university of groningen / faculty of science and engineering

Vehicle License Plate Recognition using Pixel Information

Bachelor's Project Thesis

Sandra Bedrossian, s.bedrossian@student.rug.nl,
Supervisor: Hamidreza Kasaei, hamidreza.kasaei@rug.nl

Abstract: Law enforcement and public safety relating to the recognition of vehicles from image or video footage call for the efficient and rapid performance of Automatic License Plate Recognition (ALPR) systems. ALPR systems are devised by an integration of vehicle localisation, license plate detection, license plate segmentation and character recognition. This project primarily aims to extend the ALPR system devised by Kasaei et al. [1] to discover character recognition algorithms that yield a reliable accuracy in the recognition of Persian license plates. Specifically, the performance of template matching, Gaussian-weighted template matching, naive Bayes and a multilayer perceptron within the domain of character recognition are compared. Preliminary results while testing on a data-set of plain characters via 10-fold cross validation indicate the multilayer perceptron outperforms naive Bayes, Gaussian-weighted template matching and template matching, respectively. Further testing on a dataset of 500 highway images of vehicles indicate contradictory results with template matching marginally outperforming the multilayer perceptron, naive Bayes and Gaussian-weighted template matching respectively. The differences in accuracies are however generally close with large standard deviations and should not be exaggerated; further preprocessing of the segmented characters prior to recognition should steer results to those more representative of the preliminary results on the characters dataset.

1 Introduction

The recognition of vehicles from image or video footage is vital in fields of law enforcement and public safety. Such recognition is reliant on image processing and Automatic License Plate Recognition (ALPR) systems. Automatic identification of vehicles violating speed limits or tolls with pay-per-use roads are some applications in which ALPR systems lend themselves.

These systems are derived from intelligent transportation systems (ITS) which seek to optimally integrate technology, information and communication in transportation in order to provide transportation management systems.

Within ALPR lies a dependence on computer vision and character recognition algorithms to facilitate the automatic identification of the characters within a license plate, as identified through the image of a vehicle.

The shortcomings encountered by ALPR systems are attributed to the variety of noisy conditions in which the images of the vehicles are captured. Essentially, research in the field comes to a common consensus that variations in lighting as well as objects and shadows in the background complicates the ease at which license plates are identified. The lack of consistency in license plates among nations lead to difficulties in building a single, adaptable system to recognise any, arbitrary license plate: the font, characters, colours and sizes of plates may vary [2].

Consequently, we seek to recognise and tackle some of the issues encountered in ALPR systems. We begin with the system proposed by Kasaei et al. [1] on the detection and recognition of Persian license plates and work to heighten its accuracy and robustness. In particular, we will be working on character recognition.

Kasaei et al. [1] propose the following method of ALPR on still highway images of Persian license plates:

1.1 Approach to vehicle detection

In order to isolate the vehicle from its background, the Sobel edge detector is initially used to identify the lines making up the vehicle. These are thickened through the image dilation operator and the region is filled. This results in an isolated vehicle that we can separate from the background via a bounding box.

1.2 Approach to license plate detection

Detecting the region containing the license plate involves passing a sliding window across the bounding box previously identified. The sliding window technique takes a region of a particular ratio and rolls across and down the image, scanning for features that are specific to license plates. These features are identified through horizontal feature detectors. Horizontal feature detectors search for 2 lines marking the sides of the license plate to ascertain the presence of the license plate.

1.3 Approach to license plate segmentation

The identified license plate region is preprocessed in order to correct for rotations. Binarization ensures that the characters are separated from the plate. The separation between characters as well as the heuristic that the following sequence: *2 numerals + 1 letter + 5 numerals* make up all Persian license plates is used for segmentation.

1.4 Approach to character recognition

Finally, template matching is employed to recognise the segmented and separated characters. Template matching entails the construction of a set of templates for every character class and the calculation of the Euclidean distance from each example to each character. The class to which this distance is minimized is selected as the most probable outcome for that example.

The steps in the implementation of the Kasaei et al. [1] ALPR system are depicted in Figure 1.1.

These can be cross-referenced with Figure 1.2, showing the interface of an interaction with the system. The top left corner demonstrates the steps in finding the bounding box that contains the vehicle body; the bottom left corner demonstrates the sliding window technique that localizes the license plate region and the window on the right demonstrates the processes involved in the segmentation of the license plate and consequent identification of the characters through character recognition.

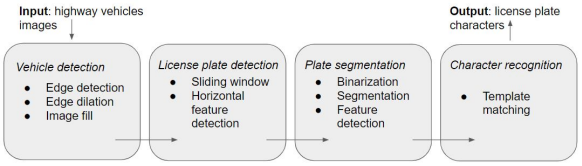

Figure 1.1: Stages in ALPR of the Kasaei et al. [1] system.

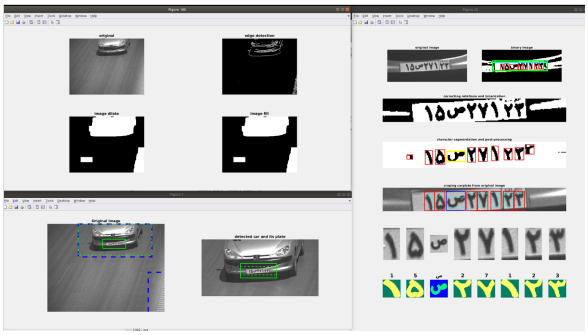

Figure 1.2: The stages and results of one iteration of the Kasaei et al. [1] ALPR system.

1.5 Our research

We propose a robust, inexpensive approach to the identification of Persian license plates from a highway dataset of 500 images of vehicles. In particular, we experimented with the ALPR system devised by Kasaei et al. [1] by focus on character recognition.

A comparison of template matching, Gaussian-weighted template matching, naive Bayes and the multilayer perceptron is made to ascertain the approach that yields most favourable character recognition.

Consequently, we are led to answer the following research questions:

- Which combination of image processing tech-

niques or character recognition algorithms make a reliable vehicle license plate detection and recognition system?

- What are the main limits and factors involved in image-processing based vehicle license plate detection and recognition?

