University of Florida

Engineering Education Department

Fall 2024

EGN 5442 – Programming for Applied DS

Semester I Project Report

Predicting Handwritten Letters

Rohit Borse (rohitborse@ufl.edu)

TABLE OF CONTENTS

Dataset

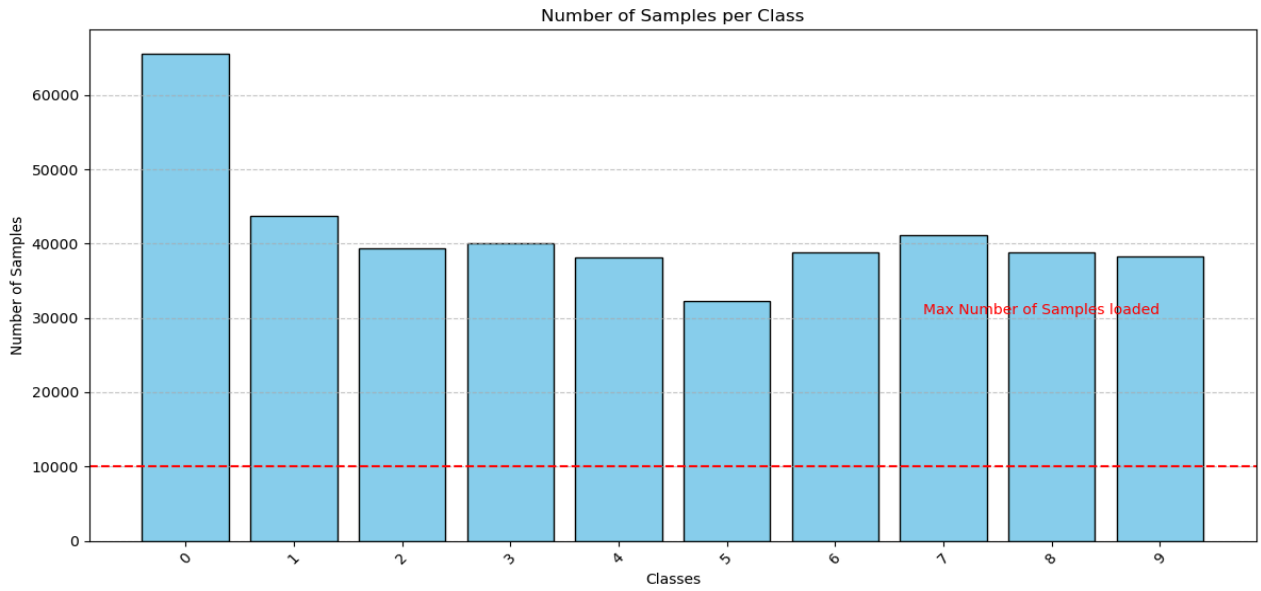
Kaggle Link: https://www.kaggle.com/datasets/vaibhao/handwritten-characters

The dataset used for this project is a comprehensive collection designed for Optical Character Recognition (OCR) tasks, containing English alphabets, digits, and special characters. The key features of the dataset are as follows:

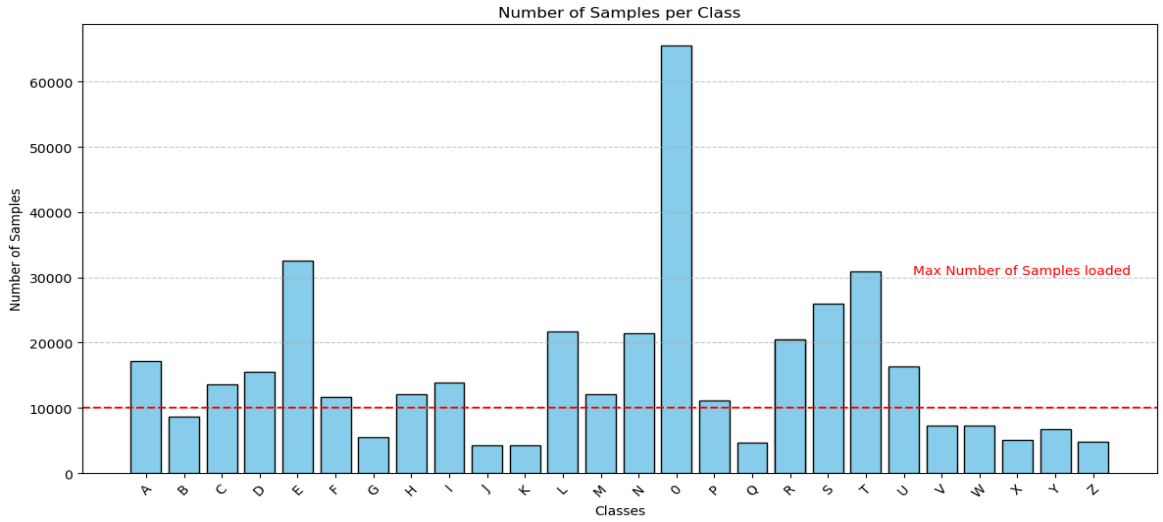
- **Context**:
  The dataset is based on the EMNIST dataset, transformed using image processing techniques to create standardized 32x32 pixel black-and-white images. Additional categories for special characters (@, #, $, &) were created and included in the dataset.
- **Content**:
  The dataset contains a total of 39 categories:
  - **Alphabets**: 26 categories for English alphabets where uppercase and lowercase letters are merged into a single class for each character.
  - **Digits**: 9 categories for numbers (1 to 9), with 0 merged into the character O to avoid misclassification.
  - **Special Characters**: @, #, $, &.
- **Transformation**:
  Image preprocessing techniques were applied to ensure consistency and reduce noise. The images were resized to 32x32 pixels to streamline input dimensions for model training.
- **Purpose**:
  This dataset was curated to minimize misclassification by merging similar-looking categories and provide a robust foundation for OCR and related applications. It serves as an excellent resource for building models to recognize handwritten characters.

Dataset Visualization

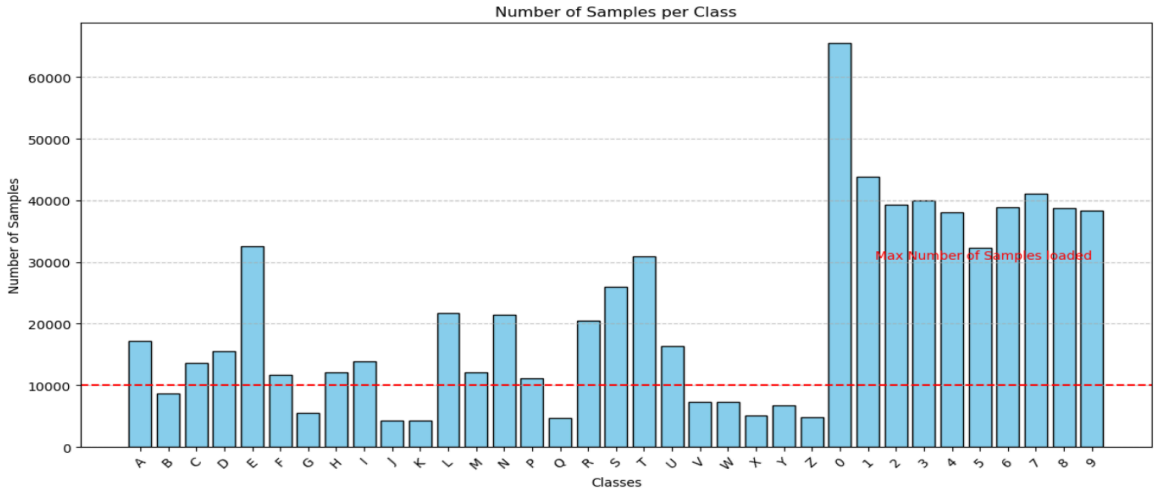
Numbers:



