Improving Automatic Number Plate Recognition

Mohammad Javad Ahmadi Amin

Stockholm University

Bachelor's Thesis in Computer Science
at Stockholm University, Sweden, 2021

Improving Automatic Number Plate Recognition

Mohammad Javad Ahmadi Amin

Bachelor's Thesis in Computer Science (15 ECTS credits)
Bachelor's Programme in Computer Science
Stockholm University year 2021
Supervisor at the Department of Mathematics was Woosok Moon
Examiner was Lars Arvestad

Department of Mathematics
Stockholm University
SE-106 91 Stockholm, Sweden

Abstract

Automatically recognizing the license plates of vehicles is done for various purposes. In many studies, the focus is on clear images whereas this study aims at improving the accuracy in recognition of the license plate when facing difficulties such as blurry image, redundant objects / pixels on the license plate, and / or inclination of the license plate.

Also, finding the best machine learning model for classifying the characters of the license plate.

In this study, character recognition of vehicle license plates was done by a few preprocessing steps and then utilizing the capability of machine learning. In preprocessing and character segmentation, to detect the characters of the license plate correctly, the connected objects were detected from binary images. Line fitting, Multi-Otsu thresholding, Sobel filter, K-means clustering, and other techniques were used to improve the localization and recognition of license plates.

Then for machine learning, Support-vector machine, Artificial neural network (ANN), and Convolutional neural network (CNN) methods were implemented. The models performed well with CNN resulting in the highest accuracy. This experiment concludes that indeed it is possible to improve the accuracy of ANPR in the detection and classification of the license plates.

Sammanfattning

Förbättring av Automatic number plate recognition (ANPR).

Att kunna läsa av registreringsskyltar på ett automatiskt sätt har använts i olika ändamål. I många studier vars arbeten fokuserar klara bilder, så skiljer sig denna studie på att förbättra noggrannheten i att läsa av registreringsskyltar trots olika svårigheter, såsom suddiga bilder, överflödiga objekt/pixlar på registreringsskylten, och/eller infallsvinkel på registreringsskylten.

Även användandet av maskininlärning för att kunna hitta den bästa modellen för att klassificera de tecknen som visas på registreringsskylten.

Innan användandet av maskininlärning så gjordes ett par förbehandlandingssteg på bilder. Vid förbehandlandlingen och segmentering av tecken så används binära bilder för att hitta anslutna objekt. Linjeanpassning, Multi-Otsu-tröskel, Sobel-filter, K-medelkluster samt andra metoder användes för att förbättra lokaliseringen och igenkänning av registreringsskyltar.

Efter att data förberetts så implementerades olika maskininlärning metoder, såsom Support-vector machine, Artificial neural network (ANN) och Convolutional neural network (CNN). CNN fick högsta noggrannhet bland alla implementerade metoder. Slutsatsen ger att det går att förbättra noggrannheten för ANPR vid identifiering och klassificering av registreringsskyltar.

Contents

1 Introduction

1.1 Background

Automatic number plate recognition, also known as ANPR is a technique using optical character recognition to read and store the vehicle's registration number (Du et al. 2013). The technology was invented in 1976 by the British police ("History of ANPR - ANPR International" n.d.).

ANPR is used by law enforcement in many countries for traffic control and surveillance. The technology is also used by governmental and private entities for collecting payments at toll stations, unattended parking lots, etc. (Wikipedia contributors 2021a).

1.2 Problem Description and Previous Work

There are several challenges in license plate recognition, such as license plates with dirt stains, blurred images, objects such as stickers that could be wrongly recognized as a character, and images taken with a camera not positioned directly in front of the vehicle resulting in a skewed license plate (Du et al. 2013). Damilare et al. (2015) have studied Nigerian license plates and have implemented machine learning to build a model to recognize the license plates of Nigerian

vehicles. The machine learning model was tested on images with license plates that were hard to read, and the outcome of these tests was very promising. The model was tested on vehicle images taken from the front therefore the inclination of the image was not considered.

For license plate localization Damilare et al. (2015), used canny edge detection, a multi-step algorithm for finding the edges in an image (Canny 1987), and then connected object detection by using pixel connectivity (Samet and Tamminen 1988). They have tested their algorithm for extraction and recognition of license plates in three categories.

1. Images with clear and easy to read license plates with distinguishable characters gave a success rate of 98 % in the extraction of 40 license plates as testing data and 96.25 % in recognition of 320 characters.
2. Blurry images due to variation in illumination condition gave a success rate of 95 % in the extraction of 20 license plates and 91 % in recognition of 160 characters.
3. Images that include dirt stains and/or shadowed characters had a success rate of 80 % in the extraction of 10 license plates and 91.25 % in recognition of 80 characters.

It is evident as the image quality worsens the success rate of extraction and recognition suffers a substantial decrease.

Oladeji (2016) has another similar work on Nigerian license plates, that uses binary images of license plates for capturing the connected objects. Neighboring white pixels in the binary images are considered as a connected object. Then the characters were segmented and resized in 20×20 pixels. The total flatten 400×1 pixels were used for training machine learning models. The accuracy for this model was not reported and this model was only implemented on the horizontal car images.

1.3 Aim

The methods used by Damilare et al. (2015) and Oladeji (2016) were studied and extended to be able to recognize the license plate characters in instances including images with inclination. This study aims to improve the methods used in the extraction and recognition of the characters in the mentioned works.

This study also implements and compares three learning models to find the optimal classification method. The learning models used are Support-vector machine, Artificial neural network, and Convolutional neural network, which are described in the next chapter.

1.4 Social Aspects

There are ethical concerns about the vast-ranging spread of the ANPR devices in countries such as the United States, where people feel a threat to their privacy as the governments

can track them for any reason. Since the data collected by the ANPR is usually stored in permanent databases and there is a lack of regulations about how this data can be used and/or shared (Crump 2013).

1.5 Delimitation

Due to limitations in time and resources, this study is restricted to Swedish license plates and the three supervised learning algorithms mentioned in the previous section.

1.6 Structure

This report contains 4 chapters where chapter 2 is divided into 3 main sections starting with preprocessing and license plate detection, and character segmentation. In the last section, the basic notions of the Support-vector machine, Artificial neural network, and Convolutional neural network and classification models, as well as their implementations, are explained. Chapter 3 presents the results of the study and in the last chapter, the obtained results and suggestions for future work are discussed.

2 License Plate Detection

This chapter describes in detail, the preprocessing models used for license plate detection and character segmentation.

2.1 Preprocessing and License Plate Detection

Two hundred vehicle images were downloaded/scraped from Blocket, a Swedish marketing website for second-hand as well as brand new vehicles. The images were taken from various angles of the exteriors and interiors therefore not all the images include a license plate.

The image quality differs as there are many private sellers with different camera qualities on the website. The algorithm will read all the images in the database one by one and select only the images where license plates were detected. Preprocessing, license plate detection, and character segmentation was done to make the images ready for classification. The process is explained in detail in the ongoing section.

The acquired database consists of RGB images, which include three channels of Red, Green, and Blue. First, to perform the analysis of license plate detection, the RGB images will be converted to grayscale that has one intensity channel. There are several methods for converting RGB images into grayscale. One of the popular methods for this

conversion is called the weighted average (Stokes et al. 1996) with the formula

$$Y = 0.2126\,R + 0.7154\,G + 0.0721\,B.$$

Where R, G, and B are the intensity of the image for the red, blue, and green channels of the RGB image.

The coefficients used in this averaging formula is based on how the human eye works. For example, since the human eye is more sensitive to green, we can see a greater coefficient for its respective channel, and the coefficient for the blue channel is the lowest as the human eye is least sensitive to the blue color (Stokes et al. 1996).

Next, a simple segmentation algorithm called Otsu's thresholding method is used for converting the grayscale image into a binary image, containing two classes where 0 indicates black/background pixels and 1 white/foreground pixels (Otsu 1979). Using the discrete histogram of the grayscale image with 256 intensity levels or bins, the threshold $0 \leq T \leq 255$segments the image into two classes, C_0or background, and C_1or foreground. This means all the pixels with grayscale intensity less than T will be considered as background and the rest of the pixels as foreground. The weight or probability of class occurrence for $C_0$and $C_1$is calculated using the formulas

$$w_0 = \frac{\sum_{k=0}^{T} F_k}{N}, \quad w_1 = \frac{\sum_{k=T+1}^{255} F_k}{N}.$$

Where, F_k is the count frequency of the k[th] bin ($0 \leq k \leq T$ for background and $T + 1 \leq k \leq 255$ for foreground), T is the selected threshold and N is the total number of pixels. For example, an image with a resolution of 600×800 includes 480000 pixels ($N = 480000$).

