Predicting Handwritten letters

Presented by Aditya Adke, Mihir Bhanasali, Jonas Kobel

Introduction

Objective:

- The project aims to develop a robust model for predicting handwritten text, focusing on the classification of letters, digits, and special characters to enhance text recognition systems.

Dataset and Scope:

- Utilizing a carefully curated dataset with 780,000 images across 39 categories, the project addresses challenges like misclassification of similar-looking characters and incorporates diverse data for alphabets, digits, and symbols.

Significance:

- By advancing handwritten text recognition, the project contributes to applications like digitization of handwritten documents, OCR systems, and assistive technologies for individuals with visual or motor impairments.

Motivation

Growing Importance of OCR:

- The ability to classify handwritten letters and numbers is a crucial component in OCR systems used in various fields such as banking, healthcare, and education.

Real-World Applications:

- This project has the potential to improve handwriting recognition technologies for tasks like digitizing documents, postal address reading, and exam grading systems.

Diverse Challenges:

- The dataset combines alphabets, digits, and special characters, providing an opportunity to address complex classification challenges like distinguishing between visually similar classes (e.g., 'O' and '0').

Skill Development:

- This project enables me to enhance my expertise in deep learning, computer vision, and image preprocessing techniques while working on a practical and impactful problem.

Dataset

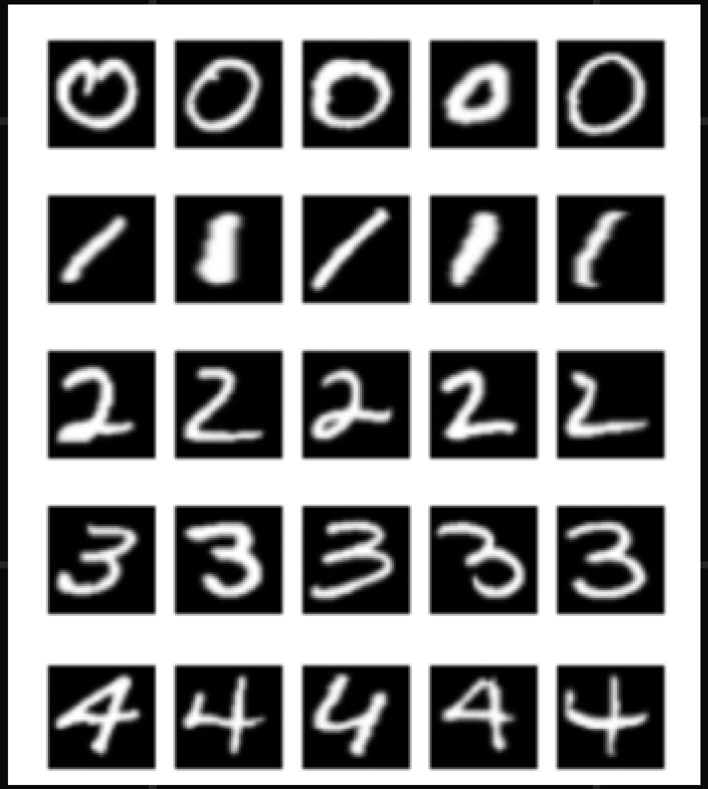
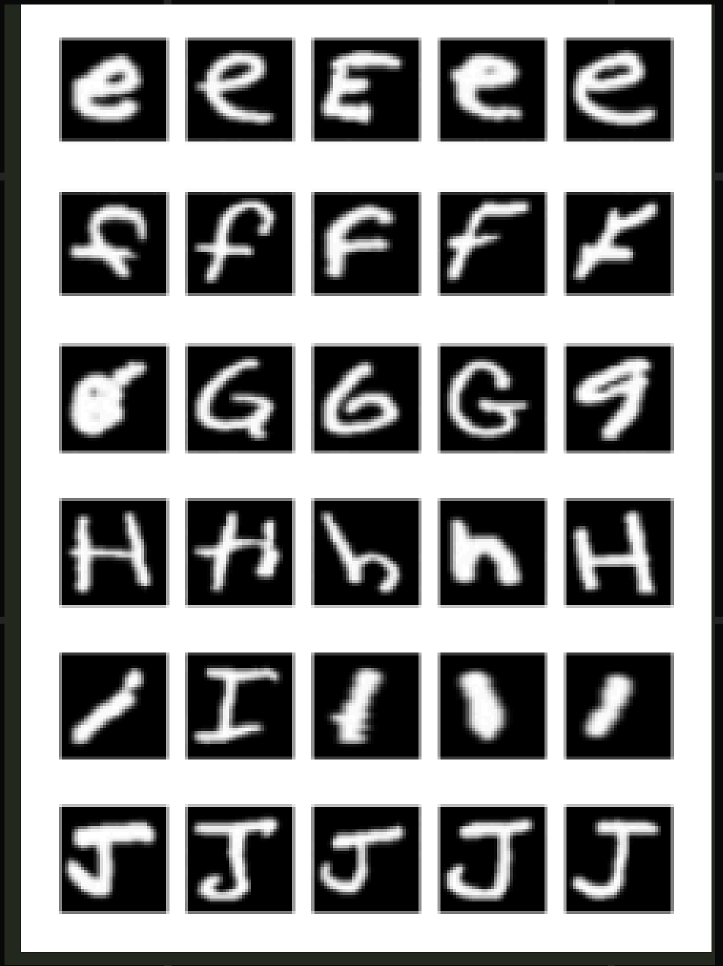
Dataset: 2.1 GB in size and contains 780,000 images, with approximately 30,000 handwritten samples per category across its 39 classes.

