CS771 Assignment 2

Challa Sai Yohitha (200289) | Ganesh Mahlawat (200368) | Harshal Mehta (200426) | Prashant Kumar Mahour (21103072)

Detecting parity of HexaCAPTCHA

CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) are popular ways of preventing bots from attempting to log on to systems by extensively searching the password space.In its traditional form, an image is given which contains a few characters (sometimes with some obfuscation thrown in). The challenge is to find parity of the hexadecimal number in that image.

Data:Each CAPTCHA image in this assignment will be 500×100 pixels in size. Each image will contain a code that is a 4 digit hexadecimal number. The font of all these characters would be the same, as would be the font size. The Latin characters A-F will always be in upper case. However, each character may be rotated (degree of rotation will always be either $0\circ$, $\pm 10\circ$, $\pm 20\circ$, $\pm 30\circ$ and each character may be rendered with a different color. The background color of each image can also change. However, all background colors are light in shade. Each image also has some stray lines in the background which are of varying thickness, varying color and of a shade darker than that of the background. These stray lines are intended to make the CAPTCHAs more "interesting" and realistic.

Question 1

> Describe the method you used to find out what characters are present in the image. Give all details such as details of the model being used (e.g. linear, decision tree, neural network, kernel etc), the training algorithm used, including hyperparameter search pro- cedures, validation procedures.

Solution: To solve this task, we use the following steps.

- **Load the train images and lables**
  1. Read the training images using **'imread'**
  2. Convert the image into grayscale or we can directly read image as grayscaled using **'imread(image path,0)'**.
  3. open the lables as a file and split and strip the each line lable to only consider first character(i.e,'EVEN/ODD)

- **Processing the training images**

  Here we do processing using OpenCV functions.
  1. At first we read images using '**imread** function.
  2. Convert the image into grayscale or we can directly read image as grayscaled using **'imread(image path,0)'**.
  3. To remove noise/lines in the background use a morphological operation called dilation followed by erosion (this method gives better results than simple erosion alone).

Preprint. Under review.

- **Extract feature vector of images**
    1. For each character in image, extract relevant features that can help in differentiating between odd and even numbers. These features can include pixel intensities, gradients, or other image descriptors.
    2. Convert processed images and lables into numpy arrays.Create an array of features of each image and lables using **np.array**.
    3. Also convert lables into integer lables as '0' for ODD and '1' for EVEN.

- **Train the features using a Logistic Regression model**
    1. Logistic Regression is a classification algorithm used to predict the probability of a binary or multi-class outcome.
    2. The logistic function, also known as the sigmoid function, is applied to the linear combination. The sigmoid function maps the linear combination to a value between 0 and 1, representing the probability of the positive class (in binary classification) or the probabilities of each class (in multi-class classification).

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

    3. The predicted probabilities are then used to make class predictions. For binary classification, a threshold (typically 0.5) is chosen, and probabilities above the threshold are classified as the positive class, while those below are classified as the negative class. For multi-class classification, the class with the highest probability is selected as the predicted class.
    4. The **'fit()'** method trains the Logistic Regression model by adjusting its parameters to find the best fit for the provided training data and labels.
    5. By fitting the Logistic Regression model to the training data, it learns the optimal coefficients that best fit the data and can be used to predict the class labels for test data

- **Make prediction on test images and find the accuracy**
    1. Load the test images as we did for training images
    2. Process the test images using function defined above for training images and also extract feature vector for test images using numpy array.
    3. Make predictions using the model defined above and compare them with ground truths.At last print the predicted labels and accuracy.Accuracy we got for dummy_test data is 0.98(98%).And the run time is 43sec

Resources used: GeeksforGeeks,stack Overflow.