Assignment 8

Name: Vaishnavi Gosavi **Roll No:** 71
Class: TY CSAI-A **Batch:** 2

Title: Use MNIST Fashion Dataset and create a classifier to classify fashion clothing into categories. Using CNN

Description:-

- **Convolutional Neural Network (CNN)**
  A Convolutional Neural Network (CNN) is a class of deep neural networks most commonly used in analyzing visual imagery. CNNs are inspired by the visual cortex of the human brain, which processes visual data hierarchically.

- **Architecture Components:**

1. **Input Layer**
   - Accepts images of shape (28x28x1) for grayscale MNIST data.

2. **Convolutional Layer**
   - Applies filters (kernels) to extract low-level features like edges, textures, and patterns.
   - Operation: Convolution between input and filter.
   - Output is called a feature map.

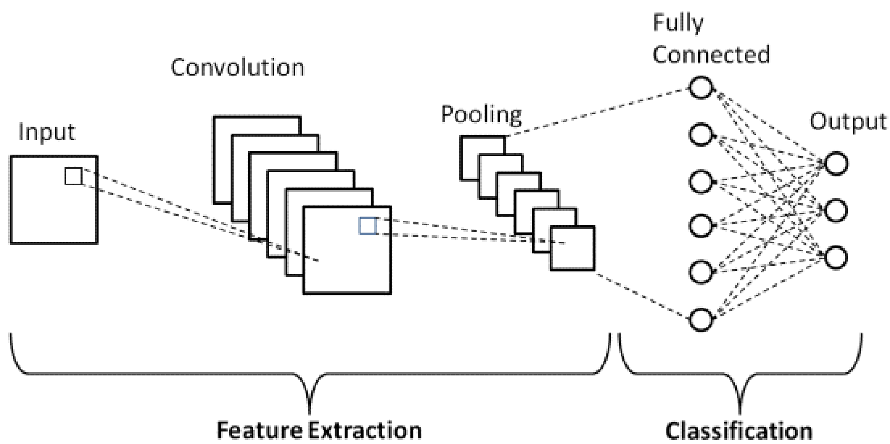
3. **Activation Layer (ReLU)**
   - Applies the ReLU (Rectified Linear Unit) function: $f(x) = \max(0, x)$
   - Introduces non-linearity and prevents vanishing gradients.

4. **Pooling Layer (Max Pooling)**
   - Reduces spatial dimensions (width and height) of feature maps.
   - Helps in making the model more efficient and invariant to minor translations.
   - E.g., 2x2 MaxPooling reduces 28x28 → 14x14.

5. **Flatten Layer**
   - Converts the 2D feature maps into a 1D vector to feed into the Dense (fully connected) layer.

6. **Dense Layer**
   - Fully connected layer where each neuron is connected to all outputs from the previous layer.
   - Learns high-level global patterns.
   - Plot actual vs. predicted stock prices.

7. **Output Layer**
   - Uses Softmax activation function to output probabilities for each of the 10 classes.
   - Softmax ensures the outputs sum to 1 and represents confidence for each class.



- **Working Mechanism of CNN on MNIST**
  1. Input image (28x28) is fed to the CNN.
  2. First Conv layer extracts edge-level features (horizontal, vertical lines).
  3. Pooling layer reduces dimensions, keeps the most important features.
  4. Second Conv + Pooling layers extract more abstract features like loops, curves.
  5. Flattening makes the data suitable for the Dense layer.
  6. Dense layers learn complex relationships and finally classify the digit using Softmax.