Source: Built upon the EMNIST dataset, transformed with image processing techniques to enhance usability.

Image Format: Contains 32x32 pixel black-and-white images, offering uniformity for machine learning models.

Categories: A total of 39 classes, including:

- 26 alphabet categories (lowercase and uppercase merged).
- 9 digit categories (0 merged with 'O').
- 4 special characters (@, #, $, &).

Our Plan of Action

Steps:

1. Train **Random Forest Classifier** for the Numbers 0 to 9
   a. Use 300 Images of each Class (Training Time ~ 1h)
2. Train a **CNN (Convolutional Neural Network)** for the Numbers 0 to 9
   a. Use 2000 Images of each Class (Training Time ~ 30 mins)
3. Train a **CNN** for the Alphabet (A to Z)
   a. Use 10000 Images of each Class (27) (Training Time ~1.5h)
4. Train **3 x CNN** for the Alphabet
   a. Use 30000 Images of each class, split the Dataset into 3 separate Datasets and train the CNN (The Python Kernel broke at more the 12000 Images). Each CNN makes their own prediction, use the majority for the outcome prediction
5. **Fine Tune** the models and **predict a hole handwritten text**

Random Forest Regression

Structure of the CNN (for Numbers):

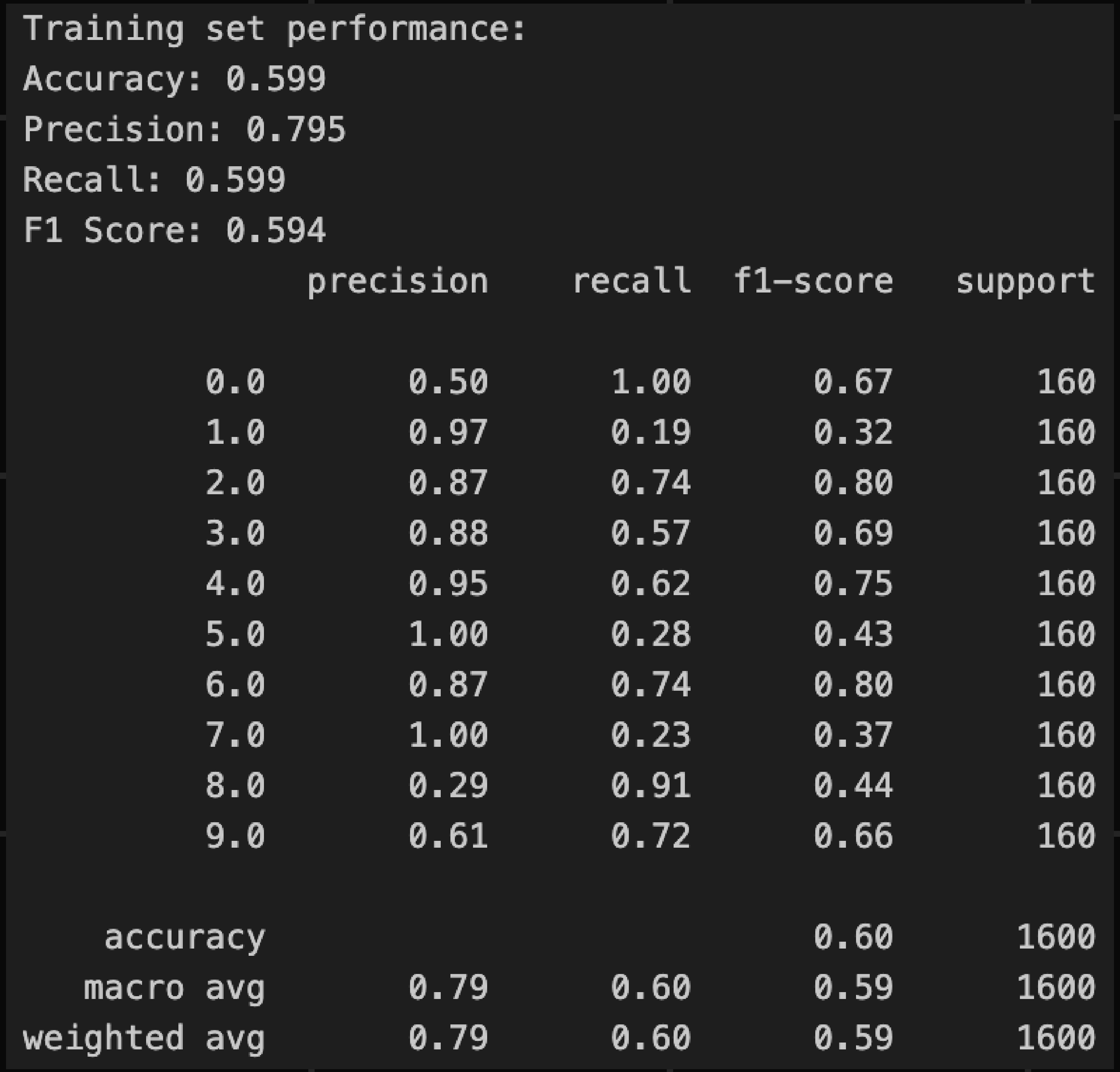
1. **MinMaxScaler**
2. **RandomForest Classifier**

Hyperparameter Tuning:

- **Criterion:** Entropy
- **Max Depth:** 20
- **Min Leaf Samples:** 2
- **Min leaf Sample Split:** 2

Random Forest Regression Result

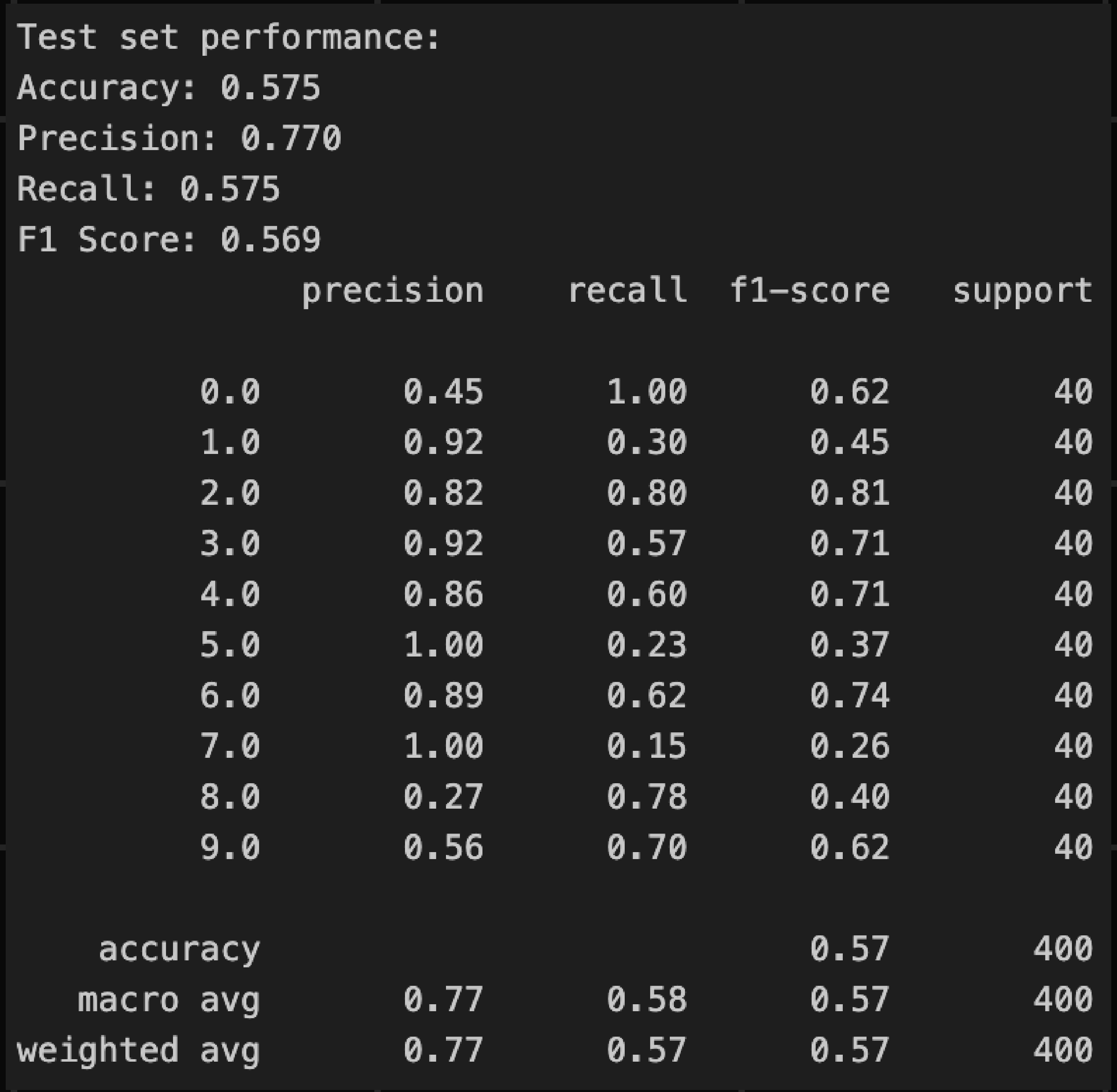
- The model performs poorly on the dataset with very low accuracy of 57.5 %



```
Training set performance:
Accuracy: 0.599
Precision: 0.795
Recall: 0.599
F1 Score: 0.594
              precision    recall  f1-score   support

         0.0       0.50      1.00      0.67       160
         1.0       0.97      0.19      0.32       160
         2.0       0.87      0.74      0.80       160
         3.0       0.88      0.57      0.69       160
         4.0       0.95      0.62      0.75       160
         5.0       1.00      0.28      0.43       160
         6.0       0.87      0.74      0.80       160
         7.0       1.00      0.23      0.37       160
         8.0       0.29      0.91      0.44       160
         9.0       0.61      0.72      0.66       160

    accuracy                           0.60      1600
   macro avg       0.79      0.60      0.59      1600
weighted avg       0.79      0.60      0.59      1600
```



```
Test set performance:
Accuracy: 0.575
Precision: 0.770
Recall: 0.575
F1 Score: 0.569
              precision    recall  f1-score   support

         0.0       0.45      1.00      0.62        40
         1.0       0.92      0.30      0.45        40
         2.0       0.82      0.80      0.81        40
         3.0       0.92      0.57      0.71        40
         4.0       0.86      0.60      0.71        40
         5.0       1.00      0.23      0.37        40
         6.0       0.89      0.62      0.74        40
         7.0       1.00      0.15      0.26        40
         8.0       0.27      0.78      0.40        40
         9.0       0.56      0.70      0.62        40

    accuracy                           0.57       400
   macro avg       0.77      0.58      0.57       400
weighted avg       0.77      0.57      0.57       400
```

Convolutional Neural Network (Numbers)

