

Introduction:

We have developed *ReadR*, a software tool which will be used for converting numbers and letters from an image of a car's number plate to a machine readable text format. Our group was inspired to create this project after noticing a police officer manually writing license plate characters/digits into his official phone. We thought that it would be much more practical and convenient to have the license plate characters converted to digital text format automatically.

This project will have massive practical use, not only in policing service, but also in task automation. For example, it can be utilized by parking lot security guards, as well as speed detection and license plate recognition cameras, making this software very useful and marketable. This is an ideal machine learning application because of the wide amount of data available to train a model.

Illustration/Figure:

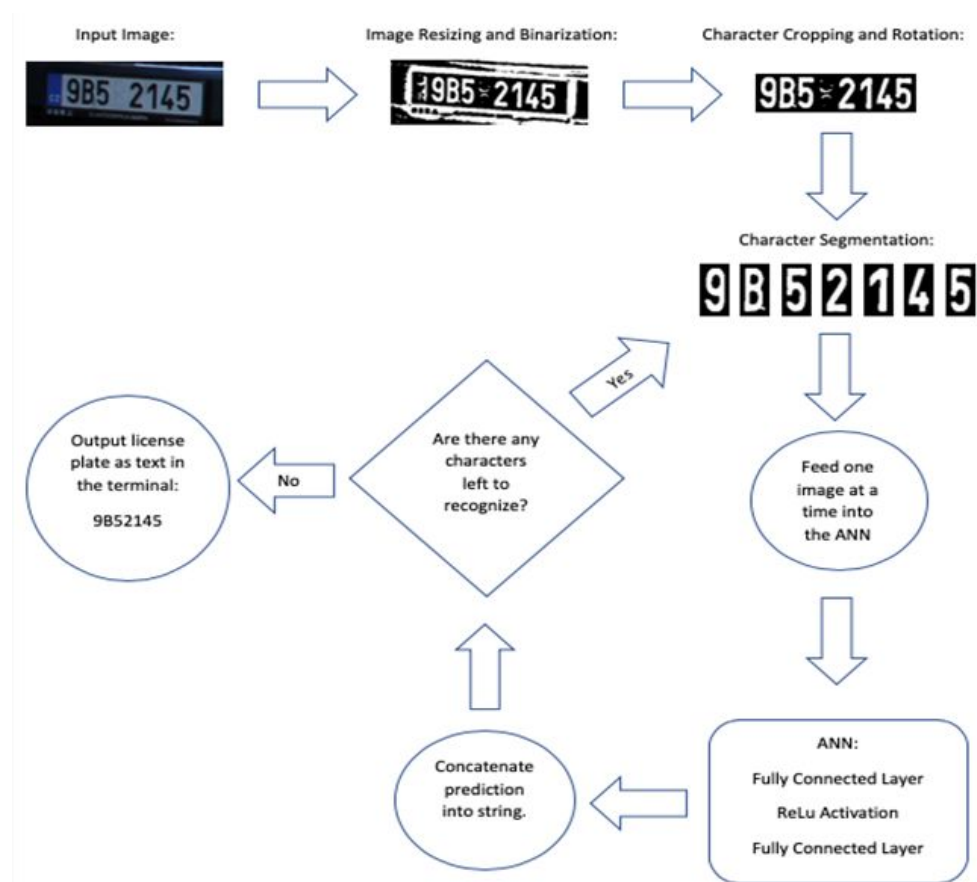


Figure 1: Diagram of the overall project with the primary model neural network, and steps to get an output.

Background and Related Work:

- An Overview of the Tesseract OCR Engine by Ray Smith [1]:

The above work contains an overview of Tesseract OCR which is used primarily for image to text conversion. This study will be useful for developing an architecture scheme for ReadR. It contains very relevant information such as algorithms for conversion of color image to grayscale.

- License Plate Detection with Shallow and Deep CNNs in Complex Environments [2]:

The above work proposes a method to detect license plates based on two CNNs, a shallow CNN and a deep CNN. The shallow CNN is used to remove the background regions, and the deep CNN is used to detect license plates in the remaining regions, lowering computational cost. This research can have a fundamental impact on the way we choose to design our neural network model.

Data Processing:

We obtained our license plate images dataset from Internet Archive (<https://archive.org/details/HDRDataset>). This dataset provided 652 HDR quality images with consistent character count (7) per plate, taken from different angles with no preprocessing.

Given below are some samples taken from the dataset. While the majority of images could be used for building our model, some posed challenges due their differences in brightness, clarity and use of graphics between characters.



