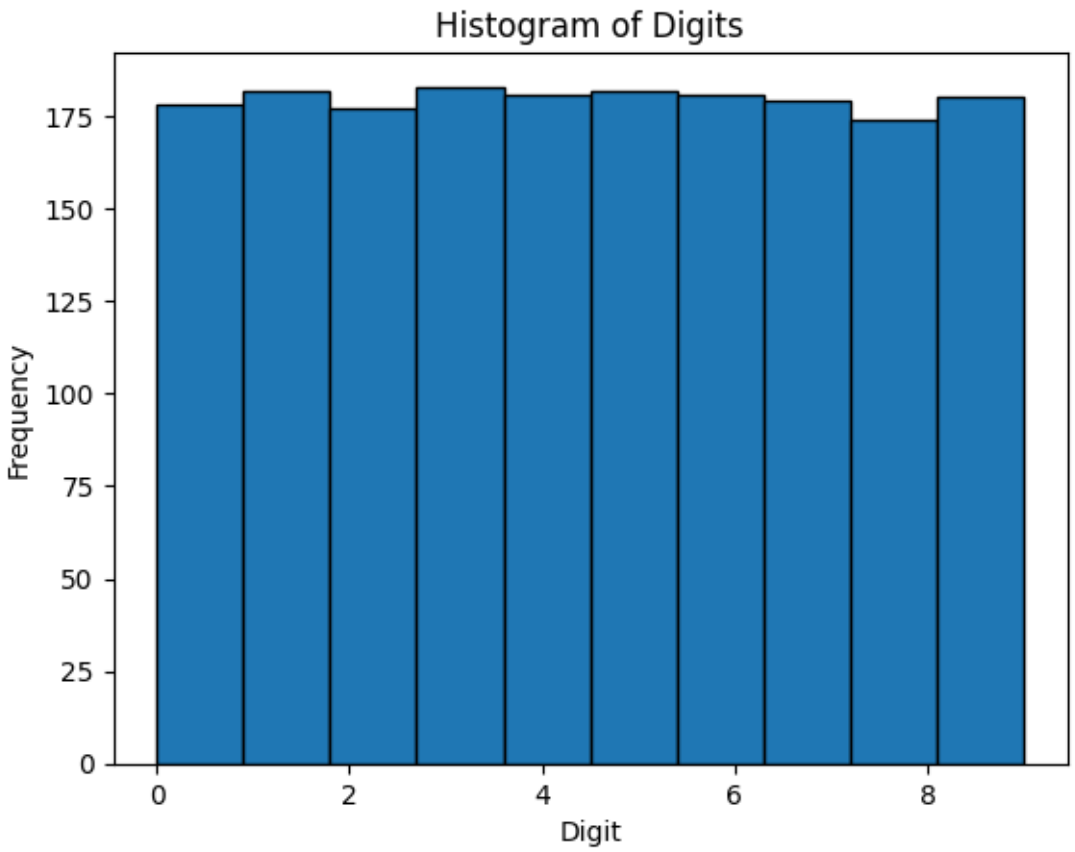