The approaches to character recognition that we will be using are listed below and will be explained in greater depth in Section 3:

- **Template matching:** This is the benchmark approach implemented in the Kasaei et al. [1] system. We will be comparing performance of subsequent approaches to this.
- **Gaussian-weighted template matching**
- **Naive Bayes**
- **Multilayer perceptron**

The reliability of an ALPR system is typically judged on the basis of vehicle detection, license plate extraction and recognition of characters. This allows a standardised metric of comparison to existing ALPR systems. In this research, we focus on character recognition accuracy, which is computed as:

$$\textbf{accuracy} = \frac{\#\text{correctly recognised characters}}{\text{total number of characters}}$$

2 Related Work

In a literature review of existing ALPR systems, Du et al. [2] assess some of the methods used in the extraction of a license plate, its segmentation into the components that contain characters and the recognition of these characters to identify the vehicle. Fundamentally, they identify the process of ALPR to be divided into the following four stages:

- Image acquisition
- License plate extraction
- License plate segmentation
- Character recognition

2.1 Research on vehicle detection

As is demonstrated by the four stages in ALPR identified by Du et al. [2], research in the field commonly glosses over the step we provide between image acquisition and license plate extraction - vehicle detection. While a majority of the approaches go straight to license plate extraction, Kasaei et al. [1][3] provide an intermediary step where the region in which to locate license plate is reduced to the region in which the vehicle probably occurs.

An approach to vehicle detection using edge detection is recommended by Wang [4]. Wang [4] uses the Sobel operator after image pre-processing in order to detect vehicle edges. Wang [4] identifies that use of the Sobel operator is susceptible to weakness with noisy input.

Similarly, Ajmal & Hussain [5] utilise the Sobel operator to detect the edges of vehicles, then perform further processing through the dilation and fill operator to identify the area in which the vehicle lies. The dilation operator is used to thicken the edges; this operator also ensures the connectivity of broken edges, while the fill operator is used to encapsulate each detected vehicle. Ajmal & Hussain [5] criticise the dilation operator in edge detection in that it may mistakenly fuse multiple vehicles that lie in close proximity.

2.2 Research on license plate detection

As mentioned earlier, some literature skip the preliminary process of vehicle detection and immediately attempt to detect the license plate. Yu & Deak Kim [6] and Sarfraz et al. [7] both take this approach and rely on the heuristic that the image of a vehicle contains more horizontal than vertical edges. Thus, their approaches utilise edge detectors in the vertical direction in order to detect the license plate. The rectangular license plates are detected on the basis of 2 vertical edges. Yu & Deak Kim [6] recognise the robustness of their approach as vertical edge detection is resilient to skewed license plates due to the angle the photo is taken at.

Following vertical edge detection, Sarfraz et al. [7] filter out unwanted regions before fitting an aspect height to width ratio to candidate regions. The region that satisfies this ratio is deemed to be

the license plate. License plate extraction yielded a 96.22% accuracy, with some difficulties in ambiguous detection or edge extraction.

Safaei et al. [8] examine the arguments Du et al.[2] make for license plate localisation on the basis of contextual features and in turn propose a method of ALPR that makes use of temporal information. They argue that failing to take into account the motion of the vehicle can lead to shortcomings as irrelevant objects in the background are picked up instead of the license plate.

The approach taken by Safaei et al. [8] in turn examines multiple frames of images in which the vehicle appears as extracted via video footage. These are integrated in order to remove the shadows of background objects and streamline attention to the relevant license plate. The method utilised by Safaei et al. [8] falls short with a high density of salt-and-pepper noise mimicking the effects of rain or snow.

2.3 Research on license plate segmentation

License plate segmentation may rely on background knowledge on the composition of license plates. Busch et al. [9] use the Sobel operator and the characteristic that license plates commonly consist of black characters on white background. Horizontal detectors determine the upper and lower boundaries of the characters in the license plate through detection of consecutive white pixels followed by white and black values. Similarly, they determine the height of the characters via vertical detectors.

As opposed to their knowledge-based implementation, Busch et al. [9] realise the potential of artificial neural networks as another means of performing the segmentation task, though they argue for the former approach due to its applicability to their research.

Gao et al. [10] also take into consideration the composition of license plates: they make use of the fact that Chinese license plate consist of 7 characters. After vertical projection of the binarised input image, the number of segments are compared to 7; if there are more than 7 segments, those with the smallest distance to their neighbour are merged.

2.4 Research on character recognition

Sarfraz et al. [7] employ the simple yet reliable strategy of template matching after normalising the segmented blocks of characters they obtain from segmentation of the license plate. The Hamming distance is used to compare each character to the template. This method is vulnerable to font changes and requires processing of irrelevant pixels [2].

An alternate method extracts features then uses a classifier. Kim et al. [11] utilise 4 support vector machines (SVM) on extracted characters in Korean license plates. These license plates have characters occurring in 2 rows; 2 SVMs classify the numerals and letters in the upper row, while the other 2 classify the numerals and letters in the lower row.

Masood et al. [12] take a novel approach in that they train 3 deep convolutional neural networks (CNN) - one for each stage of ALPR: license plate detection, character detection and character recognition. Characters are confirmed by comparing to license plate segments (positives) and background or symbols (negatives). After characters are confirmed, a deep CNN is trained with 35 characters including the alphabet (excluding O) and numerals 0 to 9.

3 Methods

In order to allow for ease of integration and comparability to the current system by Kasaei et al. [1], the character recognition algorithms we will be experimenting with are also implemented on MATLAB. To recap, the approaches we are experimenting with are template matching, Gaussian-weighted template matching, naive Bayes and the multilayer perceptron.

These approaches are trained and tested for the Farsi numbers 1 through 9 and the select group of 15 Farsi letters that are used on Persian license plates. This combination of characters matches those we expect on the license plates. The 24 classes of characters are depicted in Figure 3.1. We execute each approach on 2 sets of data: a preliminary, generated characters dataset and the final highway vehicles dataset.

The motivation for testing on a preliminary char-

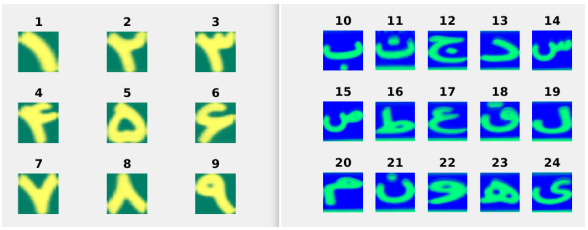

Figure 3.1: The 24 classes of characters we expect on the license plates.

acters dataset before application to the highway vehicles dataset is to be able to ascertain the performance of the algorithms on different datasets. Any difference in the performance of the algorithms while testing on each dataset can be attributed not to the algorithm in question but to the use and manipulation of the dataset. The characters dataset is more 'clean' (with less noise) and thus should show greater accuracy. We can then encounter any limitations to each approach by first examining performance on the characters dataset and then adjusting testing on the highway vehicles dataset accordingly.

