

Car Number Plate Recognition System

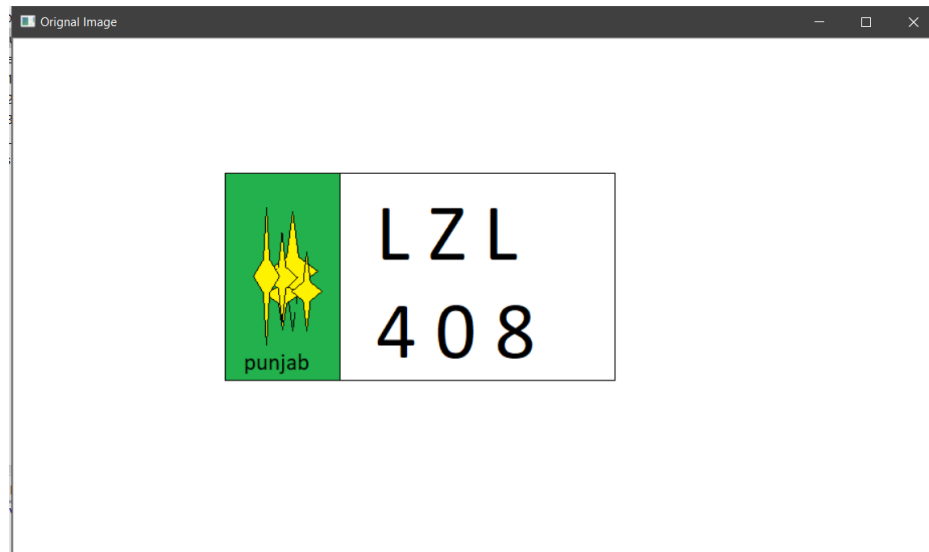
Q2. Design and implement Car Number Plate Recognition System using Artificial Neural Network for computerized plates of Punjab (Lahore). Provide solution for recognition of any other car registration plate e.g., Sindh, Baluchistan, Government cars, commercial cars, customized plates etc.

This code submitted consists of two files

1. Car number plate recognition.py —ANN is implemented using back-propagation algorithm. Testing & training
2. imageProcessing.py ---- detects and extracts the number plates. Also detects individual characters in the plate.

Main function is defined in Car number plate recognition.py. In the filename variable defined in main, the image of the car whose number plate is to be detected is provided.

The code uses a .jpg or .png file of the car provided. The limitation is that it fails to detect the license plate if the lighting is dimmed or there is noise in the picture. Code loads the image. Here image 'input1.png' is loaded. The original image is displayed upon loading in the program. In the image processing discussion below two images of number plates are considered as examples





Data sets

The data set utilized for training consists of the fonts used in number plates. Some of this data from collected from the internet and some of it was generated by our own selves. Corresponding to each character, 4 images are provided for training purpose.

Algorithm

This algorithm uses artificial neural networks for optical character recognition. It makes use of the back-propagation algorithm to train the network. The architecture is defined in sections followed. It uses the perceptron learning rule to learn. Back-propagation is just a way of propagating the total loss back into the neural network to know how much of the loss every node is responsible for, and subsequently updating the weights in such a way that minimizes the loss by giving the nodes with higher error rates lower weights and vice versa.

This document covers the following.

- Image processing and number plate extraction
- Training
- Testing
- Processing on detected characters

To see the implementation, refer to the code provided.

IMAGE PROCESSING:

Following are the steps taken to extract the number plate from the image. This extracted number plate is then used to detect the car registration number.

1. The first step in image processing is to remove all color from the image. This step converts the image from RGB to grayscale. This is done using OpenCV `cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)`.

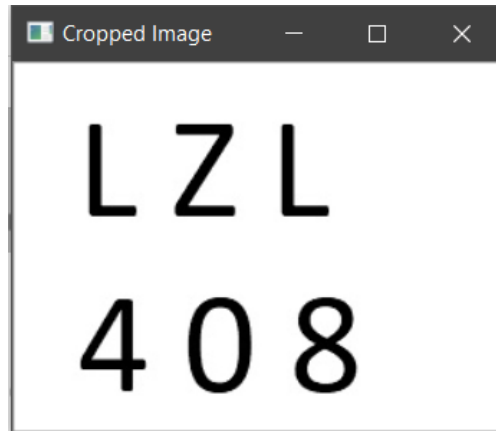


2. The next step in this regard is the noise removal from the image. For this purpose, bilateral filter is applied which removes noise while preserving the edges.
`cv2.bilateralFilter(gray, 11, 17, 17)`
3. Next step is to detect the edges of grayscale image. `cv2.Canny(gray, 170, 200)`



4. This is followed by detecting the contours in the image. These contours are detected on the basis of the edges detected beforehand. A copy of the image is saved and contours are drawn on it afterwards.

