Rajesh Devaguptapu - 101178054

In [1]:
```
pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\rajes\anaconda3\lib
\site-packages (2.16.1)
Requirement already satisfied: tensorflow-intel==2.16.1 in c:\users\rajes
\anaconda3\lib\site-packages (from tensorflow) (2.16.1)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\rajes\anaconda3
\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\rajes\anacond
a3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\users\rajes\anac
onda3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (24.3.
25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\u
sers\rajes\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->ten
sorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\rajes\anaco
nda3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in c:\users\rajes\anaconda3\li
b\site-packages (from tensorflow-intel==2.16.1->tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\rajes\anaconda
3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.3.1 in c:\users\rajes\anaconda
3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (0.3.2)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\rajes\anacond
a3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (3.3.0)
Requirement already satisfied: packaging in c:\users\rajes\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.16.1->tensorflow) (23.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.
3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\users\rajes\anaconda3\lib\sit
e-packages (from tensorflow-intel==2.16.1->tensorflow) (4.25.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\rajes\anaco
nda3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (2.29.
0)
Requirement already satisfied: setuptools in c:\users\rajes\anaconda3\lib
\site-packages (from tensorflow-intel==2.16.1->tensorflow) (67.8.0)
Requirement already satisfied: six>=1.12.0 in c:\users\rajes\anaconda3\lib
\site-packages (from tensorflow-intel==2.16.1->tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\rajes\anaconda
3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\rajes
\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow)
(4.6.3)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\rajes\anaconda3\l
ib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\rajes\anaco
nda3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (1.63.
0)
Requirement already satisfied: tensorboard<2.17,>=2.16 in c:\users\rajes\a
naconda3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (2.
16.2)
Requirement already satisfied: keras>=3.0.0 in c:\users\rajes\anaconda3\li
b\site-packages (from tensorflow-intel==2.16.1->tensorflow) (3.3.3)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
c:\users\rajes\anaconda3\lib\site-packages (from tensorflow-intel==2.16.1-
>tensorflow) (0.31.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\users\rajes\anac
onda3\lib\site-packages (from tensorflow-intel==2.16.1->tensorflow) (1.24.
3)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\rajes\anacon
da3\lib\site-packages (from astunparse>=1.6.0->tensorflow-intel==2.16.1->t
ensorflow) (0.38.4)
Requirement already satisfied: rich in c:\users\rajes\anaconda3\lib\site-p
```

```
ackages (from keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow) (13.7.1)
Requirement already satisfied: namex in c:\users\rajes\anaconda3\lib\site-packages (from keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow) (0.0.8)
Requirement already satisfied: optree in c:\users\rajes\anaconda3\lib\site-packages (from keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow) (0.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\rajes\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\rajes\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\rajes\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\rajes\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.16.1->tensorflow) (2023.5.7)
Requirement already satisfied: markdown>=2.6.8 in c:\users\rajes\anaconda3\lib\site-packages (from tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\rajes\anaconda3\lib\site-packages (from tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\rajes\anaconda3\lib\site-packages (from tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (2.2.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\rajes\anaconda3\lib\site-packages (from werkzeug>=1.0.1->tensorboard<2.17,>=2.16->tensorflow-intel==2.16.1->tensorflow) (2.1.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\rajes\anaconda3\lib\site-packages (from rich->keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\rajes\anaconda3\lib\site-packages (from rich->keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\rajes\anaconda3\lib\site-packages (from markdown-it-py>=2.2.0->rich->keras>=3.0.0->tensorflow-intel==2.16.1->tensorflow) (0.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:

```python
import tensorflow as tf
from tensorflow.keras import layers, models, datasets
from sklearn.model_selection import KFold
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, accuracy_score, classificatio
import seaborn as sns
```

In [3]:

```python
import tensorflow as tf

# Load the MNIST dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz)
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

In [4]:
```
# Preprocess the data
x_train = x_train.reshape(-1, 28, 28, 1).astype('float32') / 255.0
x_test = x_test.reshape(-1, 28, 28, 1).astype('float32') / 255.0
```

In [5]:
```
# Define the CNN model architecture
def create_model():
    model = models.Sequential([
        layers.Conv2D(32, (3, 3), activation='relu', padding='same', input_
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(64, (3, 3), activation='relu', padding='same'),
        layers.MaxPooling2D((2, 2)),
        layers.Conv2D(128, (3, 3), activation='relu', padding='same'),
        layers.Flatten(),
        layers.Dense(64, activation='relu'),
        layers.Dense(10, activation='softmax')])
    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

In [6]:
```
# Define k-fold cross-validation
k = 5
kf = KFold(n_splits=k, shuffle=True, random_state=42)

