CS771 Assignment 2

Aadvik Kumar (210002) **Ankit Kumar (210150)** **Dhruva Singh Sachan (210343)**

Khushi Agrawal (210514) **Om Shrivastava (210685)** **Zainab Fatima (211211)**

Abstract

This document describes the methodology and approach used by our group in the second assignment of the course CS771: An Introduction to Machine Learning, offered at IITK in the semester 2022-23-Summer.

1 Training Algorithm Description

1.1 K-Means Clustering

This pipeline utilised the two facts found in the dataset, i.e the background had a constant colour, with only noise being random lines and that the characters had a distinct colour from background. Thus we were able to perform colour based segmentation. To extract the most common colours present in the image we used K-Means Clustering model on the RGB (Red, Blue, Green) space.
First, the whole image is processed by K-Means, center of the largest cluster is found to be the background color. Then the image is cropped along X-axis to get a suitable region in which only the last character is present. Then this cropped image is again processed through K-Means, the most common observation being, that the centers of the three largest clusters belong to background, character boundary and character color itself. Since the colors in the characters were unique and not common with background or the lines, this fact was utilised to create a mask which was combined with the image using bitwise_and() provided by open_cv library.
This approach helped us to completely segment the character from background and noise, but since it used K-Means clustering, the inference time per image was significantly increased (0.7 sec per image), and it was observed that we don't need a very perfect segmentation to get a good accuracy, thus the method was not used in the final pipeline.

Figure 1: Result of K-Means Clustering

Preprint. Under review.

1.2 Image Processing

The image processing pipeline starts by loading the input image and converting it to grayscale. From the grayscale image, the pipeline extracts the background color by sampling regions from the corners and estimating the background distribution. Using this distribution, a threshold is applied to convert the grayscale image to binary, separating foreground and background. Erosion and dilation operations refine the binary image, followed by the extraction of the region of interest. This region is then resized and processed using the Histogram of Oriented Gradients (HOG) algorithm to extract robust features capturing local shape and gradient information. These HOG features are flattened and stored for further analysis and modeling. By applying this pipeline, valuable features are extracted from the input image, enabling various computer vision applications such as object recognition and classification.

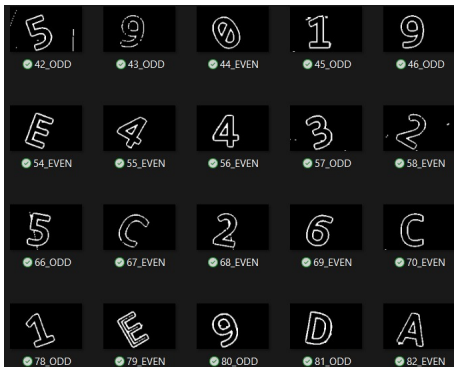


Figure 2: Result of Image Processing

2 Model Description

2.1 Decision tree

- We classified features with a decision tree model after image processing. Decision trees predict by recursively partitioning the feature space using decision rules. Leaf nodes include class labels, while inside nodes represent feature-based decisions.
- During training, the tree partitions the feature space depending on input features and class labels to determine the decision rules.
- The decision tree model follows decision rules from the root to the leaf nodes during inference. Input instance feature values determine the path. The input instance's prediction is the leaf node's class label.
- The hierarchical decision tree model captures complex feature-class label relationships. It handles categorical and continuous features, making it suitable for classification jobs. Decision trees, which represent decision rules, can also illuminate the decision-making process.
- We train the decision tree model with extracted characteristics to learn patterns and predict based on decision rules. The model's capacity to partition the feature space by relevant features helps classify new instances.
- In conclusion, the decision tree model learns decision rules from training data and uses them to anticipate unseen occurrences, making categorization easy to understand.
- The validation was done using a **split of 1600:400**.

