

# Deep Licence Plate Reader

## Technical Report

Jason Bolito    Siu Kei Muk    ChengYao Qian  
u5786025       u5721042       u5836376

### Abstract

In this work, we address the recognition problem of Australian licence plates. We apply morphological image segmentation for character image extraction, convolutional neural network (CNN) for recognition, and a scoring function for format detection and correction. Character sub-images are first extracted from the input through the character segmentation process, and fed to the CNN for recognition afterwards. Format-based correction is then applied to deal with incorrectly classified plates due to ambiguous characters. The CNN is trained by a set of 1093 character images which is a combination of segmentation results and characters generated from a special font. The network achieved an accuracy of 96.50% and 99.19% with a uniform and adaptive thresholding setting respectively. The adaptive thresholding and format correction schemes are shown to have a better performance in handling licence plates with similarly looking characters thus achieving a plate classification rate of 98.41%.

## 1 Introduction

With the rise of intelligence computing technology, automatic number plate recognition (ANPR) has been widely used in transportation systems by organisations, especially governments. The usage ranges from law enforcement, crime deterrent, enterprise security and services, to traffic control and electronic toll collection. The automation has introduced systems optimised to real-time scenarios credited to the development of optical character recognition (OCR) solutions. Several techniques have been developed in previous works, such as Centroid-to-Boundary Distances feature extraction followed by nearest neighbour classification[1], and clustering using cryptogram decoding algorithm[2]. In this paper, we present a solution that makes use of deep learning techniques.

This paper is organised as follows. Section 2 describes the details of the design and methodology of our solution. Section 3 presents the testing results and draws conclusions. The possible future works are mentioned in Section 4.

## 2 Solution Description

Our solution, Deep Licence Plate Reader (DLPR), performs licence plate recognition through three main phases. The first step is the *segmentation*

*phase*, where the initially provided licence plate image is cleaned up and preprocessed followed by the segmentation of all the characters and spaces in it. These resulting characters are fed to convolutional neural network (CNN) based recognition system in the *recognition phase*. Given a candidate character image, the CNN outputs the predicted character along with an estimate of its confidence which is accepted based on an adaptive threshold. Finally, all the recognised characters/spaces are compiled into a candidate licence plate number which is processed in the *correction phase*. In this step, a licence plate number is matched against a database of currently valid licence plate number formats and some corrections are performed if necessary. A diagram illustrating DLPR’s pipeline is displayed in figure 1.

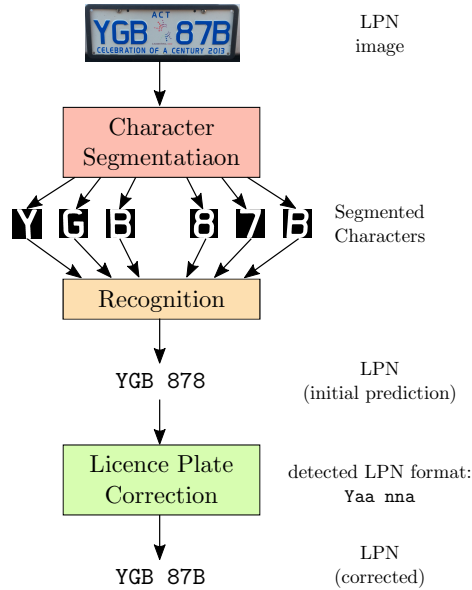


Figure 1: DLPR pipeline.

## 2.1 Character Segmentation

This is the first phase of DLPR’s pipeline and is mainly responsible for performing segmentation on a licence plate. This module attempts to extract the images of individual character and also attempts to detect spaces<sup>1</sup> found in the given licence plate number. The segmentation process can be broken down into three steps: *noise reduction and resampling*, *connected component analysis* and *space detection*.

<sup>1</sup>Spaces are detected for licence plate correction purposes, discussed further on in the report.

