fashion_MNIST

May 2, 2025

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import accuracy_score
```

[3]:
```
#Step 1: Load the data
train_df = pd.read_csv("fashion-mnist_train.csv")
test_df = pd.read_csv("fashion-mnist_test.csv")

X_train = train_df.iloc[:, 1:].values / 255.0  # Normalize pixel values
y_train = train_df.iloc[:, 0].values.reshape(-1, 1)

X_test = test_df.iloc[:, 1:].values / 255.0
y_test = test_df.iloc[:, 0].values.reshape(-1, 1)
```

[5]:
```
# Step 2: One-hot encode the labels
encoder = OneHotEncoder(sparse_output=False, categories='auto')
y_train_oh = encoder.fit_transform(y_train)
y_test_oh = encoder.transform(y_test)
```

[7]:
```
# Step 3: Initialize weights for a simple neural network
input_size = 784
hidden_size = 128
output_size = 10

W1 = np.random.randn(input_size, hidden_size) * 0.01
b1 = np.zeros((1, hidden_size))
W2 = np.random.randn(hidden_size, output_size) * 0.01
b2 = np.zeros((1, output_size))

def relu(Z):
    return np.maximum(0, Z)

def relu_derivative(Z):
    return (Z > 0).astype(float)
```

```
def softmax(Z):
    expZ = np.exp(Z - np.max(Z, axis=1, keepdims=True))
    return expZ / np.sum(expZ, axis=1, keepdims=True)

def compute_loss(y_true, y_pred):
    epsilon = 1e-10
    return -np.mean(np.sum(y_true * np.log(y_pred + epsilon), axis=1))
```

[9]:
```
# Step 4: Train the model
epochs = 100
learning_rate = 0.1
losses = []

for epoch in range(epochs):
    # Forward pass
    Z1 = np.dot(X_train, W1) + b1
    A1 = relu(Z1)
    Z2 = np.dot(A1, W2) + b2
    A2 = softmax(Z2)

    # Loss
    loss = compute_loss(y_train_oh, A2)
    losses.append(loss)

    # Backward pass
    dZ2 = A2 - y_train_oh
    dW2 = np.dot(A1.T, dZ2) / X_train.shape[0]
    db2 = np.sum(dZ2, axis=0, keepdims=True) / X_train.shape[0]

    dA1 = np.dot(dZ2, W2.T)
    dZ1 = dA1 * relu_derivative(Z1)
    dW1 = np.dot(X_train.T, dZ1) / X_train.shape[0]
    db1 = np.sum(dZ1, axis=0, keepdims=True) / X_train.shape[0]

    # Update weights
    W1 -= learning_rate * dW1
    b1 -= learning_rate * db1
    W2 -= learning_rate * dW2
    b2 -= learning_rate * db2

    print(f"Epoch {epoch+1}/{epochs} - Loss: {loss:.4f}")
```

```
Epoch 1/100 - Loss: 2.3018
Epoch 2/100 - Loss: 2.2986
Epoch 3/100 - Loss: 2.2954
Epoch 4/100 - Loss: 2.2919
Epoch 5/100 - Loss: 2.2882
```

```
Epoch 6/100 - Loss: 2.2842
Epoch 7/100 - Loss: 2.2796
Epoch 8/100 - Loss: 2.2745
Epoch 9/100 - Loss: 2.2687
Epoch 10/100 - Loss: 2.2620
Epoch 11/100 - Loss: 2.2545
Epoch 12/100 - Loss: 2.2459
Epoch 13/100 - Loss: 2.2362
Epoch 14/100 - Loss: 2.2253
Epoch 15/100 - Loss: 2.2130
Epoch 16/100 - Loss: 2.1993
Epoch 17/100 - Loss: 2.1841
Epoch 18/100 - Loss: 2.1674
Epoch 19/100 - Loss: 2.1491
Epoch 20/100 - Loss: 2.1292
Epoch 21/100 - Loss: 2.1076
Epoch 22/100 - Loss: 2.0843
Epoch 23/100 - Loss: 2.0593
Epoch 24/100 - Loss: 2.0325
Epoch 25/100 - Loss: 2.0041
Epoch 26/100 - Loss: 1.9742
Epoch 27/100 - Loss: 1.9429
Epoch 28/100 - Loss: 1.9106
Epoch 29/100 - Loss: 1.8775
Epoch 30/100 - Loss: 1.8440
Epoch 31/100 - Loss: 1.8105
Epoch 32/100 - Loss: 1.7773
Epoch 33/100 - Loss: 1.7447
Epoch 34/100 - Loss: 1.7128
Epoch 35/100 - Loss: 1.6819
Epoch 36/100 - Loss: 1.6519
Epoch 37/100 - Loss: 1.6230
Epoch 38/100 - Loss: 1.5952
Epoch 39/100 - Loss: 1.5684
Epoch 40/100 - Loss: 1.5427
Epoch 41/100 - Loss: 1.5179
Epoch 42/100 - Loss: 1.4942
Epoch 43/100 - Loss: 1.4713
Epoch 44/100 - Loss: 1.4494
Epoch 45/100 - Loss: 1.4282
Epoch 46/100 - Loss: 1.4079
Epoch 47/100 - Loss: 1.3883
Epoch 48/100 - Loss: 1.3694
Epoch 49/100 - Loss: 1.3512
Epoch 50/100 - Loss: 1.3337
Epoch 51/100 - Loss: 1.3167
Epoch 52/100 - Loss: 1.3003
Epoch 53/100 - Loss: 1.2844
```