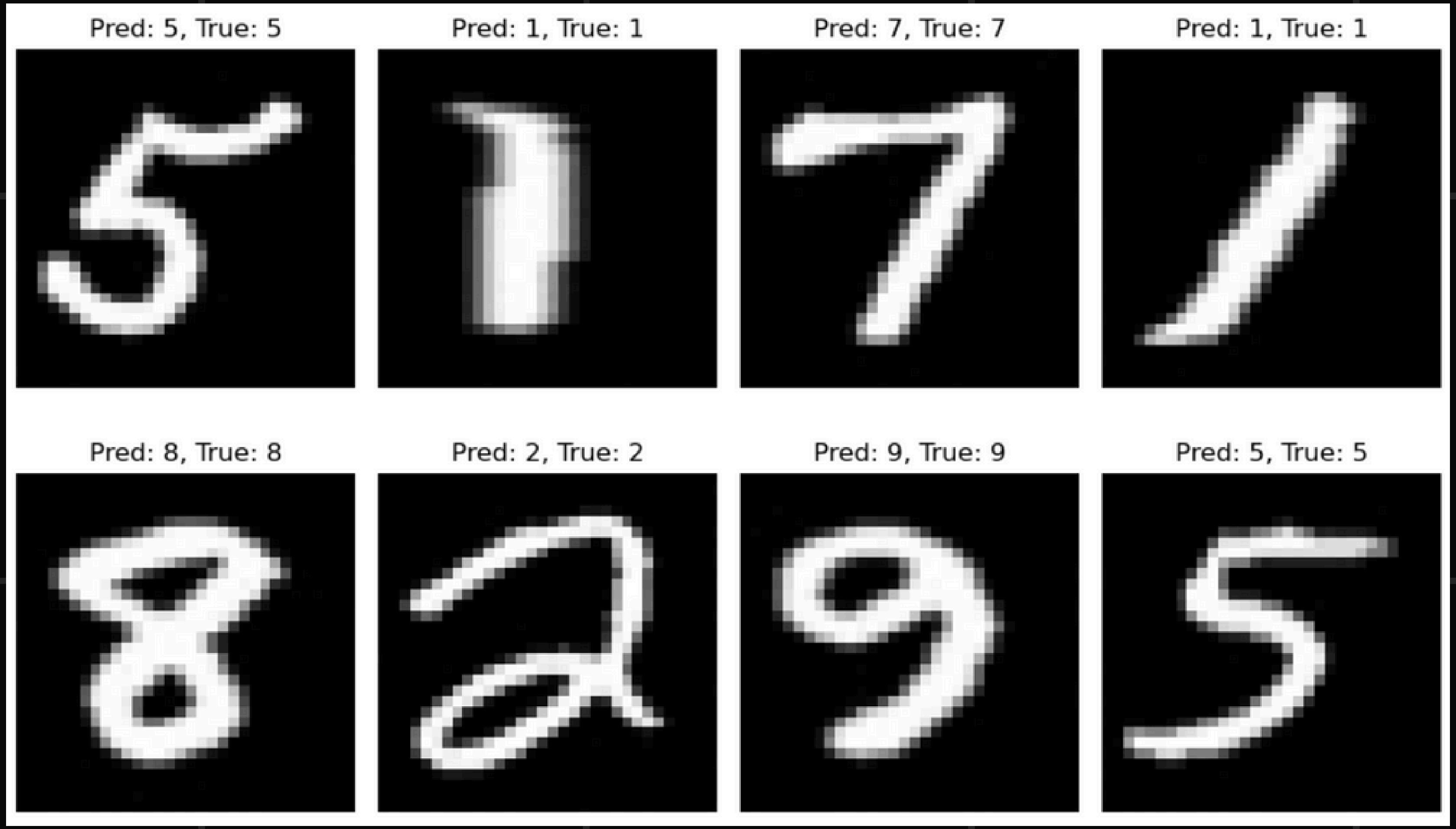
Structure of the CNN (for Numbers):

1. **Input Layer:** Input 32 x 32 x 1 (Images in Grayscale)
2. **Convolutional Layer 1:** Applies 32 filters of the size 3 x 3 over the Input
3. **MaxPooling Layer 1:** Downsampling, takes the maximum of 2 x 2 windows
4. **Convolutional Layer 2:** Applies 64 filters size 3 x 3
5. **MaxPooling Layer 2:** Downsampling, takes the maximum of 2 x 2 windows
6. **Flatten Layer:** Convert the 2D Features into 1D Vector
7. **Dense Layer 1:** Fully Connected Layer with 128 Neurons
8. **Dropout Layer:** Randomly sets 50% of the Neurons to 0 to prevent overfitting
9. **Output Layer:** Fully Connected Layer with 10 Neurons, each class 1 + Softmax Activation