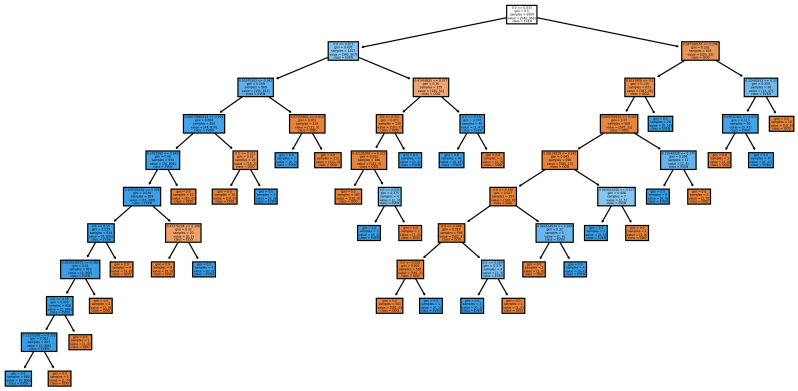


Figure 3: Decision tree

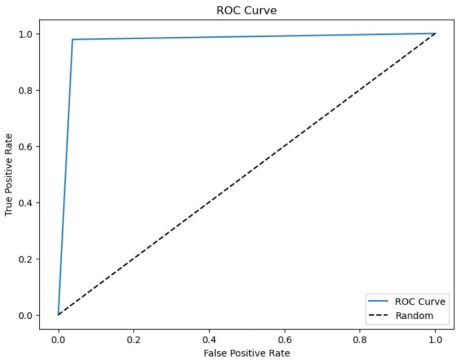


Figure 4: ROC curve

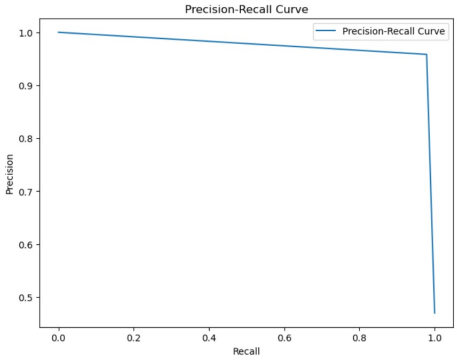


Figure 5: Precision recall curve

2.2 K-nearest neighbors (KNN)

- We used a non-parametric supervised learning technique for k-nearest neighbors (KNN). KNN identifies new instances by their feature space similarity. KNN classifies an instance by its k closest neighbors.

- The approach uses Euclidean or Manhattan distance to calculate instance distances. KNN estimates the distances to its k nearest neighbors and assigns the majority class label to the new instance.
- The number of neighbors, k, impacts algorithm performance. A smaller number of k can result in more flexible limits but may be more sensitive to noisy data, whereas a greater value may overlook local patterns but smoother decision boundaries.
- KNN can classify multi-class problems using neighbors' class labels. KNN is an intuitive classification system that predicts using proximity and local patterns.
- The validation was done using a **split of 1600:400**.

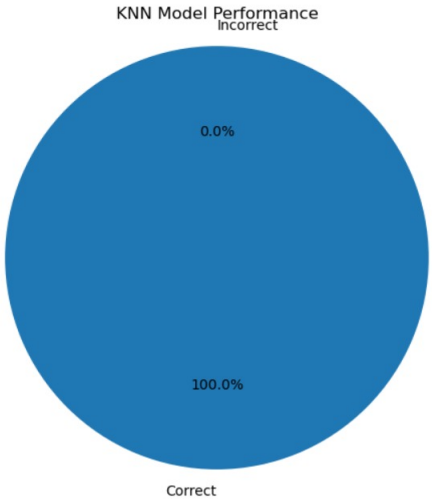


Figure 6: KNN Model performance

2.3 Convolutional Neural Networks

- We experimented with Convolutional Neural Networks (CNNs) and achieved impressive outcomes. However, we encountered challenges with the large size of the model and its weights, as well as the time required for training and inference. Despite achieving similar accuracy results as decision trees, the CNN model remained larger and had longer inference time compared to decision trees.
- Our CNN architecture followed a standard structure, employing the "Binary Cross Entropy" Loss Function and the Adam optimizer. The validation was done using a **split of 1800:200**.