The Otsu's algorithm iteratively searches for the threshold T, that minimizes the within-class variance, defined as

$$\sigma_w^2 = w_0 \sigma_0^2 + w_1 \sigma_1^2,$$

where σ_w^2 is the within-class variance, σ_0^2 and σ_1^2 are variances of the two classes.

Lastly, after finding the optimal value for T, the grayscale pixels are replaced by white in those areas, where intensity is greater than T and by black otherwise, resulting in a binary image.

This implementation might fail to consider the images of license plates as foreground, especially with white vehicles, when there is a shadow on the license plate. Multilevel Otsu threshold (Liao, Chen, and Chung 2001) which is an extension of the Otsu threshold finds several thresholds and creates multiple segments/objects in the image. Using all of these optimal thresholds helps in improving the algorithm in the license plate detection.

In the binary image, all the connected objects will be detected and labeled considering an 8-connectivity sense as shown in

figure 1. Hence all the pixels in the image that have the same value as their 8-connected neighbors will be considered as a connected region.

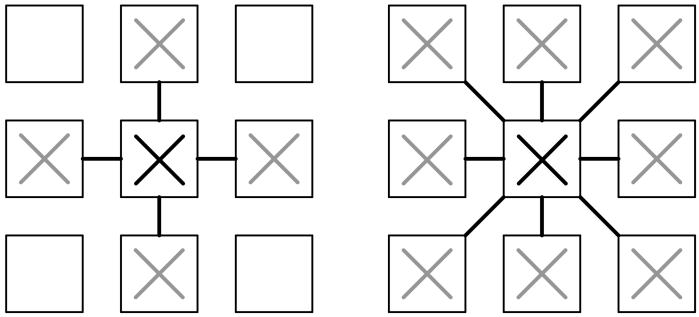

Figure 1: Difference between 4-connected pixels (left) and 8-connected (right).

After detecting all connected regions and labeling them, candidate labeled regions for the license plate are found. In some unclear images, the connected object detection fails to detect all the connected objects in the binary image. To improve the license plate detection algorithm, the Sobel filter (Sobel 2014) was used to emphasize the edges of the objects in each image.

The Sobel filter uses two 3×3 kernels, one for horizontal changes in the image and one for the vertical. The below filters are convolved over the image to find the derivative approximation in the x and y-axis.

$$K_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \qquad K_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

We call the derivative images $G_x = K_x * S$ and $G_y = K_y * S$ where S is the source image and * denotes convolution.

Then the magnitude of the gradient is found by

$$G = \sqrt{G_x^2 + G_y^2}$$

And using $G_x \; and \; G_y$ the angle of the edge is calculated as

$$\theta = atan\left(\frac{G_y}{G_x}\right).$$

Sobel filter was found to be suitable for increasing the precision of the algorithm in license plate detection, as presented in the results chapter of this study. Instances of vehicle images before and after implementation of the Sobel filter are presented in figure 2.

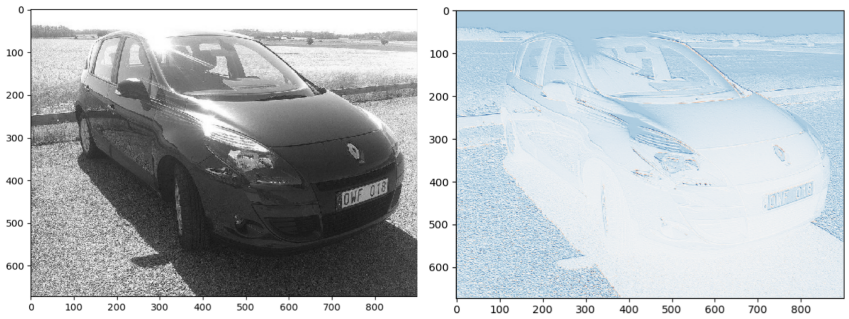

Figure 2: Vehicle before and after Sobel filter, left: normal image. right: Sobel filter.

The Sobel filter also enhanced the detection of some images with shadowed license plates, especially in bright images. In figure 3 two instances of shadowed license plates that were not detectable without the Sobel filter are presented.

As we can see, the shadow is removed and the image was brightened after the implementation of the Sobel filter so that the license plate will not be removed from consideration after using the Otsu thresholding.

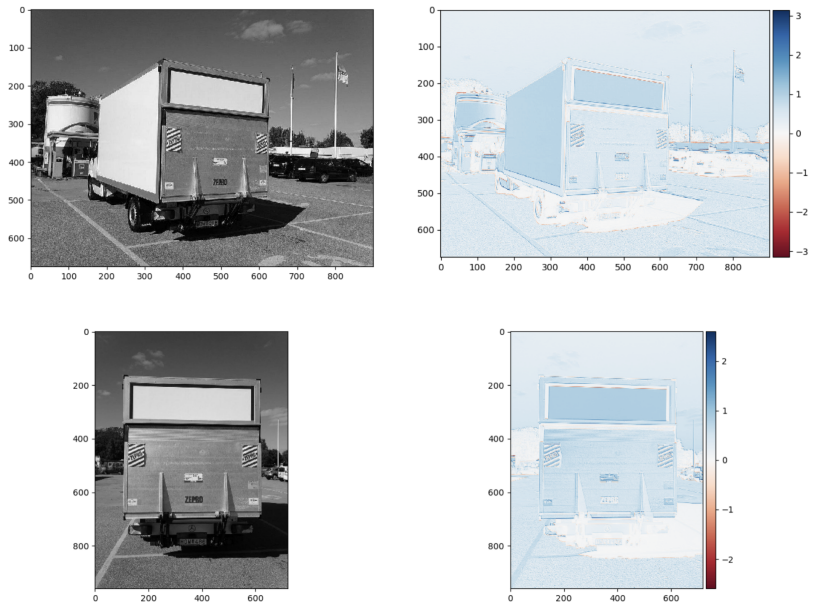

Figure 3: Left: original image, right: Sobel filter.

The license plates are rectangular, and the width of them is higher than the heights. All the objects which have a width between 5 % to 50 % of the total width of the image and

height of 2 % to 30 % of the total height of the image are a candidate for being a license plate. The presented numbers were found from the work of (Oladeji 2017) and tweaked to give the best result. In figure 4, images of three vehicles and the detected license plates are shown. In these images, although the license plates are non-horizontal, they were successfully detected. Although the calculation was done on the binarized image, for the next step, the original image of the license plate has been saved using the detected pixel locations in the original image.

Figure 4*: Original images and detected license plate.*

Multi-Otsu threshold and Sobel filter help in the detection of those license plates which are under the shadow in white vehicles, in figure 5, two instances of license plates that are correctly detected by multi-Otsu thresholding and Sobel filter are presented.

Figure 5: Detection of license plate under shadow using multi-Otsu and Sobel filter.

2.2 Character Segmentation

The found rectangles in the previous section which have sub-connected regions equal to or more than six are considered the license plates since they include six separate characters. To ensure that each character is separated, an erosion-dilation filter (Serra 1984) with a square of 2 pixels was implemented. This filter modifies any character which is connected to the boundary of the plate or an existing dirty stain on the plate. It also fixes characters that have some sort of cut due to blur making them not continuous. Erosion and dilation filters are two morphological filters that are used on binary images.

Morphological filters applied to an image can grow, shrink, remove, or fill-in regions in the image. An erosion filter is used to remove the boundary regions in the foreground pixels

as illustrated in figure 7. Hence the area of the foreground pixels (white pixels) will shrink, and the hole in the area becomes larger after implementing erosion. Dilation on the other hand acts in reverse and will increase the boundary of the foreground pixels. After implementing a dilation filter, the area of the foreground pixel will grow in size, and the holes become smaller.

Mathematically, erosion is defined as

$$A \ominus B = \{z \in E | B_z \subseteq A\} ,$$

where A is a binary image in a Euclidean space E, B is our structuring element and $B_z = \{b + z \mid b \in B\}$is the translation of the structuring element B by the vector $z \in E$.

Dilation is defined as

$$A \oplus B = \bigcup_{b \in B} A_b ,$$

where $A_b$is the translation of A by b.