Alphabets



Numbers and Alphabets

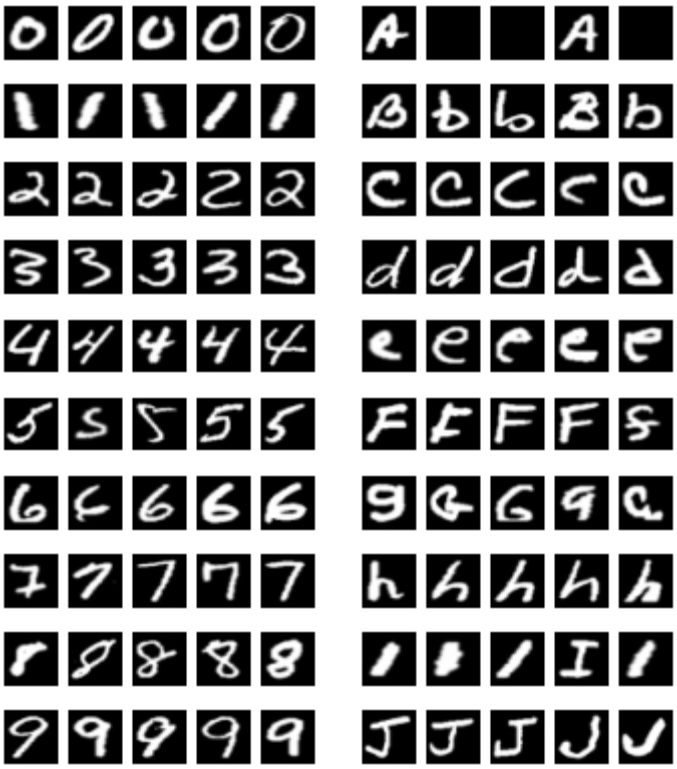


```
Loading Images "A" | Max number of Samples in Folder: 17205 | processing...
Loading Images "B" | Max number of Samples in Folder: 8666 | processing...
Loading Images "C" | Max number of Samples in Folder: 13560 | processing...
Loading Images "D" | Max number of Samples in Folder: 15509 | processing...
Loading Images "E" | Max number of Samples in Folder: 32627 | processing...
Loading Images "F" | Max number of Samples in Folder: 11636 | processing...
Loading Images "G" | Max number of Samples in Folder: 5443 | processing...
Loading Images "H" | Max number of Samples in Folder: 12134 | processing...
Loading Images "I" | Max number of Samples in Folder: 13873 | processing...
Loading Images "J" | Max number of Samples in Folder: 4261 | processing...
Loading Images "K" | Max number of Samples in Folder: 4334 | processing...
Loading Images "L" | Max number of Samples in Folder: 21648 | processing...
Loading Images "M" | Max number of Samples in Folder: 12089 | processing...
Loading Images "N" | Max number of Samples in Folder: 21421 | processing...
Loading Images "0" | Max number of Samples in Folder: 65504 | processing...
Loading Images "P" | Max number of Samples in Folder: 11095 | processing...
Loading Images "Q" | Max number of Samples in Folder: 4707 | processing...
Loading Images "R" | Max number of Samples in Folder: 20498 | processing...
Loading Images "R" | Max number of Samples in Folder: 20498 | processing...
Loading Images "S" | Max number of Samples in Folder: 25911 | processing...
Loading Images "T" | Max number of Samples in Folder: 30853 | processing...
Loading Images "U" | Max number of Samples in Folder: 16385 | processing...
Loading Images "V" | Max number of Samples in Folder: 7246 | processing...
Loading Images "W" | Max number of Samples in Folder: 7266 | processing...
Loading Images "X" | Max number of Samples in Folder: 5106 | processing...
...
```

We utilized three different datasets for the model, each varying in the number of samples per class for numbers and alphabets:

1. **10,000 samples** per class
2. **20,000 samples** per class
3. **30,000 samples** per class

However, since the datasets contain a highly imbalanced distribution of images across different characters, the model's performance may not show significant variation for certain characters, regardless of the dataset size. For instance, the character "G" appears only 5,443 times in the dataset. Consequently, the accuracy for "G" remains relatively unaffected, even as the number of samples per class increases.



Models

		Dataset	Train set Accu.	Test set Accu.	Validation set Accu.
1	Random forest		0.599	0.575	
2	CNN	10,000 (Numeric)	1	0.99	0.99
	CNN	10,000 (Alphabet)	0.93	0.92	0.92
3	Highly Dense CNN	10,000 (Num+Alpha)	0.93	0.92	0.92
	Highly Dense CNN	20,000 (Num+Alpha)	0.92	0.91	0.91
	Highly Dense CNN	30,000 (Num+Alpha)	0.92		0.91
4	CNN with Image DataGenerator		0.92	0.91	0.91
5	CNN with ResNet50	20000 (Alphabet)	0.90	0.91	0.90
	CNN ResNet50 for Alpabet + Numeric	20000 (Num+Alphabet)	N/A (Very high computation required)	N/A (Very high computation required)	N/A (Very high computation required)

1) **Random forest regression:**

1**. Hyperparameter Optimization**

- A parameter grid is defined with different values for key hyperparameters of the classifier, such as:
    - criterion (gini or entropy)
    - max_depth (5, 10, 20, etc.)
    - min_samples_split (2, 4, etc.)
    - min_samples_leaf (1, 2, etc.)
- GridSearchCV is used to systematically test combinations of these hyperparameters.
    - It uses 5-fold cross-validation to ensure robust evaluation for each hyperparameter setting.
    - The best model is selected based on the negative mean squared error as the scoring metric.

2. **Feature Importance Visualization**

- Once the best model is identified, its feature importances are extracted.

- These feature importances are reshaped into a 32x32 grid to match the input image dimensions.
- A heatmap visualization shows which features (pixels) were most influential in the model's decision-making process.

3. **Model Evaluation**

The code evaluates the model on both the training set and the test set using metrics like:

- Accuracy: Percentage of correctly predicted samples.
- Precision: How many of the positive predictions were correct (weighted across classes).
- Recall: How many of the actual positives were correctly identified (weighted across classes).
- F1 Score: Harmonic mean of precision and recall, balancing both.

The metrics are calculated for both sets and a classification report is generated, which provides detailed performance metrics for each class.

4. Observations

1. **Training Set Performance:**
    a. The training accuracy is ~59.9%, indicating the model is reasonably learning from the data but still has room for improvement.
    b. Some classes, such as "G," show higher precision and recall due to better representation, while others suffer due to class imbalance or complexity.
2. **Test Set Performance:**
    a. The test accuracy is ~57.5%, which is slightly lower than the training accuracy, suggesting slight overfitting.
    b. Imbalanced data impacts performance; for instance, the character "8" has low precision but high recall.

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

2) Classification using a Convolutional Neural Network (CNN)

a. Numbers

Dataset Preprocessing

3. **Loading the Dataset:**
    a. The dataset contains 10,000 grayscale images of handwritten digits (0-9), stored in X_10000_numbers.npy (features) and t_10000_numbers.npy (labels).
    b. Each image has a size of 32x32 pixels.
4. **Normalization**:
    a. Pixel values are normalized to a range of [0, 1] by dividing them by 255, ensuring that the input values are small and suitable for the neural network.
5. **Reshaping**:
    a. The images are reshaped to include a single channel (32x32x1) to represent grayscale format.
6. **Splitting the Dataset**:
    a. The dataset is divided into training (80%), validation (16%), and testing (20%) subsets to train, fine-tune, and evaluate the model's performance.
7. **One-Hot Encoding**:

a. Labels are converted into one-hot encoded vectors, where each digit (0-9) is represented as a binary vector for use in categorical classification.

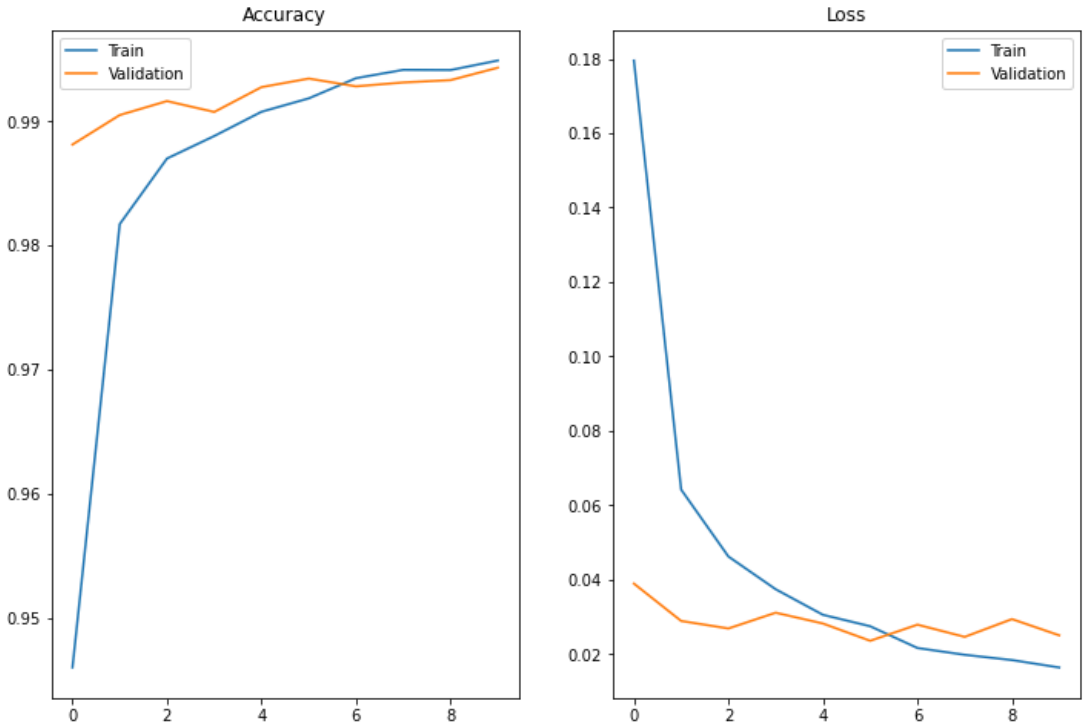
Model Architecture

The Convolutional Neural Network (CNN) is defined as follows:

1. **Input Layer**:
    - Accepts images of size 32x32x1.
2. **Convolutional Layers**:
    - Two convolutional layers extract features from the images:
        - The first layer uses 32 filters of size 3x3.
        - The second layer uses 64 filters of size 3x3.
    - Both layers use ReLU activation for non-linearity.
3. **Pooling Layers**:
    - Max-pooling layers reduce the spatial dimensions by selecting the maximum value in a 2x2 region, effectively down-sampling the image.
4. **Flatten Layer**:
    - Converts the 2D feature maps into a 1D vector for input to the dense layers.
5. **Dense Layers**:
    - A fully connected dense layer with 128 neurons applies further transformations.
    - A dropout layer (50% probability) prevents overfitting.
    - The output layer uses a softmax activation to predict probabilities for the 10 digit classes.

