License Plate Recognition and Matching Using Neural Networks

Kelvyn Sosoo [1] David Ouyang [2] Mengjun Wang [2]
Mentors: Dr. Kwai Wong [3], Dr. Lee Han [3], and Zhihua, Zhang

August 3, 2019

1.0 Abstract

This work outlines a 10-week project assigned within the National Science Foundation's *Research Experiences in Computational Science, Engineering, and Mathematics* (RECSEM) for Undergraduates program. Detailing an approach to license plate recognition and matching using Neural Networks, this system is intended to aid in traffic engineering aspects. License plate recognition and matching is a long-studied field that dates back approximately 50 years. Difficulties with this challenging project in the United States of America stems from the wide variety of license plate options offered by every state, as well as privacy issues. Our solution proposes the usage of a character segmentation technique, a Neural Network, and multiple matching options.

2.0 Introduction

License Plate Recognition (LPR) implements computer vision tactics to identify license plates on vehicles. Technology within cameras mounted on the shoulders of highways captures cars license plates. This is generally the origin of all license plate matching projects. Although, due to time restrictions of this program, data used in this project was collected in prior research performed by Dr. Han Lee in *An Online Self-Learning Algorithm for License Plate Matching* [3]. The technology not only takes several images of each license plate with different light intensities to find the ideal image but also the time, down to the millisecond, when the vehicle is passing through the station. This process is performed at two separate locations, which in the case of our research is a distance of 3.00 miles on interstate 40 in Tennessee. After the plates have been captured at each LPR station, the matching process can now begin.

3.0 Overview

The objective of this project was to find a new approach to license plate matching. Benchmark comparisons included a sixty percent readability rate for the license plate string, and a ninety-seven percent matching rate. The work discussed in this paper did not successfully reach these benchmarks.

3.1.1 Manipulation of Data

Due to the imperfections that come with technology, hundreds of images were captured that were not properly screened and were not images of license plates. Images included different sections of cars, as well as truck identification numbers, mistaken as license plates. To proceed

1.George Mason University
2.Changsha University of Science and Technology
3.University of Tennessee

with the process, it is necessary to screen out irrelevant images. Following the removal of these images, we find it essential to rename every available image as well. It is worth mentioning that prior to renaming, we need to read and save the file name of each image because it contains the specific time and date when the license plates were captured at the LPR stations.

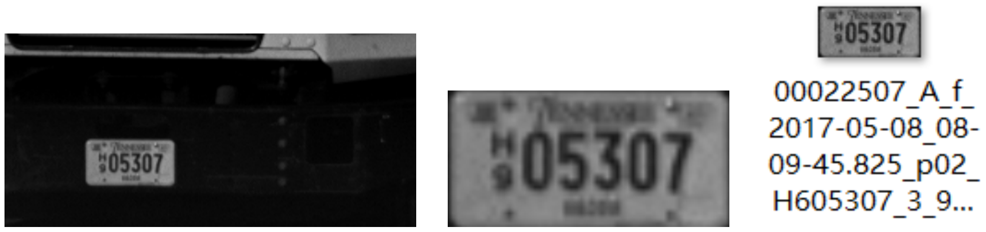


Figure 1: Original plate and Files name

3.1.2 OTSU Threshold for Binarization[1]

We perform image binarization as a precaution for more convenient character segmentation. Prior research methods involving binarization generally change pixels with a gray value between 127 and 255 to black and the remaining pixels to white. We chose to avoid the prior method because different light intensities of the images would make it extremely challenging to binarize all license plate images accurately. Another common binary processing method is to calculate the average gray value, K, of pixels. For each pixel value that is greater than K, set it to 255 (white), and if the value is less than or equal to K, set it to 0 (black). A more accurate method and the one we selected to use is the OTSU binarization method. The OTSU method performs thresholding automatically on images to separate the pixels into two categories, background, and foreground. The method thrives with bimodal images and performs calculations very quickly

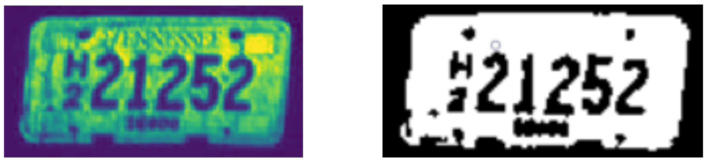


Figure 2: Original plate and Binarized plate

3.1.3 Character Segmentation

After transforming the images, they now only contain black and white pixels. Shown in the plots are how many black pixels each row has and how many white pixels each column has.