## Noise reduction and resampling

Given a licence plate image, it is somewhat naïve to assume that it will always have high quality, *i.e.*, a high resolution image with a clearly visible licence plate. For instance, images produced by law enforcement cameras will typically have low resolution images (see [3, 4]). The plates themselves might be degraded or the image itself might have some camera produced noise derived from a camera-specific setting like a high ISO value. DLPR addresses these issues by applying some noise reduction using filtering and resizing.

First, we convert the image to grayscale and compute the background<sup>2</sup> variance, using it as a crude approximation of the image’s noise level. This noise level is then used to construct variable-sized median and gaussian filters and apply them in that order. The median filter addresses the character corruption issue while the gaussian filter is meant to address low lighting/high ISO kinds of conditions. The image is then resized to avoid low resolution image issues and only occurs after filtering to avoid noise amplification. A small crop is also applied to avoid issues with the plate’s border. Our noise reduction method seems to perform rather well even with images under heavy noise as can be seen in the following figure.



Figure 2: Noise reduction and thresholding applied to a licence plate with 50% *Salt and Pepper* noise.

## Connected component analysis

After the noise reduction step the grayscale licence plate image is thresholded using Otsu’s method (see [5]) performing inversion if necessary<sup>3</sup>. *Morphological opening* is then applied as an attempt to disconnect characters from other parts of the licence plate such as holes, or remaining

---

<sup>2</sup>The background pixels of a licence plate can be easily determined using preliminary thresholding and pixel counting.

<sup>3</sup>By convention, licence plate images are thresholded such that character pixels are white and the background pixels are black. Some plates have light coloured characters on dark backgrounds resulting in black characters on a white background after thresholding. To address this issue the binary image is simply inverted in such cases.









parts of the border. We then compute the 8-connected components of the image with the method described in [6].

The characters are then extracted by analysing *admissible* connected components. A connected component is found to be admissible if its width and height fit into certain ranges that depend on the image’s size. At this point it is expected that most of the remaining connected components are characters, so we use the median of their vertical centres to filter out any misaligned blobs. This process illustrates the principle that the main characters in a licence plate are horizontally aligned, which is true for all current general-issue Australian licence plates. Some licence plates have logos or other irrelevant information in the middle. These are sieved out by applying an additional height filter. This height filter is only applied to small blobs after the first *big* character is found<sup>4</sup>. Finally, all connected components that weren’t filtered out by the previous steps are found to be *characters*, ending the character segmentation part.

### 2.1.1 Space Detection

Space detection is performed by analysing the distances between the previously found characters’ centres. Distances are marked as *spaces* based on how close they to the largest distance. This method is backed by the assumption that all licence plates use a monospace font making both character spacing and whitespace easily distinguishable cases. On the other hand, if the minimum and maximum distances between characters are too close, then the licence plate number is deemed to have no spaces.

Some sample outputs of the segmentation phase are displayed in the following table.

Initial Image	Clean-up and Thresholding	Segmentation Output
		A A 5 6 Q H
		N X S 2 0 T
		T 3 7 2 4 5
		B E A D S

## 2.2 Character Recognition

This is the second phase of DLPR’s pipeline and is responsible for performing character recognition from character images extracted in the segmentation phase. This module attempts to recognise the actual character represented in the given binary images through the application of a convolution neural network. The recognition comprises two steps: confidence level estimation and adaptive thresholding.

<sup>4</sup>Small blobs appearing before a big character might be small characters relevant to the licence plate number (like the small ‘T’ found in some trailer plates).

### 2.2.1 Confidence Level Estimation

#### Convolutional Neural Network

First, the confidence level estimation is handled by a convolutional neural network (CNN). A CNN is a feed-forward artificial neural network characterised by its unique architecture composed of stacks of different types of layers each with a specialised functionality. An illustration of an example CNN architecture is shown in Figure 3.

