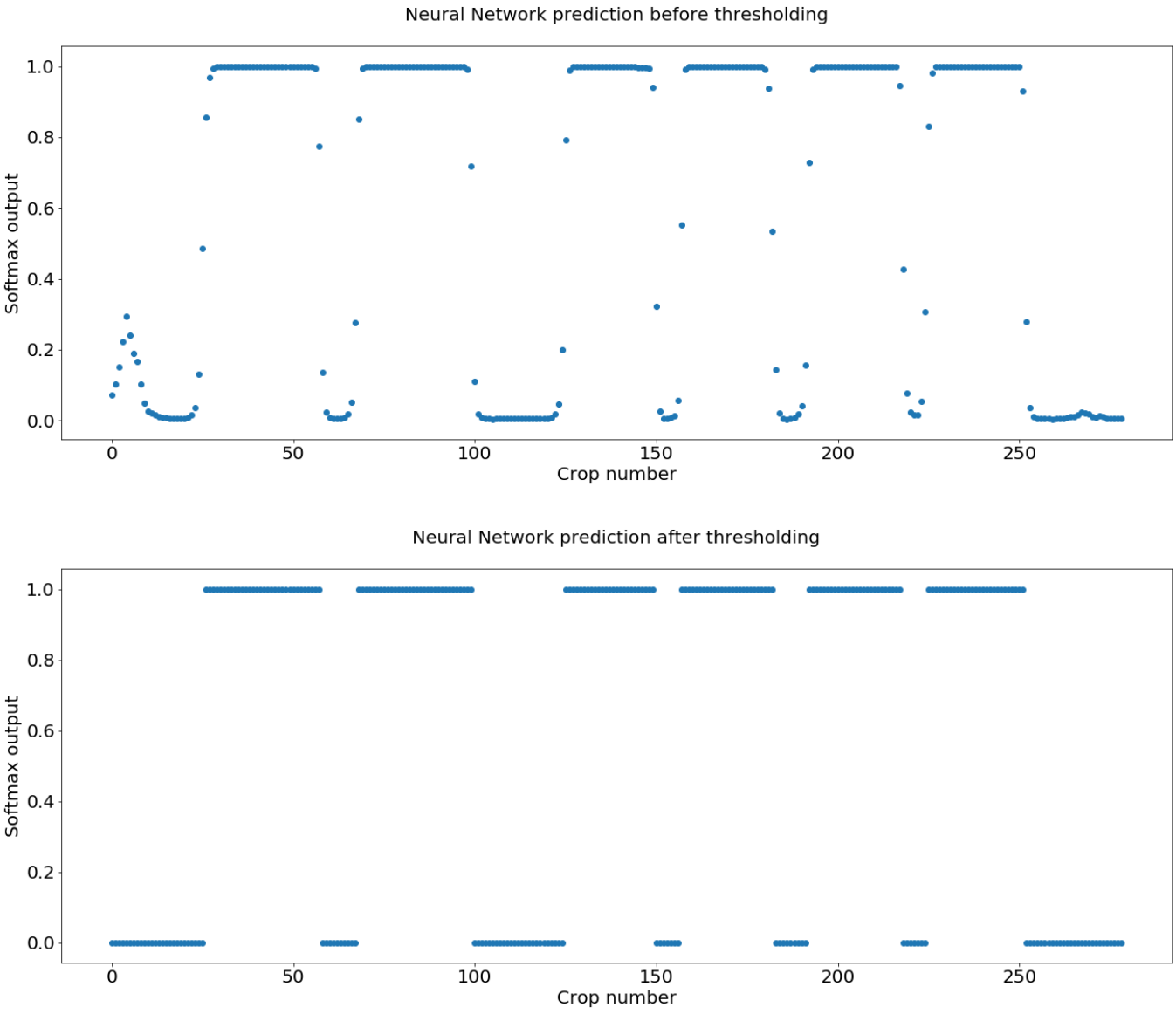

Figure 3.9: The thresholding process.

