Autonomous License Plate Recognition System Using Deep Learning Network

Rohith Polishetty, Paul Rad
Open Cloud Institute, University of Texas at San Antonio

Abstract

Autonomous License Plate Recognition (ALPR) systems capture a vehicle's license plate and recognize the license number and other required information from the captured image. ALPR systems have number of significant applications: law enforcement, public safety agencies, tollgate systems, etc. The goal of our systems is to recognize the characters and state on the license plate with high accuracy.

ALPR has been implemented using various techniques. Traditional recognition methods use handcrafted features for obtaining features from the image. Unlike conventional methods, deep learning techniques automatically select features and are one of the game changing technologies in the field of computer vision, automatic recognition tasks and natural language processing. Some of the most successful deep learning methods involve Convolutional Neural Networks. Our research applies deep learning techniques to the ALPR problem of recognizing the state and license number from the USA license plate.

Existing ALPR systems include three stages of processing: license plate localization, character segmentation and character recognition but do little for the state recognition problem. Our research not only extracts the license number, but also processes state information from the license plate. We also propose various techniques for further research in the field of ALPR using deep learning techniques.

Here in we employ deep learning techniques that are a supervised learning model. A reference supervised learning model is as follows:

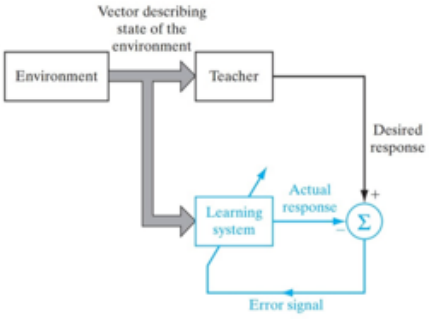

Fig1. Supervised Learning Model

1. Introduction to the deep learning

Deep learning is a new learning paradigm in deep structured learning or hierarchical learning. Deep learning is an active research area of neural networks, artificial intelligence, pattern recognition etc. Deep learning has different high-level definitions:

1. Learning or exploring automatic feature extraction from many layers of non-linear functional units has characterized deep learning. Each successive layer uses the output of one or more preceding layers as input.
2. High-level features are learned from low level features that form a hierarchical representation for supervised or unsupervised feature extraction and transformation, and for pattern recognition, analysis and classification.
3. Learning where learning of data involves modeling the complex relationships among data.

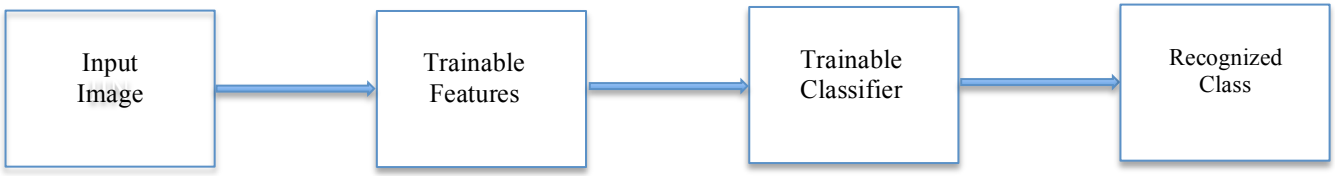

Fig2. Block diagram of Deep Learning Model

We briefly describe past ALPR research. Most of the research published in ALPR was conducted on license plates of countries, which have strict design standards. As most of the previous systems concentrate only on license number, they are ineffective in detecting the state information from USA license plates.

2. Design & Model Implementation

A typical ALPR system includes image capture, preprocessing, localization of license number and recognition of the license number. As our research concentrates on information extraction and recognition from the license plate, we review license plate segmentation and recognition.

The extraction of license plate information is divided into four stages:

1. Preprocessing the license plate
2. Binarization
3. Separation of state and license number
4. Extraction of license plate

Preprocessing is generally an important step in image processing systems, and it helps computationally in further stages of the system. We use anisotropic diffusion and histogram equalization to achieve higher quality segmentation. The basic diffusion equation provided in

$$\frac{\partial I(x,y,t)}{\partial t} = div[g(||\nabla I(x,y,t)||)]\nabla I(x,y,t)$$

Where **t** is time parameter, **I(x, y, t)** is the input image, **∇I(x, y, t)** is the gradient of image at time **t**, **g(x)** is the conductance function.