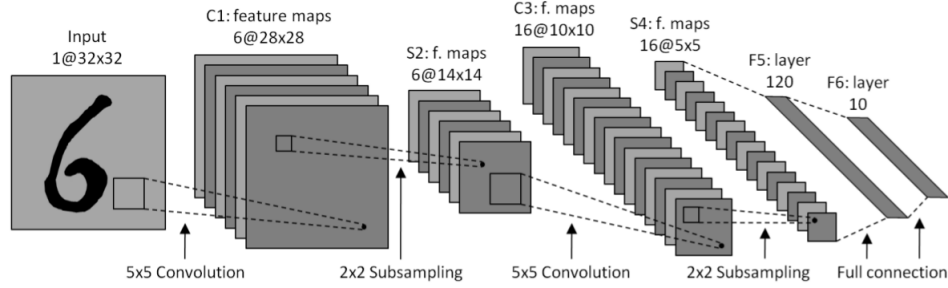


Figure 3: Example CNN architecture[7]

The network is trained in supervision by gradient descent backpropagation learning algorithm which minimises an objective loss function[8].

#### Convolutional Layer

A convolutional layer consists of a stack of filters each defined by a kernel that extracts features from the input feature map. Each filter outputs a feature map by convolving the input image or feature map from the previous layer with its kernel. The weights of the kernels are initialised randomly and learned by backpropagation.

#### ReLU Layer

A Rectified Linear Unit (ReLU) performs the non-linear mapping

$$f(x) = \max(0, x)$$

on each of the elements in every input feature map. The rectifier activation function is argued to be more biologically plausible than the traditional activation functions used in neural networks such as logistic sigmoid and hyperbolic tangent functions[9].

#### Max-Pooling Layer

A max-pooling layer performs subsampling on each of the incoming feature maps by choosing the maximum value in the sliding window. This layer aims to condense the feature responses in feature maps so that each local response is summarised by the one with the strongest signal. This provides

us with a form of translation invariance and reduces the computations for upper layers.

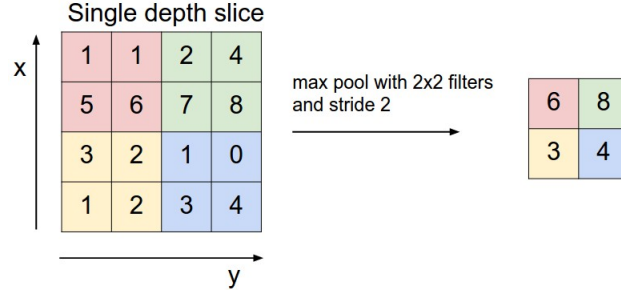


Figure 4: Illustration of max-pooling[10]

### Dropout Layer

The dropout method is used in several layers to reduce potential overfitting. This is done by adding a dropout layer between layers that randomly sets the layer's input to zero with a given probability, which corresponds to dropping a randomly chosen unit and all of its connections from the network during training. The dropout method is shown to be effective in reducing the overfitting by breaking the brittle co-adaptations that works for training data but do not generalise to unseen data[11].

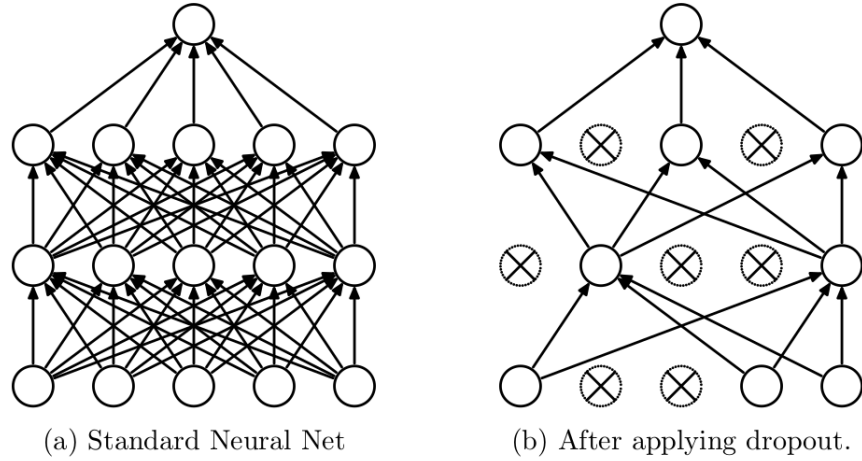


Figure 5: Illustration of effect with dropout applied.[11]

## Fully Connected Layer

Fully connected layers are identical to the hidden layers used in traditional neural networks. They are responsible for the classification task.

