License Plate Matching Using Neural Networks

Kelvyn SOSOO (GMU) David OUYANG (CSUST) Mengjun WANG (CSUST)
Mentors: Lee HAN (UTK) & Kwai WONG (UTK)

Overview

- **License Plate Recognition (LPR) technology is used to gather vehicle location data**
- **Location Data includes instances of Amber Alerts, Toll Roads Speed/Travel Time, etc.**
- **The License Plate Matching (LPM) method incorporated includes a 97% match rate of vehicles, and a 60% read accuracy**
- **Programs Used: Python, Matlab**

GOAL: Raise the 60% by using Image Processing. Find a new measure to matching plate by using supervised learning.

How It Works

Capture images → Image recognition → Plate matching

LPR 1

Procedure

Screen the License Plate images

Image Processing to segment every Character

Neural network training

Develop New method of matching

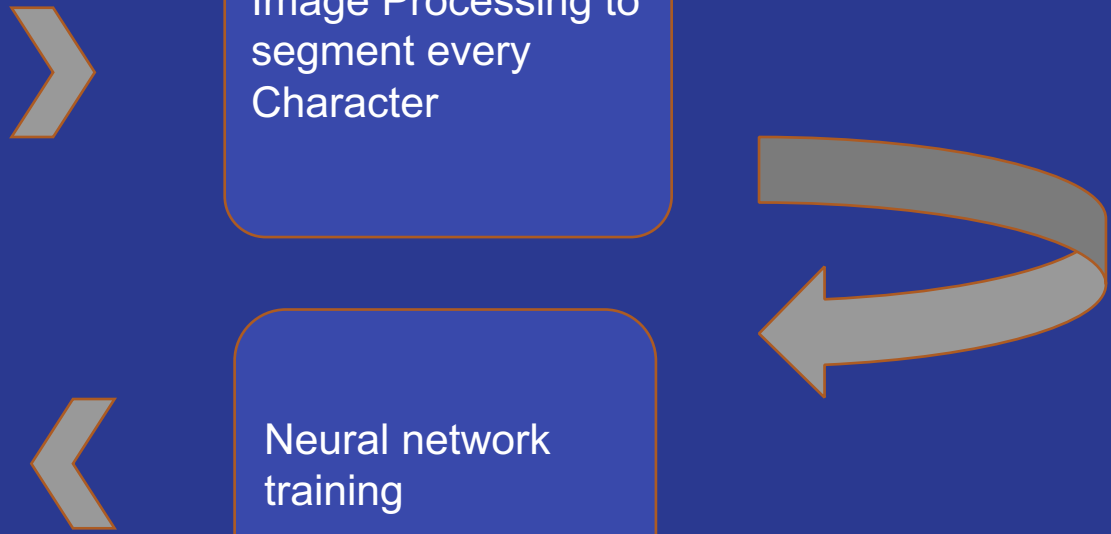
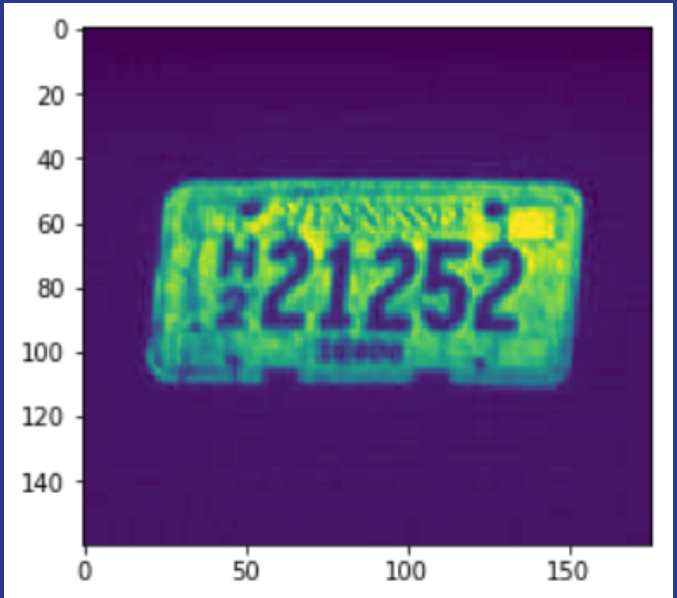