Algorithm 1 Pseudocode for Global Thresholding

Import an input image I of size $m \times n$ and the pixels range is $[0, L-1]$, where L is the maximum gray scale value.
Initialize the percentage number of total pixels in image required to be background area, let it be *background* where $background \in [0, 1]$.
$pixels \leftarrow m \times n \times background$.
$sum \leftarrow 0$
for $i \leftarrow 0, L-1$ **do**
    $bin[i] \leftarrow 0$
end for
for $i \leftarrow 0, m-1$ **do**
    for $j \leftarrow 0, n-1$ **do**
        $temp \leftarrow I[i][j]$
            $bin[temp] = bin[temp] + 1$
    end for
end for
while $i \leq L-1$ and $sum \leq pixels$ **do**
    $sum \leftarrow sum + bin[i]$
        $threshold \leftarrow i$
        $i \leftarrow i + 1$
end while
for $i \leftarrow 0, m-1$ **do**
    for $j \leftarrow 0, n-1$ **do**
        if $I[i][j] \geq threshold$ **then**
            $I[i][j] \leftarrow 1$
        else
            $I[i][j] \leftarrow 0$
        end if
    end for
end for

2.1 Edge Based Detection

Edge detection has been used in various vision applications, for instance the Canny edge detector has been used in order to find the edges for the input image and also reduces the amount of data to be processed. The Canny algorithm satisfies the following criteria:

1. Low error rate: It gives a low error rate, which means it detects all the edges in the image.

2. Good localization: The distance between the real edge pixels and detected edge pixels has to be minimized.

3. Minimal response: Only one detector response per edge.

Algorithm 2 Canny Edge Detection

1: Import the input image.
2: Apply Gaussian filter to remove noise from the image. Before locating the edges.
3: Find the intensity gradients of the image by applying the following convolution pairs at each pixel location in the image

$$G_x(i,j) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y(i,j) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Where (i, j) is the location in the image where the above operation takes place.
4: Compute the gradient value and angle using the following equations at each location in the image.

$$G(i,j) = \sqrt{G_x^2(i,j) + G_y^2(i,j)}$$

$$\theta(i,j) = tan^{-1}\left(\frac{G_y(i,j)}{G_x(i,j)}\right)$$

5: Non-Maximum suppression is applied to remove the pixels which are not part of an edge, results in thin lines at edges.
6: The final step in Canny edge detector is to use two thresholds i.e. upper and lower thresholds

1. If the pixel gradient is higher than the upper threshold it is accepted as an edge.
2. If the pixel gradient is below the lower threshold, then it is rejected.
3. If the gradient is between the upper and lower threshold then it will accepted as an edge only if it is connected to the pixel whose gradient is above the threshold.

3. Extraction of License Plate

For simplicity we assume that the state is available in the upper portion of the license plate. So the first step in our system is separation of license number and state portion of the plate. To locate and extract the state and license number, we use horizontal histogram projection. The horizontal projection calculates the sum of pixels along the horizontal direction. The summation of all the pixels along each row gives horizontal projection values. We also use vertical projection to extract characters from the license plate candidate; vertical projection obtains the coordinates of the characters. The horizontal H(y) and vertical projection V (x) for an image I of dimension M ×N is given as:

$$H(y) = \sum_x I[x,y]$$

$$V(x) = \sum_y I[x,y]$$

3.1 Summary of Binarization Techniques

Algorithm: Pseudo code for extraction of state and license number using horizontal projection

Let the binary image be I
Skeletonize the image using morphological operations.
$r \leftarrow$ number of rows in the image.
$c \leftarrow$ number of columns in the image.
for $i \leftarrow 0, r-1$ **do**
    for $j \leftarrow 0, c-1$ **do**
        $Aray[i] = I[i][j] + Array[i]$
    end for
end for

10: $window \leftarrow [0.33, 0.33, 0.33]$
11: Convolution of sum with window, $Array \leftarrow Array * window$
12: 3×1 column matrix with all the values of matrix equals to 0.33.
13: $Z_1 \leftarrow Array[0 : r/2]$
14: $Z_2 \leftarrow Array[r/2 : r - 1]$
15: $R_{start} \leftarrow$ index of array Z_1 having minimum value from the end. R_{start} is the row number where license number starts.
16: $R_{end} \leftarrow$ index of array Z_2 having minimum value from index 0. R_{end} is the row number where license number ends.
17: $license\_number \leftarrow I[R_{start} : R_{end}]$
18: $State \leftarrow I[0 : R_{start}]$

Fig4. Skeletonized result of license number portion

Fig5. Segmented characters from license number portion

4. Convolution Neural Networks

The Convolutional Neural Network CNN, first proposed by LeCun in 1988, is a neutral network model incorporating the following ideas: receptive fields, weight sharing and subsampling. It is a special type of multilayer perceptron trained in supervised mode using back propagation. It is one of the most successful machine learning architectures in computer vision and has achieve state-of-the-art results in tasks as character recognition, object recognition, face detection and pose estimation, speech recognition, license plate recognition, image preprocessing and segmentation tasks.

A CNN learns complex, high dimensional features from a large number of examples, which makes it an obvious candidate for pattern recognition tasks. CNN architectures ensure some degree of shift, scale and distortion invariance using some of the features such as local receptive fields, subsampling, shared weights etc. Unlike conventional pattern recognition tasks one of the benefits of CNN's lie in extraction features itself and which uses images as input for training, testing the network. The CNN builds complex features from large collection of examples and the complexity of learning features with more layers. This is done by successively convolving the input image with filters to build up a hierarchy of feature maps. The hierarchy results in complex

features and learning, as well as translational and distortion invariant. The whole CNN can be expressed as a score function where raw image pixels are given as input on one end and determine the category or class score at the other end.