## Softmax Layer

The softmax layer uses the softmax function to output the confidence level estimate of a particular character image:

$$P(C_r | x) = \frac{P(x | C_r)P(C_r)}{\sum_{j=1}^K P(x | C_j)P(C_j)} = \frac{\exp(a_r)}{\sum_{j=1}^K \exp(a_j)}$$

where  $0 \leq P(x | C_r) \leq 1$ ,  $\sum_{j=1}^K P(C_j | x) = 1$ ,  $a_r = \ln(P(x | C_r)P(C_r))$ ,  $P(x | C_r)$  is the conditional probability of the sample given class  $r$ , and  $P(C_r)$  is the class prior probability[12].

## Architecture

Our CNN architecture accepts an input binary image of size  $50 \times 50$ . The convolutional layers are all followed by one ReLU layer, one max-pooling layer and one dropout layer. The depth of convolutional layers increases as the network goes deeper, and the dropout probabilities are assigned in a non-decreasing fashion as the potential of overfitting grows with the complexity of layers.

The network is trained by backpropagation using stochastic gradient descent with momentum. The training set contains 1093 character images of size  $50 \times 50$ . The training set is a combination of segmentation results of normal plates and characters generated from the special Victoria font. The neural network is trained by 100 epochs, with a mini-batch size of 128 to exploit maximum efficiency of MATLAB. A learning rate scheduling which decreases the rate in a piecewise fashion every 40 epochs at a rate of 10% is adopted as the reduction of the step size is shown to be essential[13] to the convergence of stochastic gradient descent.

### 2.2.2 Adaptive Thresholding

After the confidence level estimation step, the resulting estimates are thresholded adaptively. The main reason for performing thresholding is to distinguish character objects from non-character objects detected in the segmentation process.



Figure 6: Non-character image

Two thresholds, 0.8 and 0.5, are used in the process of thresholding. For the unambiguous characters, if the given image does contain a true

Layer Type	Parameters
Input	$50 \times 50$ pixels gray-scale image
Convolution	#filters: 16, $k:3 \times 3$ , $s:1$ , $p:5$
ReLU	
Maxpooling	$p:2 \times 2$ , $s:2$
Dropout	$p:0.2$
Convolution	#filters: 32, $k:3 \times 3$ , $s:1$ , $p:0$
ReLU	
Maxpooling	$p:2 \times 2$ , $s:2$
Dropout	$p:0.3$
Convolution	#filters: 64, $k:3 \times 3$ , $s:1$ , $p:0$
ReLU	
Dropout	$p:0.3$
Fully Connected	#neurons: 200
ReLU	
Dropout	$p:0.5$
Fully Connected	#neurons: 36
Softmax	36 classes

Table 1: CNN architecture

character rather than a non-character entity, then the represented character is expected to have significantly high confidence level estimate. But for the ambiguous characters pairs, such as (0, 0), (8, B) and (I, 1), the threshold of 0.5 is used as the ambiguity between the candidate characters leads to a substantial drop in confidence. In fact, we observed that every ambiguous pair has similar confidence levels, which interestingly resembles the confusion a human would have when choosing between two similar looking characters.

### CNN challenges

We faced several problems while developing the CNN architecture. One of the problems was the network complexity. Initially, we tried an input size of  $220 \times 100$  image. The model gave an acceptable result of about 91% accuracy on the training data. However, the training took a long time to converge, and it was not able to fit in the GPU arrays due to its massive size when transferring from the MatConvNet library to MATLAB's GPU-based implementation. The model performed better after the size is reduced to its quarter,  $55 \times 25$ .

The second issue was insufficiency of information in the extracted character images. The characters were extracted by tightly fitted bounding boxes. We noticed this would be a problem when applying several invariance tests by feeding in slightly translated or rotated character images. The performance drop was severe for the characters I and 1. The reason is that the images of I and 1 are usually white in almost every pixel. This forbids the network to learn to extract information from edges of I and 1.