```
Epoch 54/100 - Loss: 1.2691
Epoch 55/100 - Loss: 1.2542
Epoch 56/100 - Loss: 1.2398
Epoch 57/100 - Loss: 1.2259
Epoch 58/100 - Loss: 1.2124
Epoch 59/100 - Loss: 1.1994
Epoch 60/100 - Loss: 1.1867
Epoch 61/100 - Loss: 1.1745
Epoch 62/100 - Loss: 1.1626
Epoch 63/100 - Loss: 1.1511
Epoch 64/100 - Loss: 1.1399
Epoch 65/100 - Loss: 1.1292
Epoch 66/100 - Loss: 1.1187
Epoch 67/100 - Loss: 1.1086
Epoch 68/100 - Loss: 1.0987
Epoch 69/100 - Loss: 1.0892
Epoch 70/100 - Loss: 1.0800
Epoch 71/100 - Loss: 1.0710
Epoch 72/100 - Loss: 1.0624
Epoch 73/100 - Loss: 1.0540
Epoch 74/100 - Loss: 1.0458
Epoch 75/100 - Loss: 1.0379
Epoch 76/100 - Loss: 1.0302
Epoch 77/100 - Loss: 1.0227
Epoch 78/100 - Loss: 1.0155
Epoch 79/100 - Loss: 1.0084
Epoch 80/100 - Loss: 1.0016
Epoch 81/100 - Loss: 0.9949
Epoch 82/100 - Loss: 0.9885
Epoch 83/100 - Loss: 0.9822
Epoch 84/100 - Loss: 0.9761
Epoch 85/100 - Loss: 0.9701
Epoch 86/100 - Loss: 0.9643
Epoch 87/100 - Loss: 0.9587
Epoch 88/100 - Loss: 0.9532
Epoch 89/100 - Loss: 0.9479
Epoch 90/100 - Loss: 0.9426
Epoch 91/100 - Loss: 0.9376
Epoch 92/100 - Loss: 0.9326
Epoch 93/100 - Loss: 0.9278
Epoch 94/100 - Loss: 0.9231
Epoch 95/100 - Loss: 0.9185
Epoch 96/100 - Loss: 0.9140
Epoch 97/100 - Loss: 0.9097
Epoch 98/100 - Loss: 0.9054
Epoch 99/100 - Loss: 0.9012
Epoch 100/100 - Loss: 0.8972
```

[11]:
```
# Step 5: Evaluate on test data
Z1_test = np.dot(X_test, W1) + b1
A1_test = relu(Z1_test)
Z2_test = np.dot(A1_test, W2) + b2
A2_test = softmax(Z2_test)

y_pred_test = np.argmax(A2_test, axis=1)
y_true_test = y_test.flatten()