Model Compilation and Training

1. **Compilation**:
    - The model uses the Adam optimizer to minimize the categorical_crossentropy loss function.
    - Accuracy is chosen as the performance metric.
2. **Training**:
    - The model is trained on the training data for 10 epochs with a batch size of 32.
    - Validation data is used to monitor the model's performance during training.



Evaluation and Results

1. **Testing Accuracy**:
    - The model achieves a testing accuracy of 99%, demonstrating excellent performance in classifying handwritten digits.

```
Training Data Shape: (64000, 32, 32, 1), (64000,)
Validation Data Shape: (16000, 32, 32, 1), (16000,)
Epoch 1/10
2000/2000 [==============================] - 4s 2ms/step - loss: 0.1795 - accuracy: 0.9460 - val_loss: 0.0389 - val_accuracy: 0.9881
Epoch 2/10
2000/2000 [==============================] - 3s 2ms/step - loss: 0.0642 - accuracy: 0.9817 - val_loss: 0.0289 - val_accuracy: 0.9905
Epoch 3/10
2000/2000 [==============================] - 3s 2ms/step - loss: 0.0462 - accuracy: 0.9870 - val_loss: 0.0269 - val_accuracy: 0.9916
Epoch 4/10
2000/2000 [==============================] - 3s 2ms/step - loss: 0.0374 - accuracy: 0.9888 - val_loss: 0.0311 - val_accuracy: 0.9908
Epoch 5/10
2000/2000 [==============================] - 3s 2ms/step - loss: 0.0305 - accuracy: 0.9908 - val_loss: 0.0282 - val_accuracy: 0.9927
Epoch 6/10
2000/2000 [==============================] - 3s 2ms/step - loss: 0.0274 - accuracy: 0.9918 - val_loss: 0.0235 - val_accuracy: 0.9934
Epoch 7/10
2000/2000 [==============================] - 3s 2ms/step - loss: 0.0216 - accuracy: 0.9935 - val_loss: 0.0279 - val_accuracy: 0.9928
Epoch 8/10
2000/2000 [==============================] - 3s 2ms/step - loss: 0.0198 - accuracy: 0.9941 - val_loss: 0.0246 - val_accuracy: 0.9931
Epoch 9/10
2000/2000 [==============================] - 3s 2ms/step - loss: 0.0184 - accuracy: 0.9941 - val_loss: 0.0294 - val_accuracy: 0.9933
Epoch 10/10
2000/2000 [==============================] - 3s 2ms/step - loss: 0.0164 - accuracy: 0.9949 - val_loss: 0.0250 - val_accuracy: 0.9943
625/625 - 1s - loss: 0.0278 - accuracy: 0.9945 - 512ms/epoch - 819us/step
Test accuracy: 0.99
```

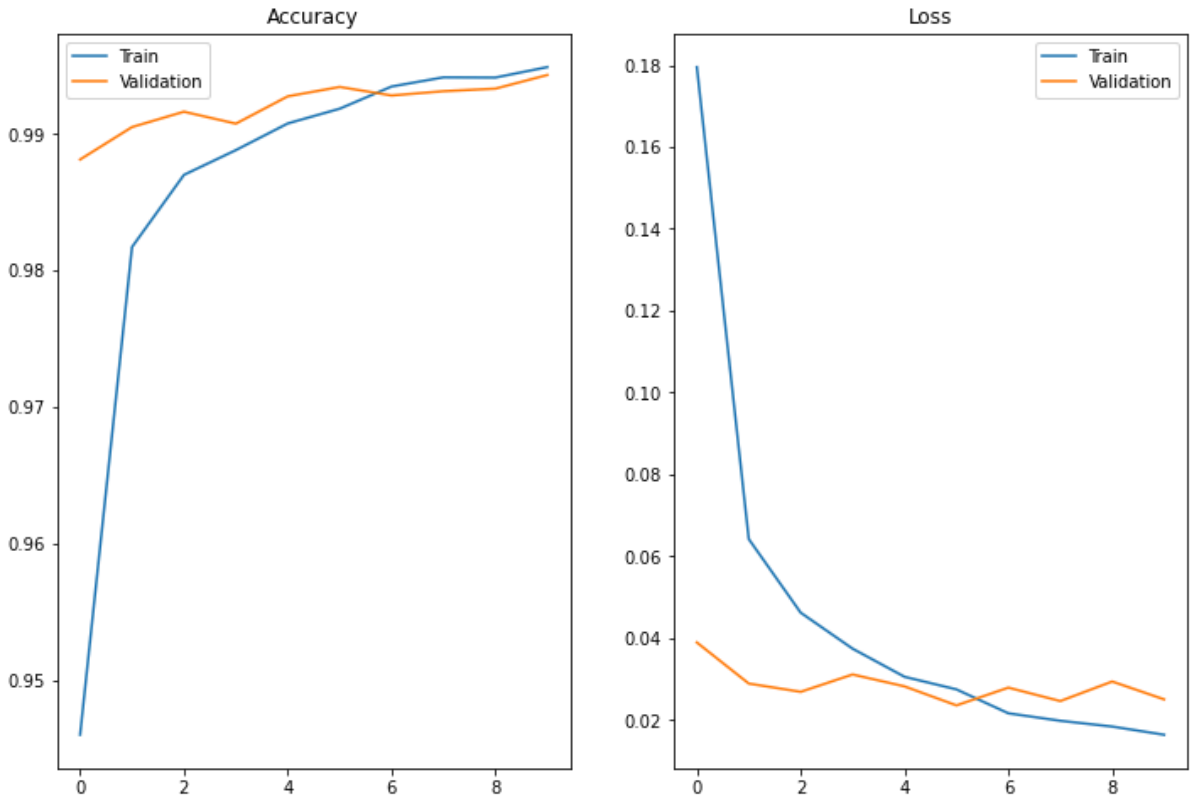
Training Set Performance

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	6396
1.0	1.00	1.00	1.00	6390
2.0	1.00	1.00	1.00	6418
3.0	1.00	1.00	1.00	6370
4.0	1.00	1.00	1.00	6399
5.0	1.00	1.00	1.00	6400
6.0	1.00	1.00	1.00	6412
7.0	1.00	1.00	1.00	6398
8.0	1.00	1.00	1.00	6406
9.0	1.00	1.00	1.00	6411
accuracy			1.00	64000
macro avg	1.00	1.00	1.00	64000
weighted avg	1.00	1.00	1.00	64000

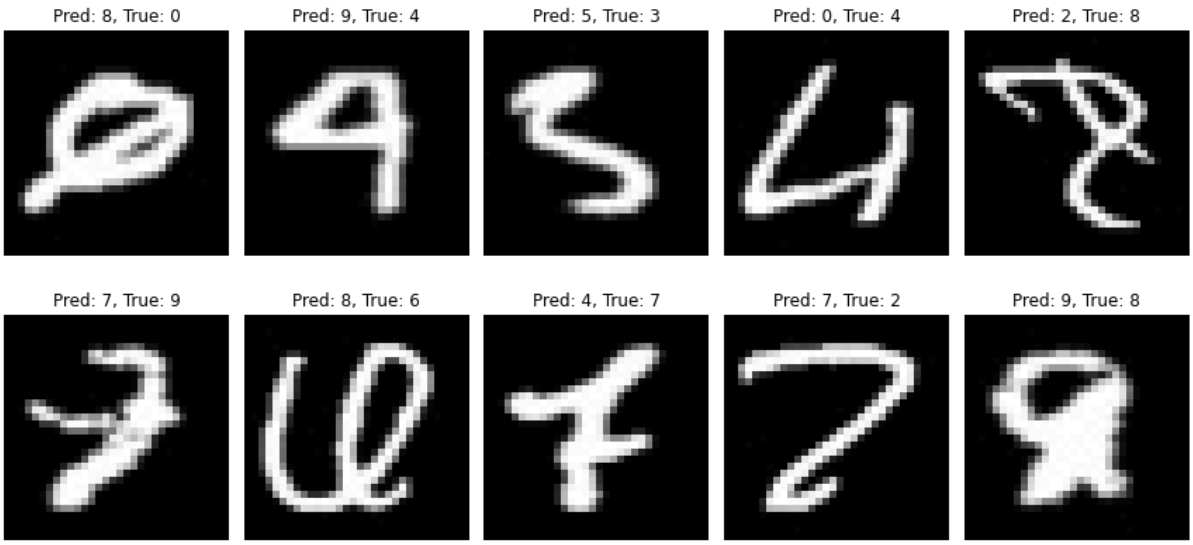
Validation Set Performance

	precision	recall	f1-score	support
0.0	0.99	1.00	0.99	1599
1.0	1.00	1.00	1.00	1597
2.0	1.00	0.99	0.99	1605
3.0	1.00	0.99	0.99	1592
4.0	0.99	0.99	0.99	1599
5.0	0.99	1.00	0.99	1600
6.0	1.00	0.99	0.99	1603
7.0	0.99	1.00	0.99	1600
8.0	0.99	1.00	0.99	1602
9.0	1.00	0.99	0.99	1603
accuracy			0.99	16000
macro avg	0.99	0.99	0.99	16000
weighted avg	0.99	0.99	0.99	16000

2. **Learning Curves**:
    - The training and validation accuracy and loss are plotted, showing consistent improvement without overfitting.