Figure 2: License plate image samples from dataset

The total number of images in the dataset provided sufficient data to train, validate and test our model even after filtering out such outliers. Our final model makes use of 80% of data for training, 10% for validation and 10% as the holdout set for final testing.

Prior to using the license plate images, we process our obtained data by employing the following steps:

- 1) Resizing 1: Input images have different dimensions. Thus, all images are scaled to have a standard height of 130 pixels. Images are resized to standardize heights while maintaining their aspect ratios.
- 2) Applying filters: Dataset images contain all images in BGR (Blue-Green-Red) format. Our implementation employs reading luminance information from the image. Consequently, formats for all images are changed to HSV (Hue-Saturation-Value) from BGR using standard openCV filters.
- 3) Binarization: Ordinary binarization involves converting a colored image to a black-white monochrome image. For our purposes, we made use of standard openCV functions to calculate a “local value threshold” by observing a fixed number of pixels surrounding every pixel. Using the threshold value, all pixels are changed to black and white to construct a carefully adjusted monochrome image.
- 4) Inversion: All monochrome images are inverted, i.e black becomes white and vice-versa. This step improves performance for our model.
- 5) Plate detection: Vertical and horizontal edges in the images are detected in order to locate the license plate. Edges that formed a “bounding rectangle” and covered the largest area represented the license plate
- 6) Contour detection and 4-point transform: Four-point transformation is applied to straighten license plate image. OpenCV functions are then used to detect and segment characters using contours and their hierarchies in the image.
- 7) Data splitting: Some images in the dataset pose challenges to detect characters due to their high brightness, low clarity and graphics between characters. Prior to feeding images to our model, we use character counts to filter out images where 7 characters are not detected per image. This step involves splitting the dataset into “Good” vs “Bad” samples. Only “Good” images are processed after this step.
- 8) Resizing 2: Segmented characters (from step 6) in different images have different dimensions. Thus, all detected character dimensions are standardized to 30x60 pixels.

9) Normalization: This step involves changing all pixel values from 0 or 255 to 0 or 1. This aids in improving performance for our model.

Below is a figure that summarizes the impact of the above processing steps on an image sample.

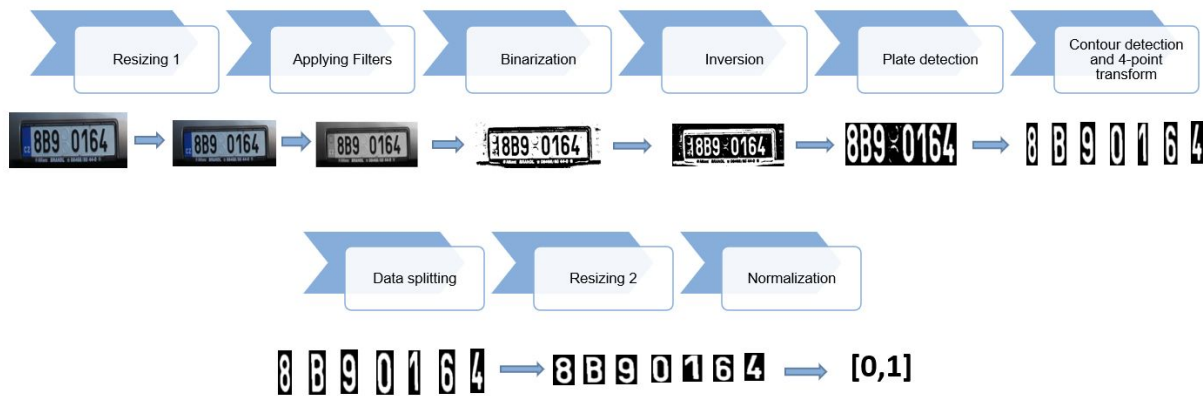


Figure 3: Data processing steps summary

Architecture:

Our final model architecture is an ANN. After the image segmentation unit binarizes and splits the license plate into characters, the binarized characters will be fed into the ANN one at a time: one image is input into the ANN, prediction is generated and concatenated into a string, until there are no more characters to recognize. Once a prediction is made for every character, the predicted license plate number is outputted. Our neural network is a two layer ANN with ReLu activation, having a 30X60 input layer, a 500 unit hidden layer, and a final output layer of 36 classes with softmax.