- **Fashion MNIST Dataset**
  - Fashion MNIST is a dataset of 28x28 grayscale images of clothing items used for multiclass image classification. It contains 70,000 images in total: 60,000 for training and 10,000 for testing, across 10 categories like T-shirt, Dress, Sneaker, and Bag.
    - Classes: 10 (e.g., T-shirt, Trouser, Coat, Sneaker)
    - Image Size: 28x28 pixels, grayscale
    - Labels: 0 to 9 (one for each class)
    - Used for: Training CNNs and benchmarking classification models

- **Algorithm**

START

1. Import necessary libraries:
   - TensorFlow, Keras, NumPy, Matplotlib (if needed)
2. Load Fashion MNIST dataset:
   - (x_train, y_train), (x_test, y_test) = load_fashion_mnist()
3. Preprocess data:
   a. Normalize pixel values:
      - x_train = x_train / 255
      - x_test = x_test / 255
   b. Reshape input data to (28, 28, 1) for CNN:
      - x_train = reshape(x_train, (-1, 28, 28, 1))
      - x_test = reshape(x_test, (-1, 28, 28, 1))
   c. Convert labels to one-hot encoding:
      - y_train = to_categorical(y_train, 10)
      - y_test = to_categorical(y_test, 10)
4. Define CNN model:
   a. Initialize Sequential model
   b. Add layers:
      - Conv2D(32 filters, 3x3 kernel, activation='relu', input_shape=(28, 28, 1))
      - MaxPooling2D(pool_size=(2, 2))
      - Conv2D(64 filters, 3x3 kernel, activation='relu')
      - MaxPooling2D(pool_size=(2, 2))
      - Flatten()
      - Dense(10 units, activation='softmax')
5. Compile the model:
   - Optimizer = 'adam'
   - Loss function = 'categorical_crossentropy'
   - Metrics = ['accuracy']
6. Train the model:
   - model.fit(x_train, y_train, epochs=5, batch_size=32)
7. Evaluate the model on test data:
   - test_loss, test_accuracy = model.evaluate(x_test, y_test)
8. Display evaluation results:
   - Print test accuracy

END

```python
import tensorflow as tf

(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.fashion_mnist.load_data()
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)

model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(28, 28, 1)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5, batch_size=32)
loss, accuracy = model.evaluate(x_test, y_test)
print('Test accuracy:', accuracy)
```

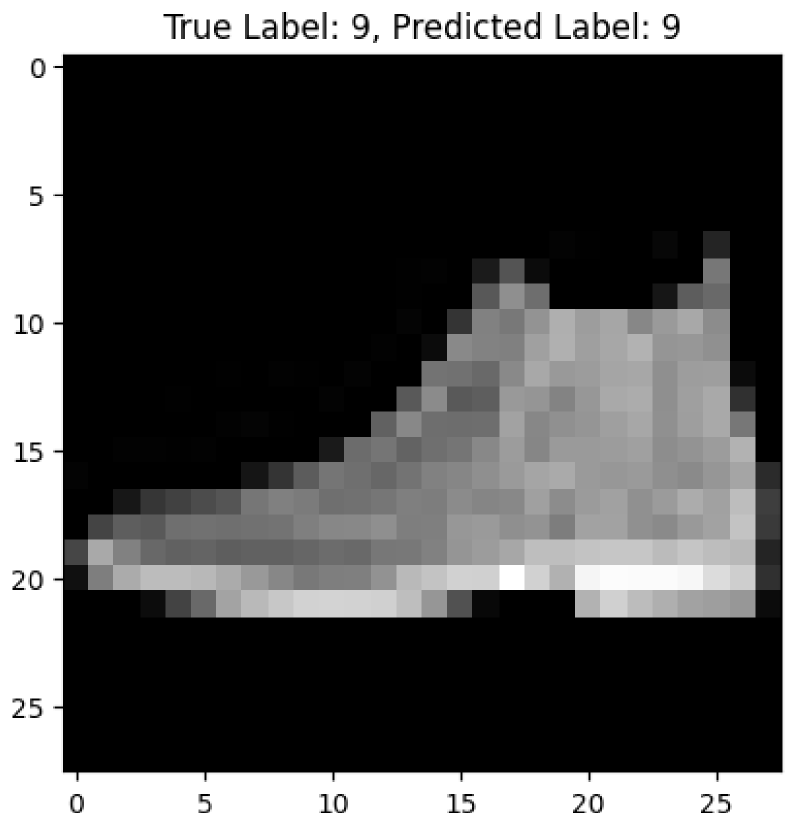
```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/
convolutional/base_conv.py:107: UserWarning:

Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.


Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 55s 29ms/step - accuracy: 0.7637 -
loss: 0.6641
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 83s 29ms/step - accuracy: 0.8774 -
loss: 0.3466
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 81s 29ms/step - accuracy: 0.8980 -
loss: 0.2858
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 81s 29ms/step - accuracy: 0.9044 -
loss: 0.2605
Epoch 5/5
```