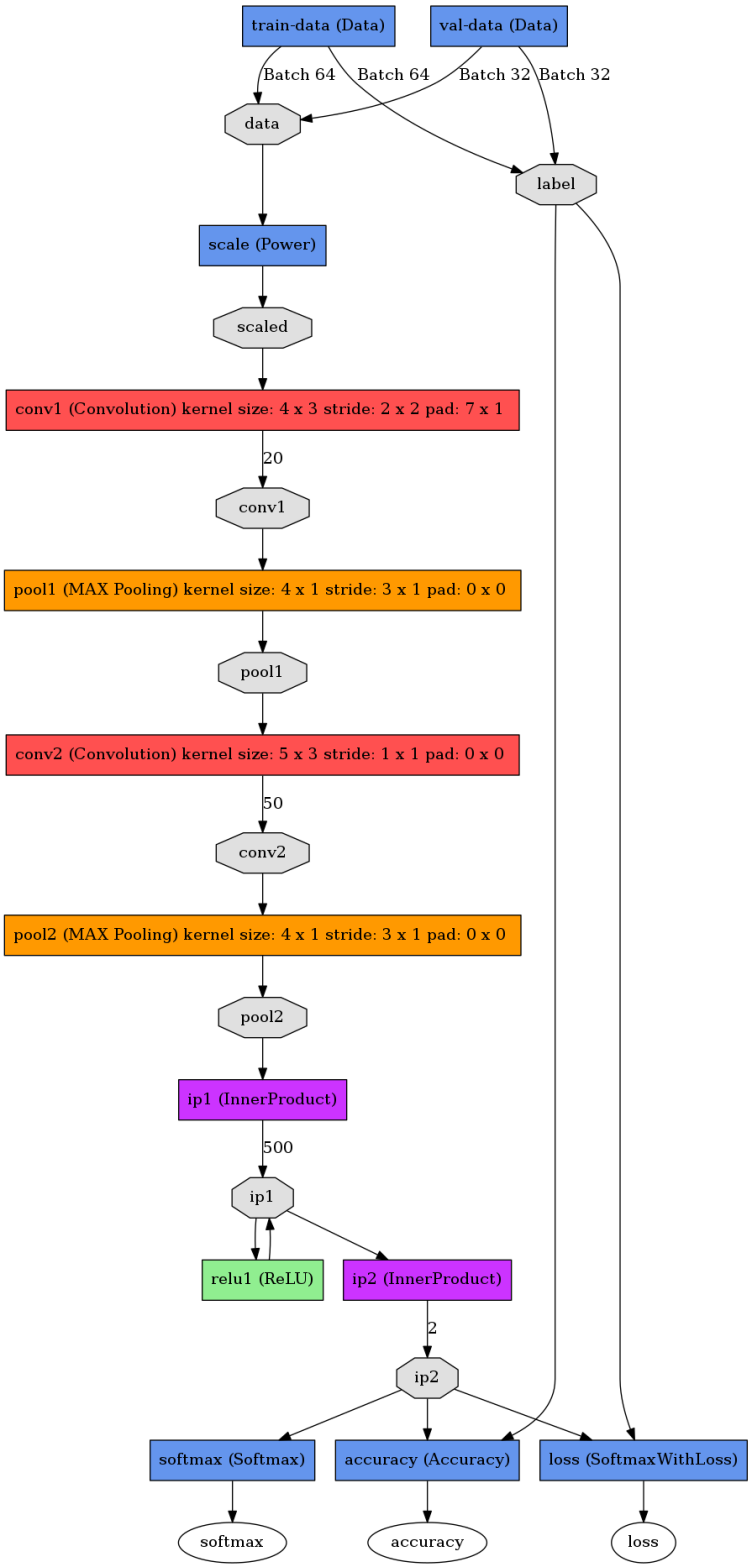

Figure 3.10: Caffe model for the extraction of the main band of characters (the text CNN or shutter CNN).