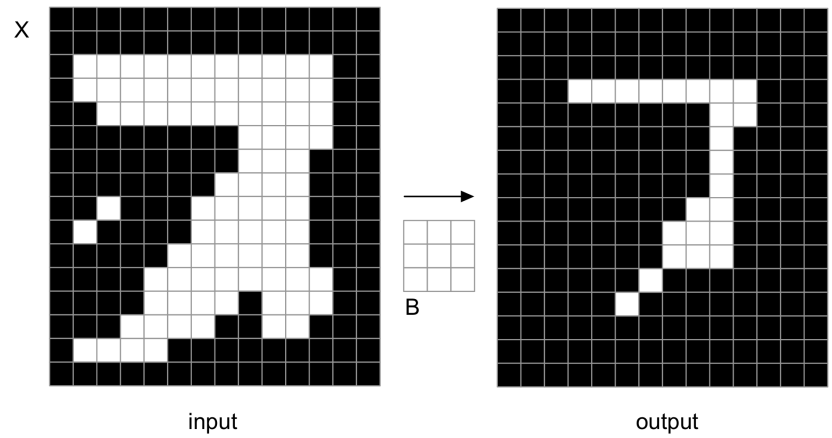

Figure 7*: Erosion on image X using structuring element B.*

Each character has a height between 30 % to 80 % and a width of 3 % to 30 % of the license plate. After the detection of each character, the characters were resized in 20×20 pixels. The presented numbers are a widened version of numbers presented in Oladeji's work. The numbers were tweaked to better perform with the algorithm of this study. The range was widened since there are procedures such as an erosion-dilation filter or line fitting which cleans up the license plate. In figure 8, the image of the license plate and detected characters are shown.

Figure 8*: license plate and detected characters.*

However, implementing the above algorithm is not enough for detecting the characters of the license plates, including angle and dirty stains or those that have character like objects inside the license plate. In figure 9, a license plate is shown that has two characters connected to the boundary with dirty stains. Implementing the above algorithm will detect 4 of 6 characters from the below image.

Figure 9*: license plate, including dirty stains.*

A technique that was used to avoid detection of more or less than six characters in the license plate was based on the fact

that in all images, whether the license plate is horizontal in the image or not, the top and bottom of all characters are settled on a straight line. So if a first-order polynomial is fitted to the upper pixel of each character and another line fitted to the bottom of the character pixels, these two lines will be tangent to all the character boundaries. As shown in figure 10, to fit a line in the upper and lower pixels of the character least square method is used.

Figure 10*: line fitted on the upper and lower boundary of characters*.

Line fitted on the upper boundary by minimizing $\sum_{i=1}^{6} e_i^2$ and lower boundary of characters by minimizing $\sum_{i=1}^{6} e'_i{}^2$ where

e_i^2 is the squared error of the ith character in upper boundary $(i = 1, 2,..., 6)$ and $e'_i{}^2$ is the squared error of i^{th} character in the lower boundary.

All the pixels above the upper tangent and below the lower tangent were removed. After removing pixels above and below of the characters by setting them to zero, the redundant pixels on the left and right side of the license plate as shown in figure 11, will also be removed because these objects now have the same size as the actual characters and will be wrongly detected as a character.

Figure 11: *License plate & detected characters after removing above and below the characters.*

Line fitting and removing the redundant objects at the start and end of the license plate improves the recognition of the license plate, however, there are still cases where the algorithm fails to detect redundant objects such as stickers in

the license plate. For instance, some stickers can be wrongly detected as a character of the license plate. But since they usually have different colors or different dimensions k-means clustering algorithm applied on the original RGB image helps detect such objects (MacQueen 1967).

Given n data points and k initial centroids, the k-means clustering algorithm iteratively places each point in a cluster or set with the nearest centroid using their Euclidean distance and then updates the cluster centroid at each step using the mean of all points in the cluster. This iteration is done until there is no more change in the centroids and therefore the algorithm has converged.

K-means clustering was implemented on the RGB image of all detected candidate characters. It uses the squared euclidean distance and maximizes the distance between various clusters. Squared Euclidean distance for the intensity matrix of RGB image is as

$$D(i,j) = (G_i - G_j)^2 + (B_i - B_j)^2 + (R_i - R_j)^2$$

where G_i are the pixels of the green channel of i^{th} character, G_j are the pixels of the green channel in j^{th} character. B_i,B_j are pixels of the blue channel, and R_i,R_j are pixels of the red channel for i^{th} and j^{th} characters.

Considering two clusters ($k = 2$), most of the characters are located in the same cluster because of their similarity in color or shape, and other objects in the second cluster are removed.

The cluster which has the most frequent items in it is considered as correctly detected characters.

This method improved the correct character detection greatly. Figure 12 shows instances of vehicle license plate images that have redundant objects even after implementing line fitting. K means clustering could correctly distinguish between the correct color of the true character from those character-like objects which have different colors or shape. For example, the middle image of figure 12 includes a sticker that is similar to the character C, and it could be distinguished after implementing K-means clustering on an RGB image.

Figure 12*: license plate instances could be correctly detected using K-means clustering.*

The preprocessing and image segmentation are fast procedures, and they could be done on testing images to detect their characters before the implementation of classification analysis.

Hence for all new images, the character detection will be done, and then the characters are ready for being predicted using the pre-trained classification model. The method of classification will be explained in the next part.

3 Classification

Classification is a supervised learning method that is built for predicting the class of each observation and classify it in its group. Classification's aim is to give the machine the ability to identify the class of each new observation (Alpaydın 2010). For character recognition, there are multi-class characters, meaning the output of the classification can only be one of the allowed characters. The algorithm should be designed to be able to distinguish between each character and recognize it correctly (Du et al. 2013).

The segmented characters images found in the previous section, just like Oladeji's work are resized to 20×20 pixels and then flattens to 400×1 pixels to be used as the input for the classification models.

The training data used by Oladeji (2016), available at GitHub, was used to train the models introduced in this chapter. This training data includes 498 images of various characters and numbers used in the license plate. The testing data are the character images detected from the 200 Blocket images, detected in the previous chapter. These characters are then labeled by hand in order to calculate the accuracy of the models.

In this study, Support-vector machine (SVM), Artificial Neural Network (ANN), and a Convolutional Neural Network

(CNN) described in detail in the following sections were implemented.

3.1 Perceptron

Although not used in the study, introducing Perceptron, aids in comprehension of more complex methods. Perceptron is a simple supervised learning algorithm for learning a binary classifier (Freund and Schapire 1999). The input of the perceptron model is a real-valued vector $\overline{x}$, and the parameter $\overline{w}$ is the weights for each of the input values. The perceptron algorithm is a linear formula that gives a binary output. $f(x) = 1$ if $w.x - threshold > 0$ and $f(x) = 0$ otherwise.

To simplify the formula the threshold can be renamed to w_0 and be included in the vector $\overline{w}$ with a corresponding $x_0 = 1$ in the $\overline{x}$ vector. After this simplification, the formula becomes simply $f(x) = 1\ if\ \overline{w}.\overline{x} > 0\ otherwise\ f(x) = 0$.

This formula gives a separating line or hyperplane depending on the space of the data. If the data is linearly separable then, the parameter $\overline{w}$gets updated as $\overline{w}' \leftarrow \overline{w} + \eta\overline{dx}$ at each step to separate the data with no misclassification as in figure 13. In the update phase, the parameter ηis called the learning rate which determines how fast we would get to a solution, and d is +1 or -1 depending on the position of the misclassified point.

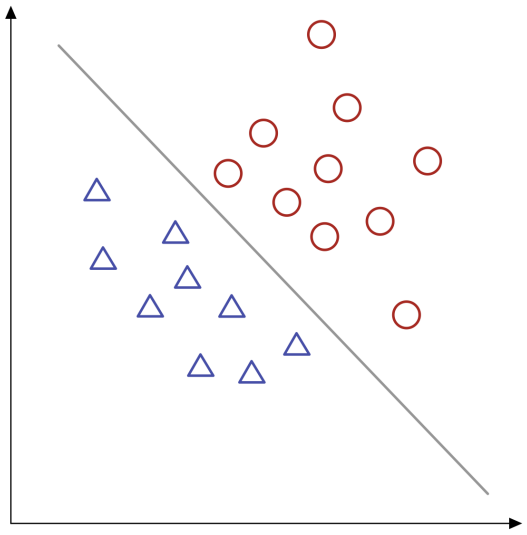

Figure 13: Perceptron used for classification of linearly separable binary data.

3.2 Support-vector Machine

This section addresses the support-vector machine model and its implementation in this study.

3.2.1 SVM on Linearly Separable Data

Support-vector machine (SVM) is an algorithm for classification of data, discriminative, and by using a hyperplane for separating the groups of the classes under study(Cortes and Vapnik 1995). This classifier is a non-probabilistic classifier that maximizes the distance between support vectors of each class for distinguishing the groups in the classification. (Platt 1999) provided a method

for transforming the output of the classification using SVM in terms of the probability distribution for the classes. The algorithm uses a hyperplane in multi-dimensional space to separate the classes from each other. If the data is linearly separable, then the optimization algorithm will calculate the hyper line somehow so that the distance between the points of each group and the hyper line is maximized as in figure 14. The parameter*w*helps in finding the maximum margin between the hyperplane and the support vectors, the points on the margin. If the nearest point to the hyperplane is called x_n, then the objective is to find the distance between this point and the hyperplane $w^T x + b = 0$(Abu-Mostafa, Magdon-Ismail, and Lin 2012).