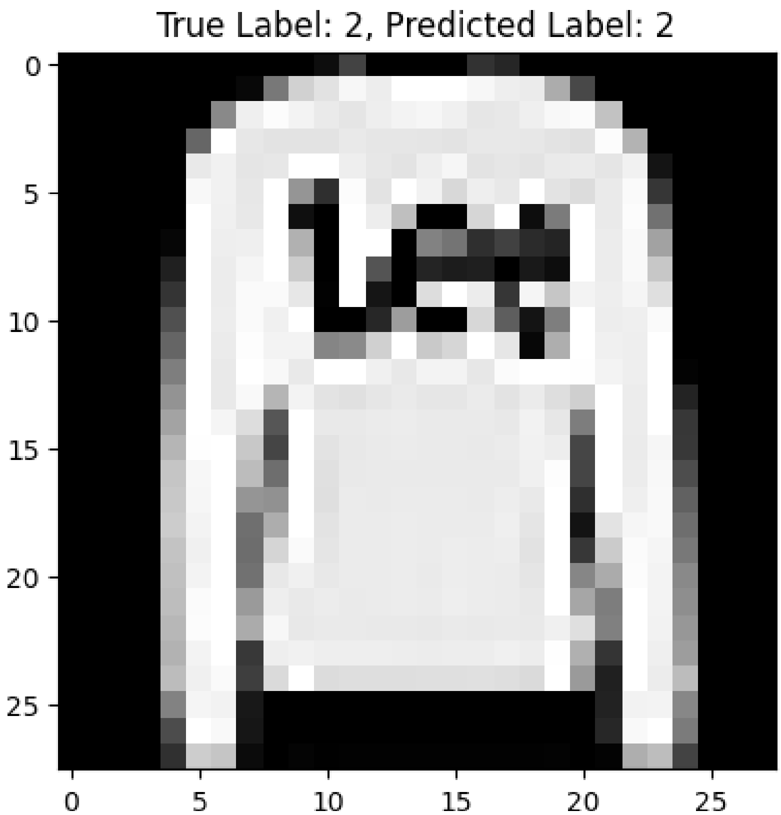
```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 81s 28ms/step - accuracy: 0.9142 -
loss: 0.2362
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - accuracy: 0.8960 - loss:
0.3008
Test accuracy: 0.8950999975204468

model.summary()

Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                          ┃ Output Shape                 ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇
━━━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                       │ (None, 26, 26, 32)           │
320 │
├───────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ max_pooling2d (MaxPooling2D)          │ (None, 13, 13, 32)           │
0 │
├───────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ conv2d_1 (Conv2D)                     │ (None, 11, 11, 64)           │
18,496 │
├───────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ max_pooling2d_1 (MaxPooling2D)        │ (None, 5, 5, 64)             │
0 │
├───────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ flatten (Flatten)                     │ (None, 1600)                 │
0 │
├───────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ dense (Dense)                         │ (None, 10)                   │
16,010 │
└───────────────────────────────────────┴──────────────────────────────┴
─────────────────┘

 Total params: 104,480 (408.13 KB)

 Trainable params: 34,826 (136.04 KB)

 Non-trainable params: 0 (0.00 B)

 Optimizer params: 69,654 (272.09 KB)

import tensorflow as tf
import numpy as np
```