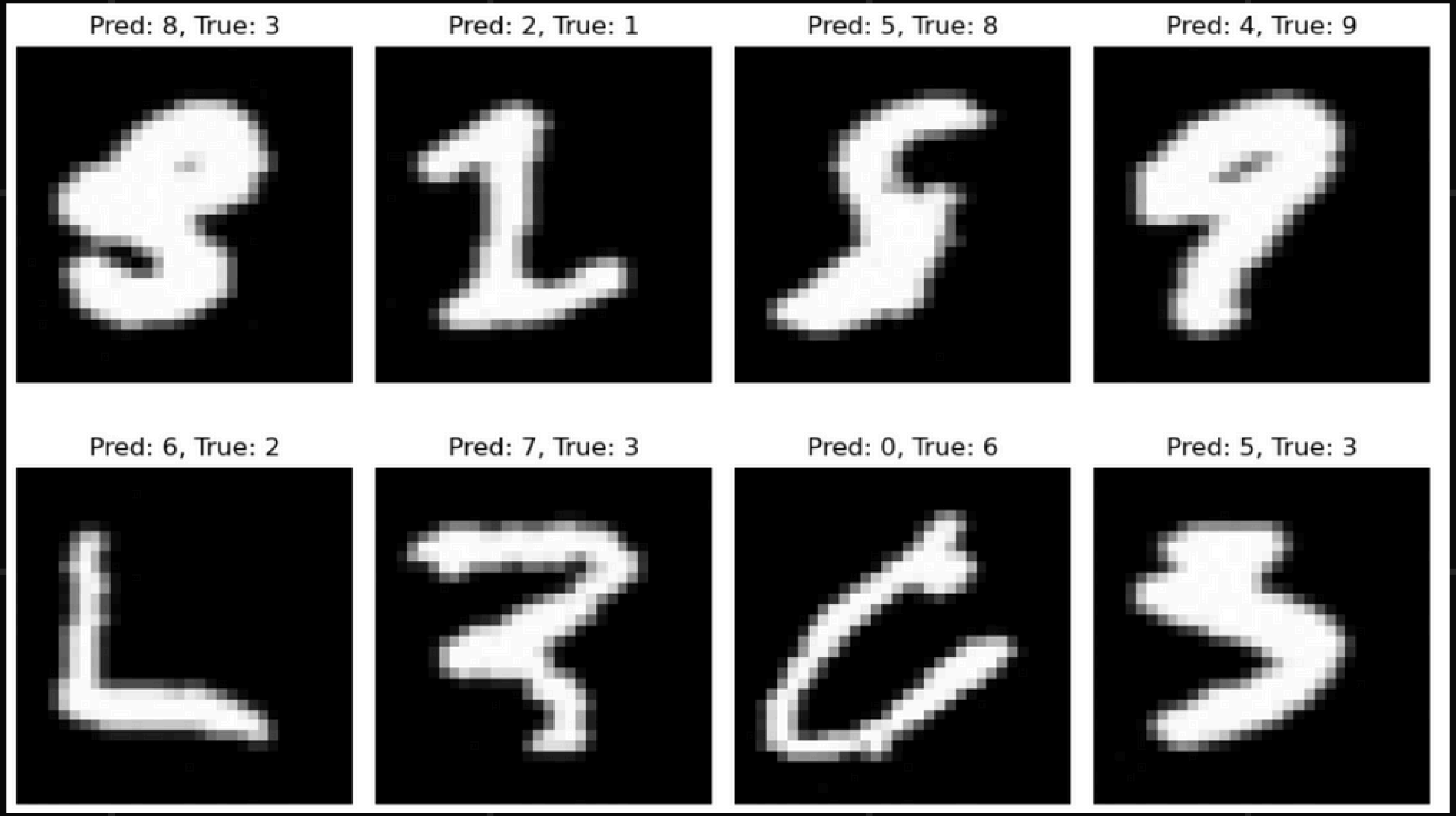
Result: Convolutional Neural Network (Numbers)

- Training the Model for 2000 Images of each Class
- Accuracy of the Test set: 99%
- Training Time: ~ 25 mins

Correct Classified Samples:



Missclassified Samples:



Convolutional Neural Network (Alphabet)