The optimization problem as a Lagrangian, where α_n is the Lagrangian multiplier, is as

$$minimize\ L(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{n=1}^{N} \alpha_n (y_n (w^T x_n + b) - 1)$$

w.r.t. w and b, and maximize w.r.t each $\alpha_n \geq 0$.

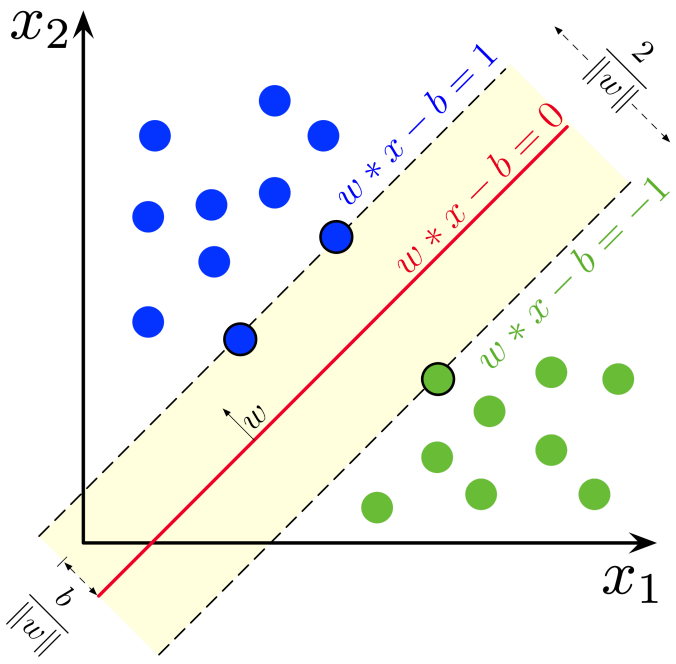

Figure 14: English: Maximum-margin hyperplane and margin for an SVM trained on two classes (Larhmam, CC BY-SA 4.0, via Wikimedia Commons).

To minimize the Lagrangian, we set the gradient w.r.t. w, equal to the zero vector and w.r.t. b, to the scalar zero.

$$\Delta_w L = w - \sum_{n=1}^{N} \alpha_n y_n x_n = \mathbf{0} \text{ and } \frac{\delta L}{\delta b} = -\sum_{n=1}^{N} \alpha_n y_n = 0.$$

Now using the above gradient the optimization problem gets free from w and b as

$$maximize\ L(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m x_n^T x_m$$

$$\text{w.r.t. } \alpha \text{subject to } \alpha_n \geq 0 \; and \sum_{n=1}^{N} \alpha_n y_n = 0.$$

Next by feeding this formula and its constraints to the

quadratic programming, the vector αis found which in turn using one of our previous constraints, $w = \sum_{n=1}^{N} \alpha_n y_n x_n$ it finds
the vector w. When $\alpha_i > 0$,it means that the point i is a support vector, it is on the margin of the separating hyperplane, otherwise, it is an interior point.

The bias term is also found using the fact that for the support vectors it holds that $y_n(w^T x_n + b) = 1$.

3.2.2 Non-Linearly Transformation

When the input data is not linearly separable it is possible to solve the optimization problem by going to a higher dimension. This can be seen visually with an example in figure 15. Looking at the optimization problem, once the space x is transformed to z the only difference in the formula is that the inner product is now in the z space and the complexity of the optimization is unchanged. After the inner product, the quadratic programming gives the α,which is then interpreted in the x space.

$$maximize\ L(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m z_n^T z_m$$

$$\text{w.r.t. } \alpha \text{ subject to } \alpha_n \geq 0 \; and \sum_{n=1}^{N} \alpha_n y_n = 0.$$

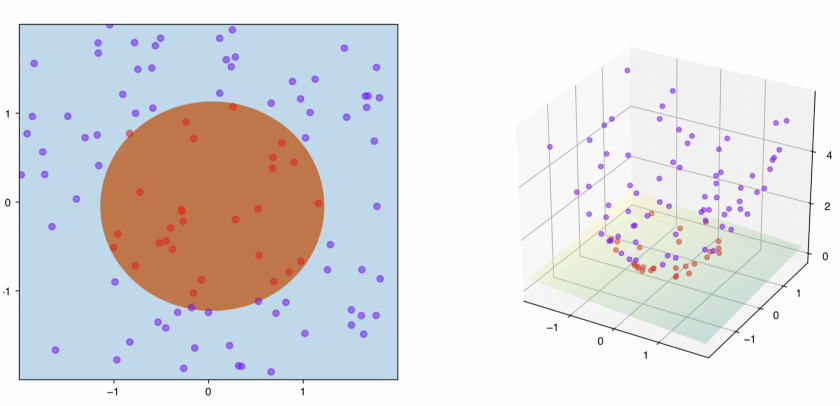

Figure 15: Transforming to a higher dimension where the data might be separable (Shiyu Ji, CC BY-SA 4.0 , via Wikimedia Commons).

3.2.3 Kernel Function

Now instead of having an inner-product in some space z, a trick called the Kernel trick or function can be used. The function of the SVM model including the kernel function is as

$$Maximize: \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n \alpha_n K\left(\vec{x}_n, \vec{x}_m\right) y_m \alpha_m$$

$$subject\ to: \sum_{n=1}^{N} \alpha_n y_n = 0\,,\ 0 \leq \alpha_n \leq \frac{1}{2N\lambda}\ ,\ n = 1\ to\ N.$$

Instead of $z_j^T z_i$ in the above objective function kernel function of x_j^T and x_i was used, which maps the coordinate into another coordinate in which the linear hyperplane could be used for separating the groups.

$$K\left(\vec{x}_i, \vec{x}_j\right) = \varphi\left(\vec{x}_i\right).\varphi\left(\vec{x}_j\right).$$

These kernel functions are used for transforming the coordinates. For example, a polynomial kernel function has formula $K\left(\vec{x}_i, \vec{x}_j\right) = (\vec{x}_i^T . \vec{x}_j + r)^d$, and the Gaussian kernel function is written as

$$K\left(\vec{x}_i, \vec{x}_j\right) = e^{-\frac{\left\|\vec{x}_i - \vec{x}_j\right\|^2}{2\sigma^2}}.$$

Solving the optimization problem subjected to the given constraint after implementing the kernel transformation in the nonlinear cases will lead to the estimation of the support vectors, which are used for distinguishing the new character observation.

3.2.4 Implementation of SVM

SVM was implemented using the Gaussian Radial Basis Function kernel also known as RBF. This kernel is usually used per default if there is no prior knowledge of the dataset ("Seven Most Popular SVM Kernels" 2020). The RBF formula is

$$K\left(\vec{x}_i, \vec{x}_j\right) = e^{-\gamma\left\|x_i - x_j\right\|^2}$$

where the value of gamma is manually chosen between 0 and 1, usually 0.1.

3.3 Artificial Neural Network

This section addresses the Artificial Neural Network model and its implementation in this study.

3.3.1 Primary Notion

The artificial neural network is also known as the Neural network, is vaguely modeled after the biological neural networks where there are connected neurons that communicate with one another via synapses (Chen et al. 2019). The mathematical and computational model for the neural network was developed by (McColloch & Pitts, 1943).

The input data of the neural network are collected in a layer, usually, the size of this layer is equal to the number of training data plus a bias term. There are several layers called the hidden layers between the input layer and the output layer containing different numbers of neurons and different activation functions. At the last layer will be the output layer, which includes the classification result or prediction of the class group for the classification model. The size of the output layer is different depending on the task which the neural network is trying to achieve. If there are L layers, in a neural network, the neurons in each layer $1 \leq i \leq L$ may be connected to some of the neurons in the next layer $i + 1$.This means that the network is not necessarily fully connected (Tch 2017).

Perceptron is the simplest ANN which consists of only one neuron. The Multi-layer Perceptron on the other hand has an input layer $x \in R^M$, an output layer $y \in R^N$, and one or more fully connected hidden layers as shown in figure 16.

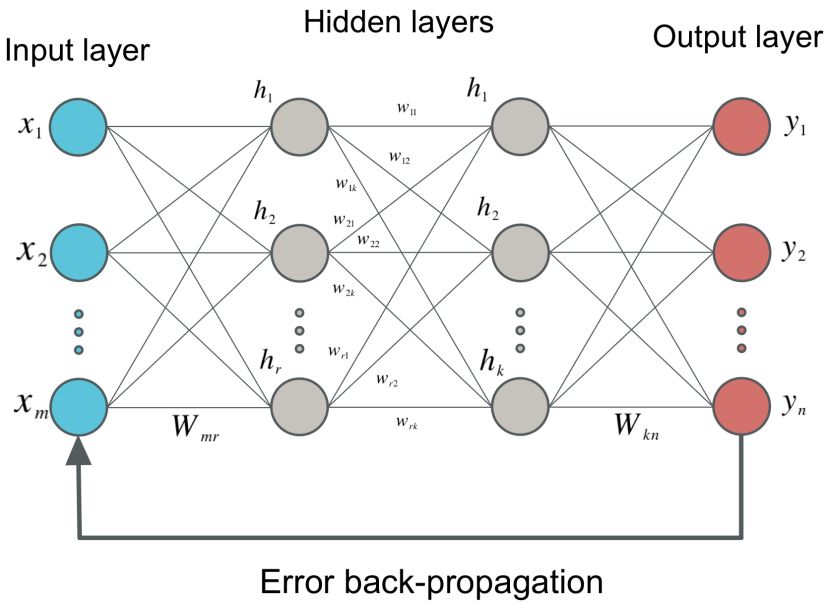

Figure 16: Architecture of multilayer artificial neural networks with error backpropagation.

A single neuron is mathematically a function $h: R^M \rightarrow R$ defined as

$$h_j = f(x, w_j) = \sigma(\sum_{i=1}^{M} w_{ij} x_i + b) = \sigma(\overline{w}^T . \overline{x}),$$

where σ: $R \rightarrow R$is an activation function and w is a vector of weights including the bias term. A hidden layer with M inputs and k neurons is a function $H_i: R^M \rightarrow R^k$ written as

$$H_i = \begin{bmatrix} h_1 \\ h_2 \\ . \\ h_k \end{bmatrix} \text{ or as matrix form } H_i = \sigma(Wx) \text{ where } W = \begin{bmatrix} W_1^\top \\ W_2^\top \\ . \\ W_k^\top \end{bmatrix}.$$

The output layer of an ANN with L layers including the input layer can be found as a nested function

$$y = \sigma^{(L)}(W^{(L)}\sigma^{(L-1)}(\ldots(\sigma^{(2)}(W^{(2)}x^{(1)})\ldots))),$$

where $x^{(1)}$is the input layer and

$x^{(l)} = \sigma^{(l)}(W^{(l)}x^{(l-1)}),\ 1 < l < L$are the hidden layers.

3.3.2 Loss Function

Finding the optimal solution of the network is done by using forward propagation and then back-propagation. In the forward propagation, the algorithm is solved from the input features to the next hidden layer and from that to the next layer until it reaches the output layer. The forward propagation learning model adjusts the weights matrix in each layer to minimizes the difference between the actual value of the output $\widehat{y}$and the output of the Neural Network y.This difference is referred to as the loss function. There are many loss functions with different use cases, for example, the categorical cross-entropy (Rubinstein and Kroese 2013) is calculated by

$$\varepsilon = -\sum_{i=1}^{N} y_i . log\, \widehat{y_i}.$$

3.3.3 Back-propagation

In back-propagation, the solution starts from the output layer and propagates backward until it reaches the input layer. Backpropagation first finds the gradient of the error function with respect to the weights extracted from the forward propagation and then uses gradient descent to find the local minima (Rojas 1996). Propagating backward from the output layer at each layer then the weights are adjusted to find the minimum loss. The weights are adjusted by $w_{ij} = w_{ij} - \alpha \frac{\delta \varepsilon}{\delta w_{ij}}$ where $\alpha$is the learning rate and $\frac{\delta \varepsilon}{\delta w_{ij}}$the gradient of the loss function.

3.3.4 Activation Function

Depending on the input the activation function decides the output of a layer (Hinkelmann, n.d.). There are two types of activation functions, linear and nonlinear. Non-linear activation functions are desirable since they grant solving complex problems. The plot of some of the commonly used activation functions and their formulas are shown in figure 17.

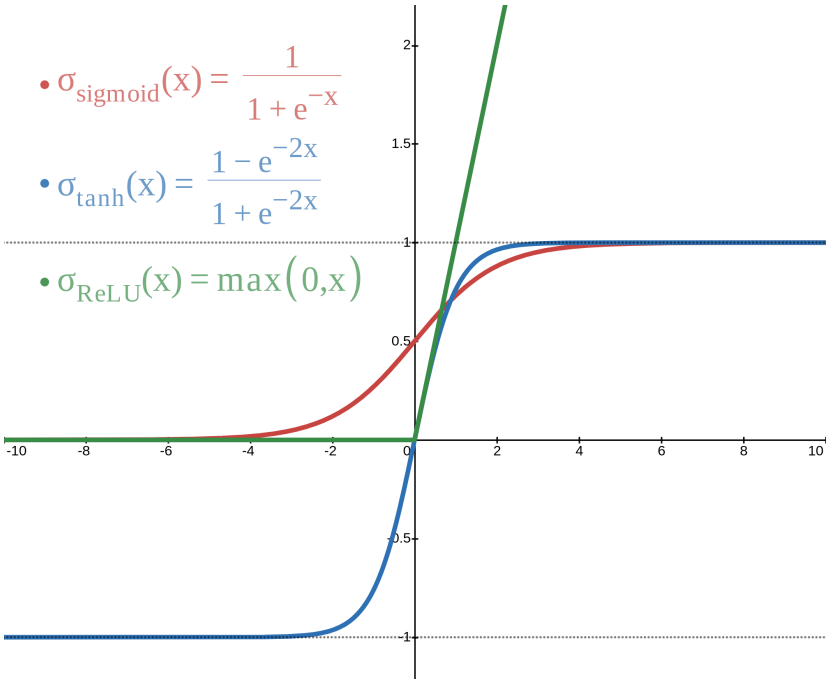

Figure 17: Plot of Sigmoid, tanh, and ReLU activation functions.

The softmax function which is a generalization of the sigmoid function is used for solving multi-class classification problems (Wikipedia contributors 2020b). Among other use cases, Softmax is often used as the last activation function in neural networks to normalize the output of the network to a probability distribution over predicted output classes so that the sum of the outputs of $\sigma(x)$ equals 1. For example, in a multi-class classification problem Softmax is used as the last activation function to rescale the output for comparison between probability distribution and the categorical cross-entropy loss function (Rubinstein and Kroese 2013).

The Softmax function is defined by

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \; for\; i = 1,\ldots,J \; and \; x = (x_1,\ldots,x_J) \in R^J.$$

3.3.5 Implementation of ANN

The Artificial neural network was fitted using one hidden layer, since in most cases, using one hidden layer is enough (Heaton 2008). The optimal size or number of neurons in the hidden layers are by some rule of thumb, usually between the size of the input and size of the output layers (Heaton 2008). The hidden layer size can be considered as 2/3 of the input features plus the number of output classes and should be less than twice the number of input features.

Pruning the hidden layer size was done after finding the values for the weights of the connections. The neurons which have a very low value in the weights of the connections could be removed from the network, and the model could be updated by checking the classification accuracy. The final model will be the model, which has the highest classification accuracy in the testing data.

The input dimension for each character is 400 pixels. The model was fitted using categorical cross-entropy as a loss function. Adam optimizer, an algorithm used instead of the classical stochastic gradient descent for updating network weights iteratively based on the training data, was used for fitting the model.

3.4 Convolutional Neural Network (CNN)

This section addresses the Convolutional Neural Network model and its implementation in this study.

3.4.1 Definition

The convolutional neural network is a deep learning method that is mostly used for image and video processing. It is the regularized form of the multilayer perceptron method, while a multilayer perceptron is a feedforward artificial neural network in which the input and the layers are fully-connected (all the neurons are connected to the previous and latter layers). The fully-connected structure causes the network to tend to overfit unless using some regularization (Caruana, Lawrence, and Giles 2001). Multilayer Perceptron is inefficient in image classification since in a fully connected neural network of an image of size $width \times height \times colour\ channel$the vector of weights in each neuron is too large.

The advantage of CNN is that each neuron only receives a small chunk of the input image and applies a small weight matrix called the kernel in the convolution process.

Convolution is the sum of the elementwise product of each cell, for example

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} * \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix} = \sum_{i=1}^{9} a_i \times b_i$$

.

The kernel is by convention quadratic, has an odd size and the number of channels in the kernel must be the same as the source image. The kernel then convolves over the image input patch by patch as shown in figure 18. The convolution operation layer might be followed by a pooling layer which downsamples the spatial size of the input and in the end, there is a fully connected layer giving the output layer. The output size of CNN depends on three parameters namely depth, stride, and zero-padding (Karpathy and Others 2016). The depth corresponds to the number of filters used, the stride is the step which we take in each convolution, for example, if we move only one pixel between each step then $S = 1$,and zero-padding is when we pad the input with zeroes for convenience.