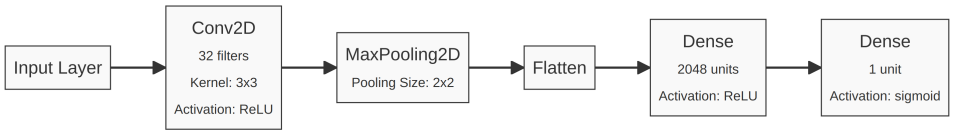


Figure 7: CNN Layers

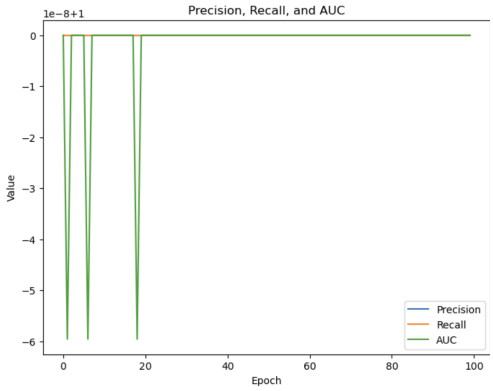


Figure 8: CNN : Precision, Recall and AUC

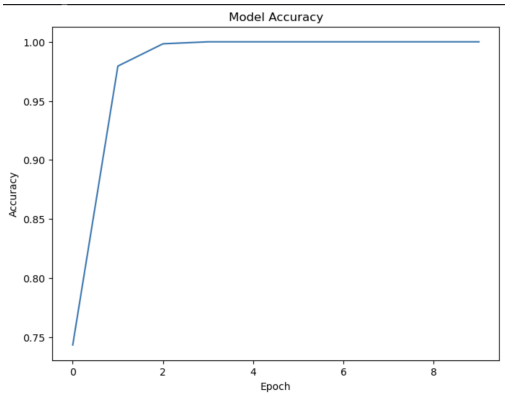


Figure 9: CNN : Model accuracy

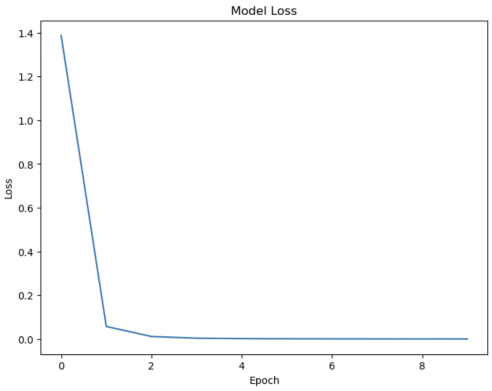


Figure 10: CNN : Model loss

2.4 Multi-Layer Perceptron

- In an attempt to minimize the model's size and reduce unnecessary processing time for inference, we opted to eliminate the Convolution and Max Pooling Layer. Instead, we utilized a simple Multi-Layer Perceptron, maintaining the accuracy.
- The model still gave longer inference times and large model size.
- Our approach involved directly taking input from each pixel of the image, bypassing the convolution and pooling kernel steps, and constructing a 3-layer model with a single hidden layer and same loss function and optimizer as before.

- The validation was done using a **split of 1800:200**.