We solved this by padding the extracted character images in such a way that the character is centred in a square image with size  $50 \times 50$ .

After further testing, we encountered some strange problems that a character **M** was recognised as **P**, and a **Y** as **H**, although they do not have any clear similarities. It turns out it was a combination of overfitting and insufficient training data. We noticed that the characters the model failed recognising was from Victoria number plates, which has its own unique font. We then discovered that the training data did not have a comparable amount of characters with this unique font. But as their differences should not be significant enough for an **M** to be interpreted as **P**, and **Y** as **H**, so this was obviously also an overfitting problem as the network failed to generalise the learned knowledge to similar situations. The problem was tackled by introducing both the inclusion of aggressive dropout layers in the network, and a set of artificially generated characters from the Victoria font.

## 2.3 Licence Plate Correction

This is the final step of DLPR’s pipeline. It is mainly responsible for validating and correcting the licence plate number predicted by the recognition module. This is accomplished using format-based validation. Given a database with all the currently valid licence plate formats, we can determine which format best matches the current prediction.

To motivate this consider the following example. Let’s assume we were given the prediction **YAB 128** by the recognition module. This number is closer to the ACT format **Yaa nna**<sup>5</sup> than any other. Hence, it is likely that the recognition module mistakingly classified the last character as an **8** rather than a **B** (which would match the format) due to ambiguity. Since the corrected number **YAB 12B** completely matches the previous format it is deemed to be a reasonable prediction and should therefore be adopted as the new licence plate number prediction.

DLPR’s format-based correction can be broken down into three steps: determining a plate’s format, correcting any ambiguous characters accordingly and assessing the correction itself.

### Scoring Function

Detecting a licence plate number’s format can be a problematic task due to overlapping formats. For example, the licence plate **YAB 12B** matches both the ACT template **Yaa nna** and the NSW format **aaa nna**. Without looking at any state information on the licence plate, a human would probably choose the ACT format because **Y** is a compulsory character in it. To mimic this behaviour we choose a scoring function approach rather than the usual format distance approach. Given a licence plate number

---

<sup>5</sup>In a format string **a** and **n** is used to denote letters and numbers respectively. All other remaining characters are used to denote required characters.

$L = (L_1, \dots, L_n)$  we adopt the format that maximises a scoring function

$$F^* = \arg \max_{F \in \text{Formats}} \text{score}(L, F),$$

$$\text{score}(L, F) = \begin{cases} 0 & \text{if } \text{length}(F) \neq \text{length}(L) \\ \sum_{i=1}^n \text{fscore}(L_i, F_i) & \text{otherwise} \end{cases} \quad \text{and}$$

$$\text{fscore}(l, f) = \text{matches}(L_i, F_i) \times \begin{cases} 5 & \text{if } F_i \text{ is a compulsory character} \\ 1 & \text{otherwise} \end{cases},$$

where  $\text{matches}(l, f)$  is 1 when  $l$  matches the format  $f$  and  $-1$  otherwise. Note that  $\text{fscore}$  represents the base score of each format (mis)match based on if the character is required or just a format indicator. This is done mainly to avoid clashes between overlapping formats. A higher compulsory character score helps us avoid clashes more effectively but increases bias towards required characters at the same time. On the other hand, a lower compulsory character score leads to more format overlapping. If we have more than one format with a maximal score

## Character correction and validation

With the licence plate's format identified we can finally attempt to correct the given licence plate number. All format mismatches are corrected according to an ambiguity map, *i.e.*, a map that relates all ambiguous character pairs discussed in the recognition section. With all the applied corrections we can finally produce a new predicted licence plate.

However we also need to check if the corrected licence plate matches the intended format. To do this we simply verify that the new prediction matches the previously chosen format and output it as our new prediction. If validation fails, DLPR rolls back and returns the licence plate number provided by the recogniser. The rollback action is crucial for custom issued licence plates as they won't fit any other format by design.

