```
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt

# Load the MNIST dataset
mnist = fetch_openml('mnist_784', version=1)

# Features and labels
X, y = mnist['data'], mnist['target']

# Convert labels to integers
y = y.astype(int)
```

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Initialize the MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(64, 64), max_iter=20, alpha=1e-4,
                    solver='sgd', verbose=10, random_state=1,
                    learning_rate_init=.1)

# Train the model
mlp.fit(X_train, y_train)
```

```
Iteration 1, loss = 0.28395107
Iteration 2, loss = 0.15718102
Iteration 3, loss = 0.13286050
Iteration 4, loss = 0.10744293
Iteration 5, loss = 0.09562792
Iteration 6, loss = 0.08628949
Iteration 7, loss = 0.08982760
Iteration 8, loss = 0.07116525
Iteration 9, loss = 0.06793491
Iteration 10, loss = 0.07117720
Iteration 11, loss = 0.07209220
Iteration 12, loss = 0.06278195
Iteration 13, loss = 0.05612610
Iteration 14, loss = 0.06040064
Iteration 15, loss = 0.06829053
Iteration 16, loss = 0.06216053
Iteration 17, loss = 0.09009062
Iteration 18, loss = 0.16276313
Iteration 19, loss = 0.09797132
Iteration 20, loss = 0.08373181
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:690: ConvergenceWarning: Stochastic Optimizer:
  warnings.warn(
▾                                MLPClassifier
MLPClassifier(hidden_layer_sizes=(64, 64), learning_rate_init=0.1, max_iter=20,
              random_state=1, solver='sgd', verbose=10)
```

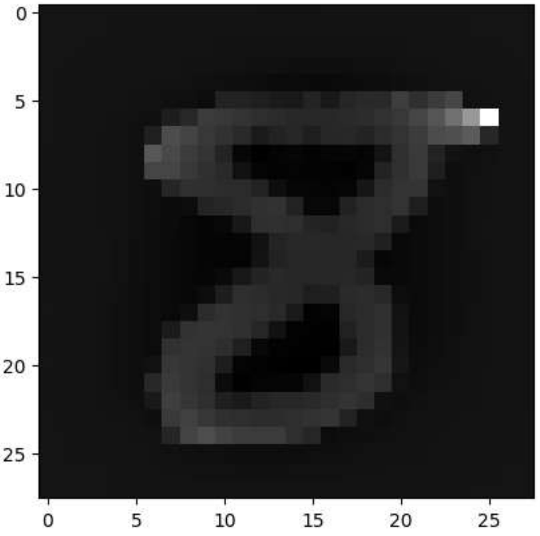
```
# Predict the labels for the test set
y_pred = mlp.predict(X_test)

# Display classification report
print("Classification Report:\n", classification_report(y_test, y_pred))

# Display confusion matrix
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Classification Report:
               precision    recall  f1-score   support
```

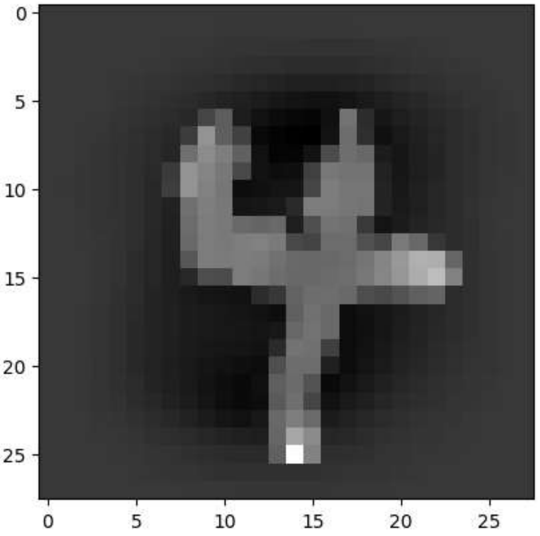
```
           0       0.97      0.98      0.97      1343
           1       0.97      0.98      0.98      1600
           2       0.89      0.96      0.92      1380
           3       0.98      0.94      0.96      1433
           4       0.96      0.97      0.96      1295
           5       0.95      0.94      0.94      1273
           6       0.95      0.96      0.95      1396
           7       0.96      0.96      0.96      1503
           8       0.98      0.90      0.94      1357
           9       0.94      0.94      0.94      1420

    accuracy                           0.95     14000
   macro avg       0.95      0.95      0.95     14000
weighted avg       0.95      0.95      0.95     14000