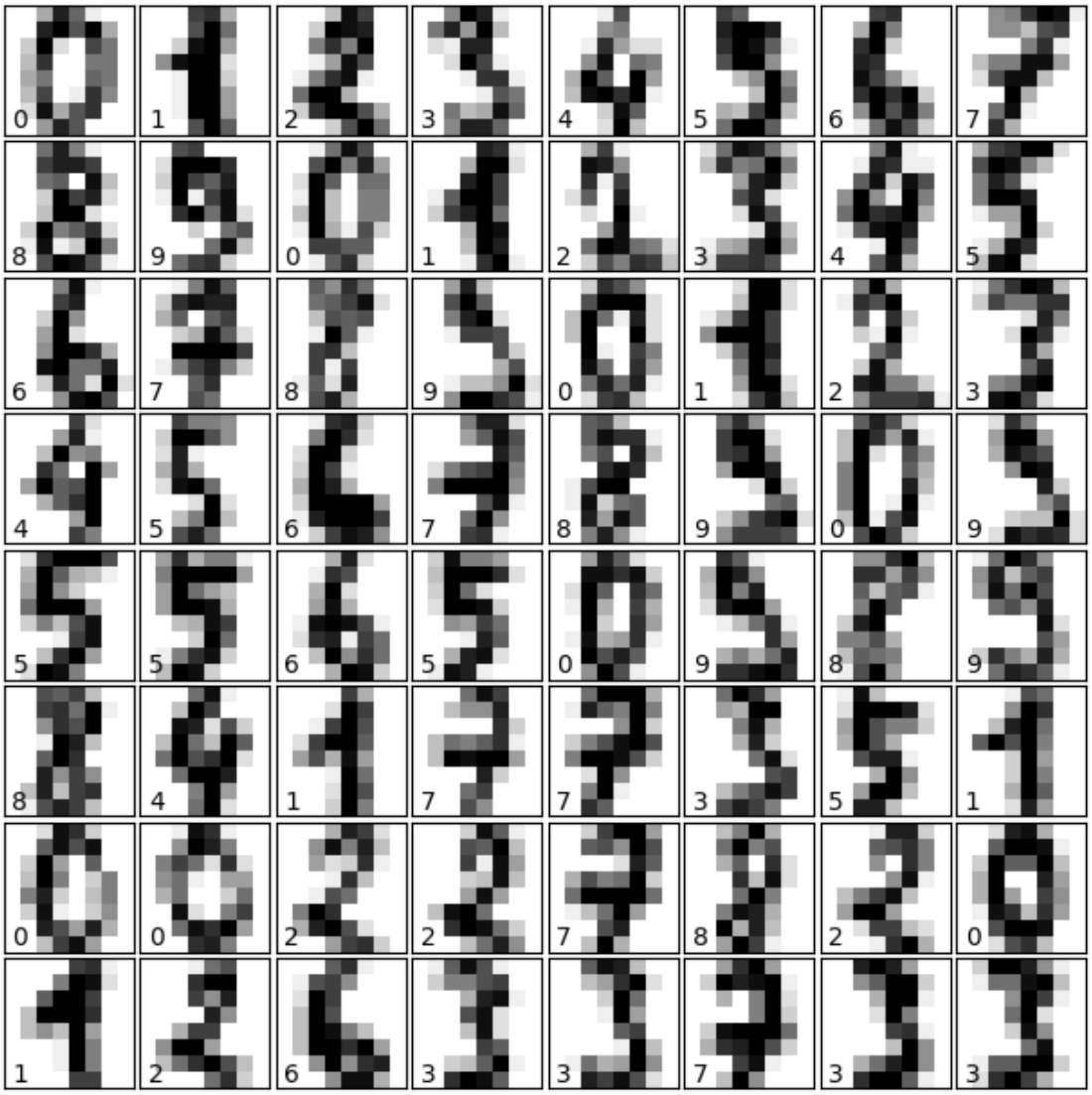
metric