acc = accuracy_score(y_true_test, y_pred_test)
print(f"\nTest Accuracy: {acc:.4f}")
```

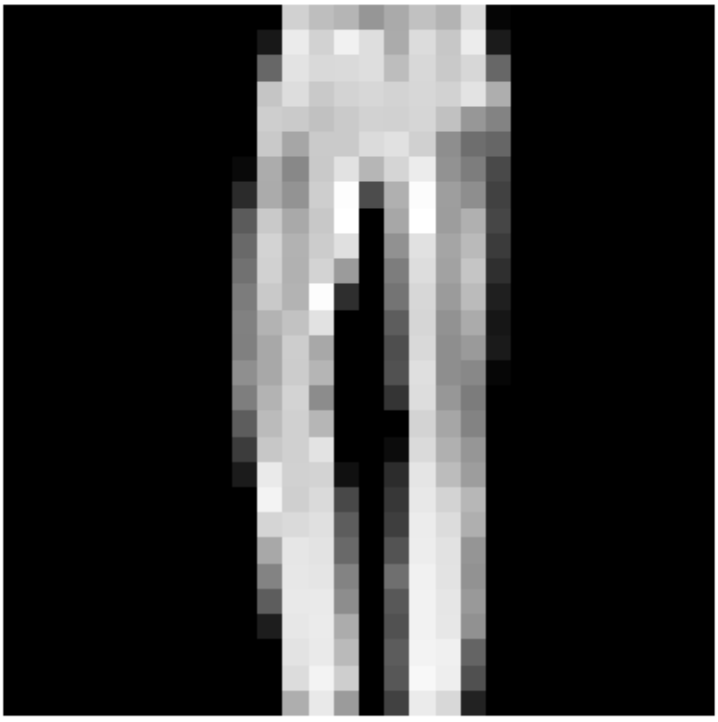
```
Test Accuracy: 0.6646
```

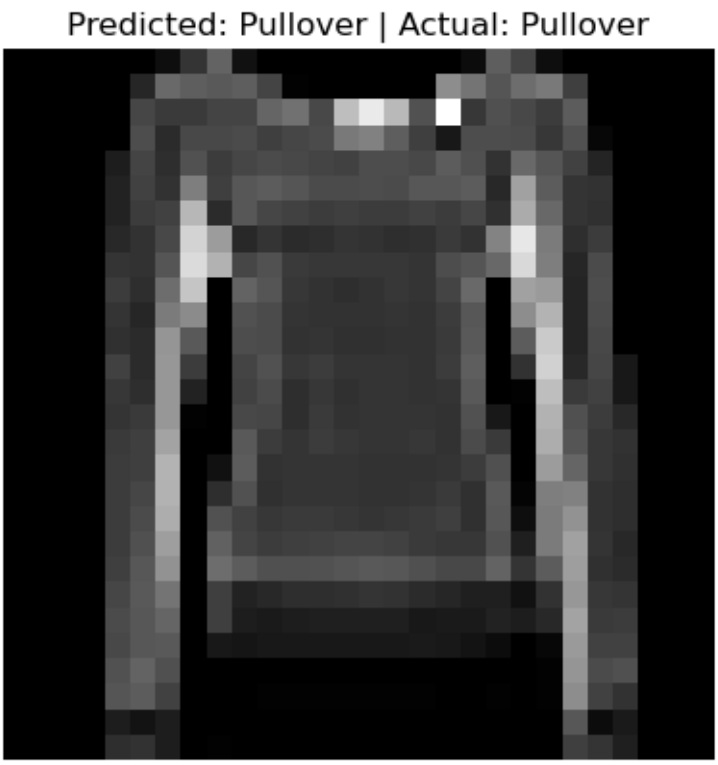
[13]:
```
# Step 6: Visualize sample predictions
label_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

