License Plate Matching Using Neural Networks

Kelvyn Sosoo (GMU), David Ouyang (CSUST), Mengjun Wang (CSUST)

Mentors: Dr. Lee Han (UTK), Dr. Kwai Wong (UTK)

Abstract

Neural Networks are systems that can be trained to predict outcomes based on what they've "learned" or been trained on. This project focuses on a semi-automated license plate matching procedure. In practice, with License Plate Recognition (LPR) technology, plates will be captured at two seperate locations to track metrics such as travel time and potential traffic. This work remains relevant to the Department of Transportation, as well as cases of Amber Alerts.

Background

Challenges arise with different fonts and designs across multiple states. Our goal is improve the accuracy of image recognition by data training, so that we can decrease the false positives. We also want to improve the efficiency of license matching through modern means.

Procedure

Preparing (MATLAB)

- Original Images are manually filtered to remove irrelevant images
- Date and time are extracted from file name and recorded into .csv file

Cutting (Python)

- Images binarization by Otsu's method
- Binary images are read horizontally and vertically
- Each character of every license plate is segmented and saved into a designated folder

Training (Python)

- Segmented characters are manually selected to be part of the training dataset
- Data Augmentation is incorporated to save several hours of labor
- The training function is created and the images are trained to be properly classified

Matching (Python and MATLAB)

- Model is used to predict characters and are recorded as a string into the .csv file
- License Plates are matched between LPR1 and LPR2 with two methods; Fuzzy Learning and MATLAB


```
cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
```

Figure 1: Original Image, and Otsu's method to get Binarized Equivalent

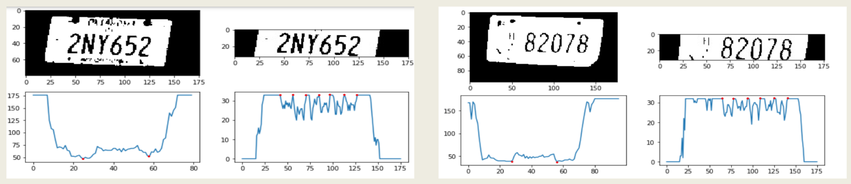

Figure 2: Read pixel along x and y axis and capture the red key points

```
idx1 = np.argmin(row_nz[0:floor(len(row_nz)/2)])
idx2 = np.argmin(row_nz[floor(len(row_nz)/2):])
```

Figure 3: Cutting code and cutting results in each folder

	FOLDER	DATE	TIME	LPR
2	LPR02_04072010	04/07/10	0:01:02	49428HZ
3	LPR02_04072010	04/07/10	0:02:41	1891
4	LPR02_04072010	04/07/10	0:02:41	BX1891
5	LPR02_04072010	04/07/10	0:02:41	S11891
6	LPR02_04072010	04/07/10	0:02:51	ZLI555
7	LPR02_04072010	04/07/10	0:05:25	9NZ853
8	LPR02_04072010	04/07/10	0:06:01	ME6STU
9	LPR02_04072010	04/07/10	0:06:15	IC1519
10	LPR02_04072010	04/07/10	0:06:15	1CISM9
11	LPR02_04072010	04/07/10	0:06:41	EWS2957
12	LPR02_04072010	04/07/10	0:09:46	AM903D
13	LPR02_04072010	04/07/10	0:09:58	H905584

Figure 4: Final .csv format for one LPR station

Training

- Dataset was augmented to include over 37,000 files for training; representing slightly over 1,000 per class
- The training model developed through Tensorflow and Keras was able to classify at a 98% success rate

Matching License Plates with Fuzzy Learning

- Fuzzy Learning is a metric based on the similarity ratio of two Strings (Levenshtein Distance); the more similar they are, then the higher the ratio
- When a car is recognized at both stations the process is able to match at a 92% rate in Python

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Figure 9: Levenshtein Disance Formula

Fuzzy Learning vs MATLAB Matching

- MATLAB code outperformed the Fuzzy Learning method due to:
  - Ability to learn based on commonly misidentified characters
  - Time Constraint consideration

Matching License Plates with MATLAB

1.Select a candidate matching set from the first station for each license plate in the second station during the time limit .

2. Calculate the cost of converting each pair of plates, i.e. the edit distance.

3. Using the edit distance to retrace the path taken to convert the pair. This path is formed by three operations: insertion, deletion, and substitution.

4. Each pair of license plates is paired, and the number of associated letters are accumulates to a 37- by-37 matrix M

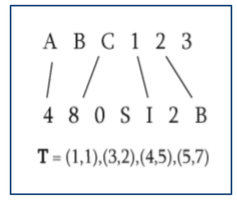

Figure 5: Trace diagram

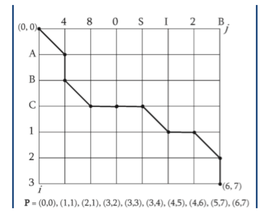

Figure 6: Path diagram

$$p(b|a) = \rho_{ab}/\rho_a$$

```
C = M./(repmat(sum(M,2),1,37));
```

Figure 7: Formula. calculate C

5. Calculate every grid in the matrix M with conditional probability to obtain the association matrix C.

6.Stop iterating until C does not change.

1) $k = k + 1$;
2) $\mathbf{M}_k = M(\mathbf{C}_{k-1})$;
3) $\mathbf{C}_k = C(\mathbf{M}_k)$;
4) Stop if $\|\mathbf{C}_k - \mathbf{C}_{k-1}\| < \varepsilon$.

7.C is used to calculate the cost of converting each pair of license plates, the smallest cost is matching license plate.

x_i	y_j	$p(y_j\|x_i)$	$\log\left(\frac{1}{p(y_j\|x_i)}\right)$
"4"	"4"	0.885	0.122
"4"	"4"	0.885	0.122
"5"	"5"	0.280	1.273
"5"	"5"	0.914	0.090
"H"	"H"	0.937	0.065
"2"	"Z"	0.055	2.906
$GED(x \to y) = \sum \log\left(\frac{1}{p(y_j\|x_i)}\right) =$			4.579

$$d(x \to y) = \min\left\{\sum_{k=0}^{n} \log\left(\frac{1}{p(i_k, j_k)}\right)\right\}$$

Figure 8: Example of edit distance calculation

Challenges and Future Work

Challenges:

- Image preprocessing
- Too much noise on plate images
- More accurate threshold value for plates under different lights

Future work:

- Improve Segmentation Process to better cut Characters
- Implement Full String analysis vs Character analysis
- Improve Matching Speed for Real Time

Acknowledgements

This project was sponsored by the National Science Foundation through Research Experience for Undergraduates (REU) award, with additional support from the Joint Institute of Computational Sciences at University of Tennessee Knoxville. This project used allocations from the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the National Science Foundation. In addition, the computing work was also performed on technical workstations donated by the BP High Performance Computing Team.