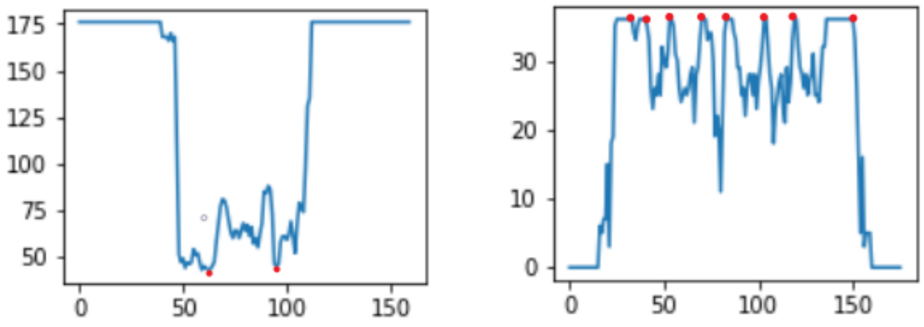


Figure 3: Image Pixel Analysis

The left plot shows the number of black pixels in each row. With the two red points indicating the region where a majority of the black pixels are concentrated. The right plot shows the number of white pixels in each column. The red key points signify the start of a new character after the white space in between is processed.

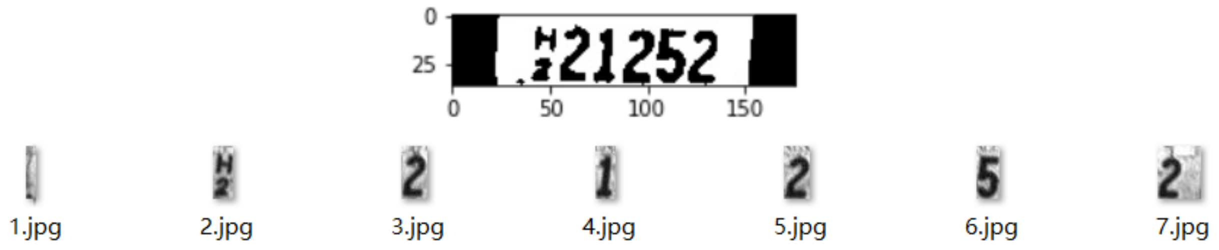
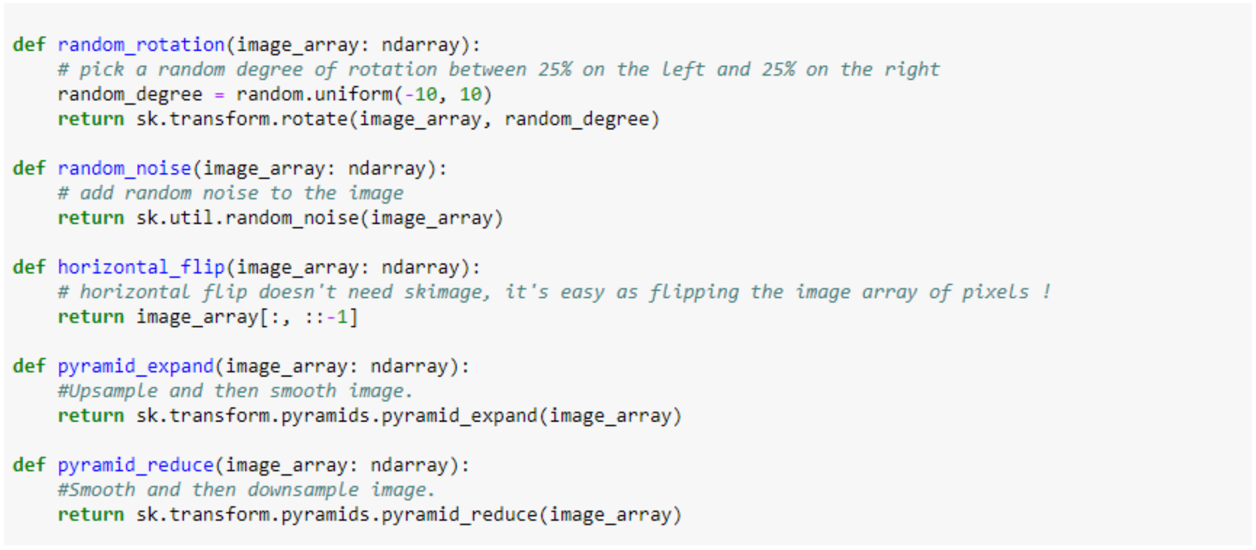


Figure 4: Outcome after Segmentation

Although this is considered a successful segmentation, there are still segmented pieces that are unneeded, such as ‘1.jpg' and ‘2.jpg.' This error is unavoidable and generally caused by noise on the plates. Under some circumstances, the noise causes two characters to be segmented as one character, and cannot be separated. This error makes it very difficult for the model to properly analyze the string correctly.

3.2.1 Data Augmentation

Due to the lengthy process of manually assigning characters to labeled 0-9 and A-Z folders, a data augmentation method was included to speed up the process. In the process to get an adequate amount of data, techniques such as random rotations, the addition of noise, and image expansion and reduction were used. These techniques helped to transform the entirety of the dataset from approximately 1,500 images to 38,000. With an adequate training dataset, the neural network was robustly trained.

