CS771A Assignment 2

The Regressors
Prabuddha Singh(200691) | prabuddhas20@iitk.ac.in
Rabindranath Das(200746) | rabindrad20@iitk.ac.in
Shrutikirti Singh(200962) | shrutiks20@iitk.ac.in
Ujesha Saigal(201057) | ujeshas20@iitk.ac.in

1 Task

Describe the method you used to determine the characters in the image. Give all details, such as the model being used (e.g., linear, decision tree, neural network, kernel, etc.), and the training algorithm used, including hyperparameter search procedures and validation procedures.

Approach : We have used **logistic regression classifier** for **character recognition**.

The logistic regression model used in this code is an instance of the **'LogisticRegression'** class from scikit-learn. It is a linear model that assumes a linear relationship between the input features and the log-odds (also known as logits) of belonging to a specific class. The logistic regression model can handle binary classification problems as well as multiclass classification problems using appropriate strategies such as one-vs-rest or softmax regression.

The training algorithm used in logistic regression is based on maximum likelihood estimation (MLE) or minimizing the logistic loss function. The specific optimization algorithm employed by scikit-learn's **LogisticRegression** class is a variant of gradient descent. The optimization algorithm aims to find the optimal weights and biases that minimize the logistic loss function, thus fitting the model to the training data.

- **Feature Preprocessing:** The input features for the logistic regression model are grayscale images of characters that have undergone preprocessing steps such as removing lines and segmenting the last letter. The resulting images are resized to a fixed size of 50x50 pixels.
- **Logistic Regression Model Creation:** An instance of the **LogisticRegression** class from scikit-learn is created. The model is set to use the default hyperparameters, which include a regularization parameter (**C**) of 1.0, a penalty type (**l2**), and a solver (**lbfgs**). These defaults are not explicitly specified in the code but are used unless specified otherwise. The logistic regression model is trained on the training data using the **fit method**. The fit method optimizes the model parameters (weights and biases) to minimize the logistic loss function based on the training data. The optimization algorithm employed by the LogisticRegression class is a variant of gradient descent.
- **Validation Procedure:** The code uses a train-test split approach for validation. The preprocessed images and their corresponding labels are split into training and test datasets using the **train_test_split** function from scikit-learn. The **test_size=0.2** argument specifies that 20% of the data will be used for testing, while the remaining 80% will be used for training. The split is performed in a stratified manner, ensuring that the class distribution is preserved in both the training and test datasets. The training dataset is used to fit the logistic regression model, and the test dataset is used to evaluate the model's performance by calculating the accuracy of the predictions using the **accuracy_score** function.
- **Reshaping the Data:** The training and test data are reshaped from a 3-dimensional array (image number, height, width) to a 2-dimensional array (image number, flattened features)

Preprint. Under review.

using the **reshape** function. This reshaping is done to match the input requirements of the logistic regression model, which expects a 2D array of features.

Procedures and Hyperparameter used: We first removed the obfuscating lines from each image using the function **vanish_lines**. We analyzed that the color of each letter is always darker than the obfuscating lines and the background color is the lightest. So only the dark pixels must remain visible and all the other pixels with colors outside that range must be updated to the same color as the background. So, to do that we converted the BGR image to HSV format. Then we found the minimum value and created a mask using a value range (minimum value- p, minimum value+ p). Here p is a **color_range** hyper-parameter that must be tuned. Using this mask, we set all the pixels outside the mask to have the color of the background.

We found an appropriate value of this hyper-parameter, p = 80. That worked perfectly in all the train images. The **color_range** hyperparameter (p) specifies the range of colors to consider as the darkest colors. The function creates a mask by thresholding the V channel of the HSV image based on the minimum value (**min_val**) and the **color_range**. The **min_val** - **color_range** and **min_val** + **color_range** values define the lower and upper thresholds for the darkest colors, respectively.

2

Submitted as predict.py

References

[1] Tutorialspoint - How to mask an image in OpenCV Python? link

[2] Kaggle - OpenCV Word Segmenting on CAPTCHA Images link

[3] Kaggle -Image Classification with Logistic Regression link