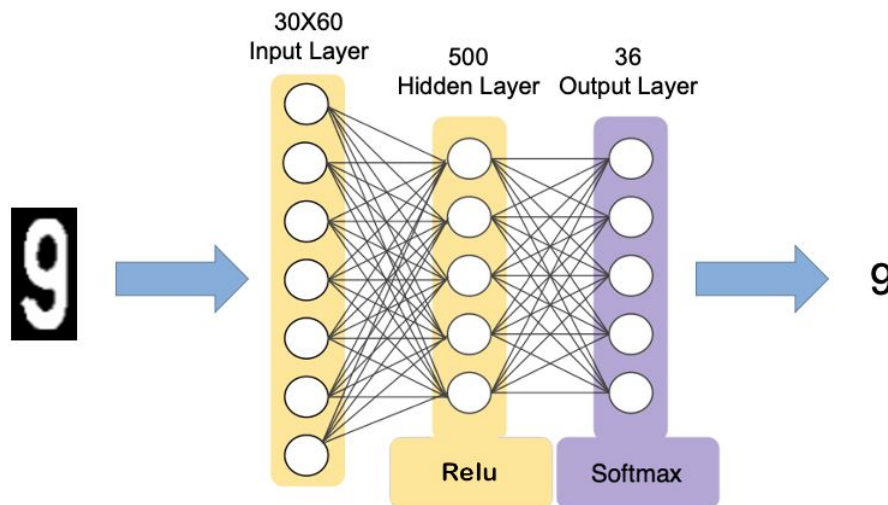


Figure 4: Diagram of Primary Model ANN

Baseline Model:

Our baseline model architecture is an ANN. After the image segmentation unit splits the license plate into characters, the RGB characters will be fed into the ANN one at a time: one image is input into the ANN, prediction is generated and concatenated into a string, until there are no more characters to recognize. Once a prediction is made for every character, the predicted license plate number is outputted. Our neural network is a single layer ANN, having a 30X60X3 input layer, and a final output layer of 36 classes with softmax.

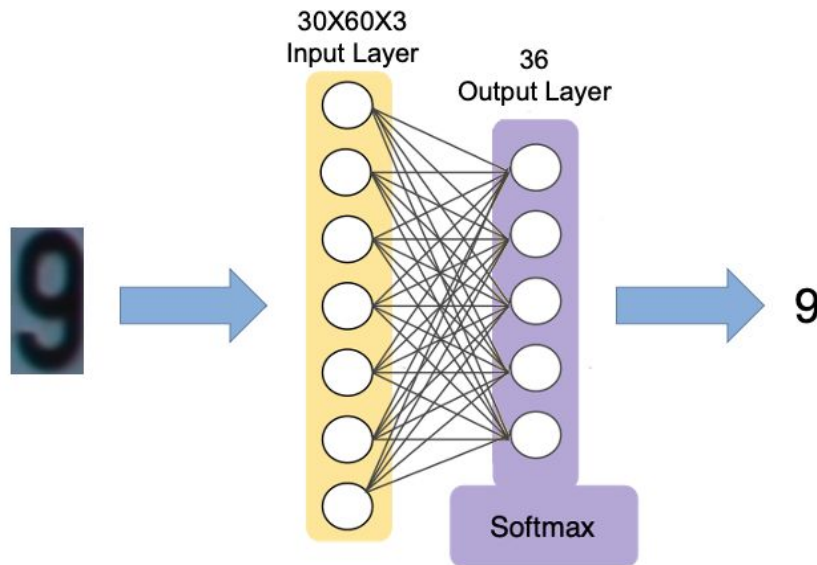


Figure 5: Diagram of Baseline Model ANN

Quantitative Results:

In this section, we will quantitatively evaluate the results of only the ANN based on training and validation accuracy. For the segmentation, the model is able to correctly segment 87% of images, while 13% of the segmentations failed due to reasons discussed in the Qualitative Results. Figure 6 shows the training and validation accuracy of our primary model ANN to predict the characters while Figure 7 shows the results using our baseline model.

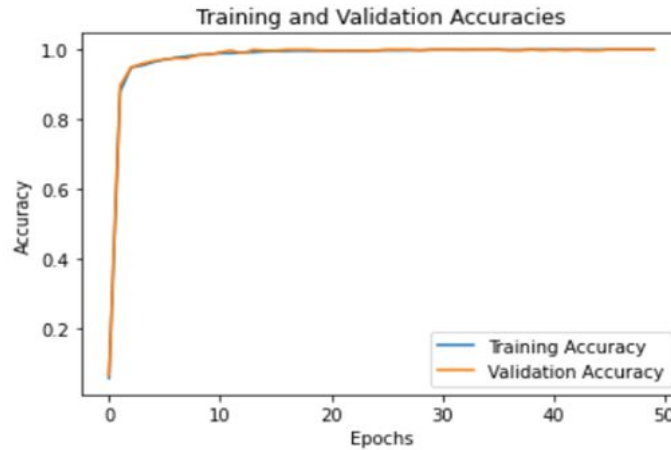


Figure 6: Plots for training and validation accuracy for the primary model. The final accuracy for training and validation is 100%