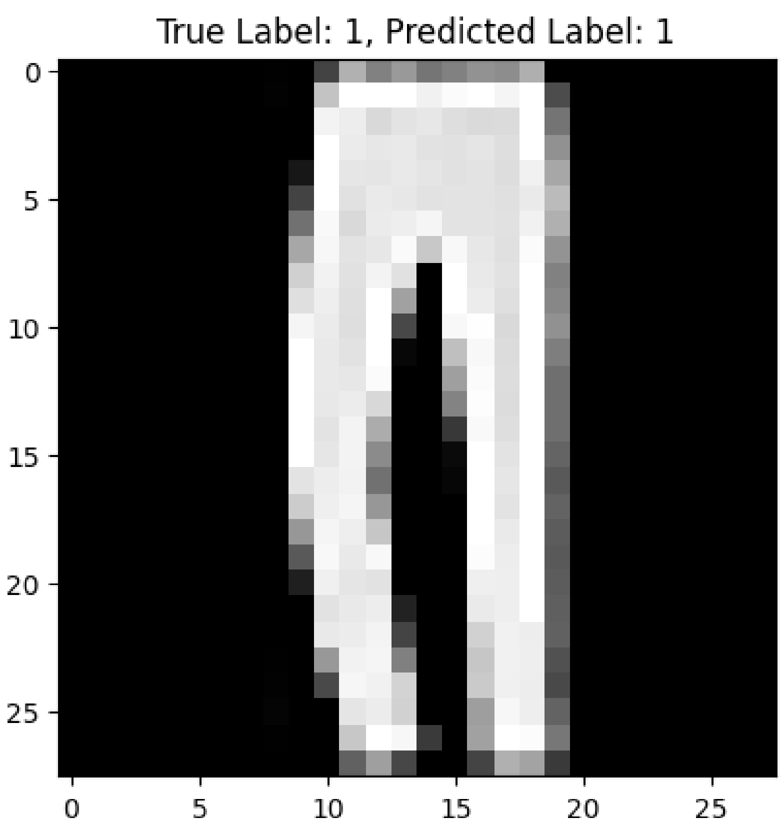
```python
from sklearn.metrics import f1_score, confusion_matrix,
accuracy_score, classification_report

y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true_classes = np.argmax(y_test, axis=1)

f1 = f1_score(y_true_classes, y_pred_classes, average='weighted')
accuracy = accuracy_score(y_true_classes, y_pred_classes)
cm = confusion_matrix(y_true_classes, y_pred_classes)
cr=classification_report(y_true_classes,y_pred_classes)
print(cr)

print("F1 Score:", f1)
print("Accuracy:", accuracy)
print("Confusion Matrix:\n", cm)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step
              precision    recall  f1-score   support

           0       0.88      0.78      0.83      1000
           1       0.99      0.98      0.99      1000
           2       0.88      0.82      0.85      1000
           3       0.86      0.93      0.89      1000
           4       0.86      0.81      0.84      1000
           5       0.98      0.98      0.98      1000
           6       0.65      0.79      0.71      1000
           7       0.95      0.96      0.96      1000
           8       0.99      0.94      0.97      1000
           9       0.97      0.95      0.96      1000

    accuracy                           0.90     10000
   macro avg       0.90      0.90      0.90     10000
weighted avg       0.90      0.90      0.90     10000

F1 Score: 0.8966458262596403
Accuracy: 0.8951
Confusion Matrix:
 [[775   1  16  44   4   1 156   0   3   0]
 [  2 979   0  16   2   0   1   0   0   0]
 [ 12   1 821  12  57   0  97   0   0   0]
 [  4   4   9 930  15   0  37   0   1   0]
 [  0   0  36  41 813   0 110   0   0   0]
 [  0   0   0   2   0 983   0   8   0   7]
 [ 77   0  48  33  50   0 788   0   4   0]
 [  0   0   0   0   0  18   0 963   1  18]
 [  5   0   8   5   1   2  32   2 944   1]
 [  1   0   0   0   0   3   0  41   0 955]]
```

