CS 274P FINAL PROJECT

Subash Chandra Kotha
Student ID: 31328473

Surya Rao Shedimbi
Student ID: 31328473

ABSTRACT

Automatic license plate recognition (ALPR) is the extraction of vehicle license plate information from an image. The extracted information can be used in many applications such as electronics payment systems at toll plazas and parking areas. It can also be used for freeway and urban traffic surveillance. ALPR as a real life application has to quickly process license plates under different environmental conditions such as day, night, rainy etc. The quality of the image captured play a major role in correctly identifying the registration number. The number plates might use different fonts and colors as well. Also some license plates can be partially occluded by dirt, lighting, and towing accessories on the car. In this project, given an image of a license plate we aim to correctly identify the registration number using Image segmentation techniques and Convolutional Neural Network Models(CNN). Later we aim to improve the prediction accuracy by data augmentation, feature engineering and ensemble machine learning methods.

ACKNOWLEDGEMENTS

We want to take some time to thank Professor Isak Bosman and our Teaching Assistants Junze Liu and Ahmad Razavi for helping us with this course. Hope that you all are well and safe during these extremely tough times. Despite an abrupt shift to remote learning, you have ensured that no student suffered in terms of learning. We have gained tremendous volume of knowledge in the field as well as learned to be disciplined when working in a quarter long project. You were extremely approachable and very friendly in your manner of teaching. We hope that we have done justice to your teachings with this project. A huge huge thanks from the core of our heart. We will always be grateful.

1 DATA SET DESCRIPTION AND EXPLORATION

The actual data set consists of license plate images, and a .csv file which is mapping between the path of the image and license plate number. The total number of images in the data-set is 182,337.

1.1 THE IMAGE

The image is a .png file which has the license plate number. It is a coloured image. A sample is shown below.

Figure 1: Sample License Plate Image

The meta data of the image is as follows:
Type of the image : class 'imageio.core.util.Image'
Shape of the image : (138, 394, 3)
Image Height : 138
Image Width : 394
Dimension of Image : 3
Image size 163116
Maximum RGB value in this image 255
Minimum RGB value in this image 13

Splitting the above image into its 3 RGB components, we get:

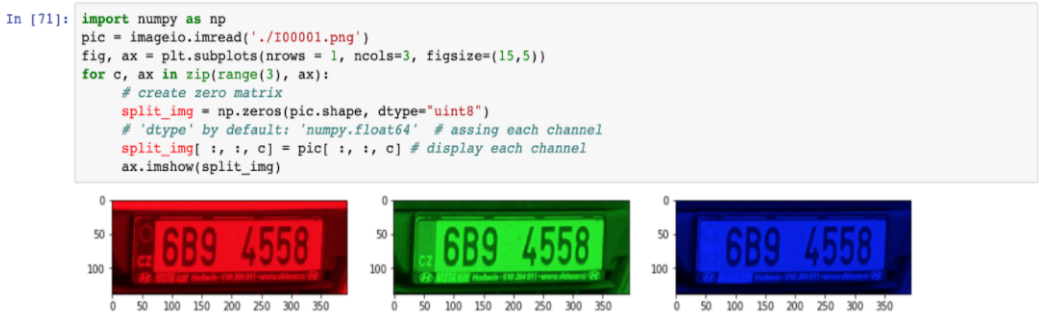

Figure 2: Image with only R, G, V values

All the images have similar dimensions and size. Since the exact shapes of the images are not the same, we need to trim the shape of the image to be able to perform numpy operations like average and hence getting the mean image. The mean image is useful because we can get useful information from it which can later be used for model preparation.

After selecting a few images by random sampling, envisioned scatter plots show the following. The data between bands are correlated

```
In [12]: _ = plt.hist2d(pixel_matrix[:,1], pixel_matrix[:,2], bins=(50,50))
```

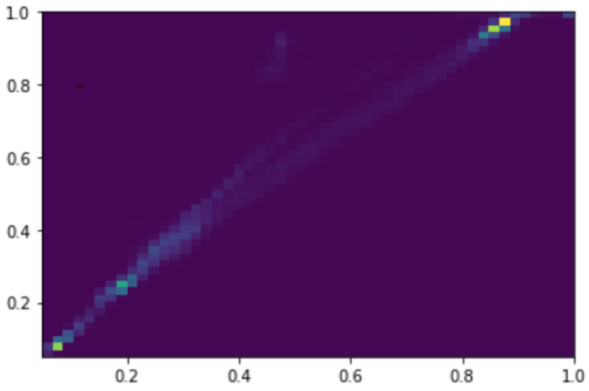

Figure 3: Scatter plot

We can apply Brightness Normalization strategy prior to using other strategies, in order for the matching algorithm to be robust across variations in illumination.

Figure 4: After Brightness normalisation

The histogram of pixel projection can be used for image segmentation or finding objects of interest in the image(in our case, numbers and alphabets). The graph for our sample image is shown below. We will be using this observation during image segmentation.

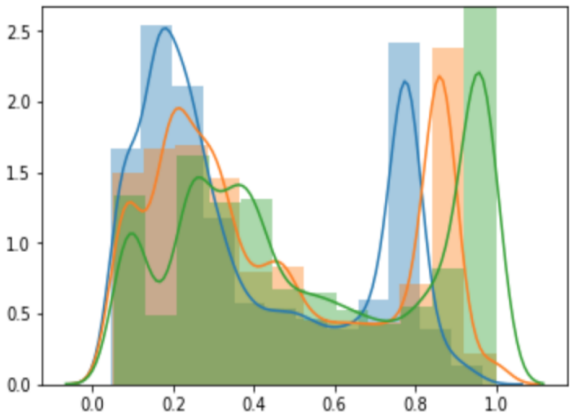

Figure 5: Pixel projection of the sample image

1.2 IMAGE-PATH

Another attribute in our dataset is image-path. It stores the location of the image in the file system directory. This attribute will not provide any useful information to the model and hence should be discarded while training the model.

1.3 LP (LICENSE PLATE) VALUE

It is the ground truth which the image refers to. This is what our model tries to classify. It is a categorical type nominal attribute in string form. Ex: 6B94558.

2 TRAINING AND VALIDATION DATA GENERATION

The initial data set contains 182,337 license plate images. Each license plate has 30 different images with variations in orientation, brightness etc. We decided to use few of those images to generate images of characters A-Z and 0-9 which will be later used as training and validation data for training our CNN model.

To achieve this objective we will be using character segmentation mentioned in the next section. We will split the image into individual characters and store them in their corresponding sub-folders inside train and val folders. We used 28,000 images out of our dataset to generate images of the characters and distributed them in the train and val folders in **80:20** ratio.

```
CS274P\working_org
├──train
│   ├──0
│   ├──1
│   ├──..
│   ├──Z
├──Val
│   ├──0
│   ├──1
│   ├──..
│   ├──Z
```

Figure 6: Training Data Directory Structure

Here we had to be sure that the image of the characters we are segmenting are going into the correct folders. We performed few checks in-order to ensure this.

1. All the license plate images contain exactly 7 characters. So when we segment license plate we check if we are able to get seven images.
2. We manually checked folders in random manner to see if the folder actually contains image corresponding to the folder label.

3 ALPR SYSTEM

Our ALPR system can be split into two stages:

- Character segmentation - extract the alphanumeric characters from the plate
- Character recognition - recognize each individual character

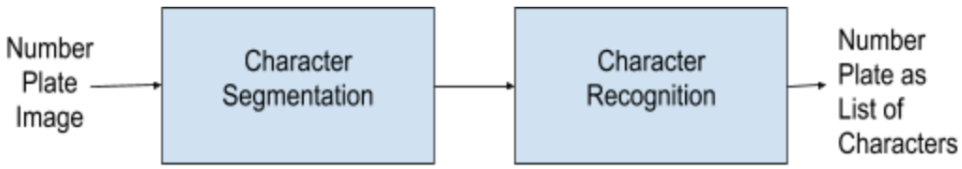

Figure 7: High level design

3.1 CHARACTER SEGMENTATION

Given an image of the license plate, we first extract individual characters present in it and feed it as input to the next stage in the pipeline. This is an important stage because if the segmentation fails, recognition will not be correct. We made use of the cv2 library to process the image and extract the characters. Before actually plotting contours we perform a sequence of steps to pre-process the image.

- **RGB to Grey** : We convert the image into grayscale format.
- Then we made use of the pixel projection shown few sections above to understand the fact that most of the crucial information about the image is at the centre and we can **crop the edges**.