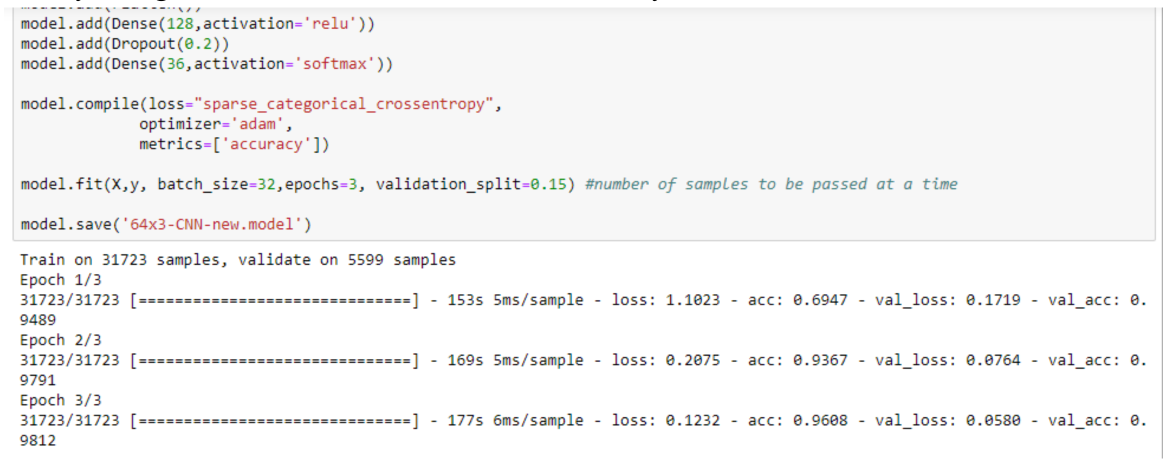


```python
def random_rotation(image_array: ndarray):
    # pick a random degree of rotation between 25% on the left and 25% on the right
    random_degree = random.uniform(-10, 10)
    return sk.transform.rotate(image_array, random_degree)

def random_noise(image_array: ndarray):
    # add random noise to the image
    return sk.util.random_noise(image_array)

def horizontal_flip(image_array: ndarray):
    # horizontal flip doesn't need skimage, it's easy as flipping the image array of pixels !
    return image_array[:, ::-1]

def pyramid_expand(image_array: ndarray):
    #Upsample and then smooth image.
    return sk.transform.pyramids.pyramid_expand(image_array)

def pyramid_reduce(image_array: ndarray):
    #Smooth and then downsample image.
    return sk.transform.pyramids.pyramid_reduce(image_array)
```

Figure 5: Data Augmentation Techniques

3.2.2 Neural Network Training

The intended approach of reading the characters derived from the license plate images is to maximize the readability of the string of characters as a whole. To maximize this potential, a Convolutional Neural Network was developed and trained using manually labeled images. Through 3 epochs of the training data, lasting slightly under 5 minutes, the model was able to efficiently recognize the characters at 98.12% accuracy.



```
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(36,activation='softmax'))

model.compile(loss="sparse_categorical_crossentropy",
              optimizer='adam',
              metrics=['accuracy'])

model.fit(X,y, batch_size=32,epochs=3, validation_split=0.15) #number of samples to be passed at a time

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

```
Train on 31723 samples, validate on 5599 samples
Epoch 1/3
31723/31723 [==============================] - 153s 5ms/sample - loss: 1.1023 - acc: 0.6947 - val_loss: 0.1719 - val_acc: 0.
9489
Epoch 2/3
31723/31723 [==============================] - 169s 5ms/sample - loss: 0.2075 - acc: 0.9367 - val_loss: 0.0764 - val_acc: 0.
9791
Epoch 3/3
31723/31723 [==============================] - 177s 6ms/sample - loss: 0.1232 - acc: 0.9608 - val_loss: 0.0580 - val_acc: 0.
9812
```

Figure 6: Outcome after Training

3.3 License Plate Reading

The model is trained to identify every potential character that may be found on a license plate and store them in a string format within a .csv file. With the license plates from two separate LPR stations stored in different cells of the file, they can then be matched using either of two different methods; a MATLAB interpretation, as well as a Fuzzy Learning approach, which involves measuring the edit distance using the Levenshtein formula.

3.4 Plate Matching (MATLAB)

We can obtain two sets of license plate numbers from two different LPR stations based on the above procedures. In this section, we try to discern whether two strings that are misread be from the same car.

3.4.1 Definition

Edit distance [2] is the cost of eliminating differences between two different strings, and there are three ways to eliminate differences: insertion, deletion, and replacement. The association matrix is a 37x37 matrix that measures the conditional probability of two characters being misread at two sites. The edit distance between them, along with the weights from the association matrix would help determine whether they are a match. The probability of confusion between different letters is different; such as 'B' to '8', and 'A' to 'B'. The probability of 'B' being read as '8' is higher than being read as 'A'. Situations as such help lead to the calculation of many weights after several iterations of the data. For example, the first letter of a certain license plate is recognized as 'A' at station 1. By calculating the association matrix, we can get the probability that the letter is recognized as '0~9', 'A~Z' or blank at station 2 respectively. Based on this, we can get the editing distance that takes into account the weight of misreading, which is more convincing than the three operations that only consider the editing distance. In this study, we believe that if there are two strings with minimum editing distance between them, then they are judged to be a set of matches.

3.4.2 Self-learning

Obtaining the ground truth of license plates from images is very time-consuming and expensive. Therefore, we use self-learning [3] and iterative methods to obtain the association matrix.

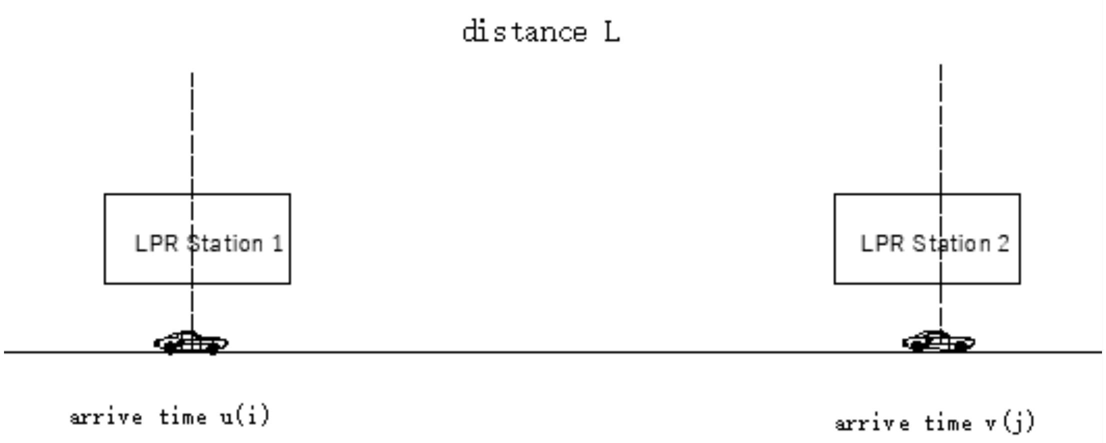


Figure 7: LPR Site Diagram