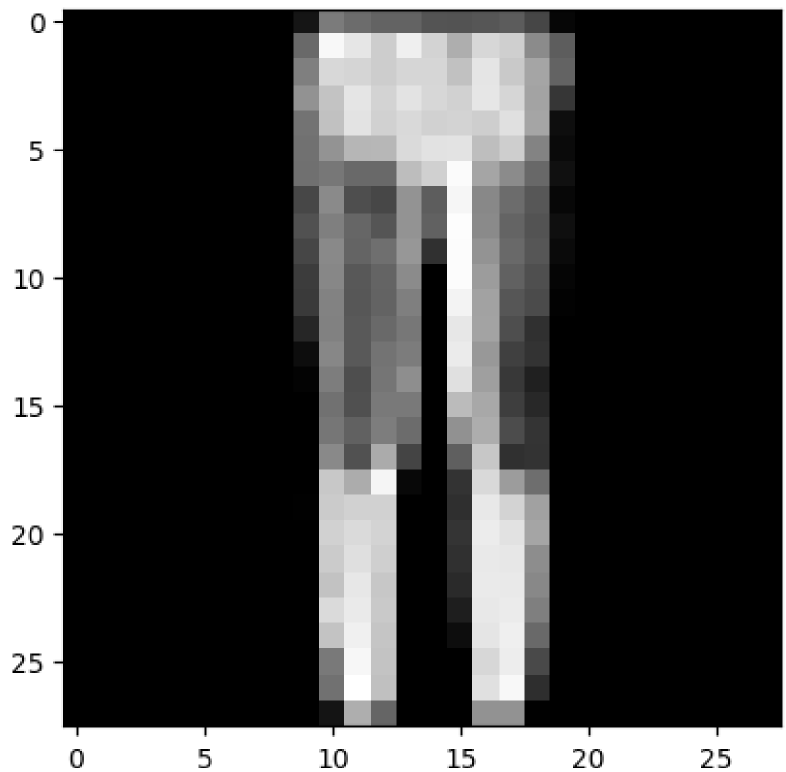
```
import matplotlib.pyplot as plt
from sklearn.metrics import f1_score, confusion_matrix, accuracy_score
import numpy as np
y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true_classes = np.argmax(y_test, axis=1)
for i in range(5):
    plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
    plt.title(f"True Label: {np.argmax(y_test[i])}, Predicted Label:
{y_pred_classes[i]}")
    plt.show()

313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step
```