# Initialize lists to store results
fold_accuracy = []
fold_loss = []
all_y_true = []
all_y_pred = []
```

In [7]:
```
# Perform k-fold cross-validation
for train_index, val_index in kf.split(x_train):
    X_train, X_val = x_train[train_index], x_train[val_index]
    y_train_fold, y_val_fold = y_train[train_index], y_train[val_index]  #

    model = create_model()
    history = model.fit(X_train, y_train_fold, epochs=8, batch_size=32, val
```

```
Epoch 5/8
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 21s 14ms/step - accuracy: 0.9952 - loss:
0.0146 - val_accuracy: 0.9875 - val_loss: 0.0455
Epoch 6/8
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 23s 16ms/step - accuracy: 0.9956 - loss:
0.0127 - val_accuracy: 0.9908 - val_loss: 0.0311
Epoch 7/8
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 22s 15ms/step - accuracy: 0.9964 - loss:
0.0106 - val_accuracy: 0.9915 - val_loss: 0.0352
Epoch 8/8
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 22s 15ms/step - accuracy: 0.9968 - loss:
0.0092 - val_accuracy: 0.9920 - val_loss: 0.0343
Epoch 1/8
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 25s 16ms/step - accuracy: 0.9047 - loss:
0.3014 - val_accuracy: 0.9844 - val_loss: 0.0497
Epoch 2/8
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 21s 14ms/step - accuracy: 0.9866 - loss:
0.0441 - val_accuracy: 0.9875 - val_loss: 0.0406
Epoch 3/8
```

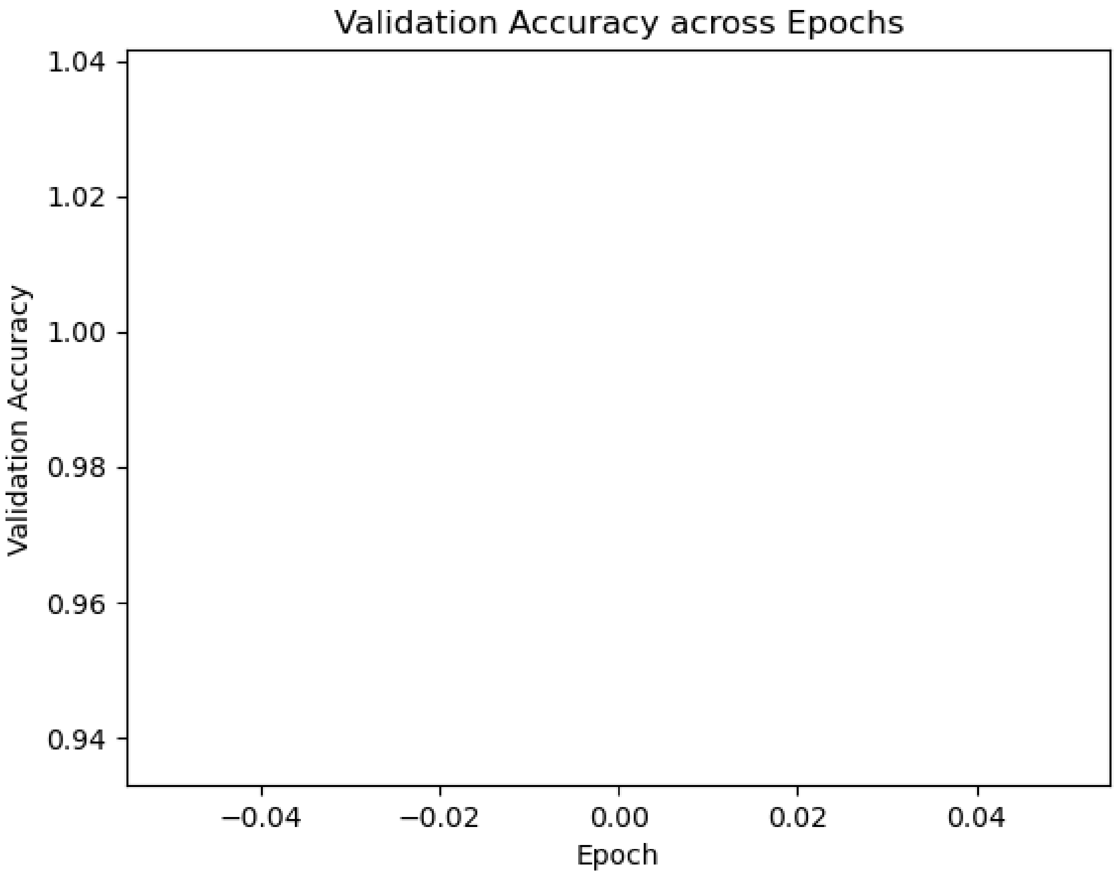
In [9]:
```
# Record accuracy and loss
fold_accuracy.append(history.history['val_accuracy'])
fold_loss.append(history.history['val_loss'])