Based on the above figure, our LPR stations are located at different positions on the highway. When the same car is driving on this road, the time it takes to pass through LPR station 1 is ‘u’, and the time to pass through LPR station 2 is ‘v.’ The maximum speed of vehicles on this road is ‘max’ and the minimum speed is ‘min.’ Travel time can thus be calculated by calculating the difference between ‘u’ and ‘v.’ This difference is what is known as a “time constraint.” In order for two plates to be considered a match, they must be within this range.

$$\frac{□}{□□□} \le □(□) - □(□) \le \frac{□}{□□□} \quad (1)$$

The above formula defines time constraints.

For example, to maximize efficiency in matching these plates, the first thing taken into consideration is whether the plate fits within the time constraint of the plate read at Station 1. Then, all the potential plates are placed into a candidate set named ‘S,’ and every string in the set is listed as S(i). We pair A with S(i), and we get a pair of plates. Look for the combinations that have the smallest edit distance required to transform each other, then choose the one which shows up first.

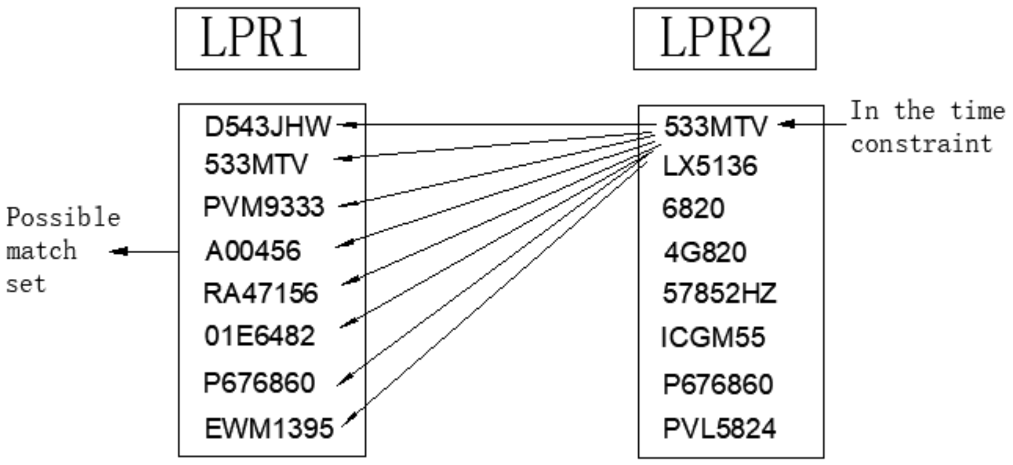


Figure 8: The candidate set.

Edit Distance details:

(1) ed ≤ $\tau_{□□□}$ where $\tau_{□□□}$ is a minimum threshold for the ED.

(2) $\tau_{□□□}$ ≤ ed ≤ $\tau_{□□□}$[3] where $\tau_{□□□}$ is a minimum threshold for the ED.

Based on the above edit distance constraints, we can figure out whether the smallest edit distance is in this range. If not, then turn to the next iteration.

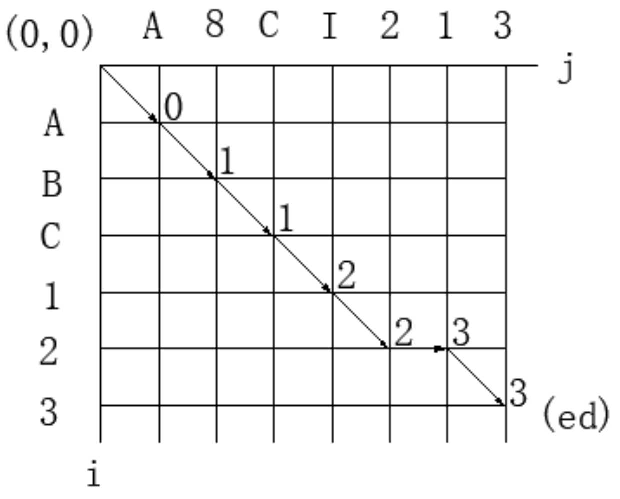
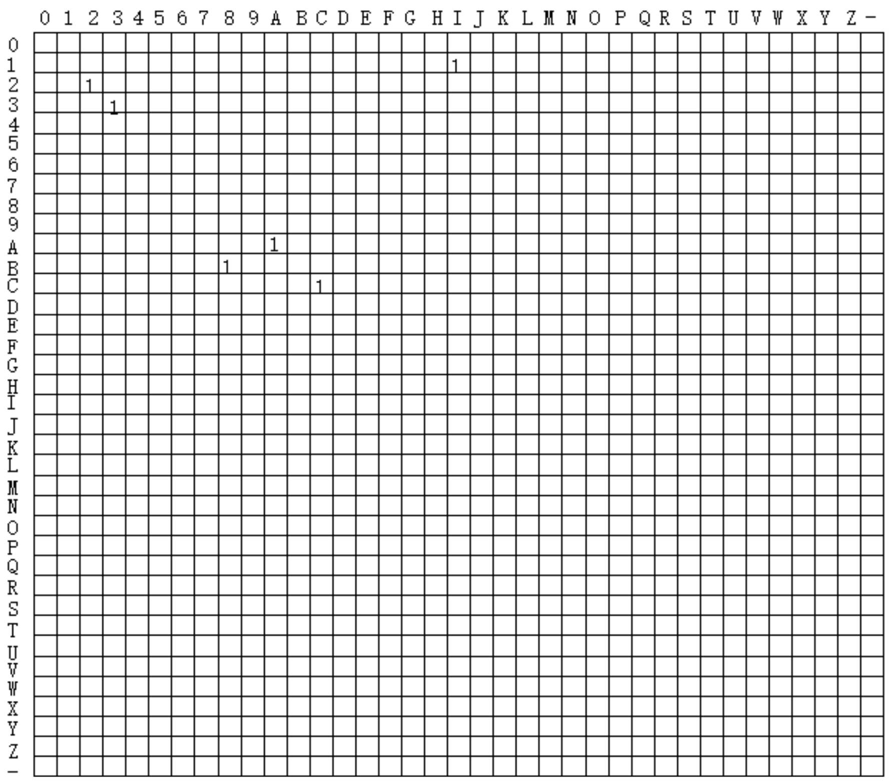


Figure 9: Example of edit distance

The above diagram described the edit distance between two different license plates and the edit paths [3]. Then, we can update the initial association matrix.

(1) Calculate the edit distance path.
(2) Find all the associated characters.
(3) Calculate the association matrix.

The matching set ‘S’ can be then used as an input to update the prior association matrix and to find an updated representation of plate patterns observed.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	-
0																																					
1																			1																		
2			1																																		
3				1																																	
4																																					
5																																					
6																																					
7																																					
8																																					
9																																					
A											1																										
B									1																												
C													1																								
D																																					
E																																					
F																																					
G																																					
H																																					
I																																					
J																																					
K																																					
L																																					
M																																					
N																																					
O																																					
P																																					
Q																																					