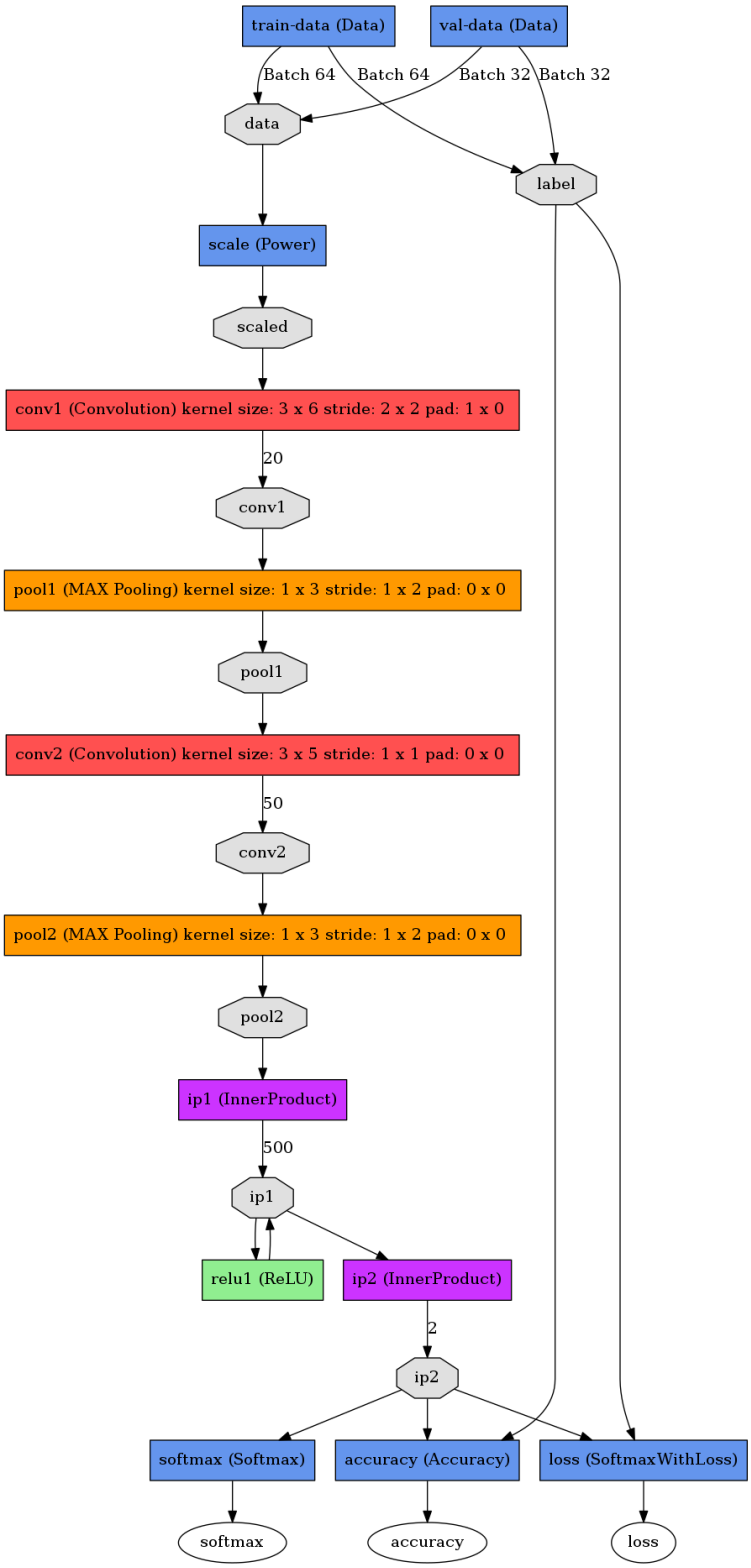

Figure 3.11: Caffe model for the character segmentation of the main band of characters (the characters CNN or curtain CNN).

Chapter 4

Results

Here are some results from this project.

4.1 Results Assessement

To know how well the whole system is performing, we compare the results on the validation set with the labels.

The measure used is the Jaccard coefficient (the pixelwise intersection over union of the bounding boxes).