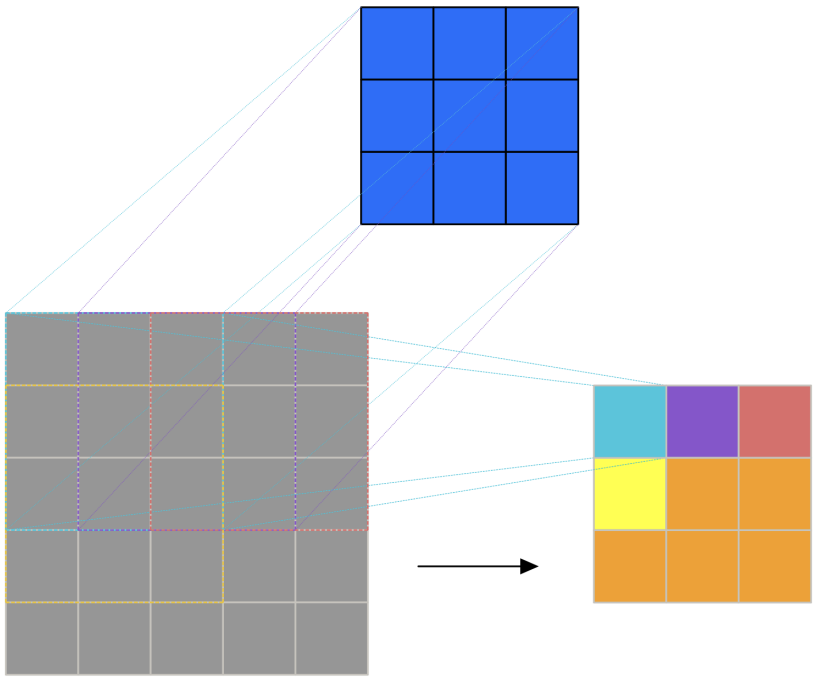

Figure 18: Convolving a single channel kernel of 3 × 3 on an image of size 5 × 5, stride=1, and with no padding.

In the CNN methods, regularization is implemented using hierarchical pattern finding in the data. By checking a smaller and smaller range of the data with the moving matrix of pixels and collecting complex patterns of the data from these small ranges in each step. This is one advantage of the CNN method, which makes it capable of finding the pattern from the pixels from their neighborhood pixels. CNN method requires much less pre-processing compared with other methods since its network could learn the features itself. This advantage of CNN makes it independent from prior knowledge about the data.

3.4.2 Max Pooling

Max Pooling down-samples the input image that means the dimensionality of the input gets reduced in the next layer. The idea is to divide the input into non-overlapping regions and select the maximum of each region as shown in figure 19.

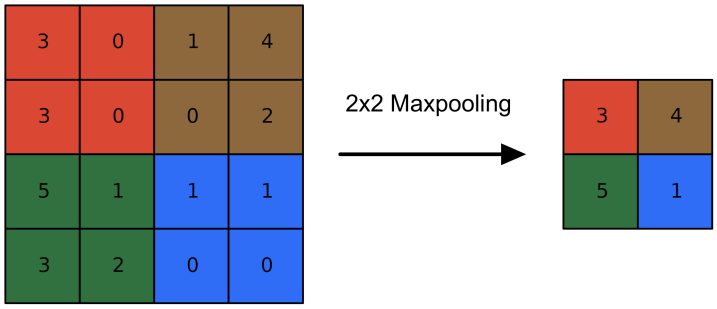

Figure 19: 2x2 max pooling with stride = 2.

3.4.3 Implementation of CNN

The convolutional neural network was used with six convolutional layers with 3×3 kernel size, followed by a 2×2max-pooling layer. The input size as in the previous classification models is $20 \times 20 \times 1$and the dimensionality of the output space or it's channel size is increased as we go deeper in the network. Starting with 32 for the first two layers, 64 for layers three and four and lastly 128 for convolutional layer five and six. As we are using max-pooling after each layer we can computationally afford

to increase the output dimension ("Convolutional Neural Network (CNN)" n.d.).

After each layer a batching normalization (Ioffe and Szegedy 2015) as a technique for training deep neural networks by standardizing the inputs to a layer was implemented by dropping out randomly 25 % of the neurons. Randomly dropping the neurons is done to reduce the effect of the weights of the neurons to reduce overfitting.

Lastly two dense layers one with size of 512 and another with size 35, representing the 35 used characters in license plates, was added to perform the classification. For the last dense layer the softmax activation function and for all the other layers the Rectifier activation function was used.

4 Results

The findings of this study are presented in this chapter.

4.1 Improvements

The code provided on GitHub by (Oladeji 2016) was compared with the methodology presented in the study. Among 127 images, the precision of the license plate algorithm using Oladeji's code and the current algorithm are presented in table 1.

Table 1*:Comparison of results of Oladeji's code with code used in this project.*

Method	*# of license plate*	*# of detection*	*# of correctly detected*	*Precision in detection*	*Precision in total images*
Oladeji (2016)	*127*	*73*	*64*	*0.88*	*0.5*
polynomial fitting	*127*	*90*	*90*	*1.0*	*0.71*
polynomial fitting + Sobel + multi-Otsu thresholding	*127*	*117*	*110*	*0.94*	*0.87*

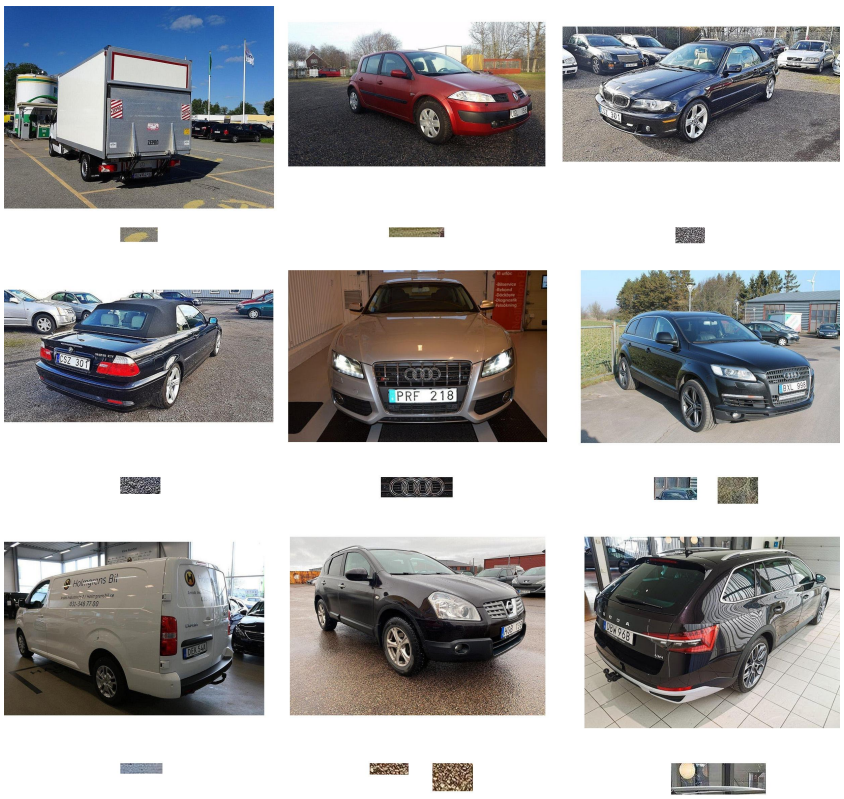

Figure 20: Wrongly detected plated of 9 images by GitHub code.

In table 1, we can see that from 127 vehicles with license plates, Oladeji's code could detect 73 of them, and 64 of them were correct. So precision for detected images is 88 %, and for all images, the precision is 50 %. Implementing the polynomial fitting method improved the algorithm, and from 127 images, 90 of them were detected. From 90 detected images, 90 of them contain a detected license plate, which shows 100 % precision in detecting images and 71 % precision for all images. Multi-Otsu thresholds (Liao, Chen, and Chung 2001) plus the Sobel filter (Sobel 2014) with polynomial fitting method increased the precision of the total detected license plates from 71 % to 87 %.

From 73 vehicle images that were detected by using code presented on GitHub, 64 of them were detected correctly whereas 9 of them were detected wrongly, images of 9github instances that were wrongly detected are presented in figure 20.

4.2 Classification Models

Three classification methods have been used for training the classification method for character recognition of Swedish vehicles. The SVM classifier with Gaussian kernel was fitted on the training data and in Table 2 the result of the classification in the testing data are presented.