R																																					
S																																					
T																																					
U																																					
V																																					
W																																					
X																																					
Y																																					
Z																																					
-																																					

Figure 10: Character-transition matrix

$$\square(\square|\square) = \square_{\square\square}/\square_{\square} \qquad (2)$$

$\rho_{\square\square}$ is the value of every grid in the Character-transition matrix.

$\rho_{\square}$ is the sum of every row in the Character-transition matrix.[3]

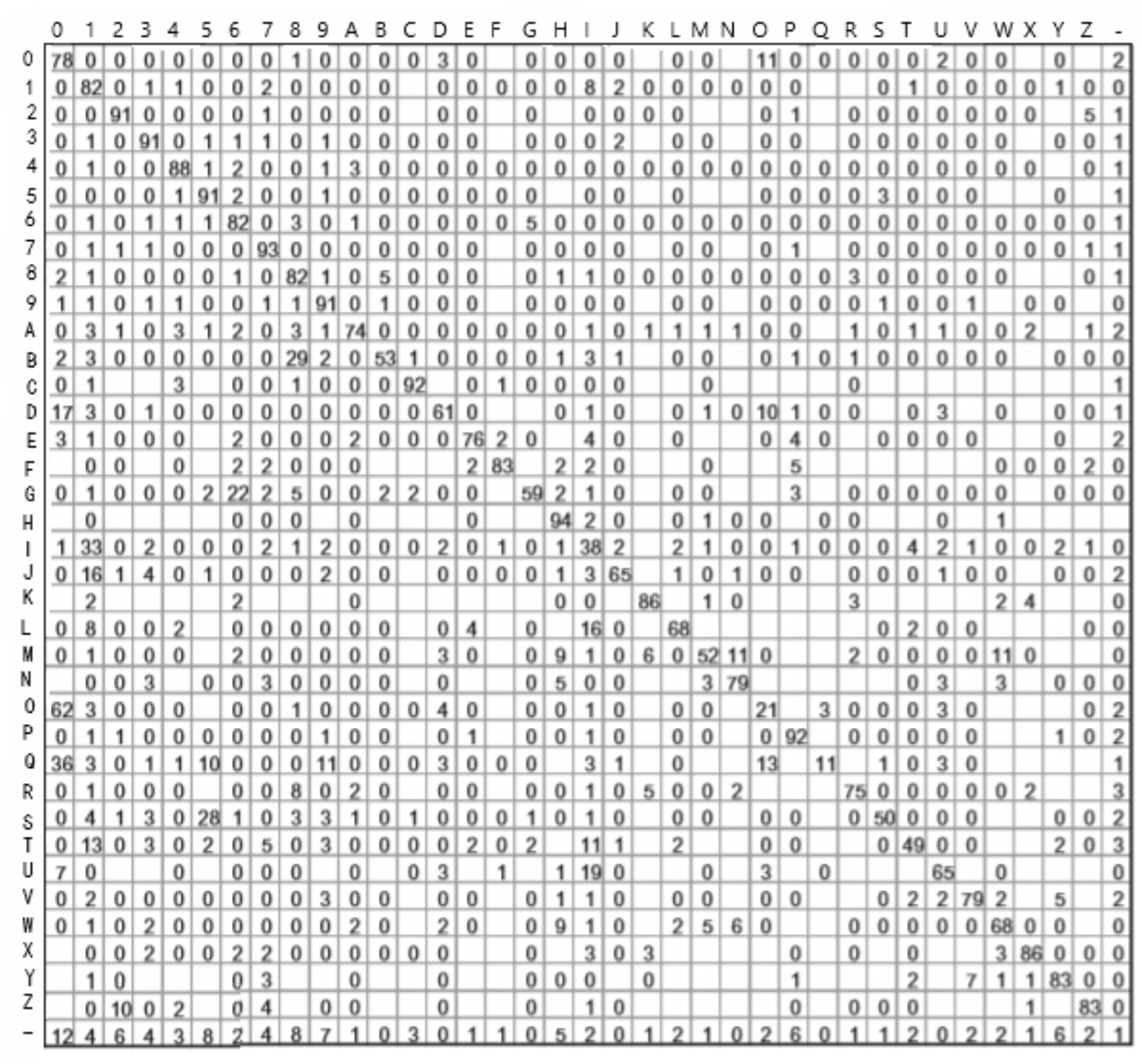
Based on the above matrix, we can obtain an association matrix by calculating the conditional probability.

3.4.3 Matching with MATLAB

After finishing the self-learning and getting the association matrix, the program recalculates the edit distance based on the final association matrix.

$$\square(\square \rightarrow \square) = \square\square\square\{\sum_{\square=0}^{\square} \square\square\square(\frac{1}{\square(\square_{\square},\square_{\square})})\} \qquad (3)$$

$\square(\square \rightarrow \square)$[8]is the cost of transforming x to y.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	-
0	78	0	0	0	0	0	0	0	1	0	0	0	0	3	0		0	0	0	0		0	0		11	0	0	0	0	0	2	0	0		0		2
1	0	82	0	1	1	0	0	2	0	0	0	0		0	0	0	0	0	8	2	0	0	0	0	0	0			0	1	0	0	0	0	1	0	0
2	0	0	91	0	0	0	0	1	0	0	0	0		0	0		0		0	0	0	0			0	1		0	0	0	0	0	0	0		5	1
3	0	1	0	91	0	1	1	1	0	1	0	0	0	0	0		0	0	0	2		0	0		0	0		0	0	0	0	0	0		0	0	1
4	0	1	0	0	88	1	2	0	0	1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	1
5	0	0	0	0	1	91	2	0	0	1	0	0	0	0	0	0	0		0	0		0			0	0	0	0	3	0	0	0			0		1
6	0	1	0	1	1	1	82	0	3	0	1	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
7	0	1	1	1	0	0	0	93	0	0	0	0	0	0	0		0	0	0	0		0	0		0	1		0	0	0	0	0	0	0	0	1	1
8	2	1	0	0	0	0	1	0	82	1	0	5	0	0	0		0	1	1	0	0	0	0	0	0	0	0	3	0	0	0	0	0			0	1
9	1	1	0	1	1	0	0	1	1	91	0	1	0	0	0		0	0	0	0		0	0		0	0	0	0	1	0	0	1		0	0		0
A	0	3	1	0	3	1	2	0	3	1	74	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0		1	0	1	1	0	0	2		1	2
B	2	3	0	0	0	0	0	0	29	2	0	53	1	0	0	0	0	1	3	1		0	0		0	1	0	1	0	0	0	0	0		0	0	0
C	0	1			3		0	0	1	0	0	0	92		0	1	0	0	0	0			0					0									1
D	17	3	0	1	0	0	0	0	0	0	0	0	0	61	0			0	1	0		0	1	0	10	1	0	0		0	3		0		0	0	1
E	3	1	0	0	0		2	0	0	0	2	0	0	0	76	2	0		4	0		0			0	4	0		0	0	0	0			0		2
F		0	0		0		2	2	0	0	0				2	83		2	2	0			0			5							0	0	0	2	0
G	0	1	0	0	0	2	22	2	5	0	0	2	2	0	0		59	2	1	0		0	0			3		0	0	0	0	0	0		0	0	0
H		0					0	0	0		0				0			94	2	0		0	1	0	0		0	0			0		1				
I	1	33	0	2	0	0	0	2	1	2	0	0	0	2	0	1	0	1	38	2		2	1	0	0	1	0	0	0	4	2	1	0	0	2	1	0
J	0	16	1	4	0	1	0	0	0	2	0	0		0	0	0	0	1	3	65		1	0	1	0	0		0	0	0	1	0	0		0	0	2
K		2					2				0							0	0		86		1	0				3					2	4			0
L	0	8	0	0	2		0	0	0	0	0	0		0	4		0		16	0		68							0	2	0	0				0	0
M	0	1	0	0	0		2	0	0	0	0	0		3	0		0	9	1	0	6	0	52	11	0			2	0	0	0	0	11	0			0
N		0	0	3		0	0	3	0	0	0	0		0			0	5	0	0			3	79						0	3		3		0	0	0