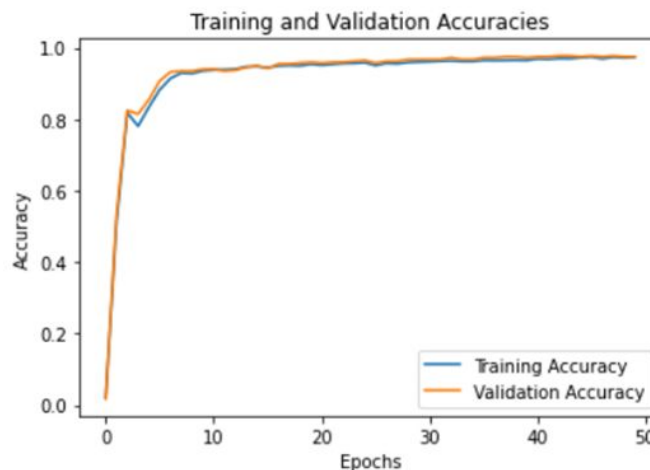






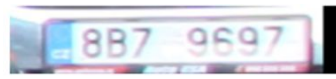


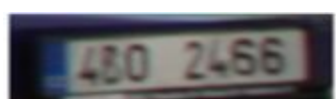




Figure 7: Plots for training and validation accuracy for the baseline model. The final accuracy for training and validation is 97%

Qualitative Results:

Given below is a table that depicts results of some license plate images fed into our model.

Table 1: Selective license plate detection results

Image	Plate Detected	Characters Segmented	Prediction
			8B9 0164
			No prediction
			No prediction
			480 2456

The table above shows instances of 4 images that give varying results when passed into our model. Image 1 represents a perfect case where all 7 characters are detected. Images 2 and 3 reflect the cases where more/less than 7 characters are detected, while image 4 represents the case where our model makes incorrect predictions on 7 detected characters. Note, that image 2 and 3 do not generate a prediction since the number of detected characters is not 7, and hence images like these are discarded by the image segmentation module.

From the 4 images above, it is observable that our model correctly detects the edges of the license plate and segments all readable characters from the processed image close to human-level accuracy. In case of example 2, the extra character appears similar to a “1” based on the processed image while in case of image 4, the second-last character also appears to be a “5” instead of a “6”.

We manually observed unprocessed versions for such outliers. In all the cases, we noticed that unprocessed images contained one or more of the following features:



Figure 8: Categories of outliers

Thus, our team collectively decided to filter out such outliers as detection of characters was hampered by improper quality of image itself. These images did not impact the performance of our model.

Evaluate Model on New Data:

Since our project's goal is to classify the entire license plate, we will test the ability of the entire model to classify license plates based on the test set which is 10% of the images in our data set. The ANN has never trained or ever seen the test set. In order to quantitatively test the entire model, we have to define a schema for what constitutes a correctly predicted license plate: a license plate is correct if every character is correctly predicted and is in the correct order of characters, hence given a score of 1, otherwise if there is any mistake in order or character identification whatsoever, it is incorrect and given a score of 0. The test set consists of 57 images, and the primary model correctly identifies 56 of those images earning a 98.2% test accuracy, while the baseline model managed to only correctly classify 44 images, earning 78% test accuracy. This is a good representation of the ability of the model to predict license plate numbers, since the schema we used is very harsh and doesn't allow for any fraction scores, so if we look at the primary model, it scored a 98.2% accuracy which means that 56 out of the 57 images were predicted completely correctly, therefore despite a very harsh schema, the primary model managed to display a very high test accuracy. Figure 9 belows shows an image from the test set and the process the model takes to get a prediction:

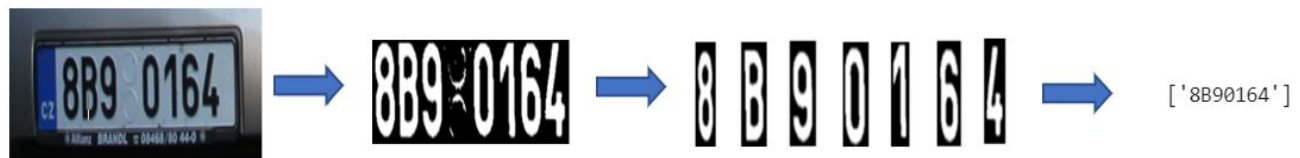


Figure 9: License plate 8B90164 fed into the model and generating prediction of '8B90164'