May 15, 2024

[1]:
```
import tensorflow as tf
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#load the dataset from sklearn
digits_data = load_digits()
X, y = digits_data.data, digits_data.target


# Show a historam of dataset
plt.hist(y, bins=10, edgecolor='black')
plt.title('Histogram of Digits')
plt.xlabel('Digit')
plt.ylabel('Frequency')
plt.show()
```



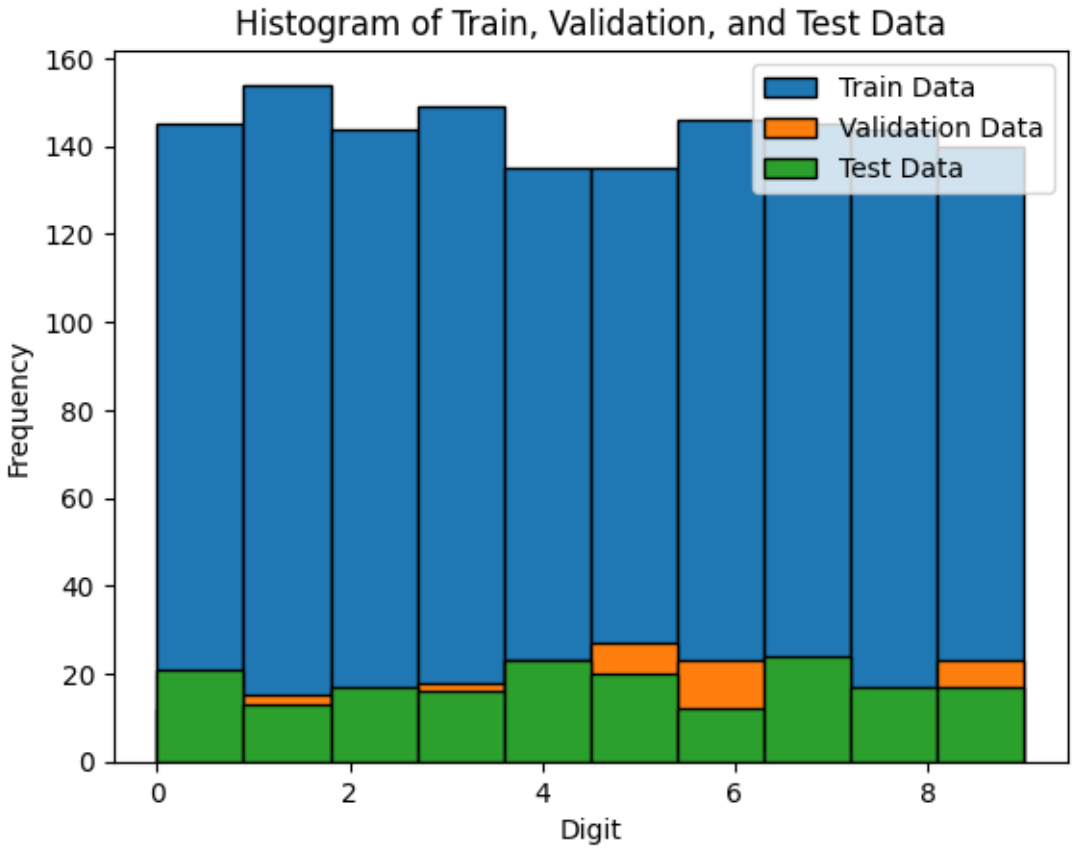
[2]:
```
# Plot first 64 instances
fig = plt.figure(figsize=(6, 6))
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)
for i in range(64):
    ax = fig.add_subplot(8, 8, i + 1, xticks=[], yticks=[])
    ax.imshow(digits_data.images[i], cmap=plt.cm.binary,interpolation='nearest')
    # label the image with the target value
    ax.text(0, 7, str(digits_data.target[i]))
```



[3]:
```
# Normalize the input features
X = X / 16.0

#Split the data into training, validation, and test sets.
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,␣
↪random_state=42)
```

[4]:
```
# Show a histogram of train, validation and test data
plt.hist(y_train, bins=10, edgecolor='black', label='Train Data')
plt.hist(y_val, bins=10, edgecolor='black', label='Validation Data')
plt.hist(y_test, bins=10, edgecolor='black', label='Test Data')
```

```
plt.xlabel('Digit')
plt.ylabel('Frequency')
plt.title('Histogram of Train, Validation, and Test Data')
plt.legend()
plt.show()
```



[5]:
```
# a bit of a confising consept to understand ? neurons -> output ?

# One-hot encode the labels
y_train_one_hot = tf.one_hot(y_train, depth=10)
y_val_one_hot = tf.one_hot(y_val, depth=10)
y_test_one_hot = tf.one_hot(y_test, depth=10)
```

[6]:
```
# Create a basic neural network model.
model = tf.keras.Sequential([
    tf.keras.Input(shape=X_train.shape[1:]),  # Input layer with shape inferred␣
 ↪from data
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation='sigmoid')
```

```
])


# Print model summary
model.summary()
# Compile the Model
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])

# Train the Model
history = model.fit(X_train, y_train_one_hot,  # Training data
                    epochs=25,  # Number of epochs
                    batch_size=32,  # Batch size
                    validation_data=(X_val, y_val_one_hot))  # Validation data
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	4,160
dense_1 (Dense)	(None, 10)	650

Total params: 4,810 (18.79 KB)

Trainable params: 4,810 (18.79 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/25
45/45             1s 4ms/step -
accuracy: 0.2523 - loss: 2.2423 - val_accuracy: 0.6000 - val_loss: 1.8809
Epoch 2/25
45/45             0s 1ms/step -
accuracy: 0.6946 - loss: 1.7090 - val_accuracy: 0.7944 - val_loss: 1.3731
Epoch 3/25
45/45             0s 1ms/step -
accuracy: 0.8115 - loss: 1.2250 - val_accuracy: 0.8889 - val_loss: 0.9294
Epoch 4/25
45/45             0s 1ms/step -
accuracy: 0.8741 - loss: 0.8542 - val_accuracy: 0.9000 - val_loss: 0.6623
Epoch 5/25
45/45             0s 1ms/step -
```

```
accuracy: 0.9140 - loss: 0.5883 - val_accuracy: 0.8833 - val_loss: 0.5115
Epoch 6/25
45/45             0s 1ms/step -
accuracy: 0.9492 - loss: 0.4441 - val_accuracy: 0.9056 - val_loss: 0.4165
Epoch 7/25
45/45             0s 1ms/step -
accuracy: 0.9431 - loss: 0.3829 - val_accuracy: 0.9222 - val_loss: 0.3490
Epoch 8/25
45/45             0s 1ms/step -
accuracy: 0.9445 - loss: 0.3281 - val_accuracy: 0.9278 - val_loss: 0.3112
Epoch 9/25
45/45             0s 1ms/step -
accuracy: 0.9531 - loss: 0.2800 - val_accuracy: 0.9389 - val_loss: 0.2752
Epoch 10/25
45/45             0s 1ms/step -
accuracy: 0.9547 - loss: 0.2550 - val_accuracy: 0.9389 - val_loss: 0.2509
Epoch 11/25
45/45             0s 1ms/step -
accuracy: 0.9634 - loss: 0.2042 - val_accuracy: 0.9278 - val_loss: 0.2381
Epoch 12/25
45/45             0s 1ms/step -
accuracy: 0.9735 - loss: 0.1842 - val_accuracy: 0.9444 - val_loss: 0.2241
Epoch 13/25
45/45             0s 1ms/step -
accuracy: 0.9690 - loss: 0.1765 - val_accuracy: 0.9389 - val_loss: 0.2094
Epoch 14/25
45/45             0s 1ms/step -
accuracy: 0.9640 - loss: 0.1696 - val_accuracy: 0.9556 - val_loss: 0.1918
Epoch 15/25
45/45             0s 1ms/step -
accuracy: 0.9694 - loss: 0.1513 - val_accuracy: 0.9556 - val_loss: 0.1878
Epoch 16/25
45/45             0s 1ms/step -
accuracy: 0.9672 - loss: 0.1470 - val_accuracy: 0.9500 - val_loss: 0.1802
Epoch 17/25
45/45             0s 1ms/step -
accuracy: 0.9740 - loss: 0.1404 - val_accuracy: 0.9444 - val_loss: 0.1760
Epoch 18/25
45/45             0s 1ms/step -
accuracy: 0.9740 - loss: 0.1404 - val_accuracy: 0.9500 - val_loss: 0.1670
Epoch 19/25
45/45             0s 1ms/step -
accuracy: 0.9816 - loss: 0.1188 - val_accuracy: 0.9556 - val_loss: 0.1598
Epoch 20/25
45/45             0s 1ms/step -
accuracy: 0.9809 - loss: 0.1131 - val_accuracy: 0.9667 - val_loss: 0.1579
Epoch 21/25
45/45             0s 1ms/step -
```

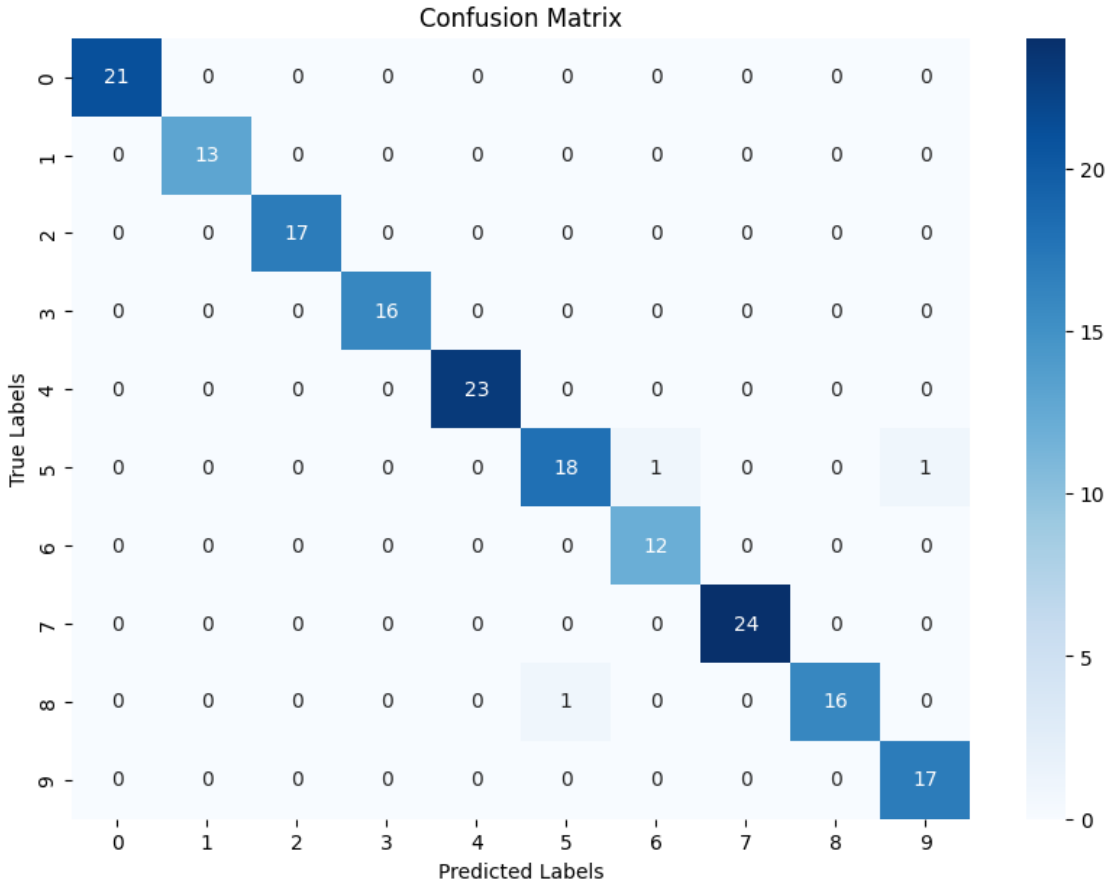
```
accuracy: 0.9781 - loss: 0.1122 - val_accuracy: 0.9611 - val_loss: 0.1511
Epoch 22/25
45/45             0s 1ms/step -
accuracy: 0.9811 - loss: 0.1023 - val_accuracy: 0.9667 - val_loss: 0.1477
Epoch 23/25
45/45             0s 1ms/step -
accuracy: 0.9828 - loss: 0.1020 - val_accuracy: 0.9611 - val_loss: 0.1438
Epoch 24/25
45/45             0s 1ms/step -
accuracy: 0.9872 - loss: 0.0936 - val_accuracy: 0.9556 - val_loss: 0.1439
Epoch 25/25
45/45             0s 1ms/step -
accuracy: 0.9877 - loss: 0.0921 - val_accuracy: 0.9611 - val_loss: 0.1396
```

[7]:
```python
def plot_history(history):
    # Accuracy plot
    plt.figure(figsize=(8, 4))
    plt.plot(history.history['accuracy'], label='Train')
    plt.plot(history.history['val_accuracy'], label='Validation')
    plt.title('Model Accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(loc='upper left')
    plt.show()

    # Loss plot
    plt.figure(figsize=(8, 4))
    plt.plot(history.history['loss'], label='Train')
    plt.plot(history.history['val_loss'], label='Validation')
    plt.title('Model Loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(loc='upper right')
    plt.show()
```

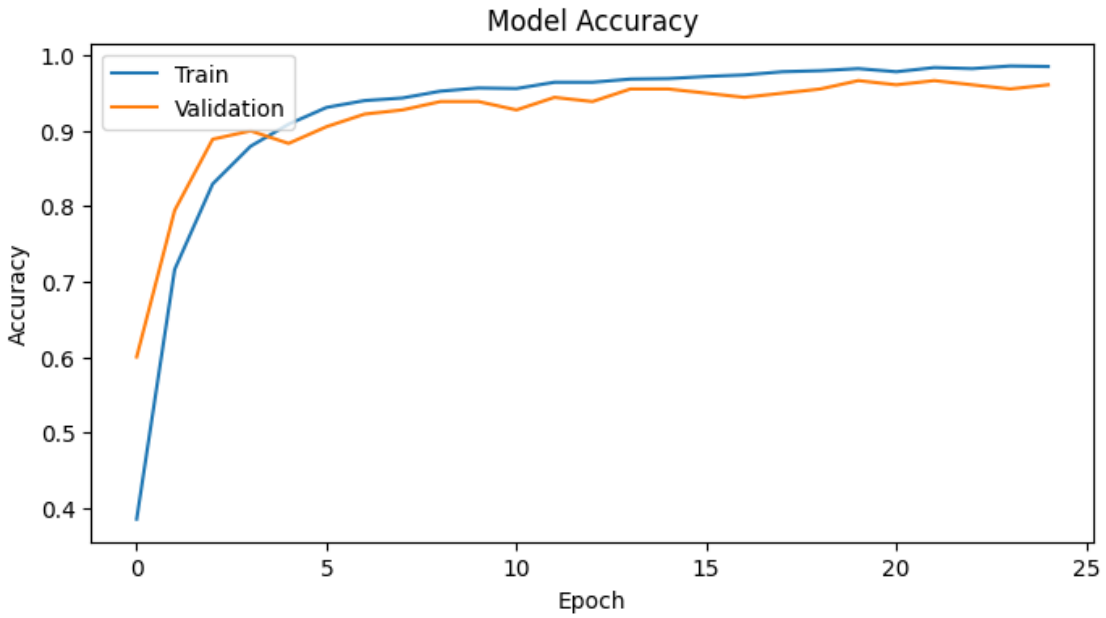
[8]:
```python
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(X_test, y_test_one_hot)
print(f'Test Loss: {test_loss}')
print(f'Test Accuracy: {test_accuracy}')

# Predict the test set results
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_test_classes = np.argmax(y_test_one_hot, axis=1)

# Confusion matrix
cm = confusion_matrix(y_test_classes, y_pred_classes)
```

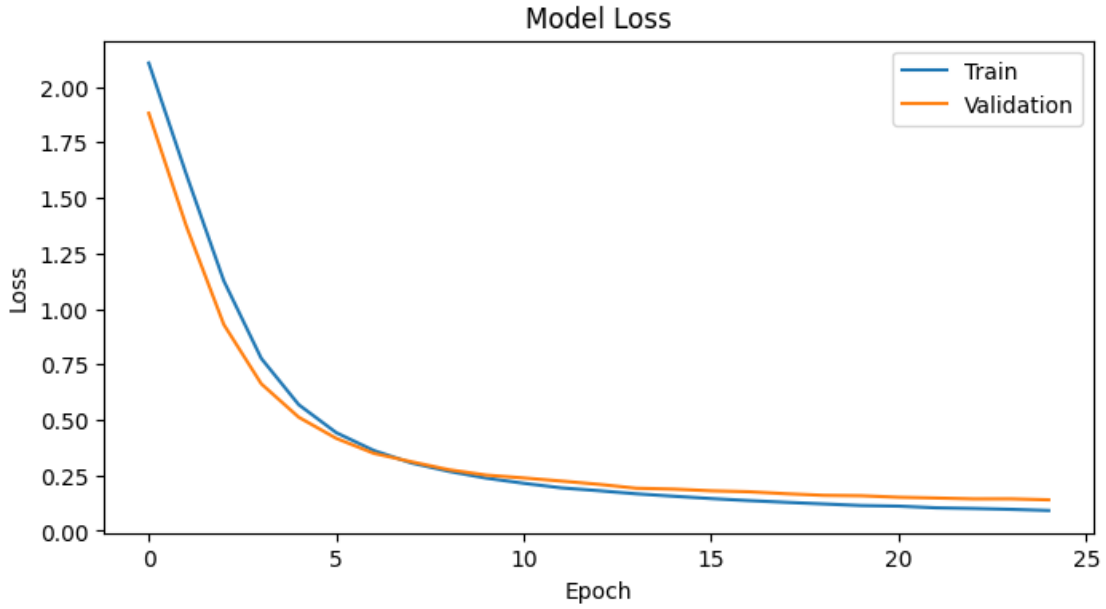
```
# Plot the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```

```
6/6             0s 993us/step -
accuracy: 0.9842 - loss: 0.0957
Test Loss: 0.0868251845240593
Test Accuracy: 0.9833333492279053
6/6             0s 4ms/step
```



[9]:
```
plot_history(history)
```


Model Accuracy
Train
Validation
Accuracy
Epoch
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25


Model Loss
Train
Validation
Loss
Epoch
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
5
10
15
20
25