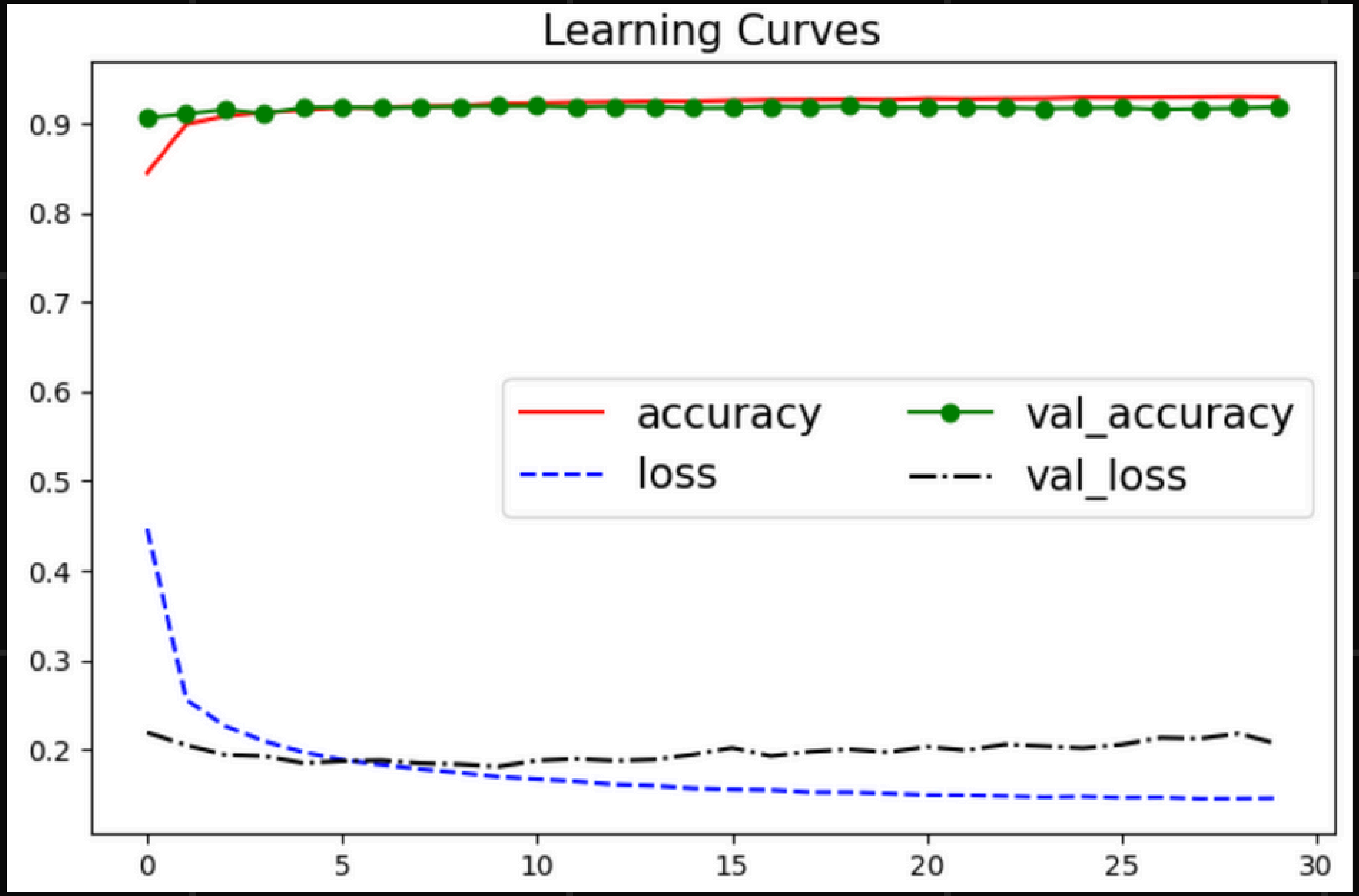
Structure of the CNN (for Alphabet): Contains to Layers more (Conv2D + MaxPooling) and 256 Neurons for to get more precise results

1. **Input Layer:** Input 32 x 32 x 1 (Images in Grayscale)
2. **Convolutional Layer 1:** Applies 32 filters of the size 3 x 3 over the Input
3. **MaxPooling Layer 1:** Downsampling, takes the maximum of 2 x 2 windows
4. **Convolutional Layer 2:** Applies 64 filters size 3 x 3
5. **MaxPooling Layer 2:** Downsampling, takes the maximum of 2 x 2 windows
6. Convolutional Layer 2: Applies 128 filters size 3 x 3
7. MaxPooling Layer 2: Downsampling, takes the maximum of 2 x 2 windows
8. **Flatten Layer:** Convert the 2D Features into 1D Vector
9. **Dense Layer 1:** Fully Connected Layer with 256 Neurons
10. **Dropout Layer:** Randomly sets 50% of the Neurons to 0 to prevent overfitting
11. **Output Layer:** Fully Connected Layer with 10 Neurons, each class 1 + Softmax Activation