3. **Classification Reports**:
    - Performance metrics (precision, recall, F1-score) and confusion matrices for the training and validation sets confirm the model's high accuracy across all digit classes.
4. **Misclassified Samples**:
    - Misclassified test samples are displayed along with their true and predicted labels, providing insight into the model's errors.



b. **Alphabets**

Data Preprocessing

1. **Normalization:**
   The input image data is normalized by dividing the pixel values by 255.0, converting them into a range between 0 and 1. This helps the model learn more efficiently.
2. **Reshaping:**
   The image data is reshaped to include a channel dimension (32x32x1) because the images are grayscale. The channel dimension is required for processing by the convolutional neural network (CNN).
3. **Dataset Splitting:**
   The dataset is split into:
   - Training set (to train the model).
   - Validation set (to check model performance during training).
   - Test set (to evaluate final performance).
     Splitting ensures data used for testing is unseen during training.
4. **One-Hot Encoding:**
   The target labels are converted to one-hot encoded vectors. For example, label A becomes [1, 0, 0, ..., 0]. This format is required for multi-class classification.

Model Architecture

1. **Convolutional Neural Network (CNN):**
   A CNN is defined with the following layers:
   - **Convolutional layers:** Extract features like edges or shapes from the images.
   - **Pooling layers:** Reduce the spatial size of the features, making computations faster and reducing overfitting.
   - **Dense layers:** Fully connected layers process the extracted features and classify the images.

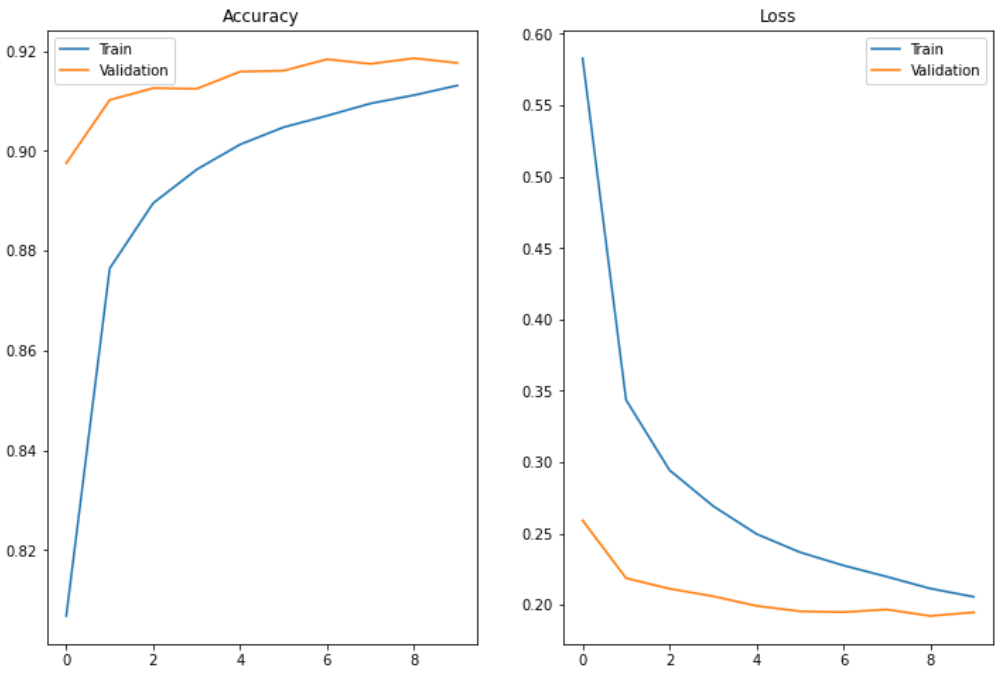
- **Dropout layer:** Randomly turns off some neurons during training to prevent overfitting.
- **Output layer:** Uses a softmax activation to output probabilities for each class.

Model Training and Evaluation

1. **Compilation:**
   The model is compiled using:
   - **Adam optimizer:** Adjusts learning rates dynamically for faster convergence.
   - **Categorical cross-entropy:** Loss function for multi-class classification problems.
   - **Accuracy metric:** Evaluates model performance.
2. **Training:**
   The model is trained using the training data while validating performance on the validation set over multiple epochs. Batch size defines the number of samples processed together during one training step.
3. **Evaluation:**
   After training, the model's performance is evaluated on the test set using accuracy and loss metrics.
4. **Saving the Model:**
   The trained model is saved as alpha_classifier_10000.h5 for reuse without retraining.

Performance Visualization

1. **Learning Curves:**
   Plots showing the training and validation accuracy and loss across epochs help in understanding the model's learning progress and identifying overfitting or underfitting.



2. **Confusion Matrix and Classification Report:**
   These tools evaluate how well the model performs on each class:
   - **Precision, Recall, F1-Score:** Measures of the model's prediction quality for each class.
   - **Confusion Matrix:** Shows the number of correct and incorrect predictions for each class.

```
Epoch 1/10
5400/5400 [==============================] - 10s 2ms/step - loss: 0.5830 - accuracy: 0.8068 - val_loss: 0.2592 - val_accuracy: 0.8975
Epoch 2/10
5400/5400 [==============================] - 9s 2ms/step - loss: 0.3436 - accuracy: 0.8765 - val_loss: 0.2188 - val_accuracy: 0.9102
Epoch 3/10
5400/5400 [==============================] - 9s 2ms/step - loss: 0.2944 - accuracy: 0.8895 - val_loss: 0.2113 - val_accuracy: 0.9125
Epoch 4/10
5400/5400 [==============================] - 9s 2ms/step - loss: 0.2693 - accuracy: 0.8962 - val_loss: 0.2060 - val_accuracy: 0.9124
Epoch 5/10
5400/5400 [==============================] - 9s 2ms/step - loss: 0.2497 - accuracy: 0.9013 - val_loss: 0.1993 - val_accuracy: 0.9159
Epoch 6/10
5400/5400 [==============================] - 10s 2ms/step - loss: 0.2369 - accuracy: 0.9047 - val_loss: 0.1955 - val_accuracy: 0.9160
Epoch 7/10
5400/5400 [==============================] - 9s 2ms/step - loss: 0.2276 - accuracy: 0.9070 - val_loss: 0.1949 - val_accuracy: 0.9183
Epoch 8/10
5400/5400 [==============================] - 9s 2ms/step - loss: 0.2197 - accuracy: 0.9095 - val_loss: 0.1968 - val_accuracy: 0.9174
Epoch 9/10
5400/5400 [==============================] - 9s 2ms/step - loss: 0.2115 - accuracy: 0.9111 - val_loss: 0.1922 - val_accuracy: 0.9185
Epoch 10/10
5400/5400 [==============================] - 9s 2ms/step - loss: 0.2057 - accuracy: 0.9131 - val_loss: 0.1948 - val_accuracy: 0.9176
1688/1688 - 2s - loss: 0.2019 - accuracy: 0.9167 - 2s/epoch - 1ms/step
Test accuracy: 0.92
```

Training Set Performance

	precision	recall	f1-score	support
A	0.99	1.00	1.00	32776
B	0.99	0.99	0.99	5538
C	0.99	0.98	0.99	6416
D	0.98	0.99	0.98	6378
E	0.98	0.99	0.98	6402
F	0.99	0.99	0.99	6396
G	0.93	0.88	0.90	3495
H	0.99	0.97	0.98	6363
I	0.77	0.78	0.78	6398
J	0.98	0.97	0.97	2790
K	0.98	0.99	0.99	2789
L	0.77	0.77	0.77	6386
M	1.00	0.99	0.99	6394
N	0.96	0.99	0.97	6391
0	0.99	0.98	0.98	6377
P	0.99	0.99	0.99	6404
Q	0.90	0.92	0.91	3014
R	0.50	0.80	0.61	6407
R	0.51	0.20	0.29	6408
S	0.98	1.00	0.99	6402
T	0.99	0.99	0.99	6422
U	0.97	0.97	0.97	6435
...				
accuracy			0.93	172800
macro avg	0.93	0.93	0.92	172800
weighted avg	0.93	0.93	0.93	172800

Validation Set Performance

	precision	recall	f1-score	support
A	0.99	0.99	0.99	8194
B	0.97	0.98	0.98	1384
C	0.98	0.97	0.98	1604
D	0.95	0.97	0.96	1595
E	0.97	0.97	0.97	1601
F	0.98	0.97	0.97	1599
G	0.88	0.82	0.85	874
H	0.97	0.95	0.96	1591
I	0.76	0.78	0.77	1599
J	0.96	0.96	0.96	697
K	0.96	0.98	0.97	697
L	0.76	0.75	0.76	1597
M	0.99	0.98	0.99	1599
N	0.94	0.99	0.96	1598
0	0.98	0.96	0.97	1594
P	0.98	0.99	0.98	1601
Q	0.85	0.85	0.85	753
R	0.48	0.76	0.59	1601
R	0.47	0.20	0.28	1602
S	0.97	0.99	0.98	1601
T	0.97	0.98	0.97	1605
U	0.96	0.95	0.96	1609
...				
accuracy			0.92	43200
macro avg	0.91	0.91	0.91	43200
weighted avg	0.92	0.92	0.91	43200

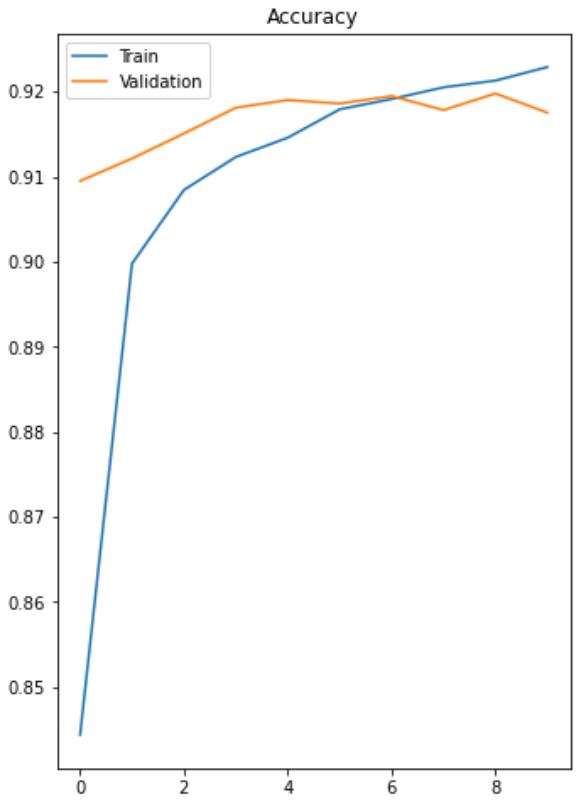
Misclassified Samples

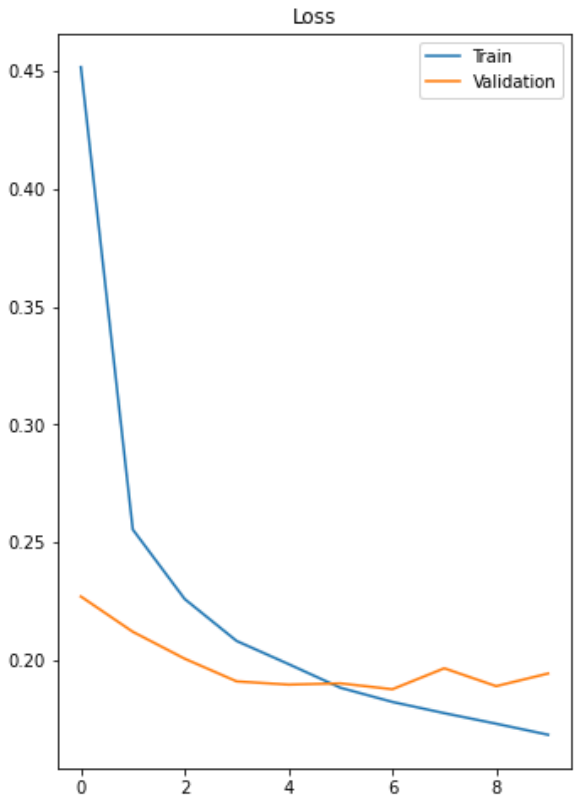
1. **Identifying Errors:**
   The code identifies samples where the predicted label does not match the true label.
2. **Visualization:**
   A random selection of misclassified images is displayed along with the predicted and true labels. This helps in understanding the model's mistakes, which can be useful for debugging or improving the model.

3) Classification using a Highly dense Convolutional Neural Network (CNN)

a. For the Dataset with 10000 Letters of each Class

1. **Data Preprocessing:**
   - Normalized pixel values to [0, 1].
   - Reshaped images to (32, 32, 1) to include the channel dimension.
   - Splits data into training, validation, and testing sets.
   -
2. **Model Architecture:**
   - Three convolutional layers with ReLU activation, each followed by max-pooling.
   - Two fully connected (dense) layers with a dropout of 0.5 for regularization.
   - Output layer uses the softmax activation function to predict probabilities for each of the 27 classes.
3. **Training:**
   - Compiled with Adam optimizer and categorical cross-entropy loss.
   - Trained for 10 epochs with a batch size of 32.
4. **Performance Metrics:**
   - Achieved **92% test accuracy**.
   - Learning curves (accuracy and loss) show consistent training and validation performance.
5. **Evaluation:**
   - Classification reports and confusion matrices highlight performance per class.
   - Visualized misclassified examples to analyze specific challenges in predictions.





Results:

```
Epoch 1/10
5400/5400 [==============================] - 11s 2ms/step - loss: 0.4518 - accuracy: 0.8444 - val_loss: 0.2270 - val_accuracy: 0.9095
Epoch 2/10
5400/5400 [==============================] - 10s 2ms/step - loss: 0.2554 - accuracy: 0.8998 - val_loss: 0.2120 - val_accuracy: 0.9121
Epoch 3/10
5400/5400 [==============================] - 10s 2ms/step - loss: 0.2259 - accuracy: 0.9084 - val_loss: 0.2005 - val_accuracy: 0.9151
Epoch 4/10
5400/5400 [==============================] - 10s 2ms/step - loss: 0.2081 - accuracy: 0.9123 - val_loss: 0.1909 - val_accuracy: 0.9181
Epoch 5/10
5400/5400 [==============================] - 10s 2ms/step - loss: 0.1983 - accuracy: 0.9146 - val_loss: 0.1895 - val_accuracy: 0.9190
Epoch 6/10
5400/5400 [==============================] - 10s 2ms/step - loss: 0.1882 - accuracy: 0.9179 - val_loss: 0.1901 - val_accuracy: 0.9186
Epoch 7/10
5400/5400 [==============================] - 10s 2ms/step - loss: 0.1822 - accuracy: 0.9191 - val_loss: 0.1876 - val_accuracy: 0.9195
Epoch 8/10
5400/5400 [==============================] - 10s 2ms/step - loss: 0.1774 - accuracy: 0.9205 - val_loss: 0.1965 - val_accuracy: 0.9178
Epoch 9/10
5400/5400 [==============================] - 10s 2ms/step - loss: 0.1729 - accuracy: 0.9213 - val_loss: 0.1889 - val_accuracy: 0.9198
Epoch 10/10
5400/5400 [==============================] - 10s 2ms/step - loss: 0.1682 - accuracy: 0.9229 - val_loss: 0.1942 - val_accuracy: 0.9175
1688/1688 - 2s - loss: 0.1961 - accuracy: 0.9174 - 2s/epoch - 1ms/step
Test accuracy: 0.92
```

Training Set Performance

	precision	recall	f1-score	support
A	0.99	1.00	1.00	32776
B	0.99	0.99	0.99	5538
C	0.99	0.99	0.99	6416
D	0.98	0.99	0.98	6378
E	0.98	0.99	0.99	6402
F	0.99	0.98	0.99	6396
G	0.95	0.85	0.89	3495
H	0.98	0.99	0.98	6363
I	0.76	0.80	0.78	6398
J	0.99	0.97	0.98	2790
K	0.99	0.99	0.99	2789
L	0.79	0.73	0.76	6387
M	1.00	0.99	1.00	6394
N	0.97	0.99	0.98	6391
0	0.99	0.98	0.99	6377
P	0.99	0.99	0.99	6404
Q	0.87	0.94	0.90	3014
R	0.50	0.73	0.59	6406
R	0.51	0.27	0.36	6408
S	0.98	1.00	0.99	6402
T	0.98	0.99	0.99	6422
U	0.97	0.97	0.97	6435
...				
accuracy			0.93	172800
macro avg	0.93	0.93	0.92	172800
weighted avg	0.93	0.93	0.93	172800

Validation Set Performance

	precision	recall	f1-score	support
A	0.99	0.99	0.99	8194
B	0.98	0.98	0.98	1384
C	0.97	0.97	0.97	1604
D	0.96	0.97	0.97	1595
E	0.96	0.98	0.97	1601
F	0.99	0.96	0.97	1599
G	0.92	0.81	0.86	874
H	0.97	0.97	0.97	1591
I	0.74	0.76	0.75	1599
J	0.96	0.94	0.95	697
K	0.97	0.98	0.98	697
L	0.76	0.72	0.74	1596
M	0.99	0.99	0.99	1599
N	0.96	0.98	0.97	1598
0	0.96	0.96	0.96	1594
P	0.99	0.99	0.99	1601
Q	0.84	0.89	0.86	753
R	0.48	0.72	0.58	1602
R	0.47	0.24	0.32	1602
S	0.97	0.99	0.98	1601
T	0.97	0.97	0.97	1605
U	0.96	0.96	0.96	1609
...				
accuracy			0.92	43200
macro avg	0.91	0.91	0.91	43200
weighted avg	0.92	0.92	0.92	43200

Performance Analysis

1. **Strengths:**
    - The architecture generalizes well, evidenced by similar train and validation accuracy (~92%).
    - High precision and recall for most classes like 'A', 'B', 'C', and others, reflecting the model's ability to distinguish well-defined patterns.
2. **Challenges:**
    - Classes like 'R' have low precision and recall due to:
        - Overlap or similarity in features with other classes.
        - Imbalanced training samples or noise in data.
3. **Insights from Misclassifications:**
    - Letters with visually similar shapes may confuse the model.
    - Classes with fewer samples or high intraclass variance contribute to errors.

