**Report**

Q.1 Problem statement :

Implement a suitable classical machine learning algorithm to detect characters in these signs and display detected text finally.

Approach:

The provided sample data have too much of noise, irregular shape, and undistinguishable characters within each image.

Proposed pipeline:

1. Train a ML model for classifying individual letters
2. Read image -> Segment out every letter from an image -> Provide this letter to Trained ML (1) model for class detection -> get predicted class of letter -> Update .txt file for each image

To solve this problem, I preferred image processing approach. In this approach I used various combinations of techniques which are listed below.

1. Color conversion: RGB to Gray, RGB to HSV, RGB to LAB
2. Thresholding: Global & Adaptive
3. Filtering: Gaussian, Laplace, Custom
4. Edge detection: Canny
5. Contour detection

Various combinations and experiments were conducted to extract every letter from the input image, but the optimal combination was not achieved.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

To create a test dataset for ML model performance analysis, manually some letters from the provided images were cropped and stored in ‘.png’ format.



**ML model training:**

For classifying identified letter images w.r.t. each representative letter, ML approach was used. Platform used for training the ML model was Kaggle.

Dataset used: <https://www.kaggle.com/datasets/preatcher/standard-ocr-dataset>

Approach:

According to pipeline, user will provide image to ‘Image processing module’ which will return individual letters from the given image. Those individual letters will be provided to trained ML model for class prediction, and the resultant class will be updated in .txt file.

Libraries used: sklearn, numpy, os, opencv, matplotlib

Dataset: The dataset is itself divided into two subfolders ‘data/training\_data’ & ‘data/testing\_data’

Each folder has subfolders of characters: 0-9 & A-Z; total of 36

Preprocess:

Images are imported in test & train variables, individual class distribution is checked; it was found to be equal.

Images were normalized so that processing became faster. The image classes were encoded in the numeric class format.

In the training and analysis,

SVM, KNN and Decision Tree models were trained and experimented with.

In the performance, every model reported accuracy of 97+ % in each class. But for testing on sample data which was manually used, it did not perform that well.

REFERENCES

http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/#download