3.1 Preprocessing

The character images we use are of pixel size 96×96 for template matching, Gaussian-weighted template matching and naive Bayes. For the multilayer perceptron, we use images of size 15×15 for the characters dataset or 18×18 for the highway vehicles dataset.

The downsizing of image for the multilayer perceptron is due to memory constraints. The slightly larger size used and testing on the highway vehicles dataset versus the characters dataset is selected in a bid to potentially improve accuracy by providing as much pixel information as possible.

Prior to use with each dataset, the images are thus resized, binarized and any interfering pixel information such as a horizontal line at the bottom of the image obtained from a portion of the license plate is removed. The image is then set to fit as tight to the borders as possible. The preprocessing process is visually depicted in Figure 3.2.

3.2 Template matching

Template matching is the original algorithm implemented in the ALPR system by Kasaei et al. [1]. This method of character recognition is commonly used for its simplicity and ease to implement [2].

Training template matching entails reading in each binarized image partitioned into the training data-set pile and summing and normalizing over each pixel in a given position to find the most representative 'template' for a given class. To test template matching, we compute the distance between each 'template' image for a class and our current test image. This is done via calculating the sum of squared errors, as shown in Equation 3.1:

$$D(T, I) = \sum_{i=1}^{96} \sum_{j=1}^{96} (T_{(i,j)} - I_{(i,j)})^2 \qquad (3.1)$$

where $T_{i,j}$ denotes the pixel in position i, j of the template and $I_{i,j}$ denotes the pixel in position i, j of the image.

Once this distance to each class's template image has been computed, the class representing the template image yielding the minimum distance to our current training image is selected as the class in which to classify the current image.

3.3 Gaussian-weighted template matching

In order to increase the accuracy of template matching, in accordance to the findings discovered by Min Wong [13], we perform variants of template matching where the sum of squared errors is multiplied by certain weights. Thus, the Equation in 3.1 becomes the Equation in 3.2:

$$D(T, I) = \sum_{i=1}^{96} \sum_{j=1}^{96} W_{(i,j)} (T_{(i,j)} - I_{(i,j)})^2 \qquad (3.2)$$

(a) (b) (c)

Figure 3.2: Prepossessing steps: (a) the original image, (b) the cropped and binarized image, (c) the image isolated in a bounding box with bordering information removed.

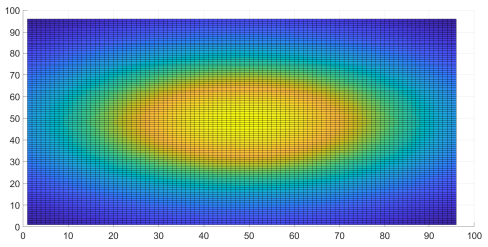

Figure 3.3: Ellipse-shaped Gaussian weight map with the pixels shaded in warmer colors demonstrating regions of higher weight, and those in cooler colors demonstrating regions of lower weight.

where $W_{i,j}$ denotes the weight of the pixel in position i, j of the weight map.

The weights are determined in accordance to the ellipse-shaped Gaussian distribution. The function mapping the ellipse-shaped Gaussian distribution is shown in Equation 3.3:

$$f(x,y) = e^{-\frac{(i-i_0)^2}{2\sigma_i^2} + \frac{(j-j_0)^2}{2\sigma_j^2}} \tag{3.3}$$

where i_0 denotes the position of the center pixel in the x-direction (=46), j_0 stands for the position of the center pixel in the y-direction (=46), σ_i paramater is selected as 26 and σ_j paramater is selected as 35.

A depiction of the Gaussian weight map is demonstrated in Figure 3.3 and its result on an arbitrary character in Figure 3.4; it is evident that the largest weights are given to the central pixels of the character.

The parameters in Equation 3.3 are selected after optimization through trial and error, where σ_j is larger than σ_i so that the weighting is such that the pixels in the central vertical direction and not horizontal, carry the most weight.

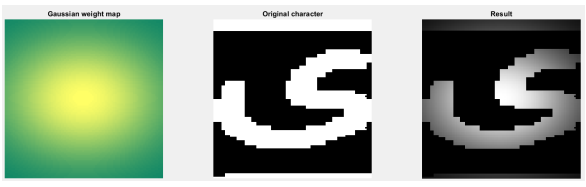

Figure 3.4: Ellipse-shaped Gaussian weight map, the original character (class 21 in Figure 3.1), and the result to the character when multiplying by the weight map.

The motivation for use of Gaussian-weighted template matching lies in that there is a tendency for noise at the top and bottom borders of each character (see class 11 for instance, of Figure 3.1). This is where the license plate meets the body of the car. The weight distribution with the above parameters seeks to eliminate this. Note that this window is also dependant on the region in which the character lies within the dimensions of its bounds.

3.4 Naive Bayes

Another means of a fast and easy-to-implement classification algorithm is through naive Bayes. This approach seeks the most probable class based on pixel values. In order to calculate class conditional likelihoods, we begin by counting a prioris and pixel conditional likelihoods. A priori class probabilities, $P(C_i)$ are calculated as in 3.4:

$$P(C_i) = \frac{\text{number of images in class } i}{\text{number of images in all classes}} \tag{3.4}$$

where C_i denotes i^{th} class where $i \in \{1, ..., 24\}$. Note that we assume a prioiri class probabilities to be the same for all C_i since each character has an equal likelihood of occurrence. Pixel conditional likelihoods, $P(p_j|C_i)$ are calculated as in 3.5:

$$P(p_j|C_i) = \frac{\text{number of times } p_j = 1 \text{ in } C_i}{\text{number of pixels} = 1 \text{ in } C_i} \tag{3.5}$$

where p_j denotes pixel j where j $\in \{1, ..., 96 \times 96\}$. If the above evaluates to zero, we replace the probability with a very small number defined by epsilon:

$$\epsilon = \frac{1}{\text{number of pixels} = 1 \text{ along all classes}} \tag{3.6}$$