b. Results for Dataset having upto 20000 samples in dataset

```
3]

..  2024-12-11 19:36:51.102784: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep N
    To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
    2024-12-11 19:36:51.906811: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1525] Created device /job:localhost/replica:0/task:0/dev
    Epoch 1/10
    2024-12-11 19:36:54.458773: I tensorflow/stream_executor/cuda/cuda_dnn.cc:366] Loaded cuDNN version 8201
    2024-12-11 19:36:56.477349: I tensorflow/stream_executor/cuda/cuda_blas.cc:1774] TensorFloat-32 will be used for the matrix multiplicat
    8423/8423 [==============================] - 22s 2ms/step - loss: 0.3835 - accuracy: 0.8583 - val_loss: 0.2192 - val_accuracy: 0.9023
    Epoch 2/10
    8423/8423 [==============================] - 18s 2ms/step - loss: 0.2380 - accuracy: 0.8991 - val_loss: 0.2012 - val_accuracy: 0.9088
    Epoch 3/10
    8423/8423 [==============================] - 18s 2ms/step - loss: 0.2165 - accuracy: 0.9042 - val_loss: 0.1989 - val_accuracy: 0.9106
    Epoch 4/10
    8423/8423 [==============================] - 18s 2ms/step - loss: 0.2046 - accuracy: 0.9073 - val_loss: 0.1935 - val_accuracy: 0.9111
    Epoch 5/10
    8423/8423 [==============================] - 18s 2ms/step - loss: 0.1970 - accuracy: 0.9100 - val_loss: 0.1885 - val_accuracy: 0.9121
    Epoch 6/10
    8423/8423 [==============================] - 18s 2ms/step - loss: 0.1920 - accuracy: 0.9107 - val_loss: 0.1937 - val_accuracy: 0.9108
    Epoch 7/10
    8423/8423 [==============================] - 18s 2ms/step - loss: 0.1873 - accuracy: 0.9122 - val_loss: 0.1930 - val_accuracy: 0.9124
    Epoch 8/10
    8423/8423 [==============================] - 18s 2ms/step - loss: 0.1849 - accuracy: 0.9134 - val_loss: 0.1951 - val_accuracy: 0.9107
    Epoch 9/10
    8423/8423 [==============================] - 18s 2ms/step - loss: 0.1814 - accuracy: 0.9129 - val_loss: 0.1911 - val_accuracy: 0.9124
    Epoch 10/10
    8423/8423 [==============================] - 18s 2ms/step - loss: 0.1805 - accuracy: 0.9141 - val_loss: 0.1891 - val_accuracy: 0.9132
    2632/2632 - 3s - loss: 0.1891 - accuracy: 0.9120 - 3s/epoch - 1ms/step
    Test accuracy: 0.91
```

Training Set Performance

	precision	recall	f1-score	support
A	0.99	1.00	1.00	61482
B	0.99	0.98	0.99	5574
C	0.99	0.98	0.98	8694
D	0.98	0.98	0.98	9864
E	0.99	0.99	0.99	12806
F	0.99	0.98	0.98	7418
G	0.95	0.83	0.89	3448
H	0.98	0.98	0.98	7765
I	0.77	0.65	0.71	8900
J	0.98	0.96	0.97	2723
K	0.98	0.99	0.99	2786
L	0.78	0.86	0.82	12826
M	0.99	1.00	1.00	7749
N	0.98	0.99	0.99	12794
0	0.99	0.99	0.99	12770
P	0.99	0.99	0.99	7110
Q	0.85	0.92	0.89	3030
R	0.50	0.95	0.65	12770
R	0.48	0.05	0.09	12748
S	1.00	0.99	1.00	12867
T	0.98	1.00	0.99	12957
U	0.98	0.97	0.98	10450
...				
accuracy			0.92	269506
macro avg	0.93	0.92	0.91	269506
weighted avg	0.92	0.92	0.91	269506

Validation Set Performance

	precision	recall	f1-score	support
A	0.99	1.00	0.99	15371
B	0.99	0.97	0.98	1394
C	0.97	0.97	0.97	2173
D	0.97	0.97	0.97	2466
E	0.98	0.98	0.98	3202
F	0.98	0.97	0.98	1855
G	0.92	0.77	0.83	862
H	0.96	0.97	0.96	1941
I	0.77	0.65	0.71	2225
J	0.96	0.95	0.95	681
K	0.96	0.97	0.97	696
L	0.77	0.86	0.81	3207
M	0.99	0.99	0.99	1937
N	0.97	0.98	0.97	3199
0	0.97	0.98	0.98	3193
P	0.98	0.99	0.98	1777
Q	0.83	0.88	0.85	757
R	0.49	0.94	0.65	3192
R	0.47	0.05	0.08	3187
S	0.99	0.99	0.99	3217
T	0.97	0.98	0.98	3239
U	0.97	0.95	0.96	2612
...				
accuracy			0.91	67377
macro avg	0.91	0.91	0.90	67377
weighted avg	0.91	0.91	0.90	67377

c. Results for Dataset having upto 30000 samples in dataset

```
2024-12-10 01:16:48.831591: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFl
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-12-10 01:16:49.735714: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1525] Created d
Epoch 1/10
2024-12-10 01:16:52.334398: I tensorflow/stream_executor/cuda/cuda_dnn.cc:366] Loaded cuDNN ve
2024-12-10 01:16:54.018057: I tensorflow/stream_executor/cuda/cuda_blas.cc:1774] TensorFloat-3
8423/8423 [==============================] - 22s 2ms/step - loss: 0.3893 - accuracy: 0.8561 -
Epoch 2/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.2463 - accuracy: 0.8962 -
Epoch 3/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.2248 - accuracy: 0.9017 -
Epoch 4/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.2121 - accuracy: 0.9055 -
Epoch 5/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.2036 - accuracy: 0.9072 -
Epoch 6/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.1996 - accuracy: 0.9087 -
Epoch 7/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.1946 - accuracy: 0.9097 -
Epoch 8/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.1920 - accuracy: 0.9099 -
Epoch 9/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.1888 - accuracy: 0.9112 -
Epoch 10/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.1868 - accuracy: 0.9118 -
2632/2632 - 3s - loss: 0.1983 - accuracy: 0.9090 - 3s/epoch - 1ms/step
Test accuracy: 0.91
```

Training Set Performance

	precision	recall	f1-score	support
A	0.99	1.00	0.99	35922
B	0.98	0.99	0.98	5574
C	0.98	0.97	0.98	8694
D	0.99	0.97	0.98	9864
E	0.99	0.99	0.99	19193
F	0.99	0.98	0.99	7361
G	0.94	0.84	0.88	3487
H	0.96	0.99	0.97	7825
I	0.78	0.62	0.69	8887
J	0.97	0.97	0.97	2739
K	0.99	0.99	0.99	2764
L	0.78	0.88	0.83	13882
M	1.00	0.99	0.99	7705
N	0.99	0.98	0.98	13761
0	0.99	0.99	0.99	19185
P	0.99	0.99	0.99	7070
Q	0.87	0.91	0.89	2974
R	0.46	0.02	0.04	13104
R	0.50	0.97	0.66	13215
S	1.00	0.99	1.00	16649
T	0.99	0.99	0.99	19193
U	0.98	0.98	0.98	10442
...				
accuracy			0.92	269506
macro avg	0.93	0.92	0.91	269506
weighted avg	0.92	0.92	0.91	269506

Validation Set Performance

	precision	recall	f1-score	support
A	0.99	0.99	0.99	8981
B	0.96	0.98	0.97	1394
C	0.97	0.96	0.97	2173
D	0.98	0.96	0.97	2466
E	0.98	0.99	0.98	4798
F	0.98	0.97	0.98	1840
G	0.90	0.82	0.86	872
H	0.95	0.97	0.96	1956
I	0.76	0.62	0.69	2222
J	0.95	0.94	0.95	685
K	0.98	0.97	0.97	691
L	0.77	0.87	0.82	3471
M	0.99	0.99	0.99	1926
N	0.98	0.97	0.98	3440
0	0.98	0.99	0.98	4797
P	0.98	0.99	0.98	1768
Q	0.84	0.86	0.85	743
R	0.46	0.02	0.04	3276
R	0.50	0.97	0.66	3304
S	0.99	0.99	0.99	4162
T	0.98	0.98	0.98	4799
U	0.96	0.98	0.97	2610
...				
accuracy			0.91	67377
macro avg	0.91	0.91	0.90	67377
weighted avg	0.91	0.91	0.90	67377

The results for all the datasets remain almost similar because the sample in the dataset is less than 10000 for a significant number of the samples. Therefore the testing and validation set accuracies do not change significantly.

4) CNN with Image DataGenerator

Data Normalization and Reshaping:

- The data X_alp is normalized to scale pixel values between 0 and 1 by dividing by 255.0.
- The dataset is reshaped to include a channel dimension, making the images compatible with CNN input requirements (32x32x1 for grayscale).

Splitting Data:

- The dataset is split into training and testing sets using an 80-20 split.
- The training set is further split into training and validation sets with stratification to maintain class balance.

One-Hot Encoding:

- Labels are converted into one-hot encoding format for 27 classes (A-Z excluding 'O').

Data Augmentation:

- An ImageDataGenerator is used to apply random augmentations such as rotation, shifting, shearing, zooming, and horizontal flipping to the training data.
- The generator is fitted on the training data to prepare for augmentation.

Model Definition:

- A CNN model is built with:
    - Three convolutional layers followed by max-pooling layers.
    - A flattening layer to convert feature maps into a vector.
    - Dense layers for classification, with dropout to prevent overfitting.
    - An output layer with softmax activation for multi-class classification.

Model Compilation:

- The model is compiled using the Adam optimizer, categorical cross entropy loss, and accuracy as the evaluation metric.

Model Training:

- The model is trained using the augmented data, with validation on a separate dataset, for 30 epochs and a batch size of 32.