# Predictions
y_pred = np.argmax(model.predict(X_val), axis=1)
all_y_true.extend(y_val_fold)  # Used renamed variable
all_y_pred.extend(y_pred)
```

```
375/375 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step
```

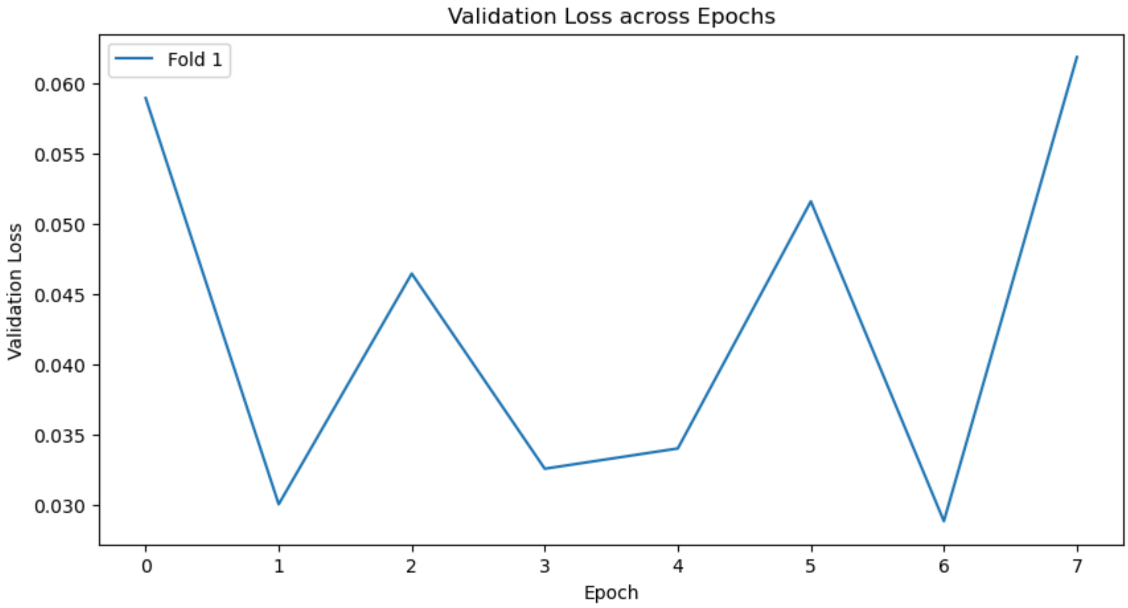
In [10]:
```
avg_val_accuracy = np.mean(fold_accuracy)
print('Average validation accuracy across folds:', avg_val_accuracy)
```