In order to obtain suitably sized numbers, we apply the log for each of a priori and class conditionals obtained above. Once we have trained the classifier, the testing entails evaluating the sum in 3.7 for ev-

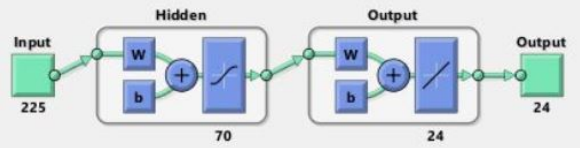

Figure 3.5: Neural network architecture with 225 nodes in the input layer representing a 15×15 pixel image, 1 hidden layer of nodes in the range of 30 to 70 in increments of 5 and an output layer of 24 nodes (classes). The hidden layer is connected to the output layer via the tan-sigmoid transfer function while the output layer is connected through a linear transfer function.

ery class:

$$logP(C_i|I) = logP(C_i) + \sum_{j=1}^{96*96} I_j \cdot log(P(p_j|C_i)) \tag{3.7}$$

where I stands for the current image and I_j denotes the j^{th} pixel of the current image. Once we have obtained such a sum for every class, we seek the largest; the class which finds this maximal sum is found to be the most probable class for the given image.

3.5 Multilayer perceptron

We design a 2-layer neural network with an input layer describing a pixel-by-pixel representation of the characters, a hidden layer with a varying number of hidden neurons and an output layer with 24 nodes attributed to the 24 classes.

Due to memory constraints, the image size is downsized from 96×96 pixel to 15×15 pixel in the implementation of the multilayer perceptron for the characters dataset, and 18×18 for the highway vehicles dataset. Thus, if we consider the architecture of the MLP for the characters dataset, we recognize an input layer of 225 input nodes. The hidden layer neurons are varied from 30 to 70 in increments of 5.

The output layer takes its input through the tan-sigmoid transfer function from the hidden layer while the final output takes its input from the linear transfer function from the output layer. The neural network is depicted in Figure 3.5.

The network is trained using Levenberg-Marquardt backpropagation due to its efficiency on MATLAB over other algorithms [14]. The network performance is evaluated via the mean squared error and weight and bias values. The performance goal is set to 0.003 after experimentation for faster convergence. The network is evaluated on its performance via an identity target matrix.

4 Experimental results

In order to gauge the adequate character recognition algorithm to use in the license plate detection and recognition system, we began with a comparison of the performances of template matching, Gaussian-weighted template matching, naive Bayes and a multilayer perceptron using the classes in Figure 3.1. The dataset to be used for training and testing simply consisted of generated instances of each class.

4.1 Results with the characters dataset

As explained in Section 3, we are to train and test on 24 classes. Each class of the 24 contain 100 instances of the relevant character. In order to be able to make a fair judgement, we utilise K-fold cross validation for a less biased prediction of which algorithm performs better*. We implement K-fold cross validation with K=10 in the following manner:

We begin by partitioning the data into K=10 folds; this value is a standard within literature. This partitioning is done per class, so we obtain 10 folds for each class with each fold containing 10 images. After obtaining this structure, in each iteration through K we select 1 fold (of 10 images per 24 classes) for testing and the remaining 9 folds (each of 10 images per 24 classes) for training. Training and testing is done on each of the algorithms explained in Sections 3.2, 3.3, 3.4 and 3.5.

In order to gauge preliminary performance of the character recognition algorithms, we computed a mean global accuracy over K=10 folds. To assist interpretation of the results, we will examine confusion matrices showing the classifications according to each character recognition algorithm. Note that in order to be able to obtain deterministic and comparable accuracies, the data has been partitioned

*https://machinelearningmastery.com/k-fold-cross-validation/

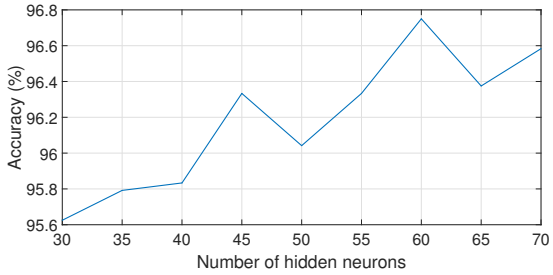

Figure 4.1: Variation of hidden neuron nodes with accuracy of the multilayer perceptron.

into the exact same folds with each algorithm in accordance to K-fold cross validation.

4.1.1 Accuracies of the four algorithms

Before comparing the accuracies of the four algorithms, we will discuss the paramaters that we have tuned to get these results. In particular, we will focus here on selection of the multilayer perceptron's number of hidden nodes.

In order to determine which number of hidden layer neurons yield the best performance, we experiment with numbers between 30 to 70 in increments of 5. This range is chosen as it is between the number of nodes in the input layer (15×15) and output layer (24), but not too large that it causes poor generalisation. The results of this experiment is demonstrated in Figure 4.1. Our network trained with 60 hidden neurons yields the highest accuracy, therefore this configuration is chosen when assessing the accuracy of the multilayer perceptron in comparison to the other approaches.

We can now compare the accuracies of the four approaches. Table 4.1 demonstrates the multilayer perceptron outperformed naive Bayes, template matching and Gaussian-weighted template matching, obtaining a higher accuracy in the classification of the 100 images per class. The same is reflected in the confusion charts in Figures 4.2 to 4.5.

Table 4.1: Accuracy and training time of different character recognition approaches testing on the characters dataset

Approaches	Accuracy(%)	Training Time(s)
Template matching	83.5 ± 2.3	6.8 ± 0.3
Naive Bayes	94.1 ± 4.6	8.3 ± 1.1
Gaussian-weighted template matching	85.0 ± 4.3	6.8 ± 0.3
Multilayer perceptron	96.8 ± 1.7	2961.0 ± 285.2

Table 4.1 demonstrates that the standard deviation for naive Bayes is larger than that for any of the other algorithms while the multilayer perceptron has the smallest standard deviation. The multilayer perceptron is thus the approach showing most consistent results.

Note also that Gaussian-weighted template matching did perform slightly better than unweighted template matching, yet at the expense of a comparatively much larger standard deviation.

A closer look at Figure 4.2 demonstrates that the classes suffering the most in performance (more than half of the images missclassified) are '2' '4' and '15'. Class '2' was more often classified as '3' according to unweighted template matching, class '4' was also more often classified as a '3', while class '15' was more often classified as '14'.

Referencing Figure 3.1 for each of these cases, the similarity between mistaken characters is readily

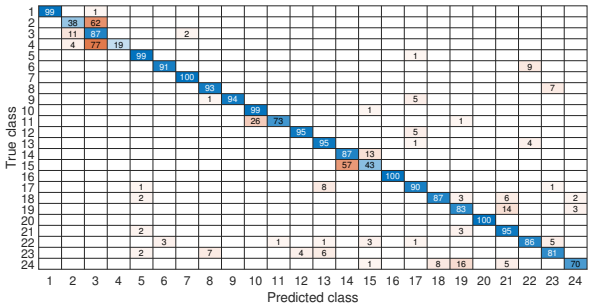

Figure 4.2: Confusion chart using 10-fold cross validation on template matching, where classes 1 to 24 represent the characters depicted in Figure 3.1 respectively.

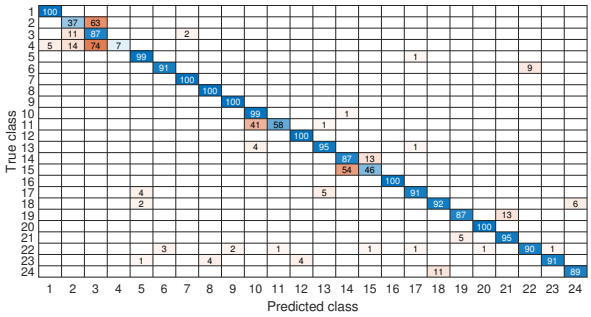

Figure 4.3: Confusion chart using 10-fold cross validation on Gaussian-weighted template matching, where classes 1 to 24 represent the characters depicted in Figure 3.1 respectively.

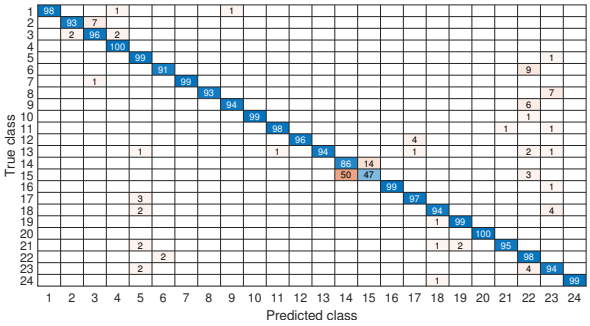

Figure 4.4: Confusion chart using 10-fold cross validation on naive Bayes, where classes 1 to 24 represent the characters depicted in Figure 3.1 respectively.

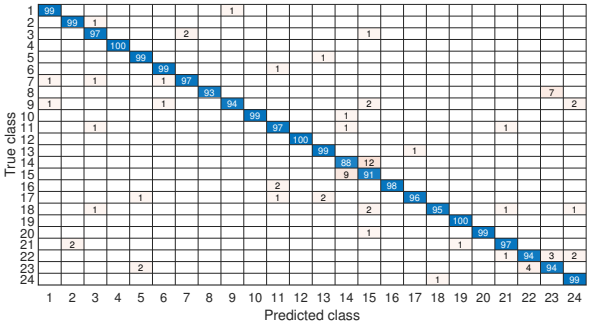

Figure 4.5: Confusion chart using 10-fold cross validation on the multilayer perceptron, where classes 1 to 24 represent the characters depicted in Figure 3.1 respectively.

noticeable. In contrast, a closer look at Figure 4.5 demonstrates that the multilayer perceptron was the only approach out of the 4 that did not suffer with character similarity.

In Table 4.1, we can see the mean training times and respective standard deviations for one iteration (over K) for each approach. It is evident that the training time using naive Bayes is greater than the training time using template matching. In this run, the standard deviation for the training time obtained using naive Bayes is greater than that obtained using template matching. Note that training template matching with and without weight maps is the same process and thus the training times are equal.

Though the accuracy of naive Bayes is not too far off the multilayer perceptron when comparing the standard deviations, there is a massive tradeoff in training time. The multilayer perceptron's training time greatly exceeds that of every other approach we experimented with.

4.1.2 Computational complexity

According to the results in Table 4.1, it is evident that naive Bayes, being a model-based learning approach has a higher time complexity than template matching, being an instance-based learning approach. This is apparent in that the latter compares to a model while the former compares to every instance of training objects - this leads to the greater training time for naive Bayes over template matching witnessed in the aforementioned table.

On the other hand, instance-based learning approaches are more memory-intensive in that they need to store the model category for each class. Due to the number of connections and the sizes of each layer, we additionally were held back by the time and memory constraints of the multilayer perceptron training on our system.

4.2 Results with the highway vehicles dataset

After testing on the pure characters dataset, we test on the input to the ALPR system: namely, segmented characters obtained from highway images of vehicles. For template matching, Gaussian-weighted template matching and naive Bayes training is done on every image in the characters dataset. The number of images per class varies from as low

Table 4.2: Accuracy against hidden nodes for the multilayer perceptron on numbers

Number of hidden nodes	Accuracy(%)
20	84.2 ± 4.1
40	86.1 ± 1.0
60	86.2 ± 0.6

Table 4.3: Accuracy against hidden nodes for the multilayer perceptron on letters

Number of hidden nodes	Accuracy(%)
20	76.9 ± 6.6
25	78.9 ± 6.0
30	80.5 ± 3.5

as 399 to as high as 2818. Due to memory constraints, the multilayer perceptron was limited to training on a maximum of 414 images per class. Class size was kept constant to eliminate bias.

Then, the ALPR system as devised by Kasaei et al. [1] is responsible for reading 500 images of highway vehicles and executing vehicle detection, license plate localization, segmentation and finally character recognition via the algorithms previously mentioned.

In effort to obtain a better accuracy when testing with the highway vehicles dataset than on the characters dataset, 2 seperate naive Bayes classifiers and 2 seperate multilayer perceptrons were designed. Thus we could delegate 1 of each to recognition of numbers and 1 to recognition of letters. This is different to the implementation with the characters dataset where only 1 classifier/ perceptron was used for both letters and numbers.

To discover the suitable number of hidden nodes in the hidden layers for the multilayer perceptrons on numbers and letters, we carried out another set of experiments. The results are shown in Tables 4.2 and 4.3. These results take the average of 10 runs for each setting.