Tables 2-4 are generated using the sklearn machine learning library (Wikipedia contributors 2021b), and here are

definitions of the terminology used in them ("API Reference — Scikit-Learn 0.24.1 Documentation" n.d.):

- **Precision:** How good the model can avoid sampling a negative sample as positive. Defined as $Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$.
- **Recall:** How good the model can find all positive samples. Defined as $Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$.
- **F1-Score:** F1_Score is the weighted average of the precision and recall. The best value for the F1-Score would be 1 and the worst value 0. Defined as $F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$.
- **Support:** Support is the number of occurrences of each label.
- **Accuracy:** How well the model performed. This is the same as the weighted average of the F1-Score.
- **Macro Avg:** The unweighted average of precision/recall/F1-Score of the classes. For example, the Macro Average for precision would be calculated by

$$Macro(P) = \frac{\sum_{i=1}^{N} P_i}{N},$$

where P_i is the precision of class i and N is the number of classes.

- **Weighted avg:** The weighted average, takes into account the number of true instances of a label. For example, the weighted average for precision is as $avg = \sum_{i=1}^{N} \frac{s_i}{S} \times P_i$, where s_i stands for support of the label i and S is the total number of data.

The total accuracy of the model on the testing data is 91 % with the macro averaged and weighted average precision, recall, and f1-score are more than 90 %. Out of 660 characters 600 were classified correctly and the only character which has a recall value (sensitivity) less than 60 % is character '3' with recall = 56 %. It means that 8 out of 18 the character '3' were wrongly assigned to other categories and that causes a reduction in the recall of this category. While the precision of this category ('3') is 100 % which means no characters were wrongly assigned to this category.

Table 3 shows the results of ANN classification accuracy for each character and the number of supporting observations in each category.

As we can see in the table the macro averaged precision and recall are 75 % and 77 % respectively. The weighted average precision and recall are 85 % and 82 %. Looking at the categories, characters '1', '3', '4', '5', 'J', 'S', and 'Z' have a recall or precision accuracy of less than 60 %.

Table 4 shows the precision, recall, and accuracy of the model for each character and a macro and weighted average accuracy of the CNN classification model.

Likewise table 4 shows the results of CNN classification accuracy for each character and the number of supporting observations in each category.

The results in table 4 show that only 2 categories of ‘z’ and ‘3’ have recall or precision of less than 60 %. The macro average precision and recall are 91 % and 90 % respectively and the weighted average precision and recall are 95 % and 93 %.

Table 2*: SVM Classification accuracy measurement for each character on the testing data.*

Character	Precision	Recall	F1-score	Support
0	0.96	0.92	0.94	60
1	0.82	0.95	0.88	44
2	0.87	0.96	0.91	49
3	1	0.56	0.71	18
4	0.85	0.85	0.85	20
5	0.72	0.95	0.82	22
6	0.81	0.81	0.81	16
7	0.96	1	0.98	27
8	0.95	0.95	0.95	38
9	1	0.98	0.99	45
A	1	0.83	0.91	12
B	0.91	0.83	0.87	24
C	0.92	0.92	0.92	12
D	0.89	0.94	0.91	17
E	1	0.86	0.92	28
F	0.9	1	0.95	19
G	0.75	0.8	0.77	15
H	0.83	1	0.91	20
J	1	1	1	12
K	0.86	1	0.92	6
L	1	0.93	0.96	14
M	1	0.78	0.88	23
N	1	0.78	0.88	9
P	1	0.96	0.98	25
R	0.82	1	0.9	9
S	1	0.91	0.95	11
T	0.89	0.8	0.84	10
U	0.77	1	0.87	10
W	1	0.94	0.97	17
X	0.75	1	0.86	6
Y	1	0.75	0.86	16
Z	0.83	0.83	0.83	6
accuracy			0.91	660
macro avg	0.91	0.9	0.9	660
weighted avg	0.92	0.91	0.91	660

Table 3*: ANN Classification accuracy measurement for each character on the testing data.*

Character	Precision	Recall	F1-score	Support
0	0.93	0.87	0.9	60
1	0.86	0.55	0.67	44
2	0.74	0.92	0.82	49
3	0.91	0.56	0.69	18
4	0.59	0.85	0.69	20
5	1	0.59	0.74	22
6	0.67	0.62	0.65	16
7	0.73	0.89	0.8	27
8	0.94	0.79	0.86	38
9	0.97	0.82	0.89	45
A	1	0.83	0.91	12
B	0.88	0.88	0.88	24
C	0.91	0.83	0.87	12
D	0.84	0.94	0.89	17
E	0.96	0.89	0.93	28
F	0.83	1	0.9	19
G	0.65	0.87	0.74	15
H	0.9	0.95	0.93	20
J	0.52	0.92	0.67	12
K	0.6	1	0.75	6
L	0.93	0.93	0.93	14
M	1	0.87	0.93	23
N	0.88	0.78	0.82	9
P	0.89	0.96	0.92	25
R	0.67	0.89	0.76	9
S	0.5	0.82	0.62	11
T	0.89	0.8	0.84	10
U	0.91	1	0.95	10
W	1	0.82	0.9	17
X	0.6	1	0.75	6
Y	0.92	0.69	0.79	16
Z	0	0	0	6
accuracy			0.82	660
macro avg	0.75	0.77	0.75	660
weighted avg	0.85	0.82	0.82	660

Table 4*: CNN Classification accuracy measurement for each character on the testing data.*

Character	Precision	Recall	F1-score	Support
0	0.97	0.97	0.97	60
1	0.95	0.86	0.9	44
2	1	0.86	0.92	49
3	1	0.56	0.71	18
4	0.95	0.9	0.92	20
5	0.69	1	0.81	22
6	1	0.88	0.93	16
7	0.87	1	0.93	27
8	1	0.97	0.99	38
9	1	1	1	45
A	0.92	0.92	0.92	12
B	1	0.88	0.93	24
C	0.86	1	0.92	12
D	0.89	0.94	0.91	17
E	0.97	1	0.98	28
F	1	1	1	19
G	1	0.87	0.93	15
H	0.95	0.95	0.95	20
J	1	1	1	12
K	0.86	1	0.92	6
L	0.93	0.93	0.93	14
M	1	0.96	0.98	23
N	0.88	0.78	0.82	9
P	1	1	1	25
R	1	1	1	9
S	0.92	1	0.96	11
T	0.91	1	0.95	10
U	1	0.9	0.95	10
W	1	0.82	0.9	17
X	1	1	1	6
Y	1	0.81	0.9	16
Z	0.4	1	0.57	6
accuracy			0.94	660
macro avg	0.91	0.9	0.89	660
weighted avg	0.95	0.93	0.94	660

4.3 Received Operating Characteristic to Multi-class

The received operating characteristic curve, also known as ROC curve, (Garrett, Lasky, and Meier 2008), for three classifiers which shows the true positive and false positive of the macro averaged of all groups, is presented in Figure 21. It can be seen that the area under the curve (AUC) of the CNN model is the highest compared with the other 2 classification methods.

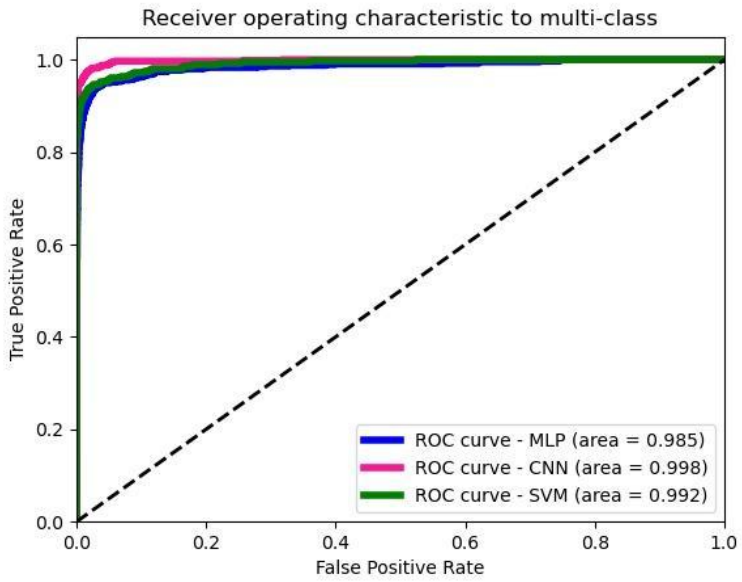

Figure 21*: ROC curve and area under the curve for three implemented classifier on testing data.*

The area under the curve (AUC) for CNN is 0.998 which is the highest. SVM result is close to CNN and the AUC is a bit lower 0.992. The lowest AUC could be seen for the ANN model.