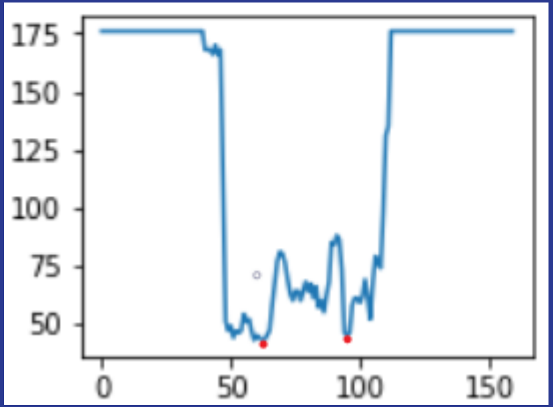
Image Processing

Step 1 : Image binarization

Method: cv2.THRESH_OTSU



Step 2 : Read the number of black pixels in the y direction

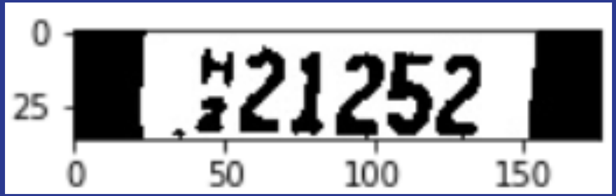


[176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176,
176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176,
176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176,
176, 176, 176, 176, 176, 168, 168, 168, 166, 170, 166, 168,
108, 51, 47, 49, 44, 47, 46, 47, 54, 51, 52, 47, 43, 45, 44,
43, **43**, 45, 47, 58, 67, 77, 81, 80, 76, 69, 63, 60, 64, 64, 60,
64, 68, 64, 67, 61, 66, 57, 60, 55, 62, 68, 85, 84, 88, 86, 73,
46, **45**, 46, 58, 61, 61, 59, 62, 69, 61, 52, 68, 79, 76, 74, 98,
131, 135, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176,
176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176,
176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176,
176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176, 176,
176, 176]

np.argmin(row_nz[0 : floor(len(row_nz)/2)]) == 59 row_nz(59) = 43

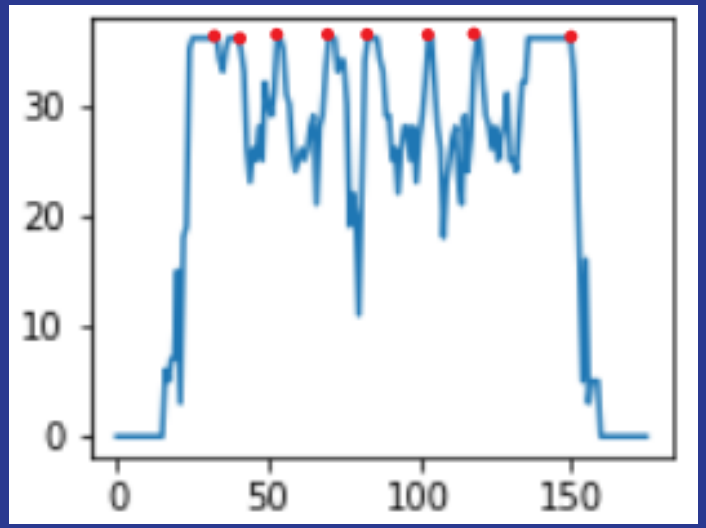
np.argmin(row_nz[floor(len(row_nz)/2) :]) == 95 row_nz(95) = 45

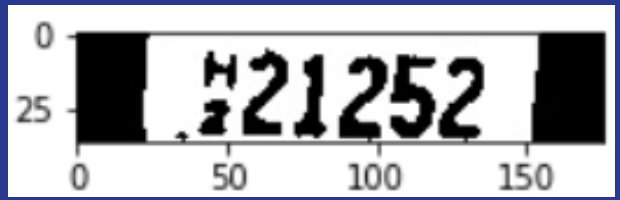
(59, 43) (95,45)