Results showed peak performance with 60 hidden nodes for the multilayer perceptron on numbers and 30 for the multilayer perceptron on letters. We note that the standard deviations also decrease as we go from 20 to 60 hidden neurons for the multilayer perceptron on numbers, as do the standard deviations for the hidden neurons for the multilayer perceptron on letters as we go from 20 to 30. The results we use for accuracy of the multilayer perceptron are thus produced via having 60 hidden nodes for the multilayer perceptron on numbers and 30 for the multilayer perceptron on letters.

Table 4.4: Accuracy of different character recognition approaches testing on the highway vehicles dataset

Approaches	Accuracy(%)
Template matching	86.1 ± 8.5
Naive Bayes	85.3 ± 8.9
Gaussian-weighted template matching	84.3 ± 9.2
Multilayer perceptron	85.5 ± 7.5

4.2.1 Accuracies of the four algorithms

As the final accuracy of the system is dependant on the performance of vehicle detection, license plate extraction and license plate segmentation, we discovered that character recognition on our chosen test set of 500 images was greatly affected by performance of the previous stages.

To correct for this, we used the results of template matching as a benchmark and removed all images where recognition of greater than half the characters out of 8 were incorrectly identified. We removed the exact same images from the results of the other 3 algorithms. Thus, the accuracies demonstrated in Table 4.4 are the character recognition accuracies of the 364 images pruned of 500.

These results conclude template matching outperforms the multilayer perceptron, followed by naive Bayes and then Gaussian-weighted template matching. This is odd as we expect a similar performance with each algorithm as we got with the characters dataset. However we also recognize that

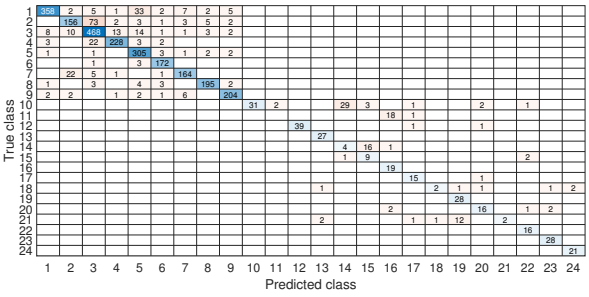

Figure 4.6: Confusion matrix of template matching performance on 364 highway images of vehicles.

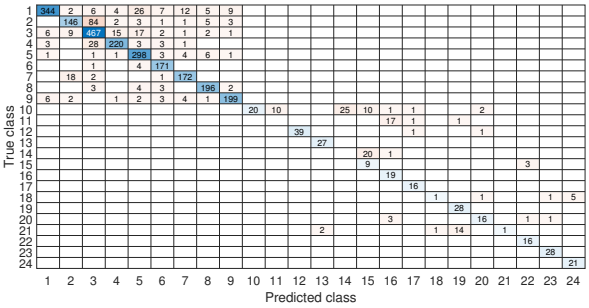

Figure 4.7: Confusion matrix of Gaussian-weighted template matching performance on 364 highway images of vehicles.

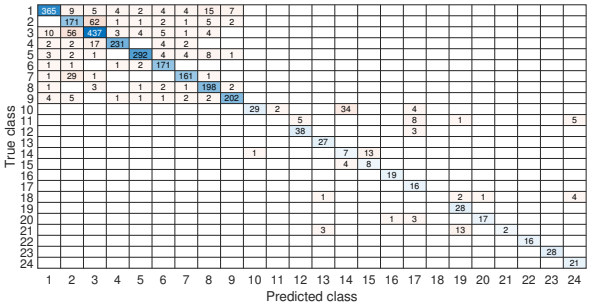

Figure 4.8: Confusion matrix of naive Bayes performance on 364 highway images of vehicles.

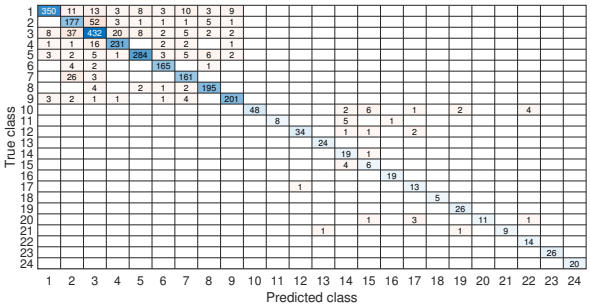

Figure 4.9: Confusion matrix of multilayer perceptron performance on 364 highway images of vehicles.

the standard deviations are large for each approach so the performance of each algorithm is close.

To determine where each algorithm goes wrong, examine the confusion matrices reflecting the accuracies in Figures 4.6 to 4.9. The confusion matrix in Figure 4.9 shows the truncated average of 10 runs while all other matrices show the results of 1 run as the results for these approaches are deterministic. The confusion matrices in Figures 4.6 to 4.8 resemble one another in that they all misclassify every instance of character 11. The confusion matrix for the multilayer perceptron in Figure 4.9 is distinct from those of the other approaches in that it is the only one that can identify at least a few instances from every class.

4.2.2 License plate recognition accuracies

In order to assess the accuracy of license plate recognition, we test on 500 highway vehicle images and determine the number of images for which the whole plate of 8 characters was recognized correctly. These results and the results for 1, 2 and 3 errors per plate are depicted in Figure 4.10.

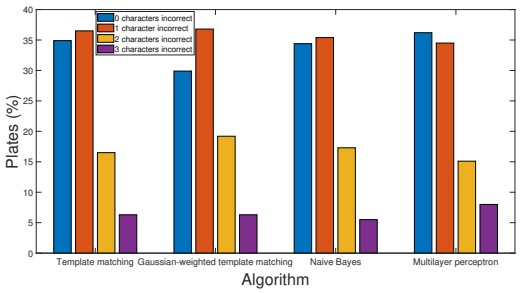

Figure 4.10: Percentage of plates that were recognized correctly or with 1, 2 or 3 errors per plate.

Greater than 50% of plates with template matching, Gaussian-weighted template matching and naive Bayes are either recognised completely correctly, or 1 character of the 8 is mistaken. The same is the case for the multilayer perceptron, however with this approach more plates are correctly recognised than plates that are classified correctly spare 1 mistake. With the former 3 algorithms, more plates occur with 1 error than fully correct plates.