5. Contours are sorted based on their area. The required portion retained is minimum required area. We have specified 30 as the minimum required contours area. anything smaller than this will not be considered.
6. Contours are sorted and top 30 of them are selected. Then we loop over our contours to find the best possible approximate contour of number plate.
7. Select the contour with 4 corners. This contour is cropped and stored as 'cropped.jpg'. the selected contour is then drawn on the original image and overwritten on 'cropped.jpg'. the cropped image can be observed as under.



This cropped image is now fed to the testing function. Where using artificial neural network we detect the image on a previously trained network. The details of it are provided in the sections following.

TRAINING

This algorithm is trained to detect 36-characters A-Z & 0-9. The data set used is these 36 characters in various fonts. 5 images corresponding to each character are used. These includes the fonts typically used on license plates.

We have set the learning rate to be 0.5 and the number of neurons in hidden layer are defined to be 100. the higher the learning rate, the faster the convergence to the global minimum. However, if the learning rate is very high like 1 or 2, the network never converges. We calculate the error at each output neuron for which a function is defined.

Initially the weights from input to hidden layer and hidden to output layer are defined randomly. However, with each iteration they are adjusted accordingly.

the training method used here is the Back-Propagation algorithm, capable of training multi-layer networks.

The neural network contains 3 layers.

- an input layer (input image pixels serve as the input to it)
- a hidden layer
- output layer

Each neuron in the output layer categorizes the input set to a classified output. Meaning, for a character 'A' to be trained, only the neuron which classifies character 'a' fires a value close to 1, the rest fires a value close to 0.

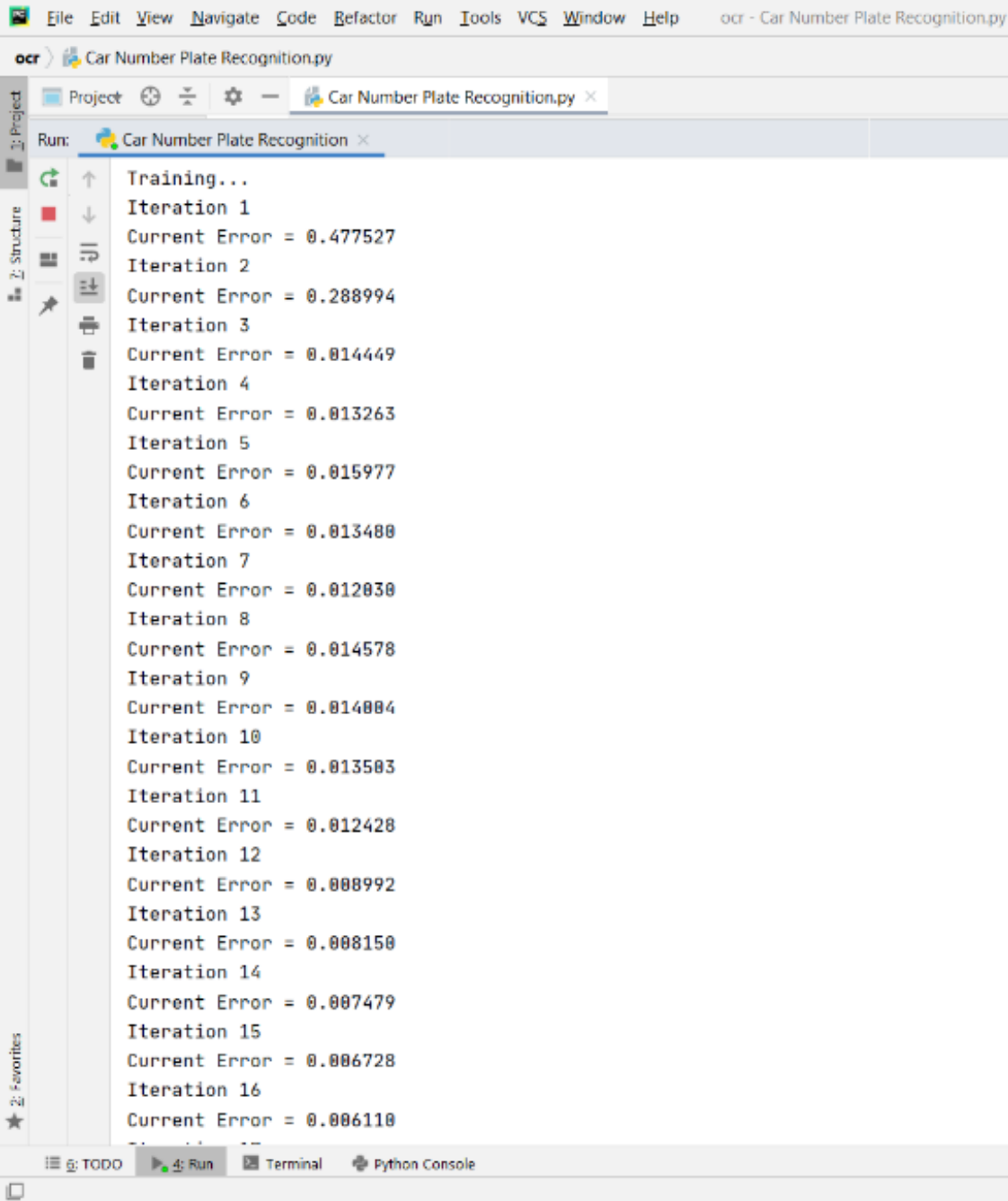
We adopt the technique called gradient decent i.e. decreasing the mean squared error. Back propagation algorithm is used so as to adjust the network's weights in such a way that the total error converges to a global minimum. The weights and biases are initialized to random numbers between -0.5 and 0.5.

