License Plate Recognition System

By

Vidushi Dwivedi

Course: DS226 Introduction to Computing for AI and ML

Instructor: Prof. Sashikumaar Ganesan

Indian Institute of Science, Bangalore
Department of Computational and Data Sciences

INTRODUCTION

Around the world and specifically in India, there is a wide variety of traffic rules that can be very easily regulated if the detection of number plates of vehicles can be automated and integrated with other systems of traffic law enforcement. A simple detection of number plates can help to detect stolen vehicles, fine violators, ensure security in specific regions and a plethora of similar applications.

The problem of number plate recognition can be solved through an amalgamation of image processing and machine learning. While image processing helps in detections of the number plate and the characters in the number plate, machine learning is involved in the recognition of those characters. In a general sense, the image processing step involves functions to create bounding boxes around required features of the image which can be done using either general libraries of image processing or specifically trained models like YOLO. On the other hand, the machine learning step involves the OCR (optical character recognition) which can be done using different models like kNN, NN, CNN and Tesseract.
Thus the problem statement of the license plate recognition system can be visualized as ***Fig1***

Fig 1

A brief description of the major techniques mentioned above is as follows:

- **kNN**

K-Nearest Neighbour is based on Supervised Learning technique.KNN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors.

kNN uses the following steps:

1. Select the number K of the neighbors
2. Calculate the Euclidean distance of K number of neighbors
3. Take the K nearest neighbors as per the calculated Euclidean distance.
4. Among these k neighbors, count the number of data points in each category.
5. Assign the new data points to that category for which the number of the neighbor is maximum.

- NN (Neural networks)

Artificial neural networks are a set of algorithms that try to mimic the function of our brain. they take a large set of data, process it and output the information we need. neural networks can learn to approximate any function. neural networks have ability to learn how to do tasks based on the data given for training or initial experience and they can create their own organisation or representation of the information it receives during learning time. Every neuron is connected with other neuron through a connection link. Each connection link is associated with a weight that has information about the input signal.

Network layers- three layers setup are most common in neural networks in which input layer is connected to hidden layer which is connected to output layer. In each layer of neural network the neurons calculate the weighted sum of the inputs add the bias and execute activation function on them

Input units represent the raw information that is fed into the network, activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units, by modifying these weights, a hidden unit can choose what it represents. Hidden layer is called so because it is invisible to external systems The behavior of the output units depends on the activity of the hidden units and the weights between the hidden

and output units. Input and output layers are must for neural networks while there could be zero or many hidden layers.

Training of neural networks-

1. Initialize some random values for our weights and bias.
2. Input a single sample of our data.
3. Compare our output values with target values.
4. Quantify the difference between our output and target value we will call it our loss.
5. Adjust the weight and biases to lower the loss.
6. Keep doing this for each sample in our data set.
7. Stop training when loss stops decreasing

- CNN (Convolution neural network)
It is a version of neural network that is mainly used for images recognition, images classifications. Objects detections, recognition faces etc. CNN uses a 3-D volume arrangements for it's neurons because the input data is and image and image data consists of height, width and depth of colors.

Convolution is the process of using kernel or filter that is applied to input. Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters

CNN layers-

1. Input layer it takes input as image data
2. Convolution layer is the most significant part of the CNN as it is the layer that learns image features which aid our classifier
3. ReLu similarly as in neural networks we use ReLu as an activation function to introduce non linearity in our CNN. It changes all negative values to zero while leaving all positive values unchanged.
4. Pool- pool layer simply reduce the dimensionality of our feature map. pooling is also known as down sampling or subsampling. In the pool layer we reduce the no parameters for training only keeping the most important ones. these three types of pooling are mostly used-max, sum, and average.
5. Fully connected all nodes of this layer are connected to outputs of next layer. This layer outputs the class probability where each layer is assigned a probability. It does so using softmax activation function.

Training of CNNs-

1. Random weight initialization in convolution kernels.
2. Forward propagate an image through our network(input>convo>ReLu-pooling-fc layer)
3. Calculate the total error
4. Use back propagation to update our weights using gradient descent.
5. Keep propagating all images through our network till one epoch is complete.

- **YOLO**

You only look once (YOLO) is a state-of-the-art, real-time object detection system. Yolo architecture is like FCNN (fully convolutional neural network) which passes the image (nxn) once through the FCNN and output is (mxm) prediction. The architecture splits the input image in mxm grid and for each grid generation 2 bounding boxes and class probabilities for those bounding boxes are generated. YOLO first takes an input image.The framework then divides the input image into grids

Image classification and localization are applied on each grid. YOLO then predicts the bounding boxes and their corresponding class probabilities for objects. Even if an object spans out to more than one grid, it will only be assigned to a single grid in which its mid-point is located.We can reduce the chances of multiple objects appearing in the same grid cell by increasing the number of grids.

Tesseract-

It is an optical character recognition engine with support for unicode and the ability to recognize more than 100 languages out of the box. It can be trained to recognize other languages. Tesseract is used for text detection on mobile devices, in video, and in Gmail image spam detection. Tesseract assumes that its input is a binary image with optional polygonal text regions defined. Processing follows a traditional step-by-step pipeline as follows-

The first step is a connected component analysis in which outlines of the components are stored. Outlines are gathered together, purely by nesting, into Blobs. Blobs are organized into text lines, and the lines and regions are analyzed

for fixed pitch or proportional text. Text lines are broken into words differently according to the kind of character spacing. Fixed pitch text is chopped immediately by character cells. Proportional text is broken into words using definite spaces and fuzzy spaces.

Recognition then proceeds as a **two-pass process**. In the **first pass**, an attempt is made to recognize each word in turn. **second pass** is run over the page, in which words that were not recognized well enough are recognized again.

METHODOLOGY

Fig 2 represents the process overview of the Licence plate recognition system.

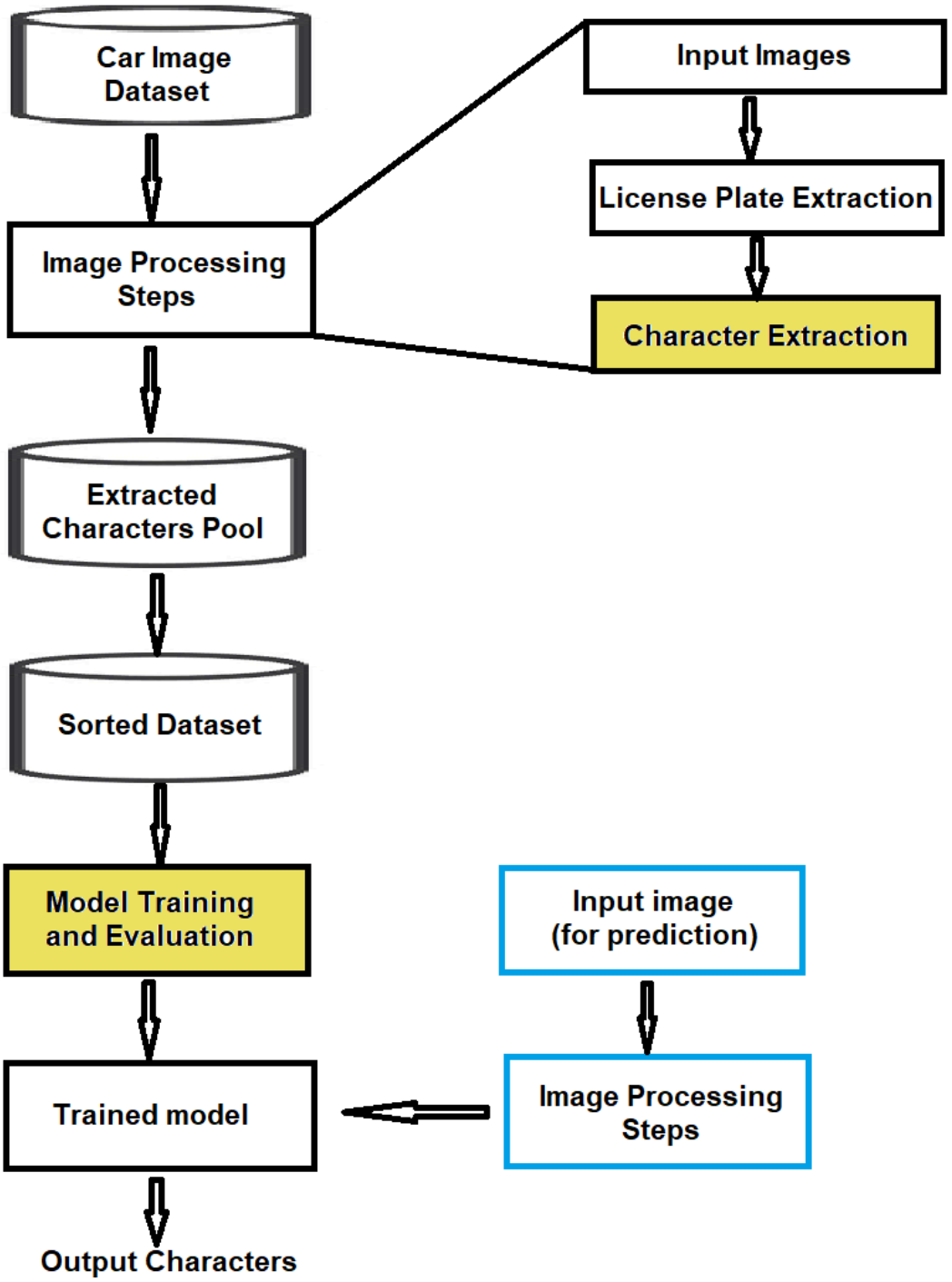

Fig 2

The very first step was to create a dataset, since the project is curated for the Indian traffic system (domestic vehicle AA-12-AA-1234 format), it was important to use the recently introduced **high-security number plates**.
Since the data from traffic cameras is considered a sensitive data as it mentions the time and location of the vehicle images it captures and the new system of number plates being recently introduced, the dataset creation needed to be done from scratch. Thus the first step was to collect images from the streets. The images then needed to be processed to extract characters from the plates. These characters were then sorted into 36 categories i.e. the 10 digits and 26 alphabets. The entire process was done manually, the details of the final dataset are:
Total Images used to train license plate detection model: 1500
Images to train character recognition model: 250 Indian vehicle images
Training images per character: 200 X 36
Test images per character: 50 X 36

The image processing steps involved:
1. Detection of number plates from images of vehicles:
This step was initially done using functions of cv2 library, the bounding box function was used after cleaning of the image using various IP techniques (i.e noise reduction using a bilateral filter, canny edge detector, finding contour, cropping an image for a license plate). Since this methodology was not as efficient as required and failed in many cases, it was replaced by a YOLO model trained on car images in order to detect license plates out of them.

2. Detection of characters in the license plates:
This step was done entirely based on the image processing techniques. It involves:
- Converting the image to grayscale and resing it
- Applying gaussian blur to smoothen the image
- Applying dilation to make regions more clear
- Finding contours and sorting them
- Finding contours of interest using height-width ratio and area

This gives us the bounding box around the characters of the plate.

The above-mentioned image processing steps give an output of characters recognized from the images provided. These characters are then used to train

and test over the character recognition model used. We picked four models to be used for character recognition steps which are:

1. Simple Neural Network:
Input Size: 28X28 images
Layers: 3 layers:
1 input (Flatten)
1 hidden (dense, 100 neurons, relu activation function)
1 output (dense, 26/10 neurons, sigmoid activation function)

2. Convolution Neural Network:
Input Size: 28X28 images
Layers: 10 layers

input_20 (InputLayer)	[(None, 28, 28, 1)]	0
conv2d_41 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_22 (MaxPooling2D)	(None, 13, 13, 32)	0
dropout_22 (Dropout)	(None, 13, 13, 32)	0
conv2d_42 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_23 (MaxPooling2D)	(None, 5, 5, 64)	0
dropout_23 (Dropout)	(None, 5, 5, 64)	0
conv2d_43 (Conv2D)	(None, 3, 3, 128)	73856
flatten_19 (Flatten)	(None, 1152)	0
dense_35 (Dense)	(None, 10)	11530

3. K Nearest Neighbors:
Value of k = 3

4. Tesseract:
The standard tesseract model without any modifications is used here.
The final output of the entire pipeline used is a series of characters i.e. the registered license plate number of the vehicle.

RESULTS and CONCLUSION

Table1 summarises the training and test accuracies of the four models used in the project and also the time taken in training of the models.

Model	Train Accuracy (%) (Digits / Alphabets)	Loss (Digits / Alphabets)	Epochs (Digits / Alphabets)	Test Accuracy (%) (Digits / Alphabets)	Training Time (Digits / Alphabets)
NN	99.63% / 98.71%	0.1150 / 0.1327	5 / 5	99.82% / 98.71%	7.42 sec / 4.2338 sec
kNN	_____	_____	_____	99.65% / 98.07%	0.0157 sec /0.0076 sec
CNN	97.66% / 97.27%	0.0624 / 0.0682	20 / 20	98.64% / 97.43%	12.04 sec / 20.36 sec
Tesseract	*	*	*	99.56% / 98.34%	*

* online available weights of pretrained model used

Training images per character: 200 X 36
Test images per character: 50 X 36

The simple neural network model shows the best accuracy while the kNN model shows the least training time. Thus the kNN model is most practical to be used as it shows both good test accuracy and less training time and hence can be used for hefty datasets like that of our Indian traffic management system.

BIBLIOGRAPHY

1. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
2. https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006
3. https://appsilon.com/object-detection-yolo-algorithm/
4. https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006
5. https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
6. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7078657&casa_token=5GLpZyOcXg0AAAAA:eaGRF0gSKOqfD4LeABMQ-_62wDHOmsoeP_DPyMYznsXSp8wq7rfzca-5ZUFbBBUvApzKGZZ06cg
7. https://iopscience.iop.org/article/10.1088/1757-899X/1084/1/012027/pdf#:~:text=In%20%5B3%5D%2C%20an%20automatic,Region%20of%20Interest%20(RoI).
8. https://towardsdatascience.com/automatic-license-plate-detection-recognition-using-deep-learning-624def07eaaf
9. https://www.pyimagesearch.com/2020/09/21/opencv-automatic-license-number-plate-recognition-anpr-with-python/
10. https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm
11. https://www.tutorialspoint.com/tensorflow/tensorflow_convolutional_neural_networks.htm
12. https://purnasaigudikandula.medium.com/a-beginner-intro-to-neural-networks-543267bda3c8
13. https://algorithmia.com/blog/introduction-to-loss-functions
14. https://medium.com/swlh/an-introduction-to-neural-networks-de70cb4305f9

15.https://medium.com/fintechexplained/understanding-neural-networks-98e94251fb97

16.https://www.researchgate.net/profile/Chirag-Patel-12/publication/235956427_Optical_Character_Recognition_by_Open_source_OCR_Tool_Tesseract_A_Cas

e_Study/links/00463516fa43a64739000000/Optical-Character-Recognition-by-Open-source-OCR-Tool-Tesseract-A-Case-Study.pdf

17. https://ieeexplore.ieee.org/abstract/document/4376991