5 Discussion

This chapter presents an analysis of the obtained results, the conclusion, and suggestions for future work.

5.1 Result Analysis

In this section, the provided results in accordance with the aim of the study are analyzed.

5.1.1 Improvement

In this study, various license plate images were tested. The challenges were that some of the images included tilted license plates, or license plates that had dirt stains or stickers on them interfering with the character detection process. Several implementations were done on the images to improve the license plate detection. Along with Otsu thresholding, multi Otsu thresholding, Sobel filter, polynomial fitting, erosion-dilation the k-means clustering algorithm on intensity images was employed to improve the license plate detection precision of the model. The algorithm was able to detect 87 % of the total license plates correctly which showed 37 % improvement after implementing these preprocessing techniques.

The procedure for license plate recognition is divided into license plate detection, character segmentation, and then classification for distinguishing between characters.

It was clear that using the simple algorithm based on connected object detection, does not work well for detecting the license plates, especially in the images which have license plates with inclination.

The implemented algorithm was useful in the improvement of license plate detection. In the 3rd algorithm mentioned in table 1, most of the license plates could be detected correctly.

5.1.2 Classification Model Analysis

Among the classification methods, the deep learning classification using a convolutional neural network (CNN) was seen to have better performance in character recognition of the testing data with 94 % accuracy. Because CNN includes filters and it uses convolution of the matrices for each layer. It studies the images by considering their neighbor pixels by moving the kernel matrix. However, the disadvantage of the CNN method is that it is a deep learning method that requires GPU for training a larger amount of data thus using the CPU is inefficient and will require much longer processing time to reach the results. Other machine learning methods are much faster compared with CNN but they require proper feature extraction methods before classification to reach an acceptable result.

ANN results were seen to be overfitting to the training data. The accuracy of the testing data was 82 %.

5.2 Conclusion

The conclusion is that these preprocessing techniques improved the recognition of the license plate and characters significantly with CNN giving the best accuracy among the other tested classification models.

5.3 Suggestions for Future Studies

It was seen that the deep learning method has a bit of improvement compared with other machine learning methods. The CNN method has the ability to study intensity images. For future study, it could be a suggestion to use the intensity images for deep learning methods to enhance the power of the machine learning model for character recognition.

Since the Support Vector Machine classifier with Gaussian Kernel shows good results which is comparable with deep learning results of CNN. Another suggestion could be improving the SVM classifier by implementing proper tuning parameters and implementing proper feature extraction methods instead of using raw binary vectors of pixels before performing the classification analysis.

Bibliography

Abu-Mostafa, Yaser S., Malik Magdon-Ismail, and Hsuan-Tien Lin. 2012. *Learning from Data*. Vol. 4. AMLBook New York, NY, USA:

Alpaydın, Ethem. 2010. "Introduction to Machine Learning, Second Editon by Ethem Alpaydin, MIT Press, 584 Pp., $55.00. ISBN 978-0-262-01243-0." *The Knowledge Engineering Review*. https://doi.org/10.1017/s0269888910000056.

"API Reference — Scikit-Learn 0.24.1 Documentation." n.d. Accessed February 3, 2021. https://scikit-learn.org/stable/modules/classes.html.

Caruana, Rich, Steve Lawrence, and C. Lee Giles. 2001. "Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping." In *Advances in Neural Information Processing Systems 13*, edited by T. K. Leen, T. G. Dietterich, and V. Tresp, 402–8. MIT Press.

Chen, Yung-Yao, Yu-Hsiu Lin, Chia-Ching Kung, Ming-Han Chung, and I-Hsuan Yen. 2019. "Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in Demand-Side Management for Smart Homes." *Sensors* 19 (9): 2047.

"Convolutional Neural Network (CNN)." n.d. Accessed February 8, 2021. https://www.tensorflow.org/tutorials/images/cnn.

Cortes, Corinna, and Vladimir Vapnik. 1995. "N.(1995). Support-Vector Networks." *Machine Learning* 20 (3): 273–97.

Crump, Catherine. 2013. *You Are Being Tracked: How License Plate Readers Are Being Used to Record Americans' Movements*. American Civil Liberties Union.

Du, Shan, Mahmoud Ibrahim, Mohamed Shehata, and Wael Badawy. 2013. "Automatic License Plate Recognition (ALPR): A State-of-the-Art Review." *IEEE Transactions on Circuits and Systems for Video Technology*. https://doi.org/10.1109/tcsvt.2012.2203741.

Freund, Yoav, and Robert E. Schapire. 1999. “Large Margin Classification Using the Perceptron Algorithm.” *Machine Learning* 37 (3): 277–96.

Garrett, Patricia E., Fred D. Lasky, and Kristen L. Meier. 2008. *User Protocol for Evaluation of Qualitative Test Performance: Approved Guideline*. Clinical and Laboratory Standards Institute.

Heaton, Jeff. 2008. *Introduction to Neural Networks with Java*. Heaton Research, Inc.

Hinkelmann, Knut. n.d. “Neural Networks, P. 7.” *University of Applied Sciences Northwestern Switzerland. Http://didattica. Cs. Unicam. It/lib/exe/fetch. Php*.

“History of ANPR - ANPR International.” n.d. Accessed September 2, 2020. http://www.anpr-international.com/history-of-anpr/.

Ioffe, Sergey, and Christian Szegedy. 2015. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” In *Proceedings of the 32nd International Conference on Machine Learning*, edited by Francis Bach and David Blei, 37:448–56. Proceedings of Machine Learning Research. Lille, France: PMLR.

Karpathy, Andrej, and Others. 2016. “Cs231n Convolutional Neural Networks for Visual Recognition.” *Neural Networks: The Official Journal of the International Neural Network Society* 1 (1). https://cs231n.github.io/convolutional-networks/.

Larhmam, Wikipedia contributors. 2020a. “Support Vector Machine.” Wikipedia, The Free Encyclopedia. September 2, 2020. https://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=976259237.

Liao, Ping-Sung, Tse-Sheng Chen, and Pau-Choo Chung. 2001. “A Fast Algorithm Form Multilevel Thresholding.” *JournalofInformation*.

MacQueen, James. 1967. “Some Methods for Classification and Analysis of Multivariate Observations.” In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–97. Oakland, CA, USA.

Oladeji, Femi. 2016. “License Plate Recognition For Vehicles.” https://github.com/femioladeji/License-Plate-Recognition-Nigerian-vehicles/.

———. 2017. “Developing a License Plate Recognition System with Machine Learning in Python.” https://blog.devcenter.co/developing-a-license-plate-recognition-system-with-machine-learning-in-python-787833569ccd.

Otsu, Nobuyuki. 1979. “A Threshold Selection Method from Gray-Level Histograms.” *IEEE Transactions on Systems, Man, and Cybernetics*. https://doi.org/10.1109/tsmc.1979.4310076.

Platt, John. 1999. “Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods.” *Advances in Large Margin Classifiers* 10 (3): 61–74.

Rojas, R. 1996. “Neural Networks: A Systematic Approach Springer-Verlag.” *Berlin, Deutschland*.

Rubinstein, Reuven Y., and Dirk P. Kroese. 2013. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer Science & Business Media.

Serra, Jean. 1984. *Image Analysis and Mathematical Morphology*.

“Seven Most Popular SVM Kernels.” 2020. December 17, 2020. https://dataaspirant.com/svm-kernels/.

Shiyu Ji, Wikipedia contributors. 2021a. “Kernel method.” Wikipedia, The Free Encyclopedia. April 13, 2021. https://en.wikipedia.org/wiki/Kernel_method

Sobel, Irwin. 2014. “History and Definition of the Sobel Operator. 2014.” *URL: Https://www. Researchgate. net/publication/239398674_An_Isotropic_3_3_Image_Gradient_Operator*.

Stokes, Michael, Mattew Anderson, Srinivasan Chandrasekar, and Ricardo Motta. 1996. “A Standard Default Color Space for the internet—sRGB, 1996.” *URL Http://www. w3. org/Graphics/Color/sRGB*.

Tch, Andrew. 2017. “The Mostly Complete Chart of Neural Networks, Explained.” *Online Medium*.

———. 2020b. “Softmax Function.” Wikipedia, The Free Encyclopedia. December 22, 2020. https://en.wikipedia.org/w/index.php?title=Softmax_function&oldid=995784016.

———. 2021a. “Automatic Number-Plate Recognition.” Wikipedia, The Free Encyclopedia. January 19, 2021. https://en.wikipedia.org/w/index.php?title=Automatic_number-plate_recognition&oldid=1001393833.

———. 2021b. “Scikit-Learn.” Wikipedia, The Free Encyclopedia. February 1, 2021. https://en.wikipedia.org/w/index.php?title=Scikit-learn&oldid=1004112336.