Model Evaluation:

- The model's performance is evaluated on the test dataset, and the accuracy is printed.

```
2024-12-10 01:16:48.831591: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Ne
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-12-10 01:16:49.735714: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1525] Created device /job:localhost/replica:0/task:0/devi
Epoch 1/10
2024-12-10 01:16:52.334398: I tensorflow/stream_executor/cuda/cuda_dnn.cc:366] Loaded cuDNN version 8201
2024-12-10 01:16:54.018057: I tensorflow/stream_executor/cuda/cuda_blas.cc:1774] TensorFloat-32 will be used for the matrix multiplicati
8423/8423 [==============================] - 22s 2ms/step - loss: 0.3893 - accuracy: 0.8561 - val_loss: 0.2271 - val_accuracy: 0.8990
Epoch 2/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.2463 - accuracy: 0.8962 - val_loss: 0.2078 - val_accuracy: 0.9059
Epoch 3/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.2248 - accuracy: 0.9017 - val_loss: 0.1985 - val_accuracy: 0.9083
Epoch 4/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.2121 - accuracy: 0.9055 - val_loss: 0.2001 - val_accuracy: 0.9086
Epoch 5/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.2036 - accuracy: 0.9072 - val_loss: 0.1954 - val_accuracy: 0.9087
Epoch 6/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.1996 - accuracy: 0.9087 - val_loss: 0.1955 - val_accuracy: 0.9096
Epoch 7/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.1946 - accuracy: 0.9097 - val_loss: 0.1946 - val_accuracy: 0.9119
Epoch 8/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.1920 - accuracy: 0.9099 - val_loss: 0.1950 - val_accuracy: 0.9109
Epoch 9/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.1888 - accuracy: 0.9112 - val_loss: 0.2014 - val_accuracy: 0.9086
Epoch 10/10
8423/8423 [==============================] - 18s 2ms/step - loss: 0.1868 - accuracy: 0.9118 - val_loss: 0.1938 - val_accuracy: 0.9114
2632/2632 - 3s - loss: 0.1983 - accuracy: 0.9090 - 3s/epoch - 1ms/step
Test accuracy: 0.91
```

Training Set Performance

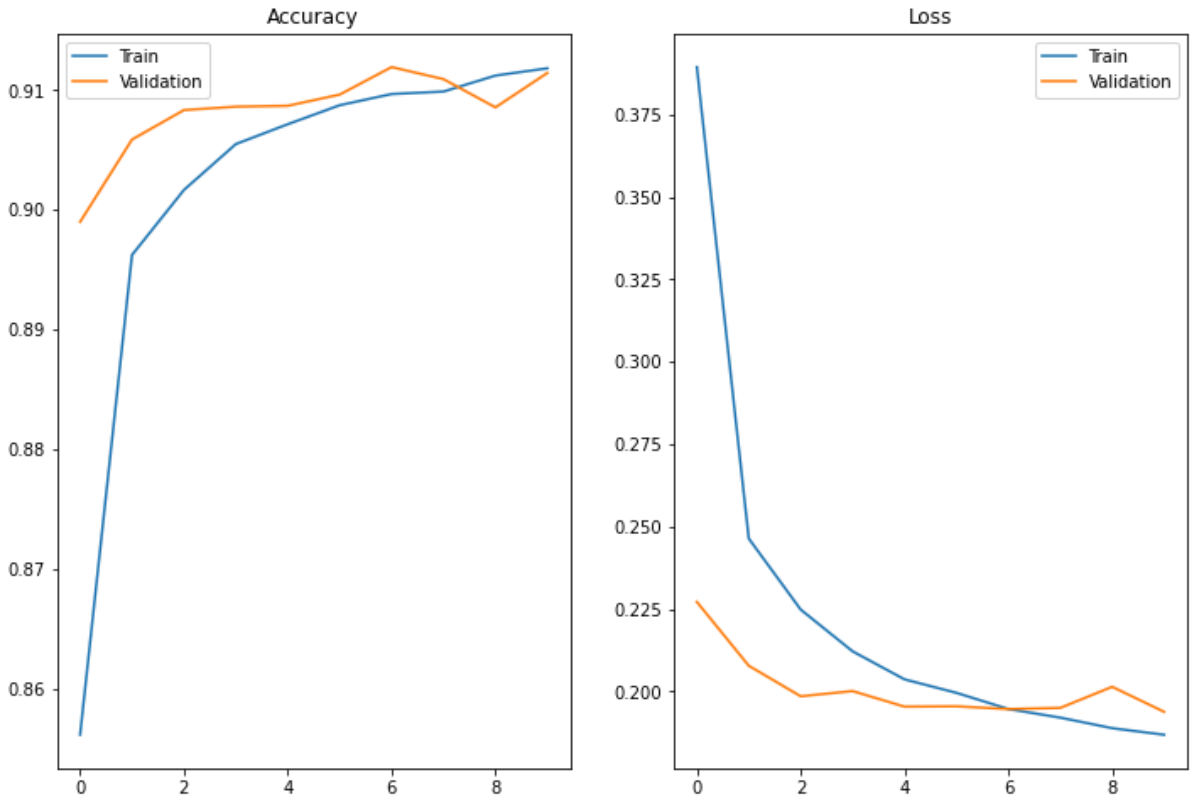
	precision	recall	f1-score	support
A	0.99	1.00	0.99	35922
B	1.00	0.98	0.99	5574
C	0.98	0.99	0.98	8694
D	0.98	0.99	0.98	9864
E	0.99	0.99	0.99	19193
F	0.99	0.98	0.99	7361
G	0.95	0.87	0.91	3487
H	0.98	0.99	0.99	7825
I	0.77	0.65	0.71	8887
J	0.98	0.97	0.98	2739
K	0.99	0.99	0.99	2764
L	0.79	0.88	0.83	13882
M	1.00	0.99	1.00	7705
N	0.98	0.99	0.99	13761
0	0.99	0.99	0.99	19185
P	1.00	0.99	0.99	7070
Q	0.89	0.92	0.90	2974
R	0.50	0.97	0.66	13104
R	0.45	0.02	0.03	13215
S	1.00	1.00	1.00	16649
T	0.99	1.00	0.99	19193
U	0.97	0.99	0.98	10442
...				
accuracy			0.92	269506
macro avg	0.93	0.93	0.92	269506
weighted avg	0.92	0.92	0.91	269506

Validation Set Performance

	precision	recall	f1-score	support
A	0.99	0.99	0.99	8981
B	0.98	0.96	0.97	1394
C	0.96	0.97	0.96	2173
D	0.96	0.97	0.96	2466
E	0.99	0.98	0.98	4798
F	0.98	0.97	0.97	1840
G	0.88	0.78	0.83	872
H	0.96	0.96	0.96	1956
I	0.76	0.65	0.70	2222
J	0.93	0.94	0.94	685
K	0.98	0.96	0.97	691
L	0.79	0.86	0.82	3471
M	0.99	0.99	0.99	1926
N	0.96	0.98	0.97	3440
0	0.98	0.98	0.98	4797
P	0.99	0.98	0.98	1768
Q	0.83	0.84	0.84	743
R	0.49	0.96	0.65	3276
R	0.45	0.02	0.04	3304
S	0.99	0.99	0.99	4162
T	0.98	0.99	0.98	4799
U	0.95	0.98	0.96	2610
...				
accuracy			0.91	67377
macro avg	0.91	0.91	0.90	67377
weighted avg	0.91	0.91	0.89	67377

Plot Learning Curves:

- Accuracy and loss curves for training and validation datasets are plotted to visualize the model's learning over epochs.



Performance Analysis:

- Predictions are made on training and validation sets.
- Classification reports are generated to display precision, recall, F1-score, and support for each class.
- Confusion matrices are plotted to analyze the model's performance on individual classes.

5) CNN with ResNet50

Dataset Preparation:

- Loaded the alphabet dataset with 20,000 images.
- Normalized the images and reshaped them to grayscale.
- Split the data into training, validation, and test sets.

Data Augmentation:

- Applied random transformations like flip, rotation, and zoom to increase dataset diversity.

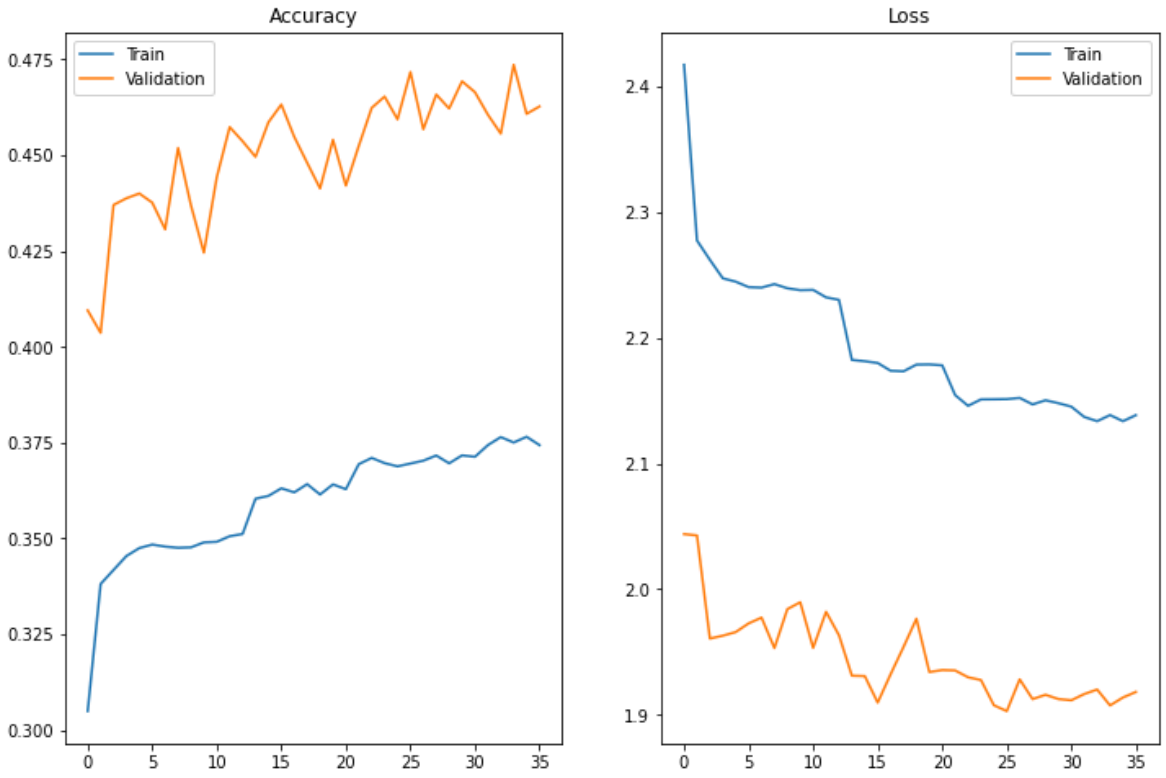
Model Setup:

