Poojitha Bolem (Data Scientist Role)

IMDA Assignment Report:

Like any data science project, a pipeline needs to be established where some of it would include data ingestion, pre-processing of data (cleaning, data transformations), exploratory data analysis, creating training, validation and tests sets, building a model and finally, evaluating the model.

The data provided included an input folder (which consists of the images) and output folder which consists of the labels of the Captcha images. I have noticed that 2 images have missing labels (input21.jpg and image100.jpg). Hence, I considered these 2 as my 'test' set (ie. the unseen captchas).

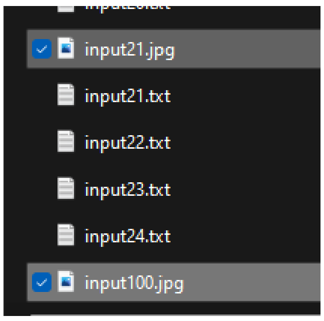
Below is brief explanation/purpose of each function for better understanding of how the problem was tackled.

clean_data function:

Runs through the input and output folders to source out the images that have labels from the images with no labels. The images with labels are placed in a new folder called 'cleaned_input'. This would come in handy if we are dealing with a large dataset to sort out the images.

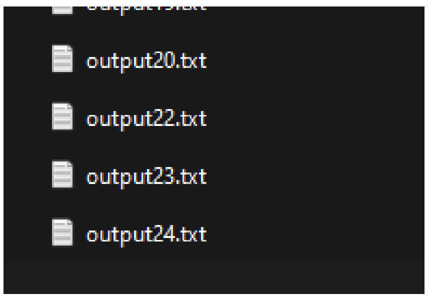
load_data function:

The paths of the other 2 unlabelled images are stored in 'unseen_captchas'. Figures 1 and 2 show how the 2 images have no labels found in the output folder. The labels in the output file were all stored in individual text files. Hence, this was extracted and stored as a list called 'labels'.



```
input21.jpg
input21.txt
input22.txt
input23.txt
input24.txt
input100.jpg
```

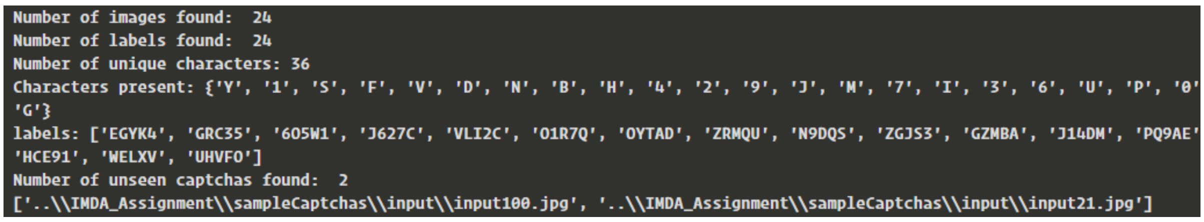
Figure 1: Images with no labels



```
output20.txt
output22.txt
output23.txt
output24.txt
```

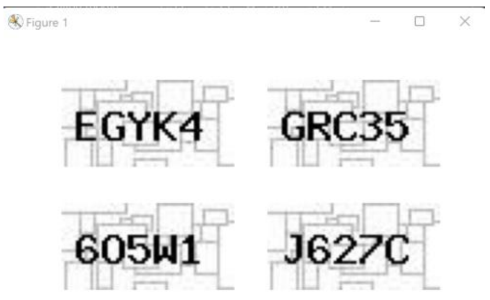
Figure 2: Output Folder (shows missing text files for input 21 and input 100 jpg)

Some stats were printed out to confirm all the characters/texts have been extracted properly (expected to see 36 characters. 26 letters and 10 numbers):



```
Number of images found:  24
Number of labels found:  24
Number of unique characters: 36
Characters present: {'Y', '1', 'S', 'F', 'V', 'D', 'N', 'B', 'H', '4', '2', '9', 'J', 'M', '7', 'I', '3', '6', 'U', 'P', '0'
'G'}
labels: ['EGYK4', 'GRC35', '605W1', 'J627C', 'VLI2C', 'O1R7Q', 'OYTAD', 'ZRMQU', 'N9DQS', 'ZGJS3', 'GZMBA', 'J14DM', 'PQ9AE'
'HCE91', 'WELXV', 'UHVFO']
Number of unseen captchas found:  2
['..\\IMDA_Assignment\\sampleCaptchas\\input\\input100.jpg', '..\\IMDA_Assignment\\sampleCaptchas\\input\\input21.jpg']
```

Figure 2: Printed Stats



4 samples of the images were plotted to ensure the images were read properly. Following which, the pre-processing of the images include:

1) Changing the color of the images to grey scale
2) Creating a threshold to convert the image to pure white and black
3) Finding contours around each letter to split the letters up in the images
4) Creating rectangles that contain the contours of the images
5) Ensure rectangle sizes does not exceed too much (split the rectangle if size is too big)
6) Sort the letters from left to right accordingly to match the original order of the text
7) Save each letter image to a separate folder with the label as its name

processing_training_data function:

Here, the aim is to create the training and validation sets to feed in the model later. We extract out each image letter and store these images in 'data' and their corresponding letters as 'labels'. We split up the data into 75% training and 25% validation sets.

Then we convert the labels (letters) into one-hot encodings so that Keras can work with it (saved the mapping to model_labels.dat). This function returns the model inputs for the next function.

nn_model function:

Here, a convolution neural network with 4 layers is being used to train the data.
Batch size = 6
Epochs = 200
categorical_crossentropy was used a loss function to deal with the multi-class classification model.
The model was then saved to *nn_model.hdf5* to use later.

read_unseen_captchas function:

Here, we are loading the unseen captchas and repeating the same pre-processing steps so that we can get predictions for each letter image. We load the model and make the predictions. We then convert the one-hot-encoded prediction back to normal letters using *model_labels.dat*. Finally, we join the different predictions into one Captcha Text.

<u>Problems Faced:</u>

Since only 24 images are provided to train the model, it is expected to get inaccurate results for the test set. However, another issue faced was when creating the contours. The findcontours function was not able to separate the different letters. I believe it was still picking up the grey lines in the background. For example, the image on the right is what was picked up for letter S.

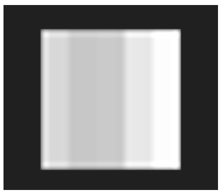


Figure 4: Letter S Image Contour

Further refinement and exploration is required to sort out this problem so that we can get accurate training images with their corresponding labels to detect unseen captchas more accurately.

Reference:
1. https://github.com/abdrabah/solve_captcha
2. https://medium.com/@ageitgey/how-to-break-a-captcha-system-in-15-minutes-with-machine-learning-dbebb035a710