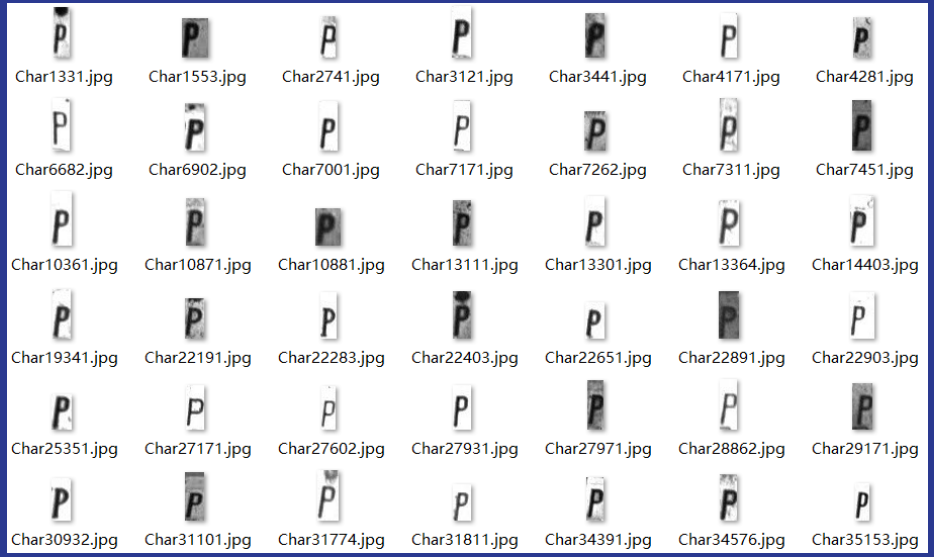
Step 3 : Read the number of white pixels in the x direction

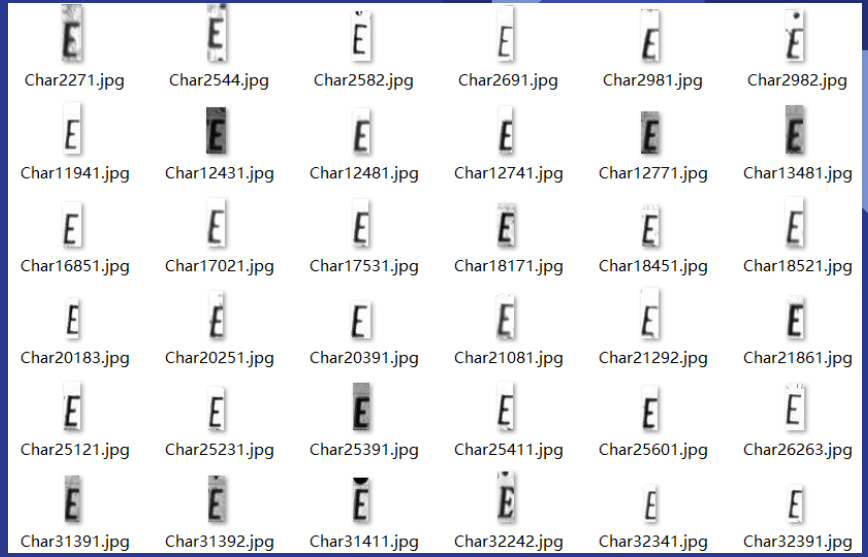
KEY POINT & CUT POINT : [33, 40, 54, 72, 86, 104, 120, 150]