For each character, from left to right, we calculate the Jaccard coefficient between the bounding box given by the labels and the bounding box given by the system (the bubbles). We then define a Jaccard threshold: a character is considered correctly recognized if its Jaccard is superior to the Jaccard threshold. The leftmost character is compared with the leftmost bubble. The second leftmost is compared with the second leftmost bubble. The rightmost character is compared with the n^{th} leftmost bubble if it exists, with n being the number of actual characters. The remaining bubbles are ignored.

The same way, we can assess the performance of the shuttering part alone, the performance of the curtaining on perfect shutters, and the performance of curtaining on the shutters given by the shuttering part.

By variating the Jaccard threshold, we can see the performance of the four processes in Figure 4.2.

4.2 Jaccard and Jaccard-Centroid Coefficient

In practice, the method used in this paper does not suffer from the problem the Jaccard-centroid coefficient is trying to solve.

Indeed, the neural networks produce bubbles that are more often too small rather than too large.

In effect, the worst case illustrated in the literature review in Figure 2.1 does not happen.

The drawback of the Jaccard centroid is the need to fix a constant empirically. That is the reason I use the regular Jaccard coefficient instead.

(a) (b)

Figure 4.1: Sample results. The red regions are false negatives, the cyan regions are false positives. (a) Jaccard coefficient of 83. (b) Jaccard coefficient of 90.

To have a better idea of what the Jaccard coefficient corresponds in terms of union and intersection, examples are provided in Figure 4.1.

4.3 Problems encountered

Because the neural networks do not reach perfect accuracy, there are some false positives and false negatives. Their effects on the accuracy of the global system vary according to their relative position in the images.

4.3.1 Main band of text cut in two

The problem with the global performance of the system for shuttering is when a bubble is cut into two because of false negatives. Some crops in the middle are sometimes not recognized, which create two bubbles instead of one. The next step is to take the biggest bubble; the other bubble is ignored, and only part of the main band of text is extracted. For the curtaining part, this band of text is scaled, so the input for the curtaining part is distorted, as shown is Figure 4.3.

4.3.2 Number of bubbles different from the number of actual characters

This problem occured when the number of detected bubbles is different from the actual number of characters. The bubbles are shifted (each character is recognized as the character before).