5 in page number 4

- **Re-sizing** : Since input images might be of different size we need to resize every image to a standard size. This is very helpful while building and designing our model which will predict the character.
- **Blurring** : In order to remove the noise, we blur the image.
- **Thresholding** : We convert the grayscale image into binary (Black white) image after sharpening if necessary. We used Adaptive threshold techniques to achieve this objective.
- We then apply the **dilation** on the threshold image. This dilates the source image using the specified structuring element that determines the shape of a pixel neighborhood over which the maximum is taken. It basically accentuates features.
- We finally segment the image by **finding contours** and then use the bounding rectangles algorithm to **segment** the characters.
- We will re-size the final image to 30*70.

3.2 CHARACTER RECOGNITION

The next stage in our pipeline deals with classifying the image into one of the 36 outputs (A-Z alphabets and 0-9 numbers). It is a black-white image so we need to convert into RGB image before feeding it into our model. The model we will be using to perform classification is CNN.

A convolutional neural network (CNN) enables data-driven learning and extraction of highly representative, hierarchical image features from appropriate training data. Generally, the working principle of a CNN is as follows: first, the convolution layer scans the input image to generate a feature vector; second, the activation layer determines which feature should activate the image under inference; third, the pooling layer reduces the size of the feature vector; and finally, a fully connected layer connects each potential tag to all outputs of the pooling layer. We will pre-train our model using images generated and stored in the train and val folders..

After building the initial model we aim to increase the prediction accuracy by adding more images to our dataset. We can achieve this by data augmentation.

3.2.1 INITIAL APPROACH

We initially built our own CNN model from scratch using PyTorch. We made use of a lot of python packages like cv2, PIL, torchvision, numpy etc. The model's input layer had 107648 neurons and output layer had 36 neurons(obviously, given our project). We made a lot of key decisions regarding the kernel size, optimiser type, number of hidden layers, activation function, etc. We made use of 2 convolutional layers followed by 2 max pooling layers. The activation function being used is RELU. We have also made use of dropout regularization, a technique used to reduce overfitting where a single model is used to simulate having a large number of different network architectures by randomly dropping out nodes during training. Finally there are three fully connected layers before the output layer. The following is a snippet of the model definition.

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
        self.conv2 = nn.Conv2d(32, 128, kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(107648, 2048)
        self.fc3 = nn.Linear(2048, 512)
        self.fc5 = nn.Linear(512, 36)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        #print(x.size())
        x = x.view(-1, 107648)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = F.relu(self.fc3(x))
        x = F.dropout(x, training=self.training)
        x = self.fc5(x)
        return x

model = Net()
model.cuda()