## Format database

The format database used is of the utmost importance. The formats stored in the database should capture the ones of the licence plates being processed. Additionally, this database should also be minimal in the sense that no extra formats are required to capture our target licence plate group. Case in point, if such a system were to be used in a law enforcement context, we would need a database with all *domestic* licence plate formats. Hence, the system wouldn't be able to correct international licence plates. However adding all international plate formats would ultimately pollute the database originating countless format clashes and would result in a poorer performance overall. The format database used in DLPR comprises all the general issue licence plates from every state and territory in Australia.

### 3 Results and Conclusion

In this section we present the accuracy of the main DLPR’s main components. The segmentation test results are omitted because they are extremely close to DLPR’s global results, given that one of DLPR’s main bottlenecks is the effectiveness of the segmentation module.

#### 3.1 Character Recognition Results

The CNN used in the recognition module was tested with 371 segmented character images, the results are presented in Table 2. The network achieved an accuracy of 99.19% when using adaptive thresholding, and 96.50% when using uniform threshold setting. We can see that the application of adaptive thresholds yields a significantly higher accuracy than its counterpart. The “Misclassified” column shows the number of each misclassified characters. Note that in the uniform threshold setting almost every characters are ambiguous and all of them are filtered out. This shows that the confidence level estimates of the ambiguous characters are significantly lower than then any unambiguous character as expected.

Thresholds	Accuracy (%)	Misclassified
$u = 0.8, a = 0.4$	<b>99.19</b>	J: filtered and 0 $\rightarrow$ 0 (2 cases)
$u = a = 0.8$	96.50	0 (5 cases); B (4 cases); 0 (2 cases); J,1: filtered (1 case)

Table 2: CNN character recognition testing results.

#### 3.2 DLPR Results

To test the accuracy of DLPR as a whole system we used 63 standard general issue licence plates. To test the system’s robustness, we included licence plates with different resolutions, lighting conditions and geometry. Furthermore, we also generated *noisy versions* of the test dataset to measure the system’s effectiveness under noisy conditions. Several kinds of noise were used to replicate real world noise-related issues. Salt and pepper noise was used to simulate plate degradation, gaussian noise to simulate lighting and camera-related issues and mixed noise<sup>6</sup> to simultaneously simulate both situations. Finally, we also tested the effect of using format-based correction (FC). The results are displayed in table 3.

#### Conclusion

The results presented above clearly allow us to conclude that our system performs very well under our initial assumptions with an overall accuracy of 98.41% competing with other well-performing solutions. The approaches behind each module are concise and intuitive leading and, in most cases, on par with actions a human would take.

<sup>6</sup>S&P and gaussian noise combined.

Noise	DLPR with FC (%)	DLPR without FC (%)	Accuracy Improvement (%)
None	<b>98.41</b>	96.83	1.58
Salt & Pepper (10%)	<b>95.24</b>	92.06	3.18
Salt & Pepper (30%)	<b>95.24</b>	90.48	4.76
Salt & Pepper (50%)	<b>79.37</b>	76.19	3.18
Gaussian (10%)	<b>96.83</b>	93.65	3.18
Gaussian (30%)	<b>90.48</b>	88.89	1.59
Gaussian (50%)	<b>84.13</b>	82.54	1.59
Mixed (10%)	<b>95.24</b>	92.06	3.18
Mixed (30%)	<b>90.48</b>	85.71	4.77
Mixed (50%)	<b>63.49</b>	58.73	4.76

Table 3: Accuracy of DLPR under different noise conditions with and without format correction.

Unlike some other ALPNR systems, DLPR is prepared to deal with similar looking characters. The use of adaptive thresholding coupled with format-based correction has proved to be a successful approach, with significant performance increases (up to 4.74%). Our solution is also performs extremely well under extremely noise conditions (79.37% and 84.13% accuracy under 50% S&P and gaussian noise respectively).

## 4 Future Work

DLPR has seems to perform very well as an ALPNR. There are however some obvious issues that need to be addressed. Some of these issues may spawn new research directions altogether.