O	62	3	0	0	0		0	0	1	0	0	0	0	4	0		0	0	1	0		0	0		21		3	0	0	0	3	0				0	2
P	0	1	1	0	0	0	0	0	0	1	0	0		0	1		0	0	1	0		0	0		0	92		0	0	0	0	0			1	0	2
Q	36	3	0	1	1	10	0	0	0	11	0	0	0	3	0	0	0		3	1		0			13		11		1	0	3	0					1
R	0	1	0	0	0		0	0	8	0	2	0		0	0		0	0	1	0	5	0	0	2				75	0	0	0	0	0	2			3
S	0	4	1	3	0	28	1	0	3	3	1	0	1	0	0	0	1	0	1	0		0	0		0	0		0	50	0	0	0			0	0	2
T	0	13	0	3	0	2	0	5	0	3	0	0	0	0	2	0	2		11	1		2			0	0			0	49	0	0			2	0	3
U	7	0			0		0	0	0		0		0	3		1		1	19	0			0		3		0				65		0				0
V	0	2	0	0	0	0	0	0	0	3	0	0		0	0		0	1	1	0		0	0		0	0			0	2	2	79	2		5		2
W	0	1	0	2	0	0	0	0	0	0	2	0		2	0		0	9	1	0		2	5	6	0			0	0	0	0	0	68	0	0		0
X		0	0	2	0	0	2	2	0	0	0	0	0	0			0		3	0	3					0		0		0			3	86	0	0	0
Y		1	0				0	3			0			0			0	0	0		0					1				2		7	1	1	83	0	0
Z		0	10	0	2		0	4		0	0			0			0		1	0						0		0	0	0				1		83	0
-	12	4	6	4	3	8	2	4	8	7	1	0	3	0	1	1	0	5	2	0	1	2	1	0	2	6	0	1	1	2	0	2	2	1	6	2	1

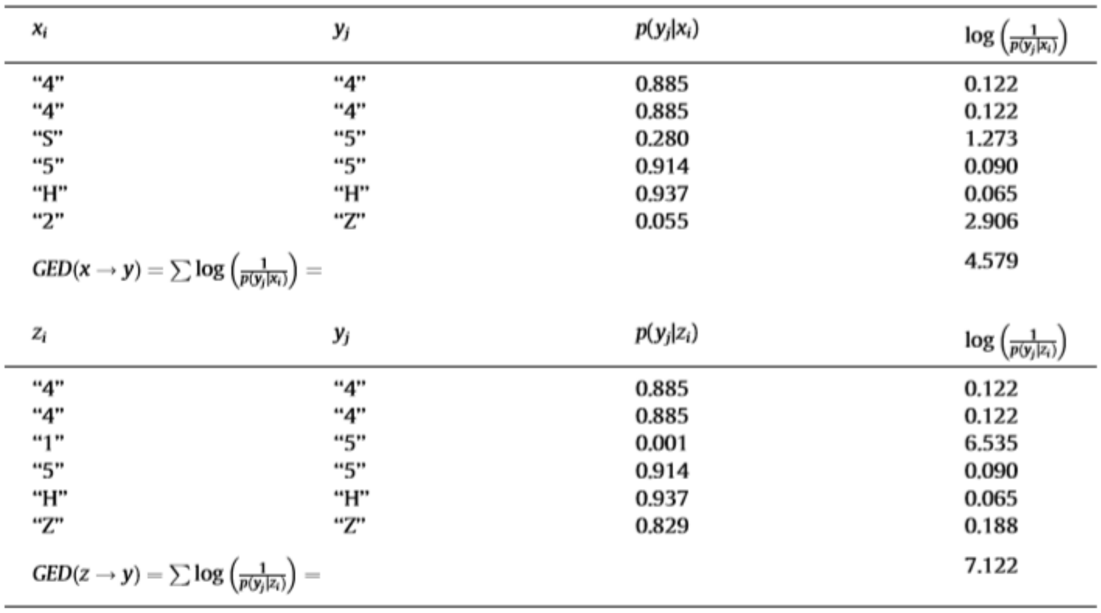
Figure 11: The association matrix

For instance, there are two pairs of license plates:

44S5H2   4455HZ

4415HZ   4455HZ

And we want to find the matching pairs. Calculating the generalized edit distance(GED), and choose the minimum one, then we call it a match.



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
"S"	"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
z_i	y_j	$p(y_j\|z_i)$	$\log\left(\frac{1}{p(y_j\|z_i)}\right)$
"4"	"4"	0.885	0.122
"4"	"4"	0.885	0.122
"1"	"5"	0.001	6.535
"5"	"5"	0.914	0.090
"H"	"H"	0.937	0.065
"Z"	"Z"	0.829	0.188
$GED(z \to y) = \sum \log\left(\frac{1}{p(y_j\|z_i)}\right) =$			7.122

Figure 12: The contrast of two pairs of strings[4]

In the above chart, we can figure out that GED(□ → □)is lower than GED(□ → □). So we defined that 'x' and 'y' is the matching pair.

3.4.4 Matching using Fuzzy Learning

FuzzyWuzzy is a library within Python that is used exactly for the task at hand, matching two strings. With this library the strings are compared using a similarity index, being assigned a score within a 100 point system. To calculate the similarity index, the algorithm uses what is known as "Levenshtein Distance" [fig. 6]. This process was the most practical for this project due to its simplicity and efficiency to compare thousands of strings to find the proper match. Although, for this exact reason, another approach to string matching for this project would make sense as well. The library is unable to self-learn, limiting its ability to help identify commonly mistaken characters. In addition, if time constraints were able to be taken into consideration, it would make the process much quicker. Instead of the entire list being scanned for a match, with a time threshold, only a certain time range would need to be scanned. Overall the system performed very well. When given a score-cutoff of seventy-five, meaning only string matches with seventy-five percent similarity, the program was able to compare nearly five thousand strings in a mere seven minutes and forty-two seconds with approximately seventy percent accuracy.

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