To illustrate 2D convolution, let the image be $x[n_1 x n_2]$ of N × N points and kernel be $h[n_1 x n_2]$ of M × M points where:

$$\begin{aligned} x[n_1, n_2] = 0 \quad & \forall \quad n_1 < 0 \quad \text{and} \quad n_1 > N-1 \\ & \forall \quad n_2 < 0 \quad \text{and} \quad n_2 > N-1 \\ h[n_1, n_2] = 0 \quad & \forall \quad n_1 < 0 \quad \text{and} \quad n_1 > M-1 \\ & \forall \quad n_2 < 0 \quad \text{and} \quad n_2 > M-1 \end{aligned}$$

4.1 Convolution Layers

Convolutional Neural Networks CNNs are often used to process 2D images to learn high dimensional, nonlinear mappings from the large set of examples. This makes them a good solution for many computer vision tasks. The CNN takes input an image and convolves it with a 2D kernel of adjustable weights.

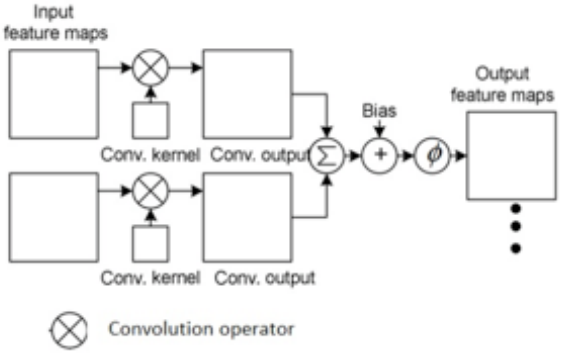

Fig6. Example of Convolutional layer.

4.2 Momentum

Momentum improves the convergence speed, where the current weight changes depends on the previous weight change. A momentum is added to improve the convergence speed.

Algorithm: Pseudocode for backpropagation

1: The input vector $\mathbf{x}$ is applied to the neural network and propagated through all the layers.
2: Calculate the error using any error function. Let us assume the error function be mean square error and the error is given as $E_{MSE}(i,j) = \frac{|\hat{y}(i,j)-x|^2}{n}$, where $\hat{\mathbf{y}}(i,j)$ be the expected result, x be the desired output and n be the total number of patterns provided for training.
3: Calculate the error gradient $\frac{\partial E}{\partial \mathbf{w}}$ for each layer.
4: Update weights

4. Results and Analysis

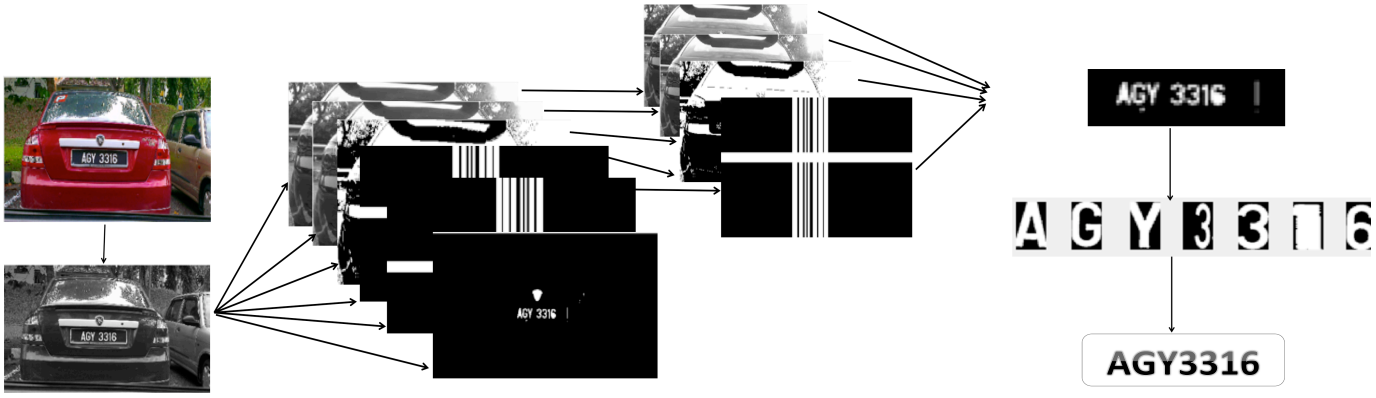

Fig7. Convolution Model for extraction and Recognition

5. Conclusion

Important aspect incorporated in this research is using convolutional neural networks for feature extraction and license plate recognition. The recognition task includes recognition of license number and state information from the license plate. Our study also includes how the data format impacts the recognition system and different techniques to optimize convolution neural networks. Our experiments show an overall accuracy of 91.1% using gray scale images. Our CNNs are implemented using Scipy, Numpy, OpenCv libraries in Python and Linux environment. In general training of CNN takes 1-6 days, which depends on the neural network architecture. Our results show the usage of deep learning techniques in the field of ALPR system.