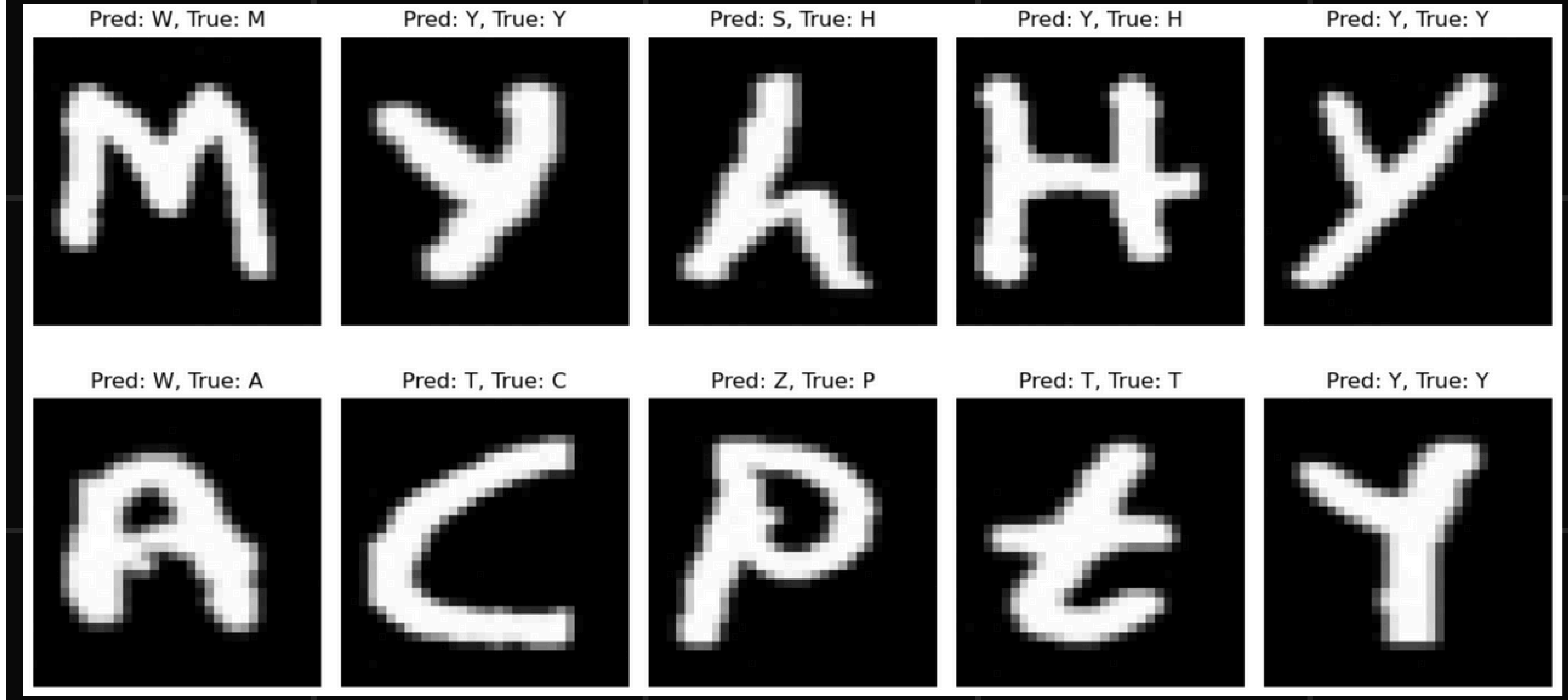
Results: Convolutional Neural Network (Alphabet)

- Training the Model for 10000 Images of each Class
- Accuracy of the Test set: 92%
- Training Time: ~ 1.5 mins

Correct Classified Samples:



Missclassified Samples:



Summary of the Initial Results

- **Random Forest Classifer:**
  - Miss some details and takes a lot of Time just for view Samples.
  - It can classify some samples but if Numbers do not have the typical shape they get misclassified
- **CNN for Numbers:**
  - Just used 2000 Images per class ~ Training time 25 mins
  - Correct predictions for almost the hole dataset (Accuracy 99%)
- **CNN for the Alphabet:**
  - Used 10000 Images per class ~Training Time 1.5 h
  - Correct predictions for the most Samples (Accuracy 92%)
- **CNN for Numbers and Alphabet:**
  - Used 10000 Images per class ~ Training Time 2h
  - Really bad performance: The Numbers and Letters are to similar for ex.: 8 and B, I and 1
  - Accuracy less then 9%

Next Steps

- **The Dataset contains 30000 to 50000 Images per class -> Train 3+ CNN for 10000 Samples of each class for a better overall accuracy. The majority of predictions counts as final output prediction.** (more samples brake the Python Kernel)

- **Finetune the Model:**
    - Use the ImageDataGenerator to random shift, rotate, zoom, flip the pictures to get more robust results
    - Adjust the Learning rate
    - Define a Learning Rate Scheduler
    - Implement a early Stopp Creteria to prevent the model form overfitting

- **Combine the Number and Letter Classifier:**
    - Setup a model to classify the Input as Numbers or Letter

- **(Interpret a Handwritten Text)**

Thank You