- Used **ResNet50**, a pre-trained deep learning model that utilizes residual connections to help improve training performance and accuracy.

- Removed ResNet50's top layer and added a custom classifier with softmax output to predict one of the 26 alphabet classes.

Training:

- Trained the model for 50 epochs, using early stopping and learning rate reduction.
- Fine-tuned the model by unfreezing ResNet50's layers and continuing training for 20 more epochs.



Results:

- Achieved 89% training accuracy and 90% validation accuracy.
- Displayed learning curves showing improvement in accuracy and loss over time.

```
Python

Epoch 1/50
5400/5400 [==============================] - 38s 7ms/step - loss: 2.4171 - accuracy: 0.3050 - val_loss: 2.0438 - val_accuracy: 0.4095 - lr: 0.0010
Epoch 2/50
5400/5400 [==============================] - 36s 7ms/step - loss: 2.2773 - accuracy: 0.3381 - val_loss: 2.0427 - val_accuracy: 0.4037 - lr: 0.0010
Epoch 3/50
5400/5400 [==============================] - 40s 7ms/step - loss: 2.2619 - accuracy: 0.3418 - val_loss: 1.9606 - val_accuracy: 0.4371 - lr: 0.0010
Epoch 4/50
5400/5400 [==============================] - 40s 7ms/step - loss: 2.2473 - accuracy: 0.3454 - val_loss: 1.9629 - val_accuracy: 0.4388 - lr: 0.0010
Epoch 5/50
5400/5400 [==============================] - 40s 7ms/step - loss: 2.2446 - accuracy: 0.3475 - val_loss: 1.9658 - val_accuracy: 0.4400 - lr: 0.0010
Epoch 6/50
5400/5400 [==============================] - 40s 7ms/step - loss: 2.2402 - accuracy: 0.3484 - val_loss: 1.9727 - val_accuracy: 0.4377 - lr: 0.0010
Epoch 7/50
5400/5400 [==============================] - 40s 7ms/step - loss: 2.2399 - accuracy: 0.3479 - val_loss: 1.9773 - val_accuracy: 0.4307 - lr: 0.0010
Epoch 8/50
5400/5400 [==============================] - 40s 7ms/step - loss: 2.2426 - accuracy: 0.3476 - val_loss: 1.9531 - val_accuracy: 0.4519 - lr: 0.0010
Epoch 9/50
5400/5400 [==============================] - 36s 7ms/step - loss: 2.2393 - accuracy: 0.3477 - val_loss: 1.9840 - val_accuracy: 0.4369 - lr: 0.0010
Epoch 10/50
5400/5400 [==============================] - 40s 7ms/step - loss: 2.2378 - accuracy: 0.3490 - val_loss: 1.9895 - val_accuracy: 0.4246 - lr: 0.0010
Epoch 11/50
5400/5400 [==============================] - 36s 7ms/step - loss: 2.2380 - accuracy: 0.3491 - val_loss: 1.9532 - val_accuracy: 0.4443 - lr: 0.0010
Epoch 12/50
5400/5400 [==============================] - 36s 7ms/step - loss: 2.2321 - accuracy: 0.3506 - val_loss: 1.9820 - val_accuracy: 0.4574 - lr: 0.0010
Epoch 13/50
...
Epoch 19/20
5400/5400 [==============================] - 99s 18ms/step - loss: 0.2703 - accuracy: 0.8910 - val_loss: 0.2541 - val_accuracy: 0.8981
Epoch 20/20
5400/5400 [==============================] - 99s 18ms/step - loss: 0.2662 - accuracy: 0.8930 - val_loss: 0.2500 - val_accuracy: 0.8978
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Training Set Performance

	precision	recall	f1-score	support
A	0.98	0.99	0.99	32776
B	0.95	0.96	0.96	5538
C	0.98	0.93	0.95	6416
D	0.92	0.97	0.95	6378
E	0.95	0.96	0.95	6402
F	0.96	0.96	0.96	6396
G	0.75	0.85	0.80	3495
H	0.96	0.94	0.95	6363
I	0.84	0.54	0.66	6398
J	0.86	0.94	0.90	2790
K	0.96	0.97	0.96	2789
L	0.65	0.87	0.74	6386
M	0.98	0.98	0.98	6394
N	0.93	0.95	0.94	6391
0	0.98	0.96	0.97	6377
P	0.97	0.96	0.96	6404
Q	0.80	0.76	0.78	3014
R	0.50	0.75	0.60	6406
R	0.50	0.20	0.29	6408
S	0.98	0.96	0.97	6402
T	0.95	0.92	0.94	6422
U	0.93	0.97	0.95	6435
...				
accuracy			0.90	172800
macro avg	0.89	0.89	0.88	172800
weighted avg	0.90	0.90	0.90	172800

Validation Set Performance

	precision	recall	f1-score	support
A	0.98	0.99	0.98	8194
B	0.94	0.96	0.95	1384
C	0.97	0.94	0.96	1604
D	0.92	0.97	0.94	1595
E	0.95	0.96	0.95	1601
F	0.95	0.96	0.96	1599
G	0.74	0.85	0.79	874
H	0.95	0.93	0.94	1591
I	0.84	0.53	0.65	1599
J	0.85	0.94	0.89	697
K	0.95	0.96	0.95	697
L	0.64	0.87	0.74	1597
M	0.98	0.99	0.99	1599
N	0.93	0.95	0.94	1598
0	0.97	0.96	0.96	1594
P	0.98	0.97	0.97	1601
Q	0.79	0.75	0.77	753
R	0.48	0.71	0.57	1602
R	0.46	0.20	0.28	1602
S	0.97	0.95	0.96	1601
T	0.95	0.92	0.93	1605
U	0.92	0.96	0.94	1609
...				
accuracy			0.90	43200
macro avg	0.88	0.88	0.88	43200
weighted avg	0.90	0.90	0.89	43200

 # Using CNN ResNet50 for Alpabet + Numeric

Despite our best efforts to implement the CNN ResNet50 model for recognizing alphanumeric characters, the limitations in computational resources hindered our ability to achieve and present any form of results.

Practical Implications and Business Communication

The findings of this project have significant implications for businesses relying on Optical Character Recognition (OCR) systems to process handwritten data. The improved accuracy achieved by the Convolutional Neural Networks (CNN) and other advanced models highlights their potential to enhance operational efficiency in various applications:

1. **Increased Automation Efficiency**:By accurately recognizing handwritten letters and characters, these models can automate processes like digitizing forms, automating data entry, and processing handwritten documents. This reduces manual effort, leading to faster turnaround times and cost savings
2. **Error Reduction**:Improved accuracy in character recognition directly reduces the error rate in systems processing handwritten input. This is particularly impactful in sensitive domains such as banking (check processing), healthcare (digitizing patient records), and legal documentation, where precision is critical.
3. **Scalability of Operations**:The ability to process large volumes of handwritten data reliably allows businesses to scale their operations without a proportional increase in resources or personnel. For example, mail sorting systems in logistics can efficiently handle diverse handwritten labels with minimal manual intervention.
4. **Improved User Experience**:Reliable character recognition enhances the user experience in applications like mobile check deposits or form submissions by ensuring seamless and error-free interactions.

Communication to Stakeholders

When presenting these findings to stakeholders, it is essential to focus on tangible business benefits rather than technical details. Here's how the results can be framed:

- **Quantifiable Benefits**: Highlight measurable outcomes, such as cost reductions, time savings, and increased throughput due to automation. For instance, "Our system processes handwritten documents 20% faster than previous methods, reducing operational costs by X%."
- **Real-World Scenarios**: Use relatable examples to demonstrate the impact of the model. For example, explain how it could streamline check processing for banks or automate the entry of handwritten survey data, enhancing productivity.
- **Visual Demonstrations**: Provide before-and-after visuals or metrics to illustrate improvements, such as reduced error rates or faster processing times. Visual aids can make the benefits more tangible and relatable.
- **Confidence in Deployment**: Emphasize how thorough model validation ensures reliability and minimizes risks, fostering trust in the system's performance and readiness for real-world applications.

Evaluation:

- Generated classification reports and confusion matrices for both training and validation sets, assessing the model's performance across different classes.