### Improve space segmentation

Space detection is currently sensitive to hardcoded relative glyph distance constraints. Most of these constraints were coded based on *intuition* and might not generalise to well with variable spacing-width monospace fonts. An interesting approach to solve this issue could be to have a neural network learn the optimal values for all these constraints.

### State text recognition

One of the shortcomings of format-based correcting is the increase of clashes as we add more formats, making the method less reliable. This happens mostly in state-based countries like Australia and the USA. If DLPR could also recognise the text with state-related information in a given licence plate we would surely end up with much less classes as we introduce more distinguishing features in our format database.

## Hidden Markov Model correction

Fixed format licence plates can be seen as recurrent and predictable strings of characters. Thus, we can reduce licence plate validation to a Hidden Markov Model problem to indicate how *likely* is a plate to be observed and come up with suitable correction candidates. This method could be used alongside format-based correction or as a standalone validation method.

## NN-based plate segmentation

Instead of incrementally improving the segmentation module, we could attempt replace it directly with a neural network backed segmentation model. This could either be implemented as a regular sliding window CNN or, more interestingly, as a recurrent neural network (RNN) with long short-term memory (LSTM) all together. The LSTM solution is more interesting because it simulates a human-like procedural segmentation method. The main advantage of developing NN-backed segmentation methods is the fact a neural network can be *optimally trained* while hard-coded algorithms and constraints are always used on the basis of *intuition* with some randomness.

## Dynamic learning rate scheduling

The current CNN learning rate scheduling scheme is static. It is specific to a particular setting, such as the maximum number of epochs being 100, or a particular training data set. As the state of the CNN varies due to a number of factors such as initial weights, current number of epochs and training data set, a dynamic learning rate scheduling scheme which allows the learning rate to be determined by the state of CNN on-the-fly is desirable as it is shown that the convergence and the local minimum obtained depend on a suitable tuning and decay of learning rate significantly[14].

## References

- [1] Mikael Laine and Olli S. Nevalainen, *A Standalone OCR System For Mobile Cameraphones*. In: The 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (2006)
- [2] Gary Huang, Erik Learned-Miller, Andrew McCallum, *Cryptogram Decoding for Optical Character Recognition* (2005)
- [3] *Mobile Speed Camera*, Western Australia Police, <https://www.police.wa.gov.au> (as of 2/06/2016).
- [4] <https://www.vitronic.com/traffic-technology/applications/traffic-enforcement/speed-enforcement/poliscan-speed-fixed.html> (as of 2/06/2016).
- [5] Otsu, Nobuyuki *A threshold selection method from gray-level histograms*, IEEE Trans. Sys., Man., Cyber. 9 (1), 1979, pp. 62-66.
- [6] Haralick, Robert M., and Linda G. Shapiro, *Computer and Robot Vision, Volume I*, Addison-Wesley, 1992, pp. 28-48.

- [7] Strigl, D., Kofler, K., Podlipnig, S.: *Performance and Scalability of GPU-Based Convolutional Neural Networks*. In: 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP 2010), pp. 317324 (2010)
- [8] LeCun, Y., et al.: *Gradient-Based Learning Applied to Document Recognition*. In: Intelligent Signal Processing, pp. 306351. IEEE Press (2001)
- [9] Xavier Glorot, Antoine Bordes and Yoshua Bengio (2011). *Deep sparse rectifier neural networks*
- [10] *Convolutional Neural Networks for Visual Recognition* <http://cs231n.github.io/convolutional-networks/>
- [11] Srivastave, N., G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research*. Vol. 15, pp. 1929-1958, 2014
- [12] <http://au.mathworks.com/help/nnet/ref/softmaxlayer.html>
- [13] H. Robbins and S. Monro, *A stochastic approximation method*, *Annals of Mathematical Statistics*, , no. 22, pp. 400407, Sep. 1951.
- [14] Tom Schaul, Sixin Zhang and Yann LeCun, *No More Pesky Learning Rates* (2012)