4.2.3 Variations

The confusion matrices obtained by testing on the characters dataset in Figures 4.2 to 4.4 demonstrated that character similarity caused a decrease in accuracy. To fix for this, the shape of the relevant characters were examined. Since characters are made up of a distinct number of successive strokes, a function counting local maxima was utilized to differentiate classes if the character class was deemed ambiguous.

Examining the confusion matrices in Figures 4.6 to 4.8, we realised that character 11 was never classified correctly by either of template matching, Gaussian-weighted template matching and naive Bayes when testing on the highway vehicles dataset. This led to the realisation that the character dataset of class 11 was not representative. The character 11 always occurs in Persian license plates with 'TAXI' written above it while in the characters dataset, this was not the case.

Thus, we generated more representative training data by concatenating together 'TAXI' as

Figure 4.11: Generated examples of character 11.

cropped from images of Persian license plates together with the character 11 stencils in our characters dataset. For demonstrative purposes, a sample of the generated characters are demonstrated in Figure 4.11 and the new template generated by template matching using this modified dataset is demonstrated among the series of letter templates in Figure 4.12.

Next, these 3 algorithms were retrained on this class. (The multilayer perceptron was implemented after this discovery, so its accuracy in Table 4.4 is obtained via training on the modified class for character 11).

As well as the above, the peak-finding function was used to distinguish between similar characters. This was used on characters 2 and 3 as well as on characters 14 and 15. After experimentation, the best results were found through modifying class 11 and implementation of the peak-finding function whenever we encountered characters 14 and 15. These results are shown in Table 4.5.

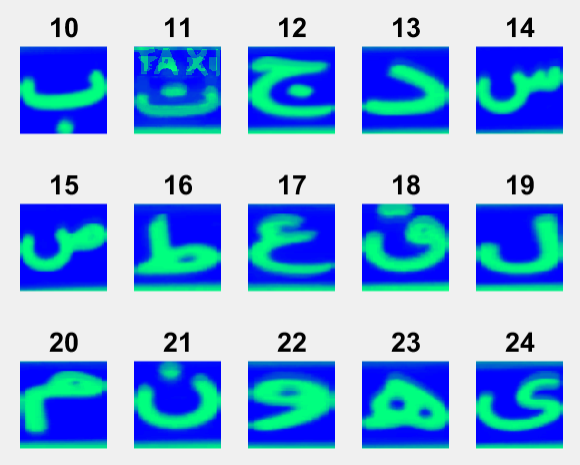

Figure 4.12: Letter templates with modified training examples in class 11.

Table 4.5: Accuracy of different character recognition approaches with modified class 11 and class 14/15 distinction function

Approaches	Accuracy(%)
Template matching	87.2 ± 6.5
Naive Bayes	85.9 ± 7.8
Gaussian-weighted template matching	85.7 ± 6.5

When comparing to Table 4.4, we see the same trend with the mean accuracy for template matching higher than naive Bayes, which is higher than Gaussian-weighted template matching. The large standard deviations demonstrate that this comparison is not too noteworthy. However, the performance for each algorithm after the modifications have indeed improved.

Figure 4.13 demonstrates the recognition with each algorithm for 4 arbitrary highway vehicle images. This figure demonstrates the detected license plate region and how each algorithm interprets the characters. Note that these accuracies are those obtained after the heuristics previously mentioned are applied.

5 Conclusion

Our research question was twofold: we sought an improvment in accuracy of the Kasaei & Kasaei [1] by focus on character recognition and we also looked for limitations along the way. Section 5.1 will examine the first research question while Section 5.2 will examine the second. Finally, any points we raise contribute to opportunities for further research. We discuss these in section 5.3.

5.1 Research question 1: exploring different approaches to character recognition

To recap, our first research question sought a character recognition approach that could improve on the accuracy of the Kasaei et al. [1] system. The Kasaei et al. [1] ALPR used template matching while we experimented with Gaussian-weighted template matching, naive Bayes and the multilayer perceptron.

Looking at the preliminary accuracies in Table 4.1 as trained and tested on the characters

Figure 4.13: 4 highway vehicles and the output from each of the 4 character recognition approaches we use: template matching (TM), Gaussian-weighted template matching (Gaus-TM), naive Bayes (NB) and the multilayer perceptron (MLP). The accuracy per plate is given in brackets.

dataset, results were promising. Gaussian-weighted template matching did prove to be an optimisation of the original template matching algorithm as it placed weight on the central sections of the character. Further so, both naive Bayes and the multilayer perceptron also showed great improvements to the accuracy.

Unfortunately, once these algorithms were asked to generalise from the training dataset to the license plate dataset, we saw much different results. The original template matching algorithm outperformed each of the 3 approaches we tried. Once we tweaked the algorithms through heuristics for similar characters and changing up the training set to be more representative (as with character 11) we got better results. However, template matching still performed better.

We must not overstate the performance of template matching though, because examination of Table 4.4 shows large standard deviations for each approach and the accuracies were quite close. Furthermore, Figure 4.10 demonstrates the potential of the multilayer perceptron in that it is the only approach of the 4 that more often gets all 8 characters of a plate correct than gets 7 out of the 8.

The multilayer perceptron also got the smallest standard deviation with testing on both datasets so could be deemed the most consistent approach. Furthermore, it did not stumble with similar characters unlike the other 3 approaches.

The reasons we do not get the results we expect can be boiled down to training on a different dataset than that which we are testing on. It seems the license plates we test on require more preprocessing than expected in order to have them resemble the training data.

5.2 Research question 2: identifying limitations when experimenting with different approaches to characters

To answer our second research question concerning the limitations encountered in the design of the ALPR with regard to character recognition, one of the problems were the low contrast instances of images. Such license plates with dark shadows led to difficulties in identifying the outline of the character. This can be seen in Figure 5.1. It is easy to see why similar characters would be mistaken for one another.

Another limitation concerns the available data; as we did not have labeled segmented data, we could not isolate the performance of character recognition from the performance of previous stages in the ALPR. Our method of pruning out plates with accuracy less than half could have also got rid of characters that had no issues with segmentation. This is a more practical limitation of the project, yet it could have affected the accuracy.