Discussion:

From our results, we believe that our model performed very well. With the model able to segment 87% of images, and correctly reading 98.2% of the testing set, we have an expected success rate of around 85%, so we expect it to perform well in real life. Our dataset chosen specifically comprised of mostly low-quality images, which allows for insightful training and more realistic real-world usage, as this allows our model to be more robust and not require perfectly taken images and conditions in order to perform well, as seen by our success rate. Also, since our processing and lightweight ANN model can process images nearly instantaneously, this can also be implemented by police not just for taking pictures of plates for parking tickets, but for real-time analysis of video from their dash cam to identify a perpetrator's license plate while driving without the need to manually enter plates or take images. This could also be used in security cameras to identify vehicles that were in the vicinity. However, there are some drawbacks with our model. Our model was trained on license plates from the Czech Republic, which has the same plate font as most European countries, however this font differs from our Canadian license plates, as evidenced when our model failed to correctly predict our example image below. In order to make our model more robust, we would need to train with samples of these plates as well. Our model should still be able to produce a similarly high success rate as the fonts while different are very similar, as seen where our model only makes slight errors on the plate below. One other drawback of our model is that it expects 7 characters in the license plate. This restriction can easily be removed from our model, allowing under or over 7 characters to be read, however this means that there could be more images that weren't segmented properly and not appropriately removed, resulting in a higher output error. However, most license plates follow the 7-character format, and normally only custom plates will have less than 7, meaning our model would still be highly successful in segmenting and reading the license plates.

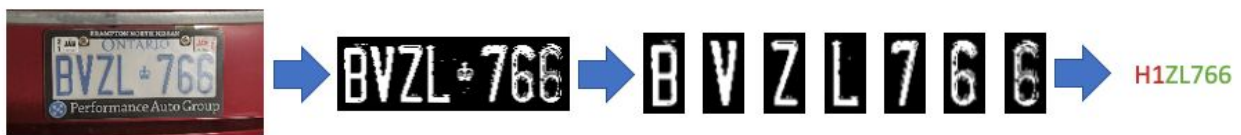


Figure 10: Example Test on Team Member's License Plate Demonstrating Error Due To Font Differences

Ethical Considerations:

The primary model developed achieves 98.2% on new (unseen) data. This does not mean that the AI is reliable enough to be commercially/practically used since it needs to be tested on a wide variety of other data. There are also other limiting factors as stated above in the Discussion section, such as the plates used in developing our model training/validation/testing were from the Czech Republic, which does not guarantee similar performance on plates from other regions, as well as the model only being able to segment and classify license plates with only 7 characters, but there are license plates with more or less than 7 characters.

Thus, in domains of law enforcement, this project's use can jeopardize justice as the model could incorrectly read the license plate, resulting in the wrong people being ticketed or charged. The training data contains license plates of actual cars. Depending on the regulations of the country, this is usually private information. This system can be misused by individuals other than the law-enforcement personnel for illegally storing license plate text to stalk vehicle owners.

Project Difficulty/Quality:

The project consists of two main components: image segmentation module and ANN. Both components had their own hyperparameters to be trained. While the image segmentation was only a computer vision problem and did not involve machine learning at all, it still had parameters that needed to be adjusted in order for the image segmentation module to work on the majority of the images. On the other hand, there was the ANN with its own hyperparameters that needed to be trained. The images that were generated by the image segmentation module were all of size 30x60 and had one input channel, so the input was not too large, and we did not have a very large data set, hence the ANN architecture did not require many weights, and only one hidden layer sufficed. On its own the ANN is not complex, however when adding a second major model to it, it complicates the project. Training the ANN is more complicated when it is fed by the image segmentation since if the image segmentation improperly segmented a character, the ANN will misclassify it and will be penalized due to the image segmentation's mistake, and that data point will throw off the training of the ANN since that point is noisy. Instances like this rarely occurred since we removed most of the bad images, and most of the data set contained very clear images.

For the overall quality of the project and results, the project exceeded our expectations. In the Evaluate Model On New Data section, it was discussed what the test accuracy was (98.2%) and how it was obtained. We anticipated a high test accuracy of ~85-90% since the image segmentation yielded clear images with consistent formatting, the images in the data set were very clear as bad images were discarded, therefore we expected to get high test accuracy.

References:

[1] Static.googleusercontent.com, 2020. [Online]. Available: <http://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/33418.pdf>. [Accessed: 14- Feb- 2020].

[2] L. Zou, M. Zhao, Z. Gao, M. Cao, H. Jia and M. Pei, "License Plate Detection with Shallow and Deep CNNs in Complex Environments", 2020.