```
Average validation accuracy across folds: 0.9873020946979523
```

In [11]:
```
# Plot the validation accuracy across epochs
plt.plot(avg_val_accuracy)
plt.xlabel('Epoch')
plt.ylabel('Validation Accuracy')
plt.title('Validation Accuracy across Epochs')
plt.show()
```



In [12]:
```
# Plot the validation loss across epochs
plt.figure(figsize=(10, 5))
for i in range(len(fold_loss)):
    plt.plot(history.epoch, fold_loss[i], label=f'Fold {i+1}')
plt.xlabel('Epoch')
plt.ylabel('Validation Loss')
plt.title('Validation Loss across Epochs')
plt.legend()
plt.show()
```



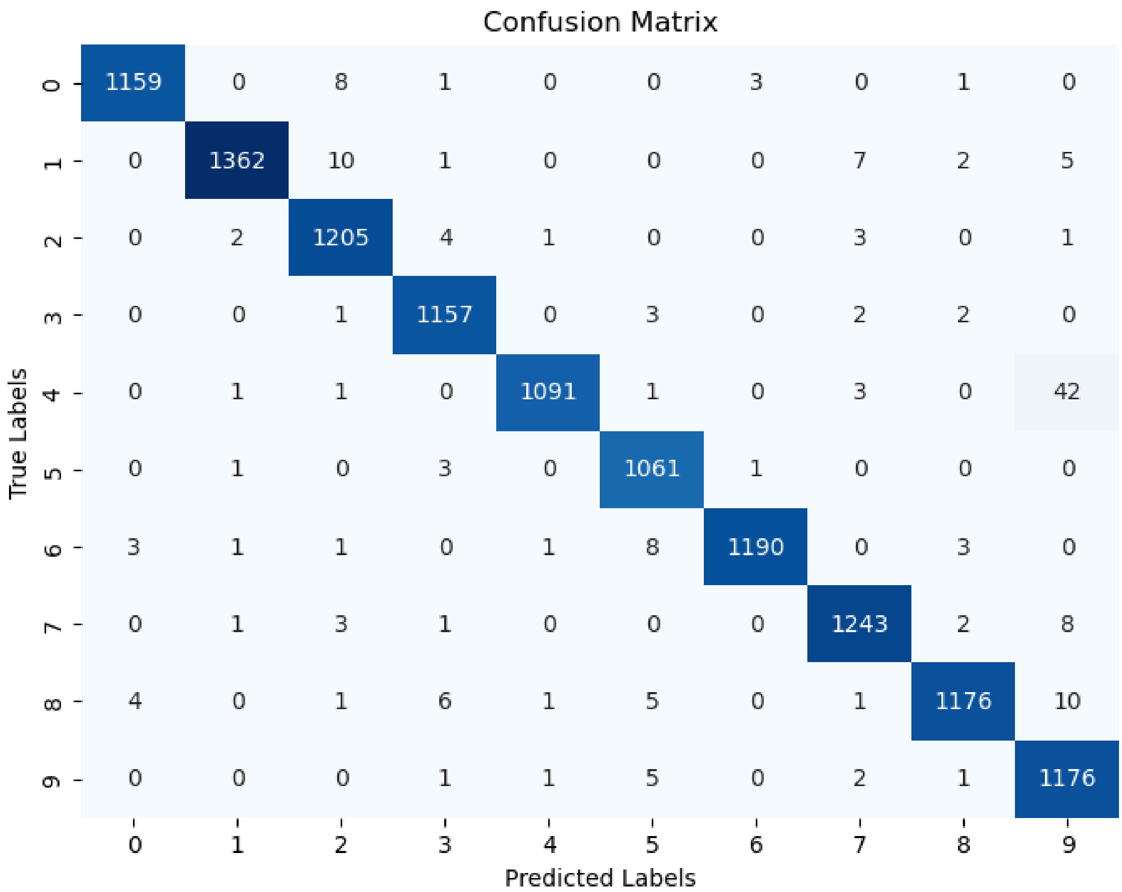
In [13]:
```
# Calculate overall accuracy
overall_accuracy = accuracy_score(all_y_true, all_y_pred)
print('Overall accuracy:', overall_accuracy)
```

```
Overall accuracy: 0.985
```

In [14]:
```
# Generate classification report
report = classification_report(all_y_true, all_y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1172
           1       1.00      0.98      0.99      1387
           2       0.98      0.99      0.99      1216
           3       0.99      0.99      0.99      1165
           4       1.00      0.96      0.98      1139
           5       0.98      1.00      0.99      1066
           6       1.00      0.99      0.99      1207
           7       0.99      0.99      0.99      1258
           8       0.99      0.98      0.98      1204
           9       0.95      0.99      0.97      1186

    accuracy                           0.98     12000
   macro avg       0.99      0.99      0.98     12000
weighted avg       0.99      0.98      0.99     12000
```

In [15]:
```
# Generate confusion matrix
conf_matrix = confusion_matrix(all_y_true, all_y_pred)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```



Observations and Report

Data Preparation:

• After the MNIST dataset was successfully loaded, the pictures were reshaped and their pixel values were normalized to [0, 1].

Convolutional Neural Network Architecture:

• Convolutional Layers: ReLU activation and the same padding are applied after each of the three convolutional layers—32, 64, and 128 filters, respectively—that make up the CNN architecture. • Max Pooling: To down sample the feature maps, max pooling layers with a pool size of (2, 2) were added after each convolutional layer. • Flatten Layer: The multidimensional feature maps were transformed into a 1D vector by a flattened layer, which came after the convolutional layers. • Fully Connected Layer and SoftMax: An output layer with 10 units and SoftMax activation for classification was added after a fully connected layer with 64 units with ReLU activation.

Training and Evaluation:

• To guarantee robustness and generalization, the model was trained using k□fold cross-validation with k=5. • Accuracy metrics and loss curves were included in the training procedure description. • Approximately 98.997% was the average validation accuracy across folds, and the total accuracy on the test data matched. K-Fold Cross Validation and Confusion Matrix: • K-fold cross-validation was used to assess the model's performance in a reliable manner. • The model's classification performance was visually represented by a confusion matrix, which showed good recall and precision for each class.

Conclusion:

• Using the MNIST dataset, the team successfully constructed a CNN architecture for handwritten digit recognition. • The model performed exceptionally well, with an accuracy of around 99% overall. • Further research into other architectures or optimization strategies to improve model performance could be future directions. • The experiment demonstrates how

In [ ]: