!pip install mlxtend

 Requirement already satisfied: mlxtend in /usr/local/lib/python3.10/dist-packages (0.23.3)

Requirement already satisfied: scipy>=1.2.1 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.13.1) Requirement already satisfied: numpy>=1.16.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.26.4) Requirement already satisfied: pandas>=0.24.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (2.2.2)

Requirement already satisfied: scikit-learn>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.6.0) Requirement already satisfied: matplotlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (3.8.0)

Requirement already satisfied: joblib>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from mlxtend) (1.4.2)

Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (1.3.1 Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (0.12.1)

Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (4.55 Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (1.4. Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (24.2) Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (11.0.0) Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (3.2.0 Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.0->mlxtend) (2 Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->mlxtend) (2024.2)

Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24.2->mlxtend) (2024.2)

Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.3.1->mlxtend)

Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.0.0->m





import numpy as np import pandas as pd

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout from tensorflow.keras.utils import to\_categorical

from sklearn.model\_selection import LeaveOneOut

from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score, recall\_score, f1\_score import matplotlib.pyplot as plt

# Load data

train\_data = pd.read\_csv('/content/emnist-bymerge-train.csv', header=None)

# Preprocess data

X\_train = train\_data.iloc[:, 1:].values.reshape(-1, 28, 28, 1).astype('float32') / 255.0 # Normalize pixel values y\_train = to\_categorical(train\_data.iloc[:, 0].values.astype('int')) # One-hot encode labels

# Limit the dataset size for quick iteration

max\_samples = 200 # Reduce dataset size further if len(X\_train) > max\_samples:

X\_train = X\_train[:max\_samples] y\_train = y\_train[:max\_samples]

# Define CNN model (simplified for faster training) model = Sequential([

Conv2D(16, (3, 3), activation='relu', input\_shape=(28, 28, 1)),

MaxPooling2D((2, 2)), Flatten(),

Dense(32, activation='relu'), # Reduce dense layer size Dropout(0.3), # Lower dropout rate

Dense(y\_train.shape[1], activation='softmax')

])

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy']) # LOOCV Setup

loo = LeaveOneOut()

n\_classes = y\_train.shape[1] y\_true = []

y\_pred = []

# Train and Evaluate using LOOCV

for train\_index, val\_index in loo.split(X\_train):

X\_train\_fold, X\_val\_fold = X\_train[train\_index], X\_train[val\_index] y\_train\_fold, y\_val\_fold = y\_train[train\_index], y\_train[val\_index]

# Train model with fewer epochs and increased batch size

model.fit(X\_train\_fold, y\_train\_fold, epochs=3, batch\_size=64, verbose=0)

# Predict

predictions = model.predict(X\_val\_fold, verbose=0) y\_true.append(np.argmax(y\_val\_fold, axis=1)[0])

y\_pred.append(np.argmax(predictions, axis=1)[0])

# Calculate metrics

loo\_confusion\_matrix = confusion\_matrix(y\_true, y\_pred) loo\_accuracy = accuracy\_score(y\_true, y\_pred)

loo\_precision = precision\_score(y\_true, y\_pred, average='weighted') loo\_recall = recall\_score(y\_true, y\_pred, average='weighted')

loo\_f1 = f1\_score(y\_true, y\_pred, average='weighted')

print("\nLOOCV Results:") print("Confusion Matrix:") print(loo\_confusion\_matrix)

print("Accuracy:", loo\_accuracy) print("Precision:", loo\_precision) print("Recall:", loo\_recall)

print("F1 Score:", loo\_f1)

# Plot confusion matrix

def plot\_confusion\_mat(cm):

plt.figure(figsize=(8, 8))

plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues) plt.title('Confusion Matrix')

plt.colorbar()

tick\_marks = np.arange(cm.shape[0])

plt.xticks(tick\_marks, tick\_marks, rotation=45) plt.yticks(tick\_marks, tick\_marks)

# Add labels in each cell fmt = 'd'

thresh = cm.max() / 2.

for i in range(cm.shape[0]):

for j in range(cm.shape[1]):

plt.text(j, i, format(cm[i, j], fmt), horizontalalignment="center",

color="white" if cm[i, j] > thresh else "black")

plt.ylabel('True Label')

plt.xlabel('Predicted Label') plt.tight\_layout()

plt.show()

plot\_confusion\_mat(loo\_confusion\_matrix)

 /usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined a

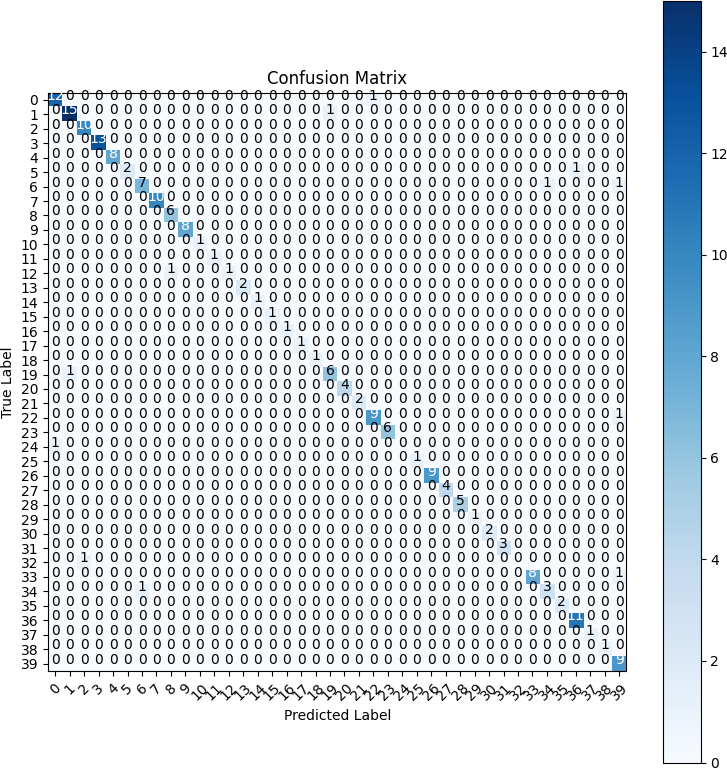
\_warn\_prf(average, modifier, f"{metric.capitalize()} is", len(result))

LOOCV Results:

Confusion Matrix:

|  |  |  |  |
| --- | --- | --- | --- |
| [[12 0 0 ... | 0 | 0 | 0] |
| [ 0 15 0 ... | 0 | 0 | 0] |
| [ 0 0 10 ... | 0 | 0 | 0] |
| ...  [ 0 0 0 ... | 1 | 0 | 0] |
| [ 0 0 0 ... | 0 | 1 | 0] |
| [ 0 0 0 ... | 0 | 0 | 9]] |

Accuracy: 0.94



Traceback (most recent call last)

Precision: 0.9347104978354978

Recall: 0.94

F1 Score: 0.9345699109075067