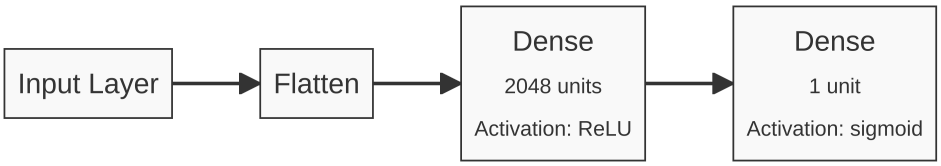
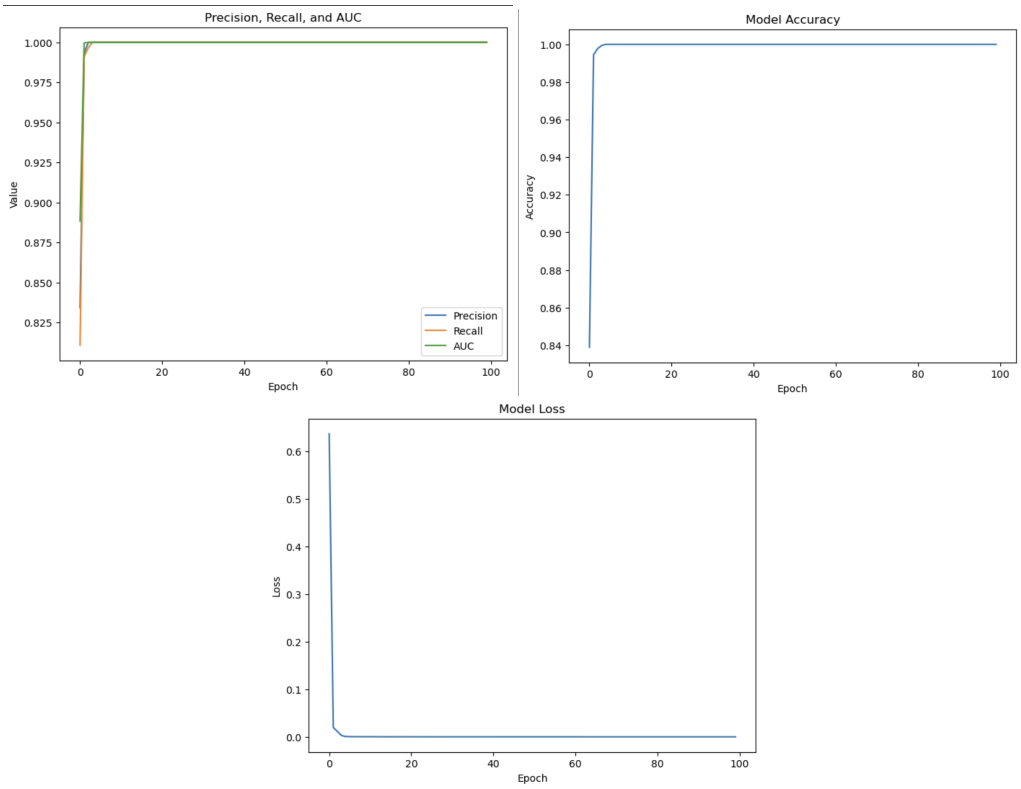


Figure 11: MLP layers



2.5 Single Layer Perceptron

- Finally, we removed all the hidden layers and kept only a single layer perceptron model, which only took inputs from all the pixels and produced an output.
- This modification significantly reduced the model's size and inference time. However, even with these improvements, the model still couldn't outperform the decision trees
- The validation was done using a **split of 1800:200**.