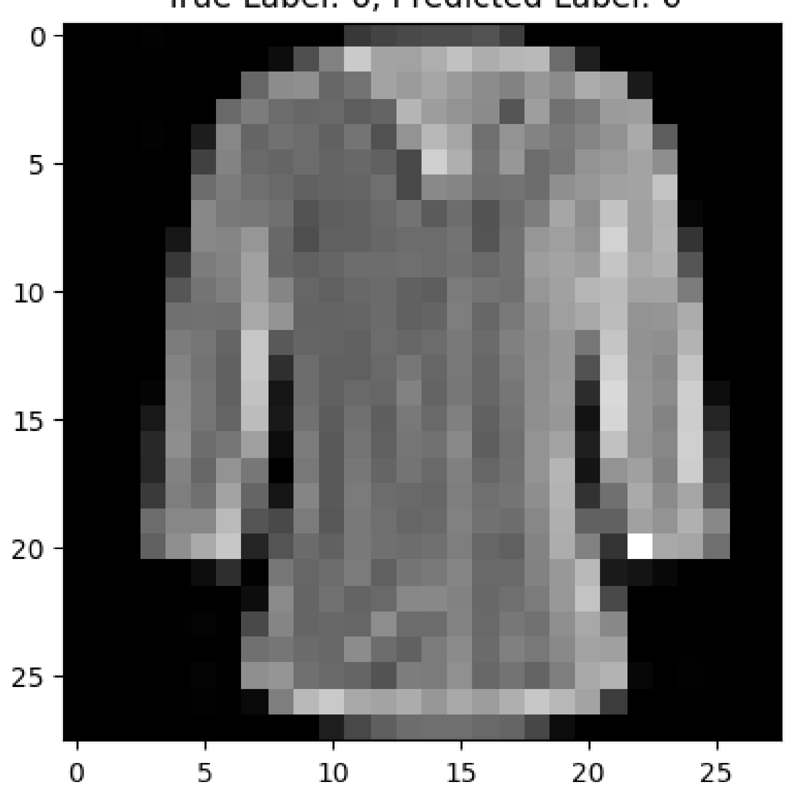

True Label: 2, Predicted Label: 2
0
5
10
15
20
25


True Label: 1, Predicted Label: 1
0
5
10
15
20
25

True Label: 1, Predicted Label: 1



True Label: 6, Predicted Label: 6



```
feature_extractor = tf.keras.Model(inputs=model.inputs,
outputs=model.layers[-2].output)
features = feature_extractor.predict(x_test)

print("Feature Shape:", features.shape)
```

```
 17/313 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step

/usr/local/lib/python3.11/dist-packages/keras/src/models/
functional.py:237: UserWarning: The structure of `inputs` doesn't
match the expected structure.
Expected: ['keras_tensor']
Received: inputs=Tensor(shape=(32, 28, 28, 1))
  warnings.warn(msg)

313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step

/usr/local/lib/python3.11/dist-packages/keras/src/models/
functional.py:237: UserWarning: The structure of `inputs` doesn't
match the expected structure.
Expected: ['keras_tensor']
Received: inputs=Tensor(shape=(None, 28, 28, 1))
  warnings.warn(msg)

Feature Shape: (10000, 1600)
```

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from tensorflow.python import keras
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense, Flatten, Conv2D,
Dropout, MaxPooling2D
from IPython.display import SVG
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import plotly.graph_objs as go
import plotly.figure_factory as ff
from plotly import tools
from plotly.offline import download_plotlyjs, init_notebook_mode,
plot, iplot
init_notebook_mode(connected=True)

IMG_ROWS = 28
IMG_COLS = 28
NUM_CLASSES = 10
TEST_SIZE = 0.2
RANDOM_STATE = 2018
#Model
```

```
NO_EPOCHS = 50
BATCH_SIZE = 128
IS_LOCAL = False
import os
from tensorflow import keras
(x_train, y_train), (x_test, y_test) =
keras.datasets.fashion_mnist.load_data()
print("x_train shape:", x_train.shape)
print("y_train shape:", y_train.shape)
print("x_test shape:", x_test.shape)
print("y_test shape:", y_test.shape)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
x_train shape: (60000, 28, 28)
y_train shape: (60000,)
x_test shape: (10000, 28, 28)
y_test shape: (10000,)
```