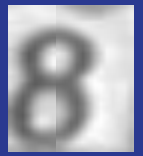


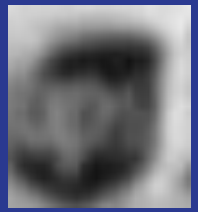


Final Outcome





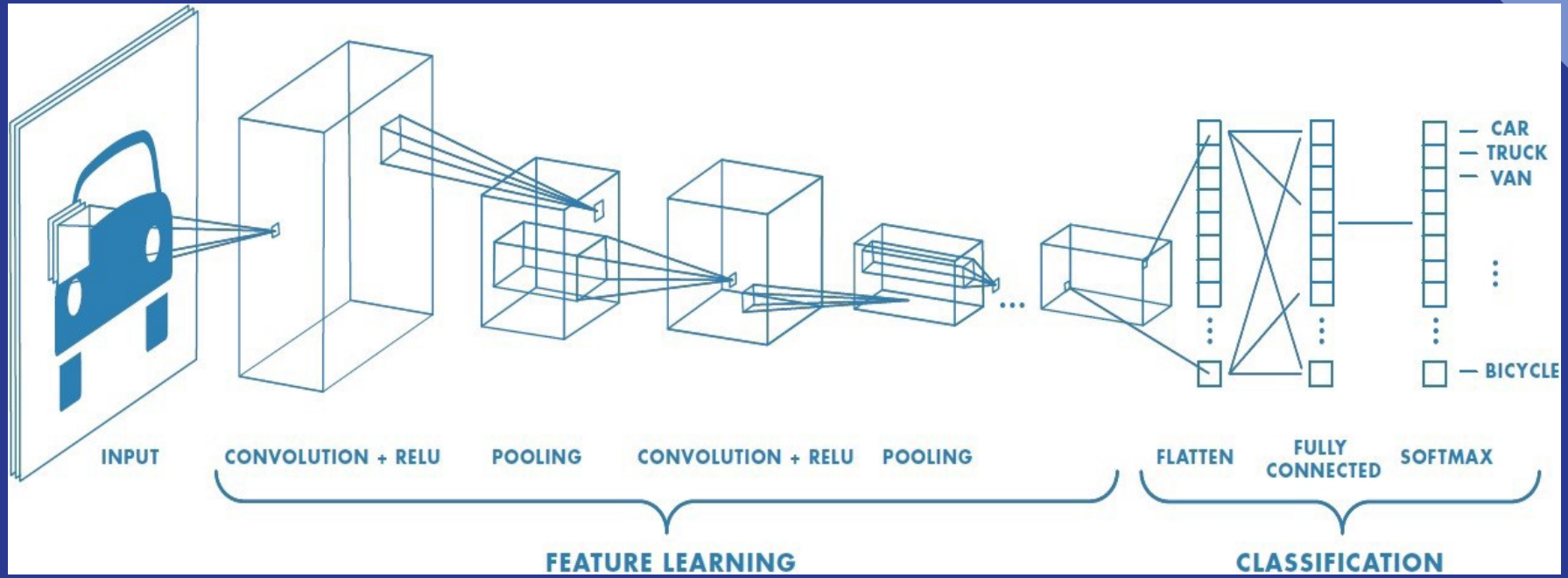




Supervised Learning: Neural Network

- Previous slide presented the outcome of Character Segmentation
  - It is very time consuming to transfer the characters to the proper label/category
- Instead of spending countless hours manually moving files, Data Augmentation was implemented
- Categories included A-Z and 0-9

Convolutional Neural Networks



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Performance

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
                metrics=['accuracy'])

model.fit(X,y, batch_size=32,epochs=4, validation_split=0.15) #number of samples to be passed at a time

model.save('64x3-CNN.model')
```

```
Using TensorFlow backend.