optimizer = optim.Adam(model.parameters(), lr=1e-4, eps=1e-4)
```

Figure 8: Model definition

We had chosen PyTorch's Cross Entropy Loss as our error function and ran our code for multiple epochs. After training, we ran tests on our model using the test set(we divided the original dataset into training, validation and test set in a 6:2:2 ratio). The accuracy of this model failed to reach our expectations with 68 percent. The accuracy was measured after putting together the predictions from all the segments from the given image of a license plate(which usually contains 7 characters). The string formed by putting all the predictions together is then matched with the target value. So if even one of the predictions is wrong, it is counted as a wrong prediction. Hence, this accuracy should not be confused with the accuracy of the neural network model.

3.2.2 TRANSFER LEARNING

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and the huge jumps in skill that they provide on related problems.

We have used this approach to come up with a new classifier with the aim to improve our accuracy. We used the famous resnet101 model as our pretrained model and adapted transfer learning.

First, we have assigned labels to the domain of target values. We have then segmented the data into two folders: train and val. Each of them have 36 folders, namely 0, 1, 2,... A, B,...Z.

Inorder to get an idea of the number of particular characters in each of the folders in total, we have calculated and plotted the occurrences of each character in the license plate in both train and val using the target labels(strings) from the original dataset. The following is the observation we have seen.

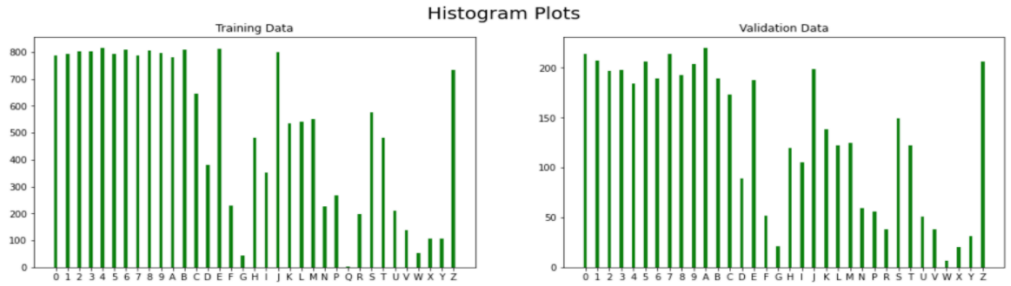

Figure 9: Frequency of appearance of characters

We then finally use the pretrained model. We set the output layers with our features(the labels described above). We decided to go with the Cross Entropy Loss as our error function.

Based on our experience with Neural Networks(which is limited to this course), we understood that the choice of optimization algorithm for your deep learning model can mean the difference between good results in minutes, hours, and days. After trying out with different algorithms, we decided to use Adam Optimiser. We made this choice because Adam is significantly faster than traditional gradient descent and combines the advantages of two other extensions of stochastic gradient descent, namely Adaptive Gradient Algorithm (AdaGrad) Root Mean Square Propagation (RMSProp).

Similarly, we have tried different values of learning rate(aka alpha in the back propagation equation) and compared all the results. We found that a learning rate of 0.001 showed best results.

We then load the data into data loaders and shuffle(to reduce bias) and set the batch size as 30 and then use the train method. This method is very similar to the one professor used in his notebook in the class Github repository. We train the model for 10 epochs and then save the model. We can now test it with a random image from the validation set.

Figure 10: Example test image

4 MODEL EVALUATION

When it comes to evaluating our model, what we are doing is taking our prediction from the model and concatenating it with the other 6 predictions all of which belong to the same initial license plate image. We then compare this string with the actual label from the dataset and see if they are equal. If they are the same, then the model has worked successfully. After running our model on 1000 images, we find that the accuracy was around **79%, a significant 16% jump from our initial model accuracy**. One thing to note would be that the model deems an input to be wrongly classified even if one of the characters does not match with the required respective target value(and it should!). Hence, if we look at the accuracy of the neural network in correctly classifying a given segment, we get,

$$(0.84)^{1/7} = 0.97540 = 97.5\%$$

We also understand that about 90% of the images were correctly segmented. So the imperfection in accuracy can also be credited to the fact that the input to the model itself was not in a very great shape.

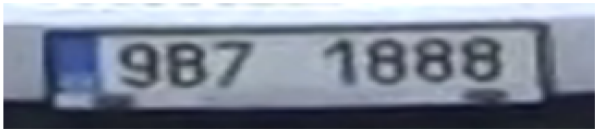

Figure 11: ALPR System in action

5 REFERENCES

[1] Shan Du, Mahmoud Ibrahim and Mohamed Shehata. Automatic License Plate Recognition (ALPR) A State-of-the-Art Review. IEEE Transactions on Circuits and Systems for Video Technology (Volume: 23 , Issue: 2 , Feb. 2013)

[2] Diogo M. F. Izidio and Antonyus P.A. Ferreria. An embedded automatic license plate recognition system using deep learning.

[3] Cohen, G., Afshar, S., Tapson, J., van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. Retrieved from http://arxiv.org/abs/1702.05373

[4]Martin Thoma, The HASYv2 dataset. Retrieved from https://arxiv.org/pdf/1701.08380.pdf

[5]Hamdaoui, Yassine. (2019, December 8). Image Data Analysis Using Python. https://towardsdatascience.com/image-data-analysis-using-python-edddfdf128f4

[6]Kamphaus, Ben.(2016, May 3).https://www.kaggle.com/bkamphaus/exploratory-image-analysis/notebookInitial-impressions

[7]Brownlee, Jason .(2017, December 17). https://machinelearningmastery.com/transfer-learning-for-deep-learning/

[8]Brownlee, Jason .(2018, December 3).https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

[9]Brownlee, Jason .(2017, July 3).https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[10]Gupta, Mehul.(2019, October).Vehicle Detection and OCR. https://medium.com/data-sci9ence-in-your-pocket/vehicle-number-plate-detection-and-ocr-tcs-humain-2019-a253019e52a1

[11]Original LP-dataset(Reference) https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode. Original-Dataset https://medusa.fit.vutbr.cz/traffic/download/512/