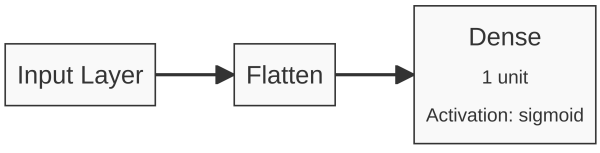


Figure 12: SLP layers

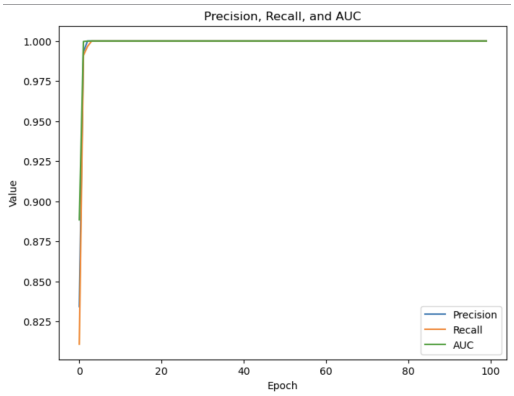


Figure 13: SLP : Precision ,Recall and AUC

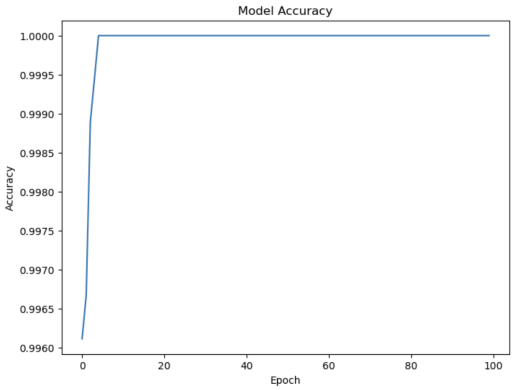


Figure 14: SLP : Model accuracy

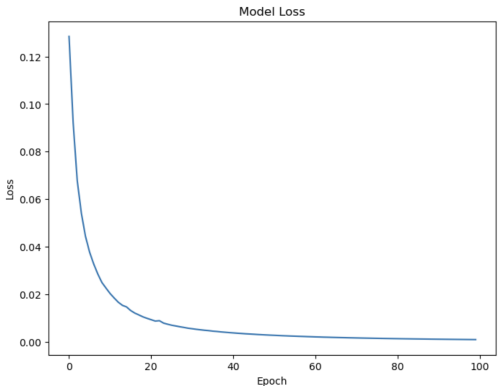


Figure 15: SLP : Model loss

3 Hyper Parameter Description

To tune these hyperparameters, we employed the grid search technique. Grid search systematically evaluates all possible combinations of hyperparameter values and selects the best configuration based on a predefined evaluation metric.

The hyperparameter description for our decision tree model includes three parameters:

- **max_depth**: It determines the maximum number of levels in the decision tree.

- **min_samples_split**: This parameter specifies the minimum number of samples required to split an internal node.
- **min_samples_leaf**: It sets the minimum number of samples required to be at a leaf node.

The parameter grid was defined as follows:

$\text{param\_grid} = \{\text{'max\_depth'} : [5, 10, 15], \text{'min\_samples\_split'} : [2, 4, 8], \text{'min\_samples\_leaf'} : [1, 2, 4]\}$

For each combination, the model's performance was assessed using a validation set on the basis of accuracy.

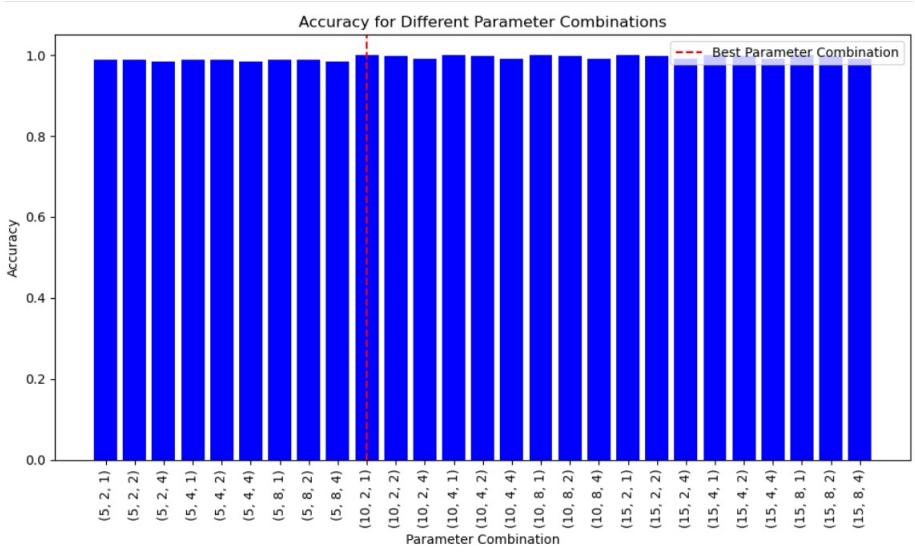


Figure 16: Accuracy for Different Parameter Combinations

4 Results

Model	Size of Weights	Avg Inference Time per Image	Accuracy
Decision Trees	5.6 kB	0.003 seconds	0.98
SLP	73.44 kB	0.0033 seconds	1.0
KNN	52 MB	0.2 seconds	1.0
MLP	129.46 MB	0.0047 seconds	1.0
CNN	955.6 MB	0.0103 seconds	1.0