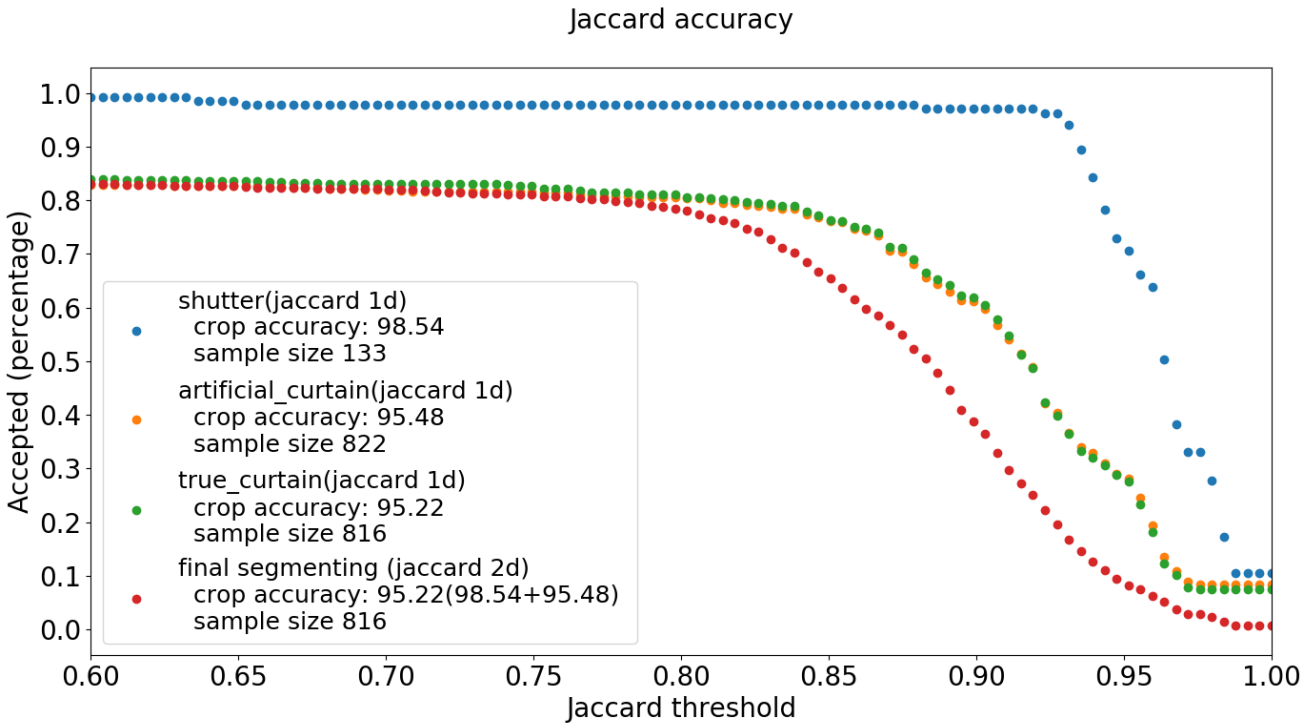

Figure 4.2: Assessement of the performance of the system as a function of Jaccard threshold.

Figure 4.3: Result after reassemblage and normalization when some crops are misclassified in the middle of the text.

Figure 4.4: In this example all the characters are considered incorrect.

For example, in Figure 4.3, the first actual character is compared with the false positives in cyan, which results in a Jaccard coefficient of 0. Then the second character is compared with the second bubble, which here corresponds to the first character resulting in a Jaccard coefficient of 0 as well. The sixth and last actual character (8) is compared with the sixth bubble (7) giving a jaccard of 0. Finally, the last bubble (8) does not have any real character to compare with.

This problem also occured when an actual character is cut into two segments. In this case, the characters on the left are not affected.

Chapter 5

Conclusion and Recommendations

Here I provide a conclusion of the project.

5.1 Possible improvements

From this work we can see that neural networks give only imperfect features. Another classifier such as a hidden Markov model would be necessary to balance the imperfections from the network.

One way to improve the accuracy would be to train the system on a larger dataset of license plates. This work has used a relatively small dataset of 545 license plates.

It must be said that this work only performs segmentation and is completely independent from the recognition part. In a real-world situation, that means more flexibility to choose the recognition system. However, the accuracy of the segmentation component can surely be increased by blending it with the recognition system.

References

Anagnostopoulos, C.-N., Anagnostopoulos, I., Psoroulas, I., Loumos, V., & Kayafas, E. (2008). License Plate Recognition From Still Images and Video Sequences: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, *9*(3), 377–391.

Franc, V., & Hlaváč, V. (2005). License plate character segmentation using hidden Markov chains. In *Joint Pattern Recognition Symposium* (pp. 385–392). Springer.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 580–587).

Gonçalves, G. R., Silva, S. P. G. da, Menotti, D., & Schwartz, W. R. (2016). Benchmark for license plate character segmentation. *Journal of Electronic Imaging*, *25*(5), 053034–053034.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., et al. (2014). Caffe: Convolutional architecture for fast feature embedding. In *ACM international conference on multimedia* (pp. 675–678).

Juntanasub, R., & Sureerattanan, N. (2005). Car license plate recognition through Hausdorff distance technique. In *IEEE international conference on tools with artificial intelligence (ICTAI)* (p. 5 pp.).

Kahraman, F., Kurt, B., & Gökmen, M. (2003). License plate character segmentation based on the Gabor transform and vector quantization. In *International Symposium on Computer and Information Sciences* (pp. 381–388). Springer.

Li, F.-F., Karpathy, A., & Johnson, J. (2016). *CS231n: Convolutional neural networks for visual recognition.*

Li, H., & Shen, C. (2016). Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs. *arXiv preprint arXiv:1601.05610.*

Lohakan, M., Airphaiboon, S., & Sangworasil, M. (1999). Single-character segmentation for handprinted Thai word. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition. International Conference on Document Analysis and Recognition (ICDAR)* (pp. 661–664).

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229.*