Though we were familiar with the issues of character similarity affecting accuracy prior to implementation, our approach of finding character peaks was effective, but not as much as we would have

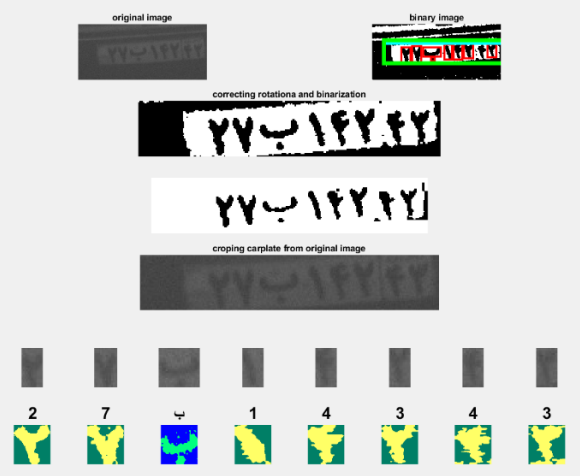

Figure 5.1: Results of low contrast license plate character recognition.

liked. This method did not work effectively for characters 2 and 3; this could be attributed to the preprocessing stage which nips off bordering information. This method sometimes chopped off the top of the character - this is detrimental to characters 2 and 3 since their distinguishing traits lie in the top portion of the character.

5.3 Further work

When comparing the accuracies obtained using the characters dataset to the accuracies obtained using the highway vehicles dataset, we realise that the training data might be a bit too 'clean' and thus less accurately represents the license plate data. To this end, further research should extract all the characters after the segmentation by the ALPR and train and test on the very same dataset. Because we did not have labeled segmented data, we could not use the same dataset from the ALPR to train and test. Ideally, this would be available.

Another approach would be to perform further data augmentation on the characters dataset. These characters have been generated with some noise, but perhaps we anticipated less than we encountered with the highway vehicles dataset. Slight rotations to the characters and adding in various random spots of noise should improve the robustness of the approach. Additionally, we could perform further preprocessing before recognition. The morphological erosion operator could thin down wide characters such as those we see in Figure 5.1.

An extension to this project could explore a multilayer perceptron with multiple hidden layers. If we were to extend the number of hidden layers from 1 to 2, we could perhaps beneficially break up the features making up a character more effectively. One could also experiment with a greater number of hidden nodes when testing on the vehicles dataset as this paper only explores 3 different settings.

In terms of algorithms used, we encountered difficulties in the resilience of our approaches. Pixel information is strongly affected by the centering of the character and because we do some preprocessing such as removing bordering information, we could offset the centering of the character. This all will affect the accuracy of the system. Future work should seek to explore local features that are pixel-position-independant.

Furthermore, the naive Bayes assumption of pixels being independant of their neighbours is probably violated as characters are fluid shapes that spill into neighboring pixels. Support vector machines look at interactions between pixels so may be more suited for character recognition.

Should further training data be generated, convolutional networks (CNN) like LeNet [15] or VGG16 [16] may be more effective in producing high accuracies. Vašěk, Franc & Urban [16] use video footage with a CNN and aggregate the results of a sequence of stills - a similar approach could reduce the probability of encountering the sort of problems we witness in Figure 4.10 where only 1 character of 8 is misrepresented.

References

[1] S.H. Kasaei and S.M. Kasaei, "Extraction and recognition of the vehicle license plate for passing under outside environment," in *2011 European Intelligence and Security Informatics Conference*, Sep. 2011, pp. 234–237.

[2] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (alpr): A state-of-the-art review," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 2, pp. 311–325, Feb 2013.

[3] S.H. Kasaei, S.M. Kasaei, and S. Monadjemi, "A novel morphological method for detection and recognition of vehicle license plates," *American Journal of Applied Sciences*, vol. 6, no. 12, pp. 2066–2070, 2009.

[4] W. Wang, "Reach on sobel operator for vehicle recognition," in *2009 International Joint Conference on Artificial Intelligence*, April 2009, pp. 448–451.

[5] A. Ajmal and I. M. Hussain, "A simple and novel method for vehicle detection and classification (a model based test)," in *Proceedings of 2012 9th International Bhurban Conference on Applied Sciences Technology (IBCAST)*, Jan 2012, pp. 58–63.

[6] M. Yu and Y. Deak Kim, "An approach to korean license plate recognition based on vertical edge matching," in *Proc IEEE Int Conf Syst, Man and Cybernetics*, vol. 4, 02 2000, pp. 2975 – 2980 vol.4.

[7] M. Sarfraz, M. J. Ahmed, and S. A. Ghazi, "Saudi arabian license plate recognition system," in *2003 International Conference on Geometric Modeling and Graphics, 2003. Proceedings*, July 2003, pp. 36–41.

[8] A. Safaei, H. L. Tang, and S. Sanei, "Real-time search-free multiple license plate recognition via likelihood estimation of saliency," *Computers Electrical Engineering*, vol. 56, pp. 15–29, 11 2016.

[9] C. Busch, R. Domer, C. Freytag, and H. Ziegler, "Feature based recognition of traffic video streams for online route tracing," in *VTC '98. 48th IEEE Vehicular Technology Conference. Pathway to Global Wireless Revolution (Cat. No.98CH36151)*, vol. 3, May 1998, pp. 1790–1794 vol.3.

[10] Q. Gao, X. Wang, and G. Xie, "License plate recognition based on prior knowledge," in *2007 IEEE International Conference on Automation and Logistics*, Aug 2007, pp. 2964–2968.

[11] K. K. Kim, K. I. Kim, J. B. Kim, and H. J. Kim, "Learning-based approach for license plate recognition," in *Neural Networks for Signal Processing X. Proceedings of the 2000 IEEE Signal Processing Society Workshop (Cat. No.00TH8501)*, vol. 2, Dec 2000, pp. 614–623 vol.2.

[12] S. Z. Masood, G. Shu, A. Dehghan, and E. G. Ortiz, "License plate detection and recognition using deeply learned convolutional neural networks," *CoRR*, vol. abs/1703.07330, 2017. [Online]. Available: http://arxiv.org/abs/1703.07330

[13] K. Min Wong, "Template-based matching using weight maps," *Computing Research Repository - CORR*, 04 2011.

[14] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963. [Online]. Available: http://www.jstor.org/stable/2098941

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.

[16] V. Vašek, V. Franc, and M. Urban, "License plate recognition and super-resolution from low-resolution videos by convolutional neural networks," in *Proc. of British Machine Vision Conference*, September 2018. [Online]. Available: ftp://cmp.felk.cvut.cz/pub/cmp/articles/franc/Vasek-LPR-BMVC2018.pdf