Figure 13: Levenshtein Distance Formula

4.0 Conclusion

Overall the project was a success, but there are still many potential directions to research and improve. In regards to segmentation, most of the cutting results came out well. Although, there are still some problems when we deal with plate images which have too much noise and symbols that cannot be recognized. If speed data could be collected when vehicles go through the LPR station, it would be helpful for the deblurring process. Once the deblurring process is successful, the noise on the license plate image will be greatly reduced.

Figure 13: Poor cutting results (noise and redundant symbol)

There are lots of details in every plate, such as the state and the date the owner applied for the license plate. Potential future research could revolve around analyzing different aspects of the vehicle, as opposed to limiting it just to the license plate. In addition, if instead of a neural network being trained for every character, a process in which the plate as a whole is analyzed, it would likely lead to much more successful results. This way states on the license plate can be recognized and the font style on the license plate of this state can be known for more targeted character recognition.

5.0 Acknowledgements

This project was sponsored by the National Science Foundation's Research Experience for Undergraduates (REU) award, with additional support from the Joint Institute of Computational Sciences at the 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.

6.0 References

**1.	Mordvintsev, A., & Revision, A. K. (2013, September 19). Image Thresholding¶. Retrieved June 25, 2019, from https://opencv-python-**

tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html

2. Oliveira-Neto, F. M., Han, L. D., & Jeong, M. K. (2009). Tracking Large Trucks in Real Time with License Plate Recognition and Text-Mining Techniques. Transportation Research Record, 2121(1), 121–127. https://doi.org/10.3141/2121-13

**3. F. M. Oliveira-Neto, L. D. Han and M. K. Jeong, "An Online Self-Learning Algorithm for License Plate Matching," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1806-1816, Dec. 2013.
doi: 10.1109/TITS.2013.2270107**

4. Francisco Moraes Oliveira-Neto, Lee D. Han, Myong K. Jeong, Online license plate matching procedures using license-plate recognition machines and new weighted edit distance, Transportation Research Part C: Emerging Technologies, Volume 21, Issue 1, 2012, Pages 306-320, ISSN 0968-090X, https://doi.org/10.1016/j.trc.2011.11.003.

**5. A. C. Roy, M. K. Hossen and D. Nag, "License plate detection and character recognition system for commercial vehicles based on morphological approach and template matching," *2016 3rd International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, Dhaka, 2016, pp. 1-6.
doi: 10.1109/CEEICT.2016.7873098**

6. Arias, F. J. (2019, February 6). Fuzzy String Matching in Python. Retrieved June 18, 2019, from https://www.datacamp.com/community/tutorials/fuzzy-string-python