Pseudocode

```
Initialize weights and biases ()
while Total Error > Target Error
    for each element E in set X
        target output = {1 for neuron Z classifying E; 0 otherwise}
        feed forward through network ()
        calculate total error ()
        back propagate through net and adjust weights ()
loop until criteria is met
```

After the network is trained on all elements in "X" it can be used for recognition. The number of output neurons depends on the number of characters to be classified. Thus, If X characters are to be classified then X neurons are present. As for the number of inputs, the network has M*N inputs, where M is the width and N is the height of the character's image. The hidden layer has 100 neurons.

Shown below is a representation of the network under training. We have set the target error to be 0.0035. On each iteration, mean squared error is calculated and iterations continue until the target is achieved.



```
ocr - Car Number Plate Recognition.py
ocr > Car Number Plate Recognition.py
Run: Car Number Plate Recognition x
Training...
Iteration 1
Current Error = 0.477527
Iteration 2
Current Error = 0.288994
Iteration 3
Current Error = 0.014449
Iteration 4
Current Error = 0.013263
Iteration 5
Current Error = 0.015977
Iteration 6
Current Error = 0.013480
Iteration 7
Current Error = 0.012030
Iteration 8
Current Error = 0.014578
Iteration 9
Current Error = 0.014004
Iteration 10
Current Error = 0.013503
Iteration 11
Current Error = 0.012428
Iteration 12
Current Error = 0.008992
Iteration 13
Current Error = 0.008150
Iteration 14
Current Error = 0.007479
Iteration 15
Current Error = 0.006728
Iteration 16
Current Error = 0.006110
```

TESTING

For testing, we first need to identify the characters in the cropped image. Then we use our trained model to check what characters those are. The cropped image is scanned to detect the lines. These lines are then scanned to detect the characters and then ultimately individual alphabets. these characters are the input to the neural network. The network will output a digital character matching the character in the image.

These scanned characters are then sent for preprocessing.

PREPROCESSING ON CROPPED IMAGE

the number of inputs to the neural network must be fixed. to allow such uniformity, the image is resized to a fixed width and height.

Following are the procedures which process the cropped image

1. **Conversion to black and white:** a method of converting the image into black and white. (i.e. black = 1, white = 0)
2. **Cropping:** detect the character's borders of the image (i.e. left, right, top and bottom borders).
3. **Normalization:** Resize to desired width and height.

Cropping and Normalizing is done on the image to have a uniform input to the ANN. After preprocessing, the detected characters in the image are inputted to the neural network to be recognized. The results obtained can be viewed as under