WARNING:tensorflow:From C:\Users\Kelvyn\Anaconda3\lib\site-packages\tensorflow\python\ops\resource_variable_ops.py:435: colo
cate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\Kelvyn\Anaconda3\lib\site-packages\tensorflow\python\keras\layers\core.py:143: calling drop
out (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
Train on 31722 samples, validate on 5599 samples
Epoch 1/4
31722/31722 [==============================] - 125s 4ms/sample - loss: 1.7344 - acc: 0.5150 - val_loss: 0.6502 - val_acc: 0.
8328
Epoch 2/4
31722/31722 [==============================] - 121s 4ms/sample - loss: 0.5751 - acc: 0.8322 - val_loss: 0.3054 - val_acc: 0.
9198
Epoch 3/4
31722/31722 [==============================] - 120s 4ms/sample - loss: 0.3669 - acc: 0.8894 - val_loss: 0.2288 - val_acc: 0.
9412
Epoch 4/4
31722/31722 [==============================] - 120s 4ms/sample - loss: 0.2713 - acc: 0.9178 - val_loss: 0.1887 - val_acc: 0.
9518
```

- After four epochs, the model was able to reach a validation accuracy of 95.18%

Next Step

- Think of a way to store License Plates with this method
- Find way to string together characters for Matching step

Plate Matching

Ex: the tag of a license plate is **4455HZ**
LPR 1 reads the plate as **44S5H2**,
LPR2 is **4455HZ**

Goal: To judge whether Different plate characters are from the same car

Constraints

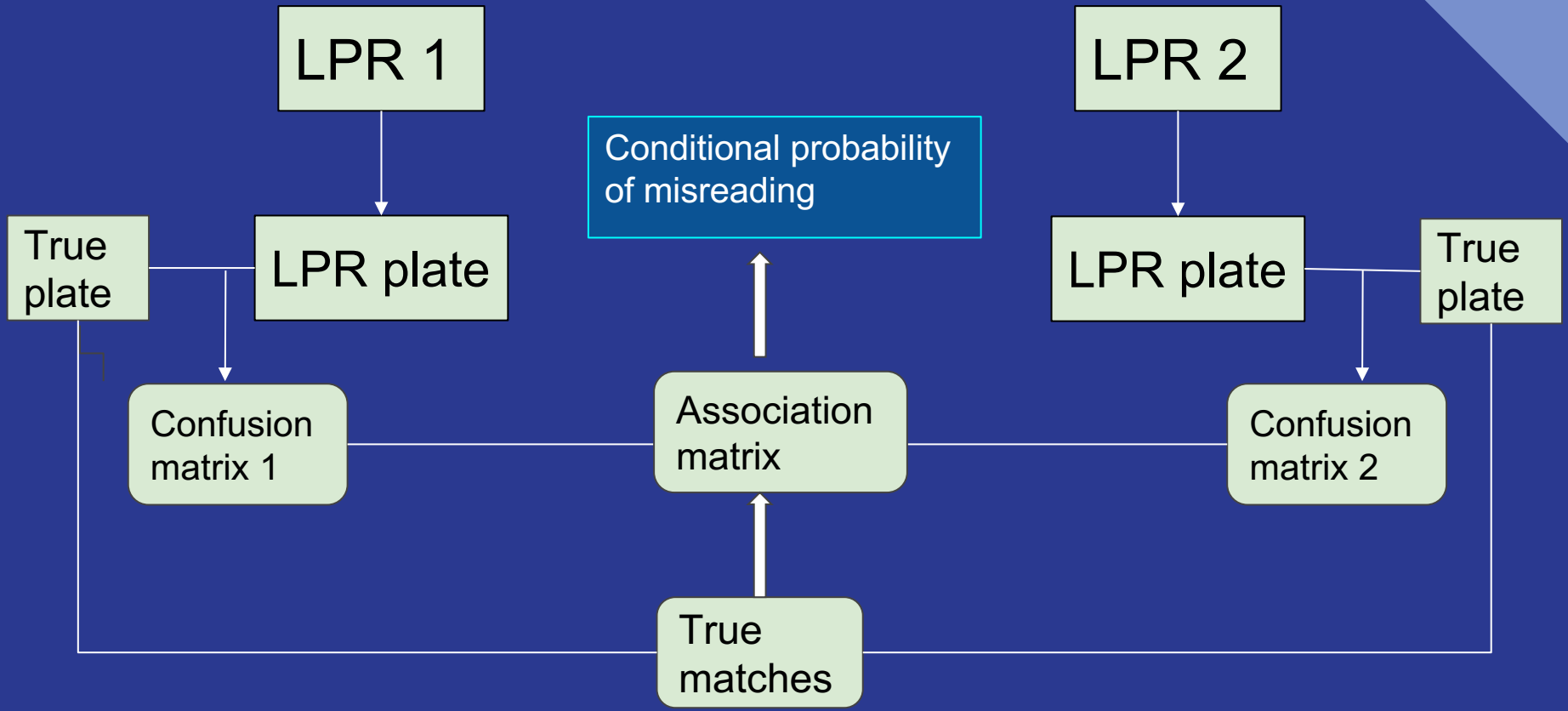
1. The travel time of LPR1 to LPR2

2. The ED of plate 1 change to plate 2

3. The conditional probabilities of character transition

Less ED, higher probability, in the time period

Association Matrix



Edit Distance Operation

$$d(x \to y) = \min\{d(i-1, j-1) + \gamma(x_i \to y_j), d(i-1, j) + \gamma(x_i \to \varepsilon), d(i, j-1) + \gamma(\varepsilon \to y_j)\}$$

Association Probability

$$p(y|x) = \frac{p(x,y)}{p(x)} = \frac{\sum_t p(x,y|t)p(t)}{\sum_{y,t} p(x,y|t)p(t)}$$

$$p(y|x) = \sum_t p(y|t)p(t|x)$$

Self-learning Matching

C(M) represents calculate association matrix from a set of matches M

1) k = k + 1;
2) Mk = M(Ck−1);
3) Ck = C(Mk);
4) Stop if Ck − Ck−1 < ε.

Research Plan

1.Work on ED algorithm, self-learning algorithm of matching

2.Work on the matching part and weight function by python

3.Calibrate the parameters and validate them

Done

1.Read papers

2.Determine the project frame

Next step

Learn matlab and transform the matlab code into python code

Challenges

Challenges Already Faced:

- Sufficient Training Data
- Character Cutting Details
- Clearer Images
- Handling of Similar Characters

Foreseeable Challenges:

- Adapting Matlab Code
- Meshing Network and Matching

THANKS FOR LISTENING, ANY QUESTIONS?