```
X_train, X_test, y_train, y_test = train_test_split(
       x_train, y_train, test_size=0.3, random_state=42
   )
```

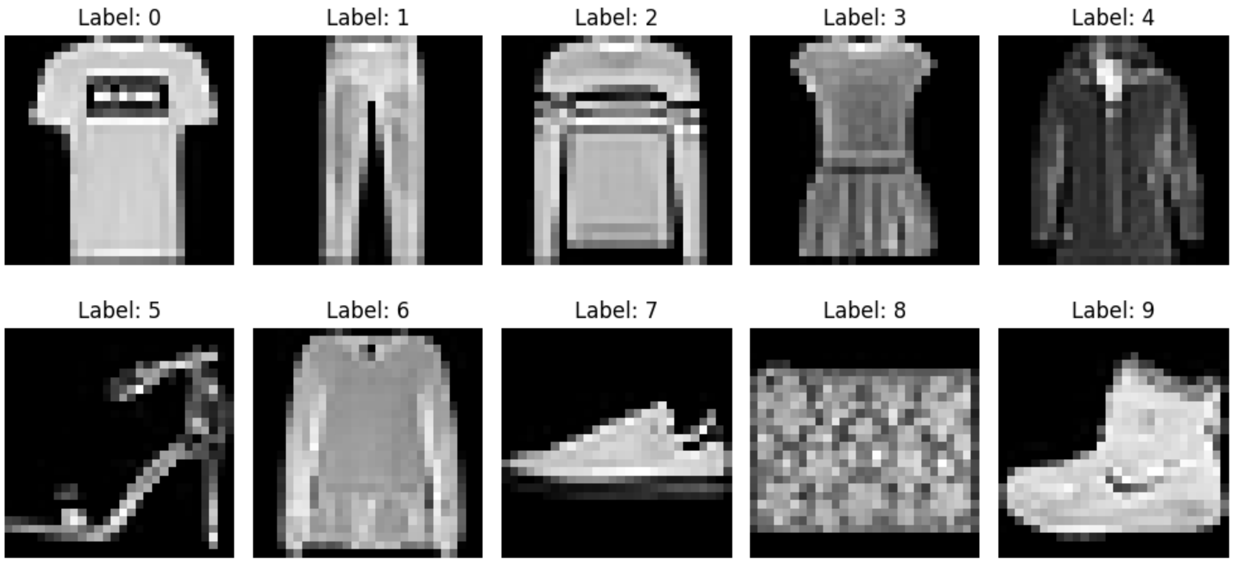
```
X_train_flat = X_train.reshape(X_train.shape[0], -1)  # Flatten the
images
X_test_flat = X_test.reshape(X_test.shape[0], -1)
train_data = pd.DataFrame(np.column_stack([X_train_flat, y_train]))
test_data = pd.DataFrame(np.column_stack([X_test_flat, y_test]))
```

```
X_train[0]
```

```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35, 109,
90,
         67,  95,  99,  89,  82,  61,   0,   0,   0,   0,   0,   0,
0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 152, 233,
244,
        226, 240, 239, 199, 135, 104,   5,   0,   0,   0,   0,   0,
0,
```

```
          178, 209, 196, 170, 180, 191, 190, 217, 164,   0,   0,   0,
0,
            0,   0],
        [  0,   0,   0,   0,   0,   0,   0, 164, 205, 186, 200, 197,
194,
          194, 205, 197, 184, 181, 188, 200, 222, 172,   0,   0,   0,
0,
            0,   0],
        [  0,   0,   0,   0,   0,   0,   0, 168, 204, 145, 158, 171,
181,
          185, 208, 196, 187, 196, 194, 186, 220, 177,   0,   0,   0,
0,
            0,   0],
        [  0,   0,   0,   0,   0,   0,   0, 154, 212, 175, 171, 171,
176,
          180, 186, 189, 187, 180, 168, 180, 215, 148,   0,   0,   0,
0,
            0,   0],
        [  0,   0,   0,   0,   0,   0,   0, 131, 243, 167, 182, 195,
194,
          189, 190, 192, 197, 197, 190, 200, 255, 186,   0,   0,   0,
0,
            0,   0],
        [  0,   0,   0,   0,   0,   0,   0,   0, 169, 156, 158, 174,
176,
          185, 190, 188, 172, 165, 149, 136, 127,  14,   0,   0,   0,
0,
            0,   0]], dtype=uint8)
```

```python
import matplotlib.pyplot as plt
import numpy as np

label_indices = {}

for i in range(len(x_train)):
    label = np.argmax(y_train[i])
    if label not in label_indices:
        label_indices[label] = i
fig, axes = plt.subplots(2, 5, figsize=(10, 5))
for label, index in label_indices.items():
    ax = axes.flat[label]
    ax.imshow(x_train[index].reshape(28, 28), cmap='gray')
    ax.set_title(f"Label: {label}")
    ax.axis('off')
plt.tight_layout()
plt.show()
```


Label: 0
Label: 1
Label: 2
Label: 3
Label: 4
Label: 5
Label: 6
Label: 7
Label: 8
Label: 9