License Plate Recognition

(DS226 Project)

Problem Statement

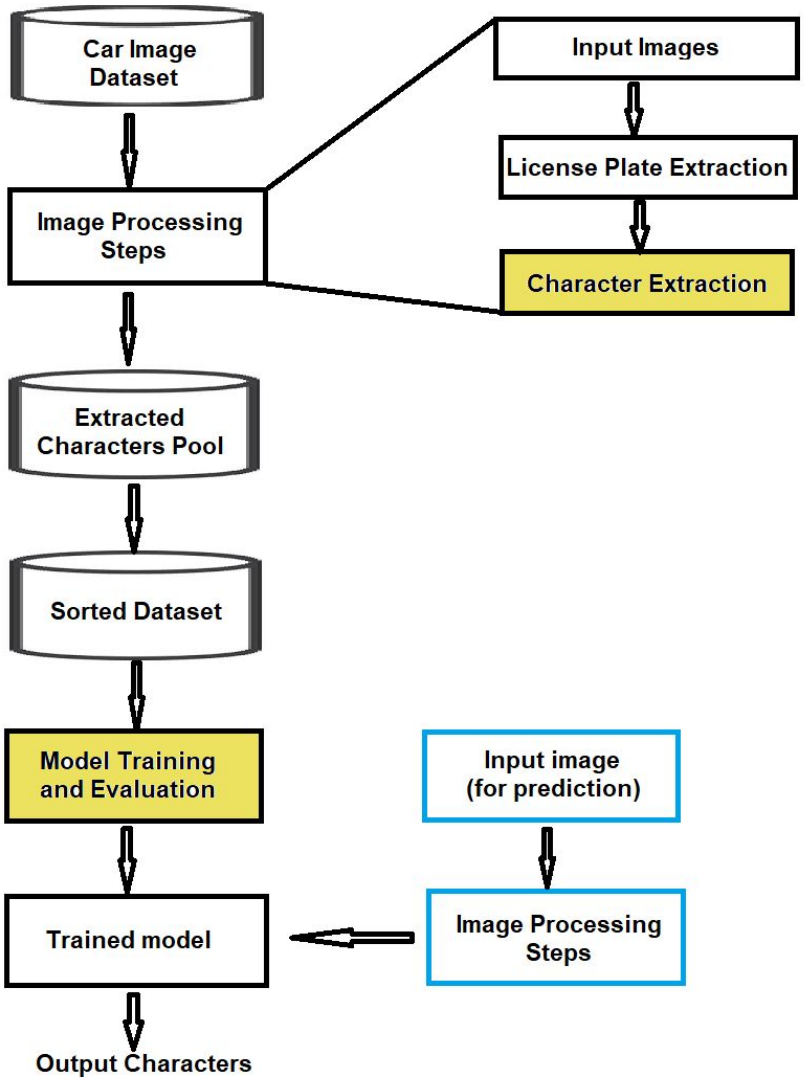
Learning practical implementation of different aspects of machine learning pipeline via developing an efficient **Licence Plate Recognition System**.

Input

Image Processing Output

7JCV393

Final Output



Pipeline of Model and Current Progress

- License plate extraction using YOLO
- Character extraction using IP entirely
- Character recognition using:
  - simple NN
  - CNN
  - kNN
  - Tesseract

Dataset details:
- Car Image Dataset:
 1500 images (for YOLO)
 250 images (for character recognition)
 - Sorted dataset:
36 Classes (10 digits+ 26 alpha)
(200 train + 50 test)

Steps Involved in the Pipeline

Input image

Detected License Plate (using YOLO)
Total params - 64003990
Trainable params - 63937686

Initial Version of License plate detection
- bilateral filter (noise reduction)
- canny edge detector
- finding contours
- finding best contours to suit license plate size
-cropping image accordingly

Image Processing Steps for Character Detection

Gray image

1. Resizing and converting to grayscale

Gaussian blur to smoothen image

2. Using gaussian blur

Otsu Threshold

3. Thresholding (Otsu)

Dilated image

4. Dilation

5. Finding and sorting contours

6. Finding contours of interest

Detected Character

Details of Models used

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

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 licence plate number of the vehicle

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

Results and Conclusion

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

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

Possible Future Work

- Using image processing to detect difference in vehicle from registered model thus helping in -

    Theft detection
    Detection of illegal changes in vehicles
    Detection of overloaded vehicles

- Similar techniques can be used near toll booths to detect vehicles with non-functional brake lights which are a major cause of accidents at nights.