Confusion Matrix:
 [[1314    2    9    0    0    1   11    3    2    1]
 [   1 1570    8    4    3    3    3    6    2    0]
 [   3    3 1325    4    7    5   10   11    8    4]
 [   0    1   32 1342    1   29    2   13    2   11]
 [   2    3   12    0 1250    1    6    3    2   16]
 [   3    5   11    5    3 1199   35    1    4    7]
 [  16    1   24    0    7    3 1345    0    0    0]
 [   4    2   17    3    6    4    0 1448    1   18]
 [  10   18   30    9    3   15   10    9 1223   30]
 [   5    8   18    8   22    6    1   18    5 1329]]
```

```
# Function to display an image from the MNIST dataset
def plot_sample(X, y, index):
    plt.imshow(X[index].reshape(28, 28), cmap='gray')
    plt.title(f"True Label: {y[index]}")
    plt.show()

# Display a few test samples with predicted labels
for i in range(5):
    plot_sample(X_test, y_pred, i)
```

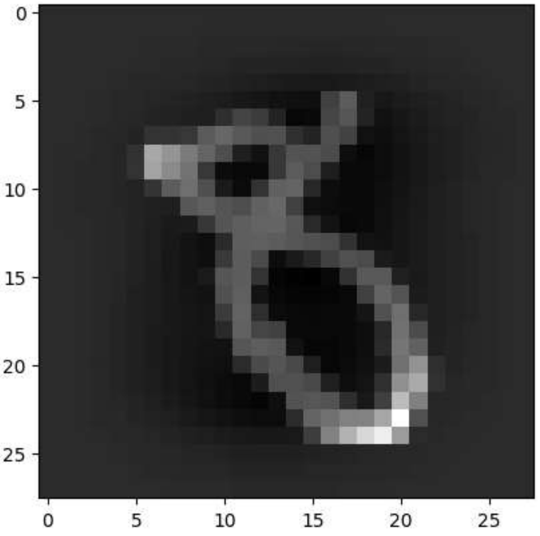
True Label: 8



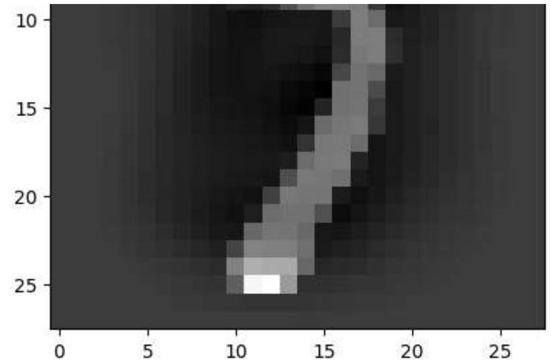
True Label: 4

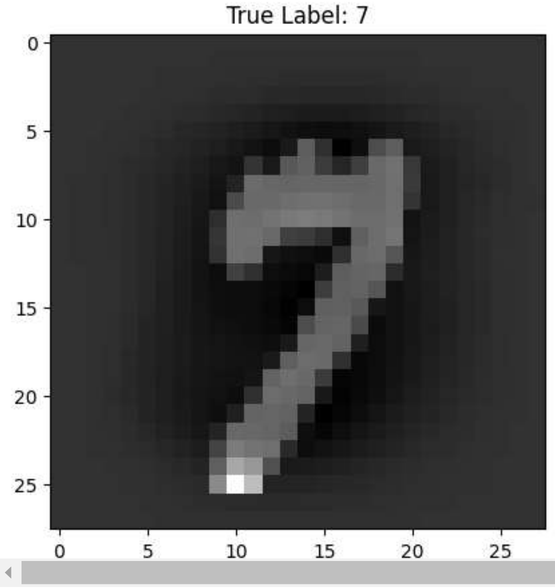


True Label: 8



True Label: 7





```
import matplotlib.pyplot as plt
# Plot the loss curve during training
plt.plot(mlp.loss_curve_)
plt.title("Loss Curve during Training")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.show()
```