for i in range(5):
    img = X_test[i].reshape(28, 28)
    plt.imshow(img, cmap='gray')
    plt.title(f"Predicted: {label_names[y_pred_test[i]]} | Actual:␣
 ↪{label_names[y_true_test[i]]}")
    plt.axis('off')
    plt.show()
```

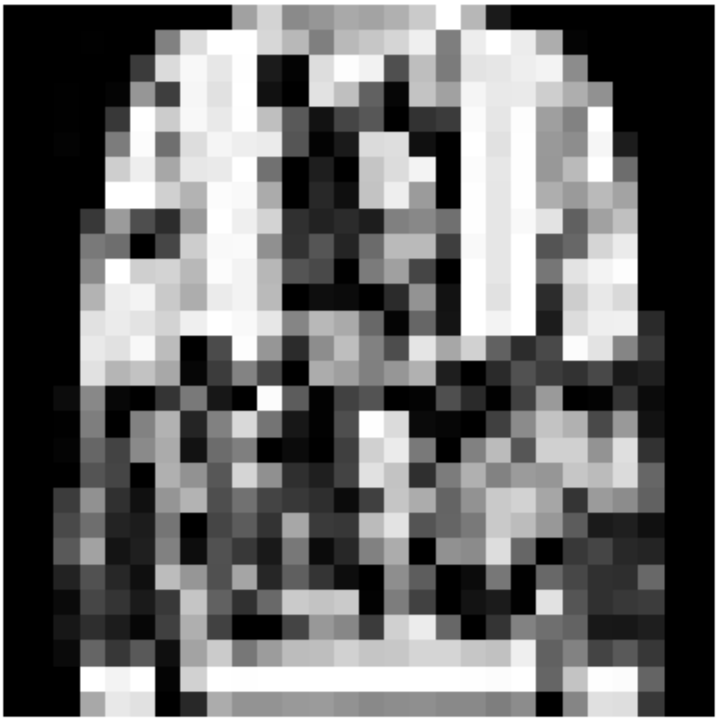
Predicted: T-shirt/top | Actual: T-shirt/top

Predicted: Trouser | Actual: Trouser

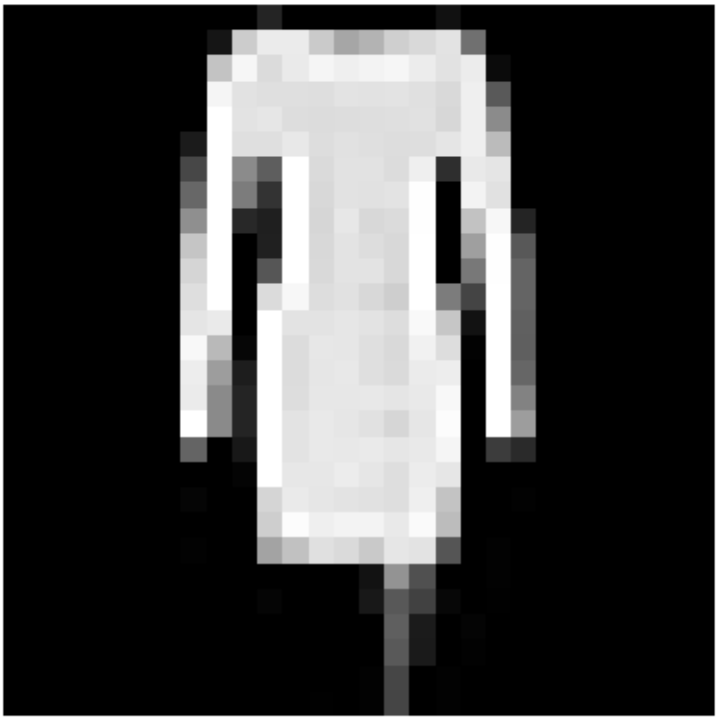



Predicted: Pullover | Actual: Pullover

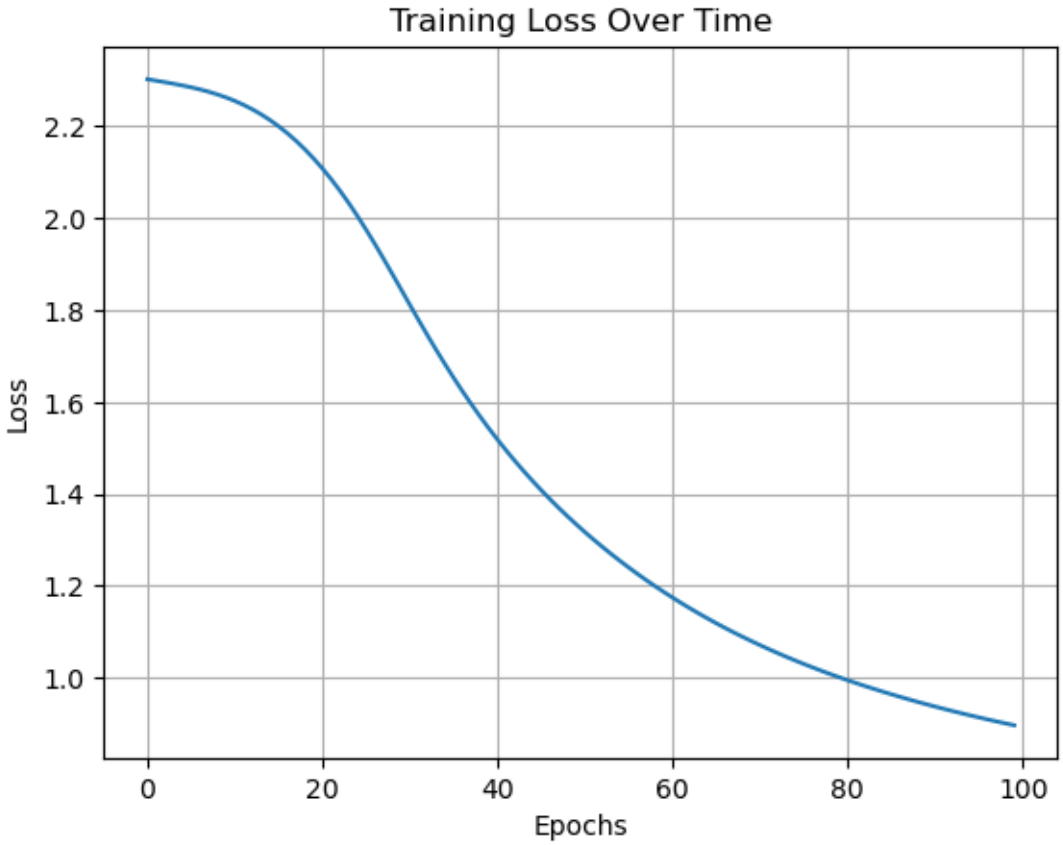
Predicted: T-shirt/top | Actual: Pullover



Predicted: Trouser | Actual: Dress



[15]:
```
# Step 7: Plot loss over epochs
plt.plot(losses)
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Training Loss Over Time")
plt.grid(True)
plt.show()
```



[ ]: