Fashion Apparel Classification using CNN

Group members :

- Youssef ELMOUMEN **FVW745**
- Doha Djellit **VFOAXE**
- Thabet Asmaa A. H. Naima **ZWGD9E**
- Abed Al Hadi Ali **YTUK16**

Introduction

Objective:

Train a CNN that accurately classifies different types of apparel from the Fashion MNIST dataset.

Importance:

Benefits in inventory management, trend analysis, and enhancing customer experience in the fashion retail industry

Dataset Details:

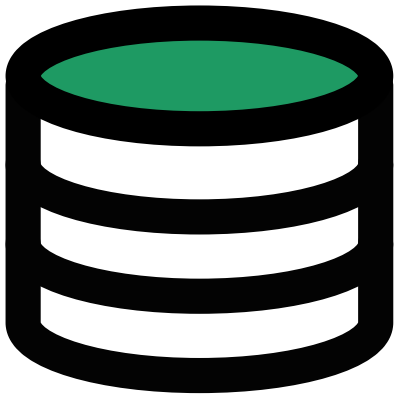
Source: Zalando Research

Content: 70,000 grayscale images (28x28 pixels) of 10 fashion categories.

Classes:

T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot

Table of contents



01
Fashion MNIST Dataset

About the dataset

- Developed by Zalando Research, a branch of Zalando SE, a European e-commerce company.
- Publicly released as an open-source dataset in 2017.
- Collection of 70,000 grayscale images in 10 categories.
- Available in a compressed format with training and test set labels and images.

Fashion MNIST Properties

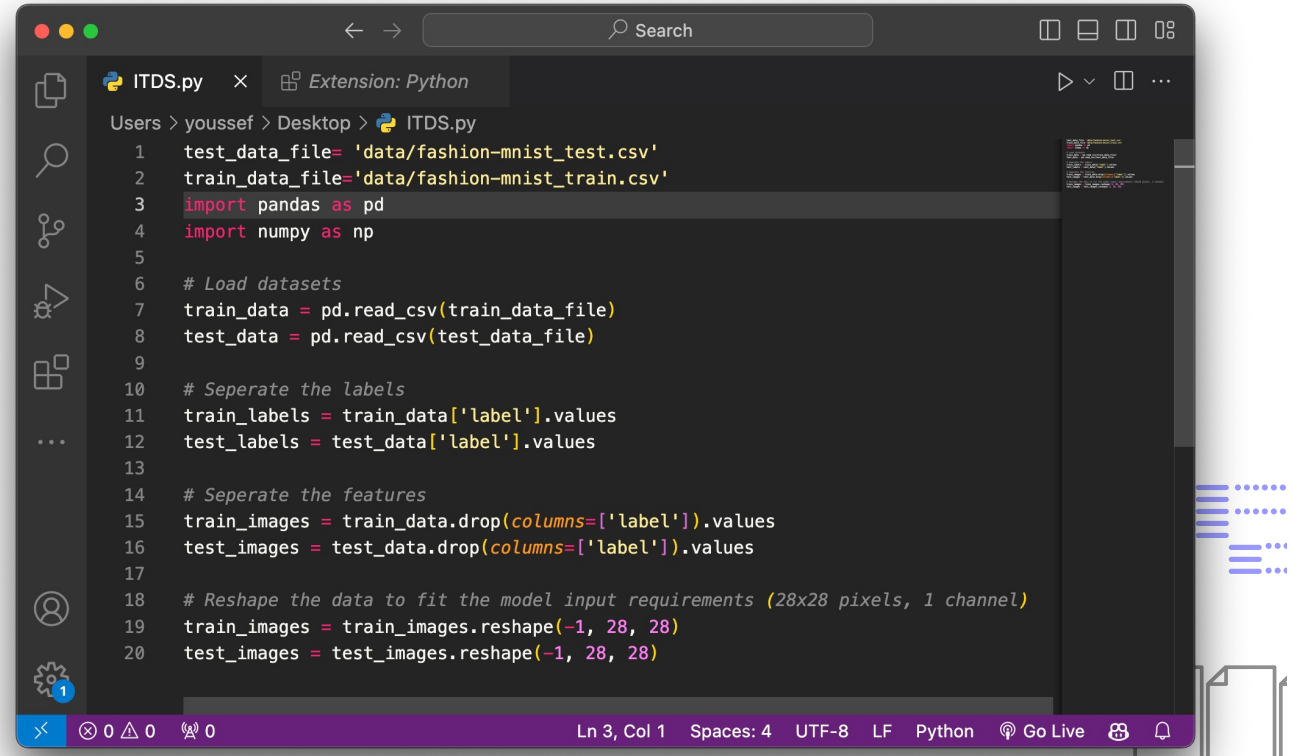
An MNIST-like dataset of 70,000 28x28 labeled fashion images

Purpose and Usage:	Designed as a more challenging replacement for the traditional MNIST dataset of handwritten digits. Commonly used for benchmarking machine learning algorithms for image classification.
Dataset Composition	Split into 60,000 training images and 10,000 test images. Uniform distribution of classes (each class has 7,000 images).
Classes in the Dataset	28x28 grayscale image
Classes in the Dataset:	T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot.
Image details	Images are 28x28 pixels, grayscale, with each pixel value ranging from 0 to 255.
Features	785 Feature
Source	https://www.kaggle.com/datasets/zalando-research/fashionmnist

02 Data Loading

We begin by loading the Fashion MNIST dataset, which consists of 60,000 training images and 10,000 test images. Each image is a 28x28 pixel grayscale representation of a fashion item, categorized into 10 distinct classes. As mentioned before

- Pandas library used For reading CSV files and data manipulation
- NumPy For numerical operations
- and data manipulation



- **Data Sources**: Utilizes 'data/fashion-mnist_train.csv' for training and 'data/fashion-mnist_test.csv' for testing.

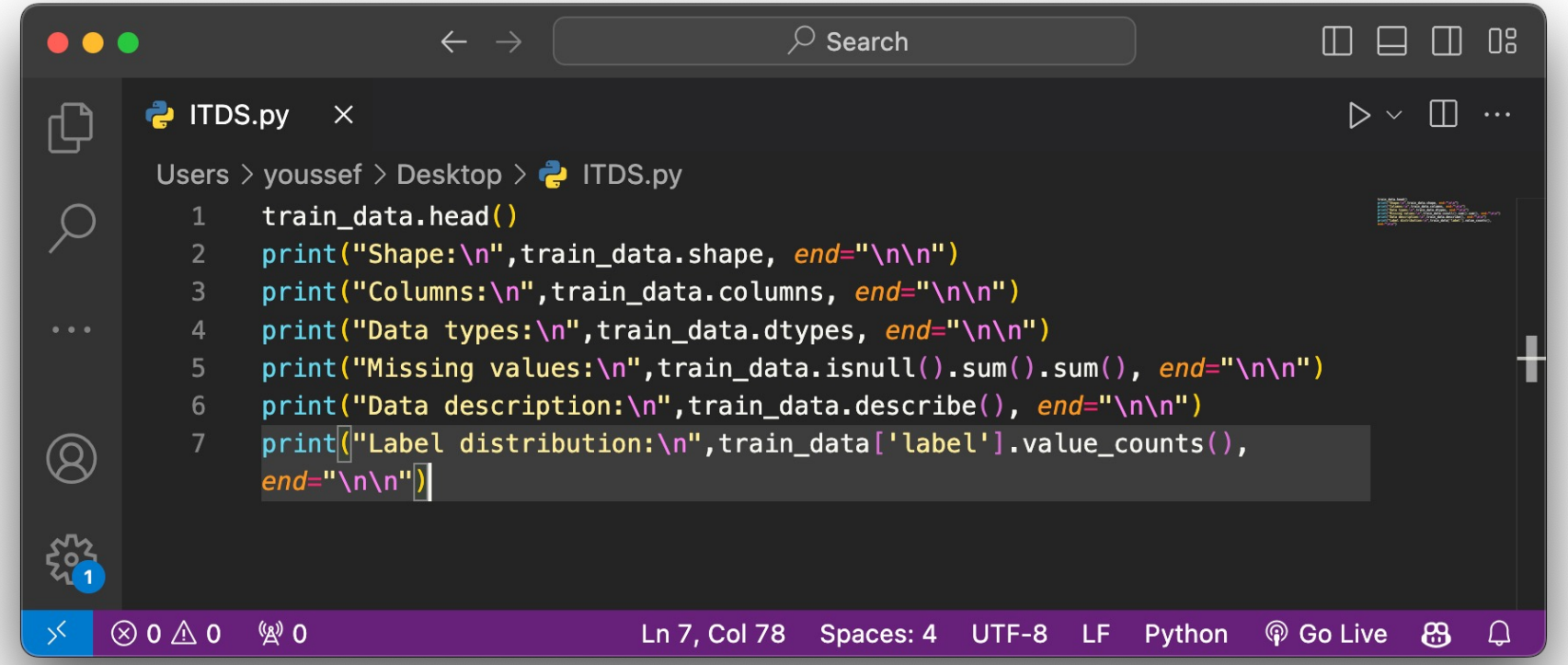
- **Data Loading**: CSV files are loaded into DataFrame objects for both training and test datasets.

- **Separating Labels and Features**: Labels are extracted into train_labels and test_labels, while pixel features are stored in train_images and test_images after removing the 'label' column.

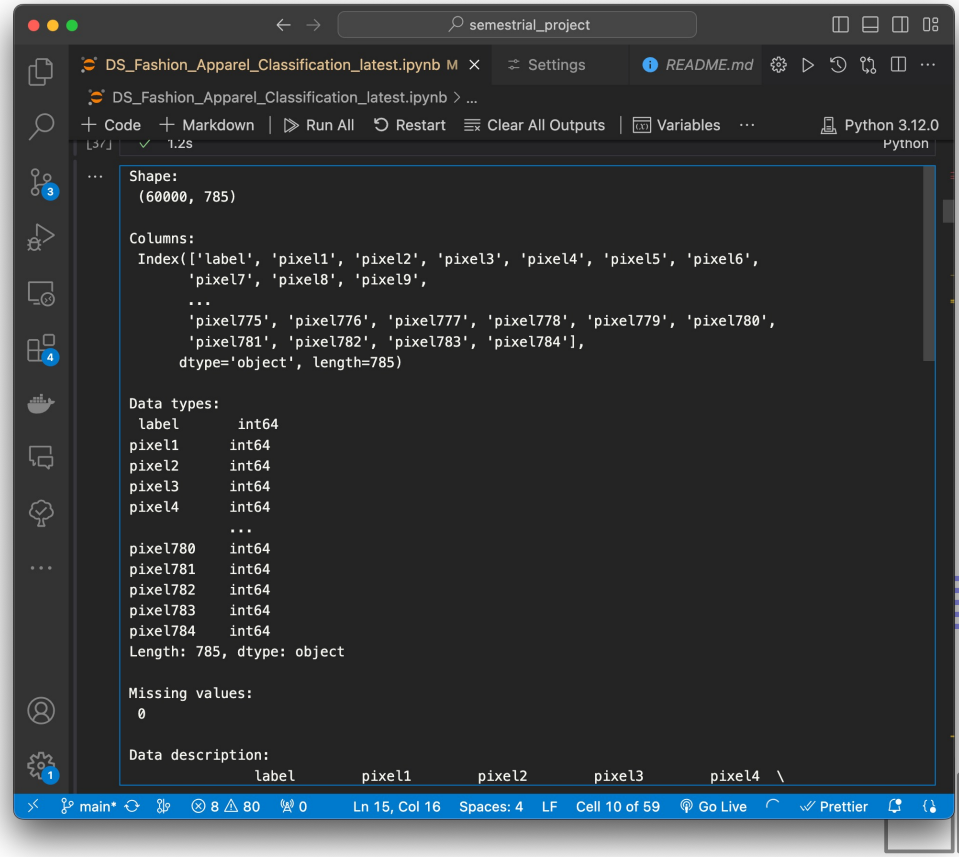
- **Reshaping Data**: Images are reshaped to 28x28 pixels to match neural network input requirements, resulting in 60000 x 28 x 28 dimensions for training images and 10000 x 28 x 28 for test images.

03
Data Visualization & Exploration

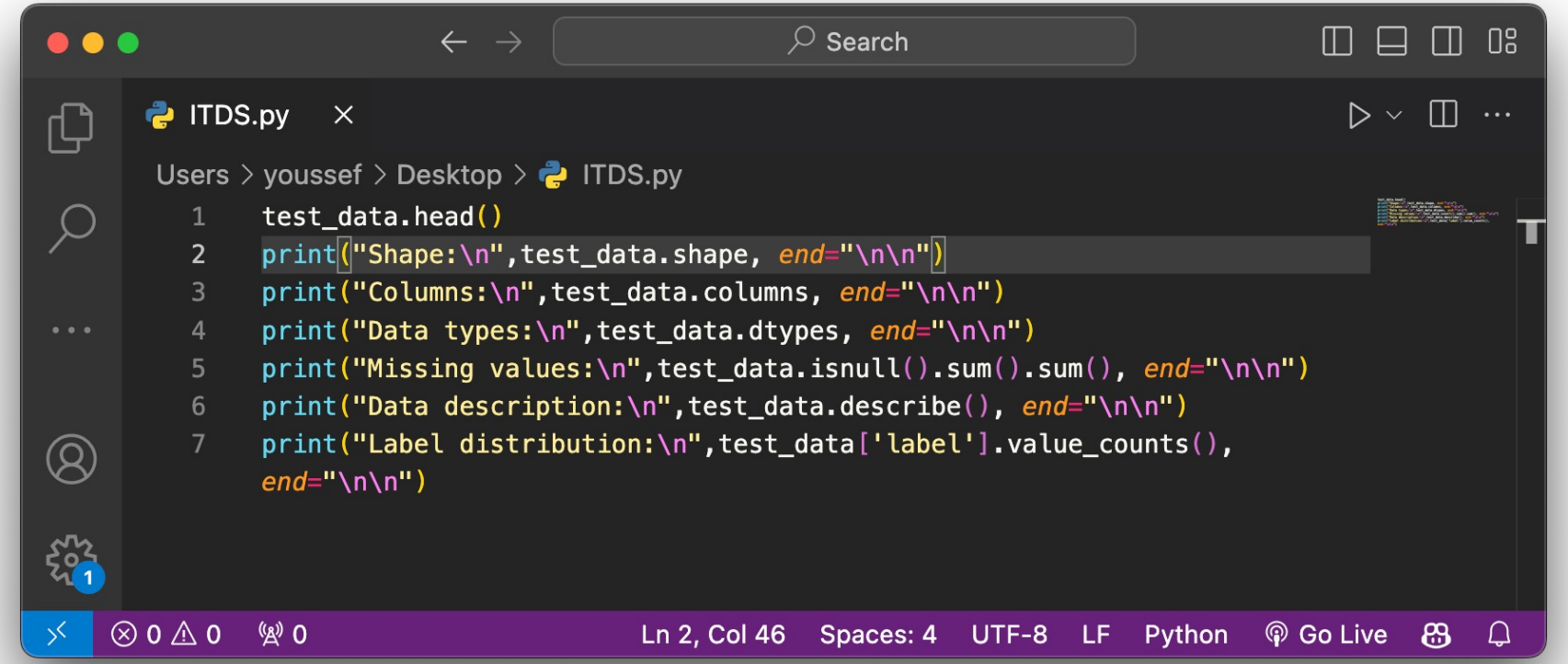
Exploring the train data



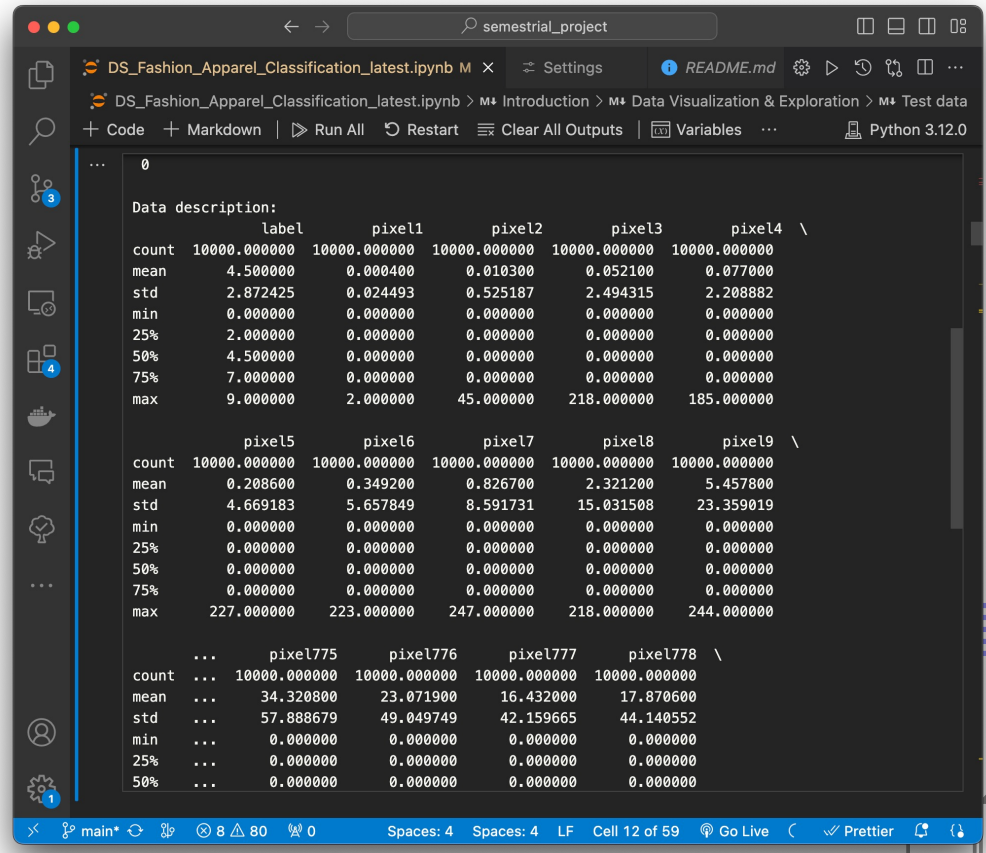
- The dataset consists of 60,000 samples, each with 785 columns.
- The columns include one 'label' column and 784 pixel columns (pixel1 to pixel784), corresponding to the 28x28 pixel values of the images.
- All columns are of integer data type (int64).
- There are no missing values in the dataset.
- The pixel values range from 0 to 255, with most of the pixels having values close to 0, suggesting that many pixels are background.

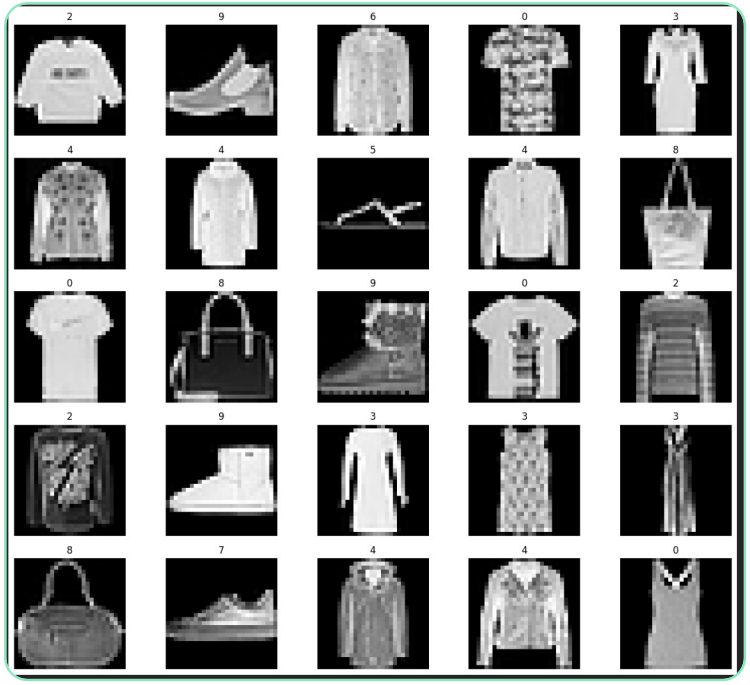


Exploring the test data



- The dataset consists of 10,000 samples, each with 785 columns.
- The columns include one 'label' column and 784 pixel columns (pixel1 to pixel784), corresponding to the 28x28 pixel values of the images.
- All columns are of integer data type (int64).
- There are no missing values in the dataset.
- The pixel values range from 0 to 255, with most of the pixels having values close to 0, suggesting that many pixels are background.

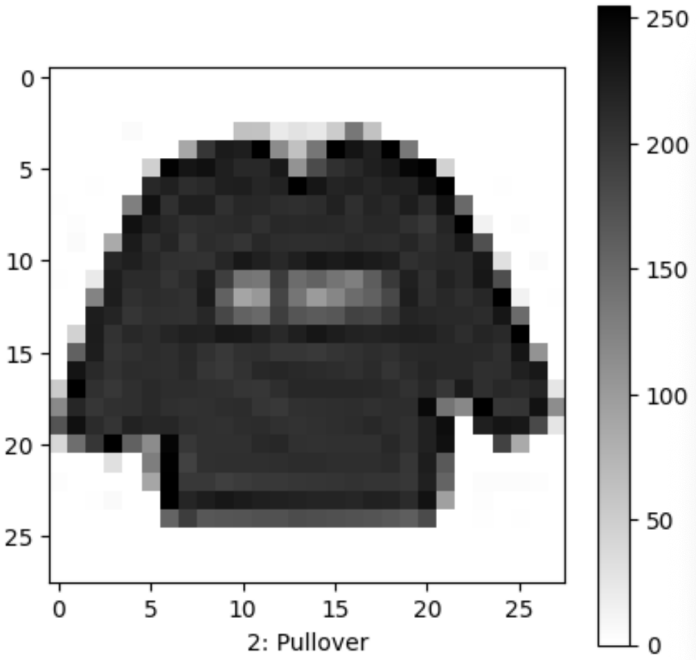




To gain insights into the Fashion MNIST dataset, we visualize various apparel classes to understand their distribution and individual image characteristics.

Matplotlib used For creating visualizations.

Sample representation of the data from the Fashion MNIST Dataset labled



Search

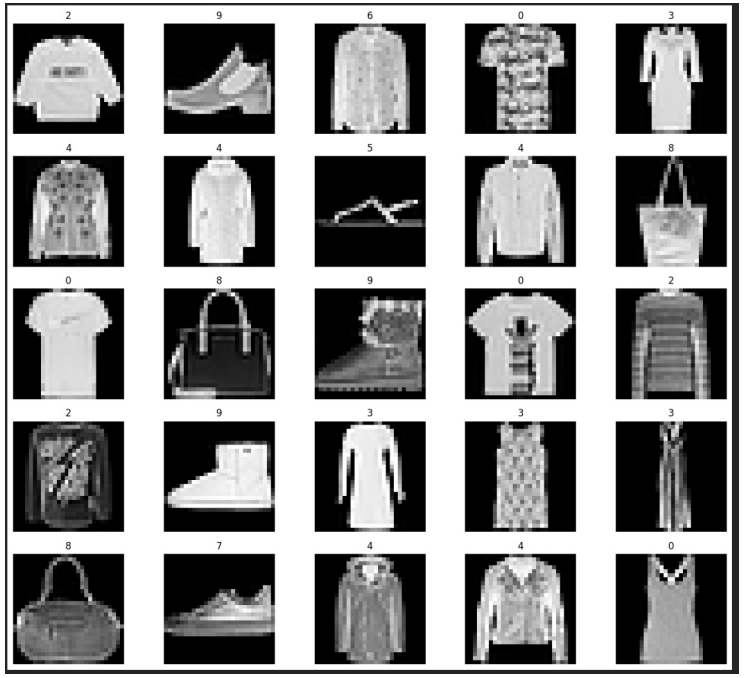
ITDS.py

Users > youssef > Desktop > ITDS.py

```
target_image_index = 0
plt.figure(figsize=(5, 5))
plt.imshow(train_images[target_image_index],
cmap=plt.cm.binary)
plt.xlabel("{}: {}".format(
    train_labels[target_image_index],
    lables_dict[train_labels
    [target_image_index]]))
plt.colorbar()
plt.show()
```

0 0 0 UTF-8 LF Python Go Live

Single image representation with the color bar with its code snippet



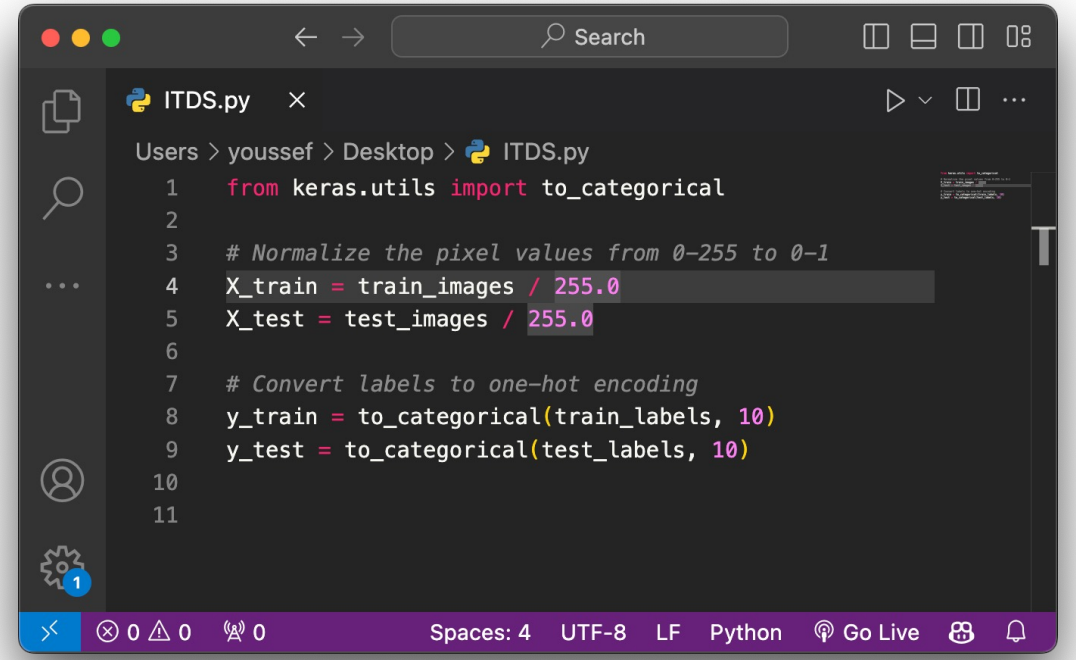
Sample Images from the Fashion MNIST Dataset with Labels

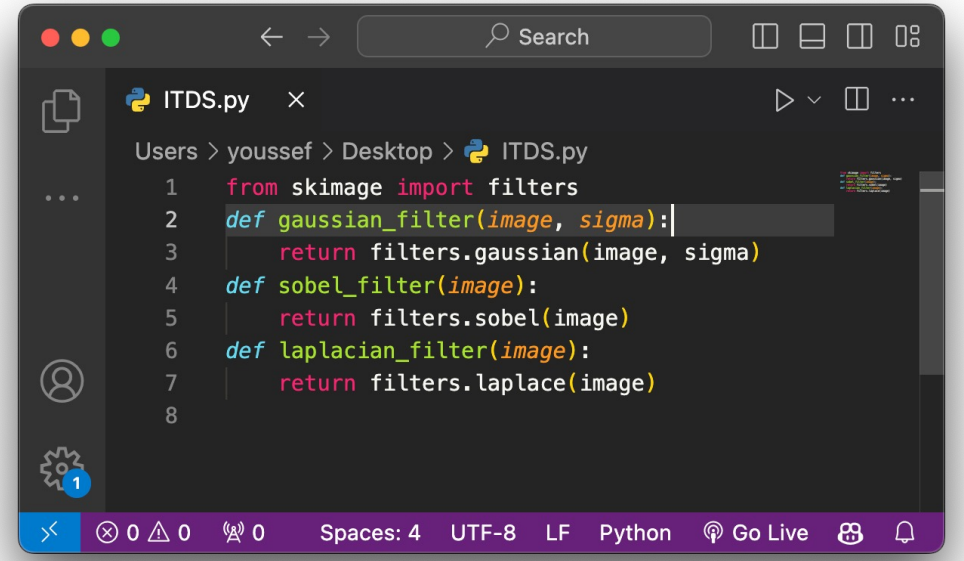
- To gain insights into the Fashion
- MNIST dataset, we visualize
- various apparel classes to
- understand their distribution and
- individual image characteristics.
- Matplotlib used For creating
- visualizations.

04
Data Preprocessing

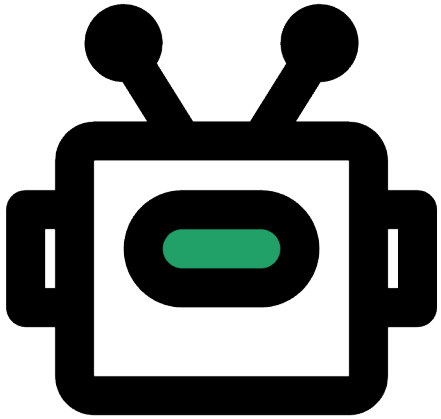
To prepare the dataset for training, We:

- didn't do any extra transformation as there were no null or missing values.
- normalized the image pixel values
- converted the labels using one Hotshot method





We also tested various filters, including Laplacian, Gaussian, and Sobel, to enhance the image clarity. However, we observed that the model achieved higher accuracy without the application of these filters.

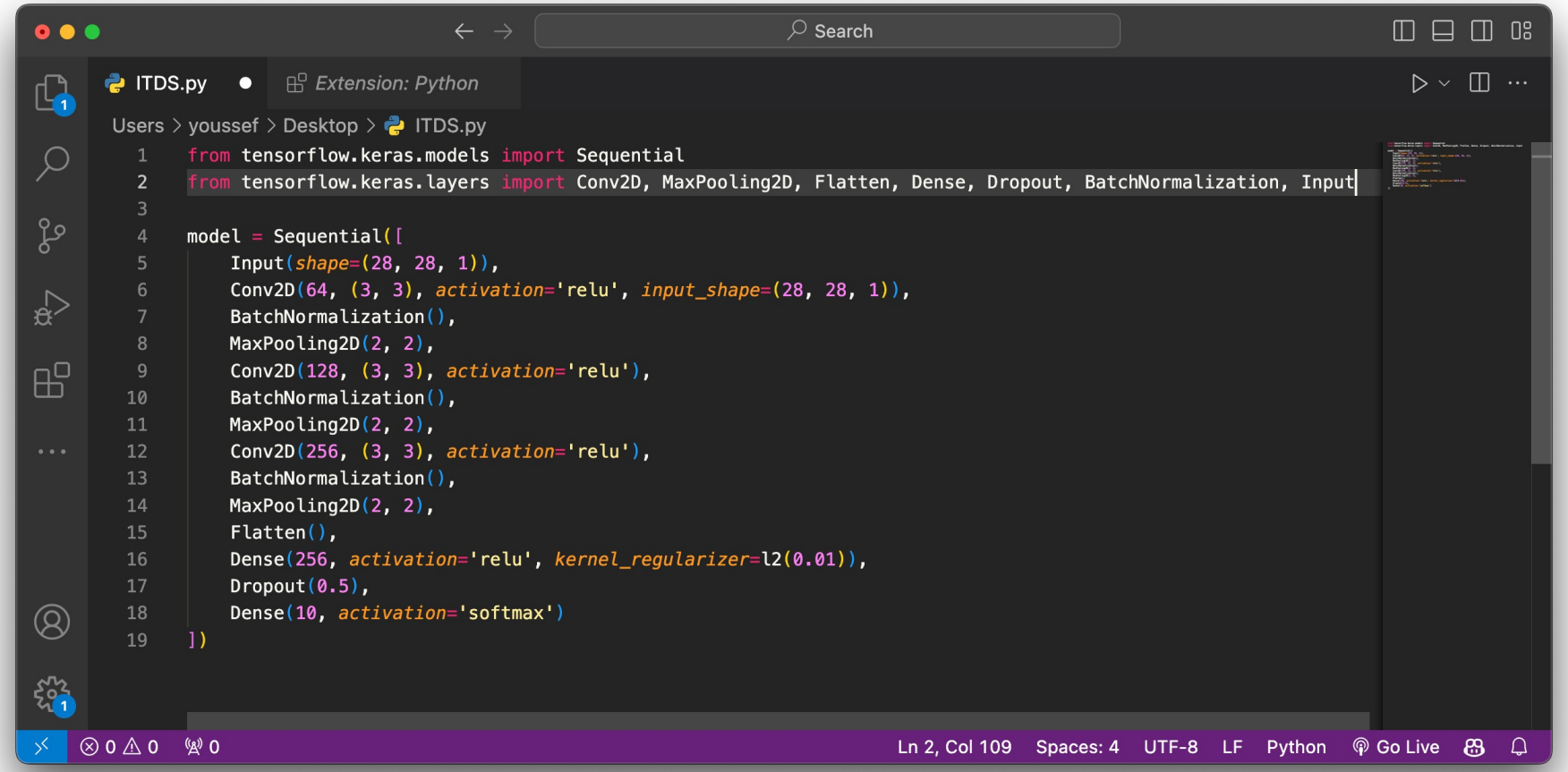


05 Model Training

Introduction

- The framework used in this project is : TensorFlow
- A Convolutional Neural Network (CNN) was designed to classify the fashion apparel images in the Fashion MNIST dataset.
- The architecture was refined through experimentation with various layers and hyperparameters.

CNN Model Development



CNN Model Development

Overview

- Utilizes a Sequential model from TensorFlow's Keras API.
- Receives input images of size 28x28 pixels, with 1 channel (grayscale).

Layers

- **Input Layer:**
  Input(shape=(28, 28, 1)): Defines the model to receive input images of size 28x28 pixels, with 1 channel (grayscale).

CNN Model Development

- **First Convolutional Layer:**
  64 filters of size 3x3.
  Activation function: ReLU (Rectified Linear Unit).
  Followed by Batch Normalization and Max Pooling (2x2).

- **Second Convolutional Layer:**
  128 filters of size 3x3.
  Activation function: ReLU.
  Includes Batch Normalization and Max Pooling (2x2)

- **Third Convolutional Layer:**
  256 filters of size 3x3.
  Activation function: ReLU.
  Includes Batch Normalization and Max Pooling (2x2).

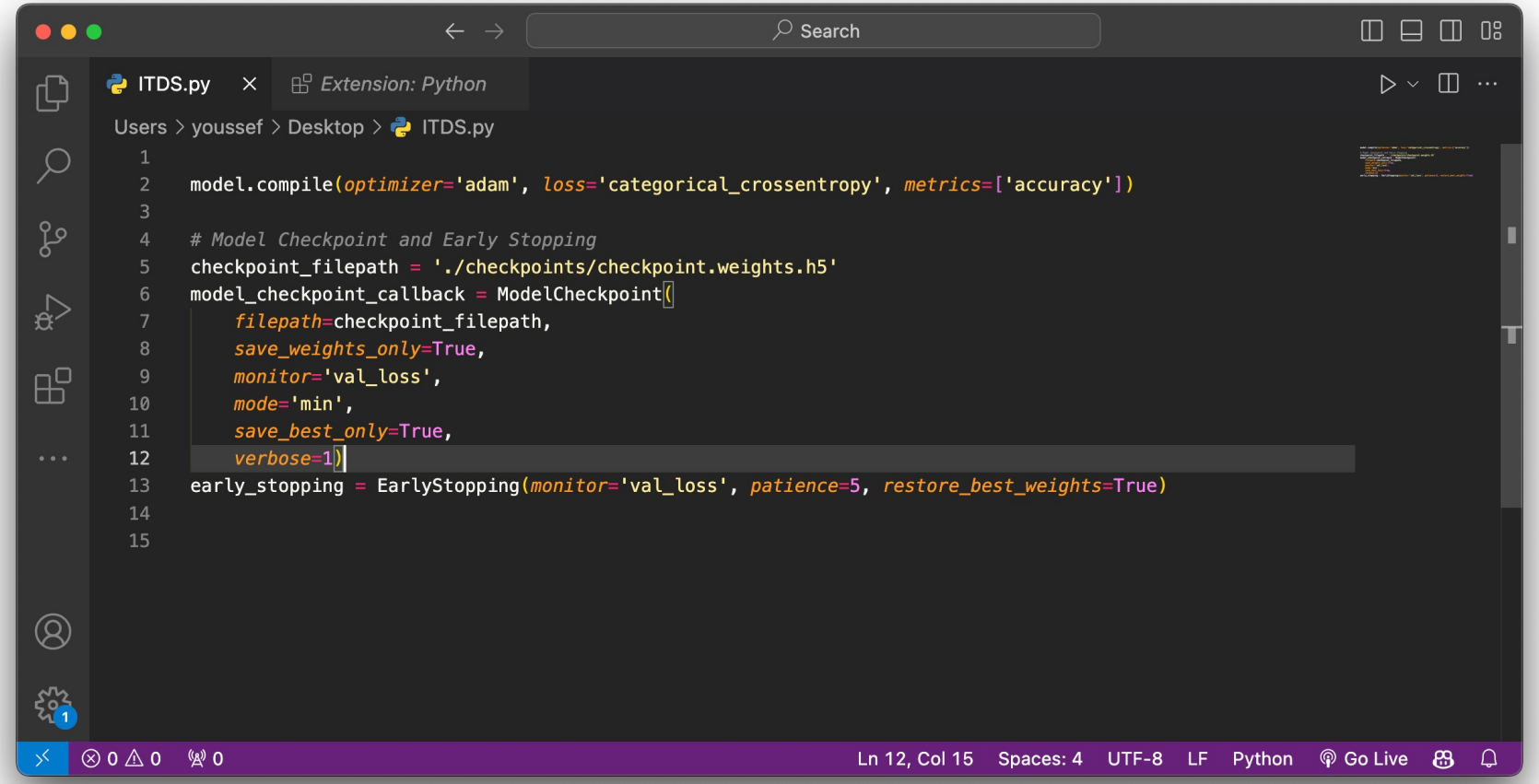
CNN Model Development

- **Flattening Layer:**
  Converts the 2D feature maps into a 1D feature vector.

- **Dense Layer:**
  256 neurons.
  Activation function: ReLU.
  Includes L2 regularization (weight decay coefficient of 0.01) to combat overfitting.
  Dropout layer with a rate of 0.5 to prevent overfitting.

- **Output Layer:**
  10 neurons (one for each clothing category).
  Activation function: Softmax for multi-class classification.

CNN Model Development

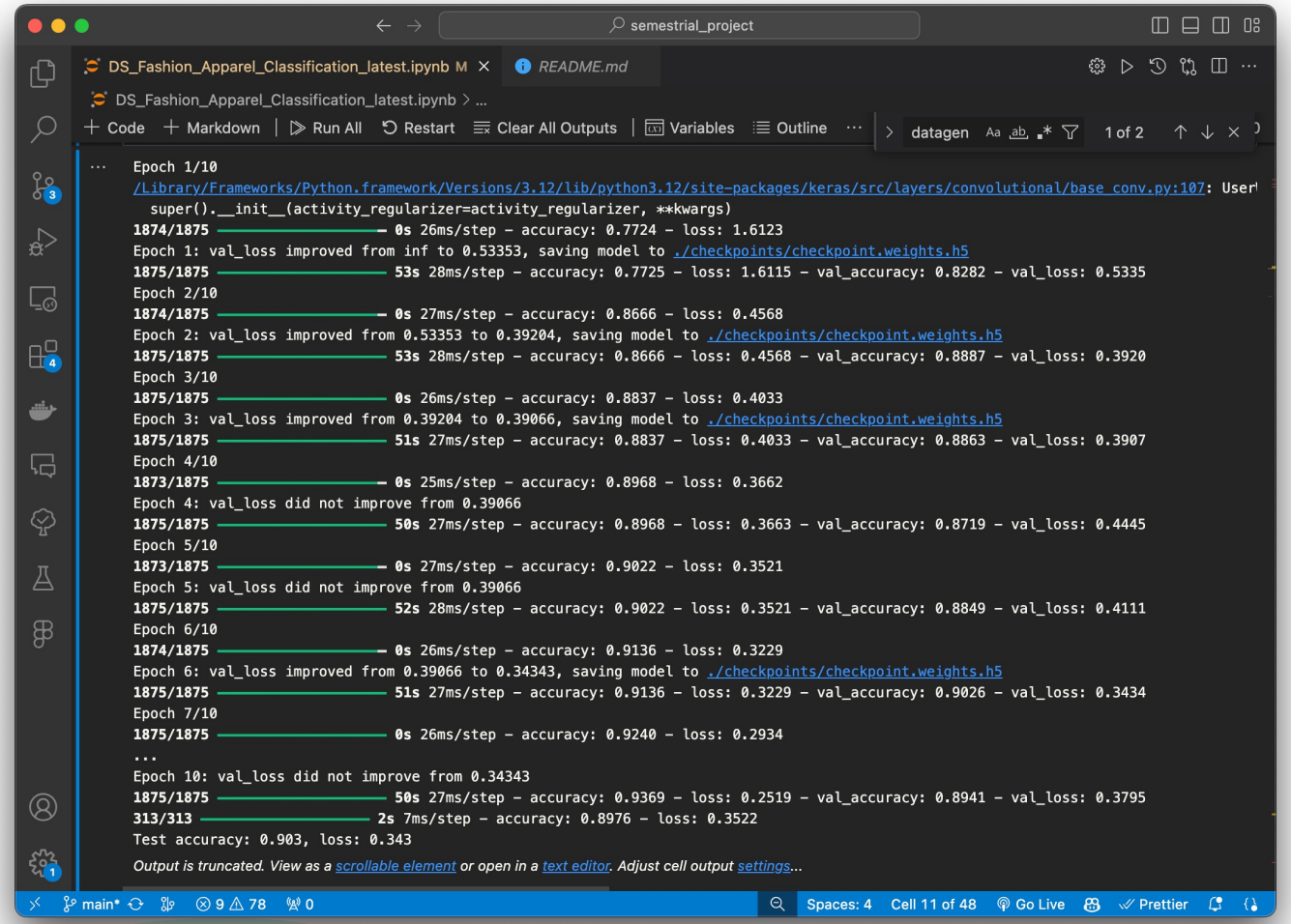


CNN Model Development

- **Compilation of the Model:**
  Optimizer: Adam.
  Loss function: Categorical Crossentropy.
  Evaluation metric: Accuracy.

- **Training Process:**
  Early stopping to terminate training when validation loss decreases, to prevent overfitting.
  Model checkpointing to save the model with the best validation loss.

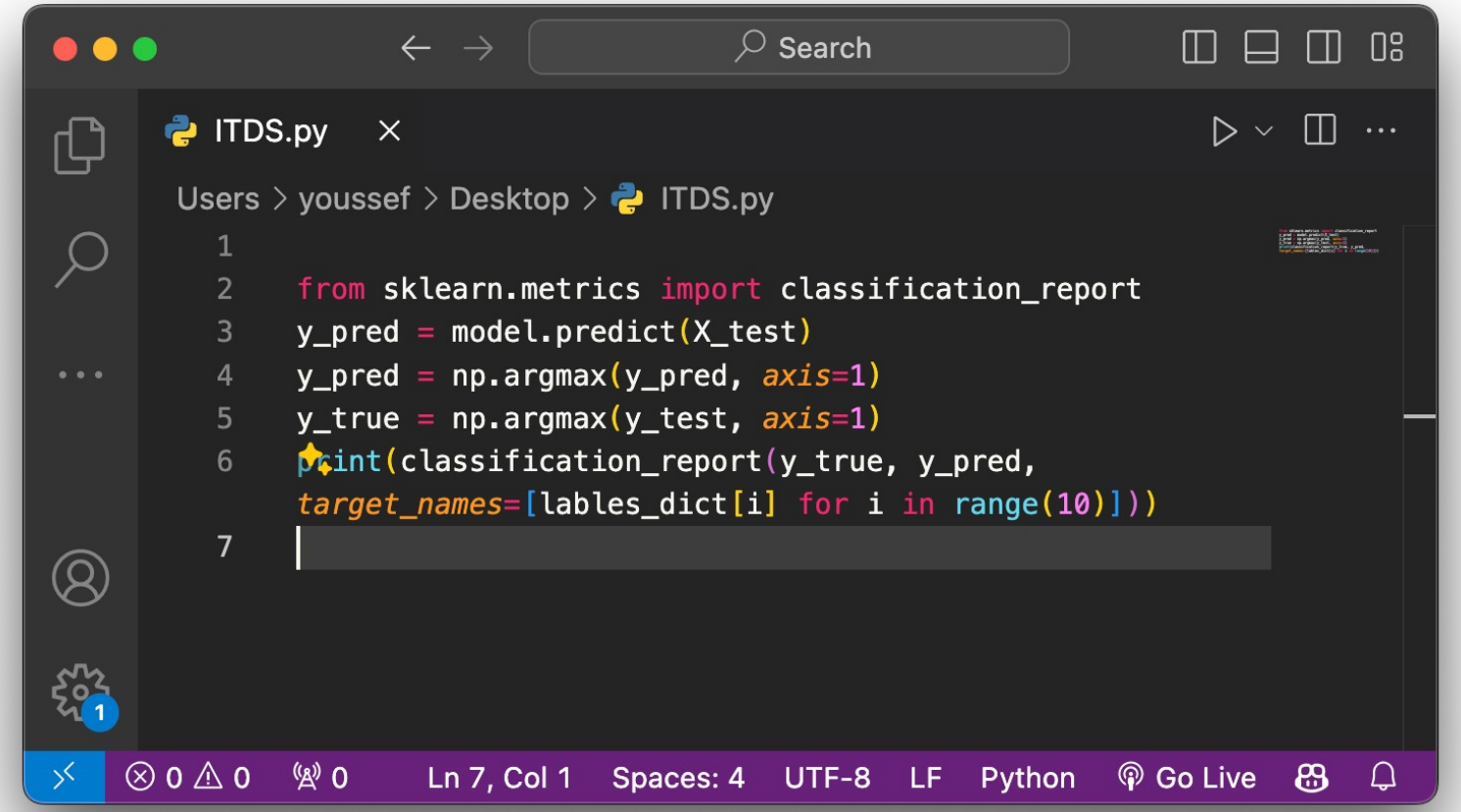
Training and Results



Training and Results

- Trained over 10 epochs with initial accuracy of 77.25% and loss of 1.6115.
- Achieved peak training accuracy of 93.69% with a loss reduced to 0.2519 by the final epoch.
- Best validation loss recorded at 0.34343 during Epoch 6, with the highest validation accuracy reaching 90.26% by the end of training.
- Final test performance demonstrated a robust accuracy of 90.3% and a loss of 0.343.

Performance Evaluation - Code

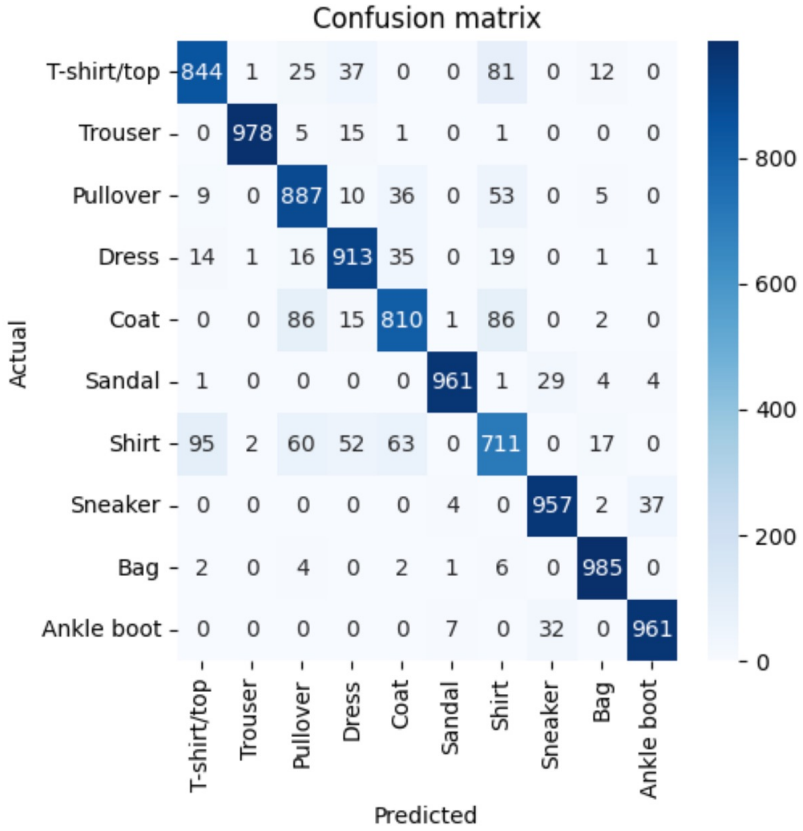


Performance Evaluation - Report

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

	precision	recall	f1-score	support
T-shirt/top	0.87	0.84	0.86	1000
Trouser	1.00	0.98	0.99	1000
Pullover	0.82	0.89	0.85	1000
Dress	0.88	0.91	0.89	1000
Coat	0.86	0.81	0.83	1000
Sandal	0.99	0.96	0.97	1000
Shirt	0.74	0.71	0.73	1000
Sneaker	0.94	0.96	0.95	1000
Bag	0.96	0.98	0.97	1000
Ankle boot	0.96	0.96	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

Performance Evaluation
Confusion Matrix



Performance Evaluation - Analysis

Overall Accuracy: The model achieved a high overall accuracy of 90%, showing it performs well across all categories.

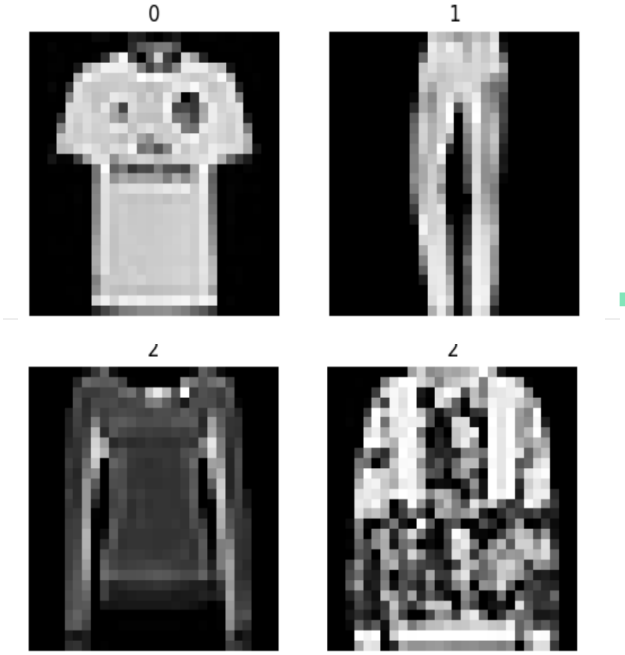
Top Performers: The Trouser category excelled with perfect precision of 100% and very high recall of 98%. The Sandal, Bag, and Ankle boot categories also performed excellently, with both precision and recall over 95%.

Areas for Improvement: The Shirt category had the weakest results, with a precision of 74% and a recall of 71%. This indicates a need for further adjustments to the model to improve its ability to correctly identify shirts.

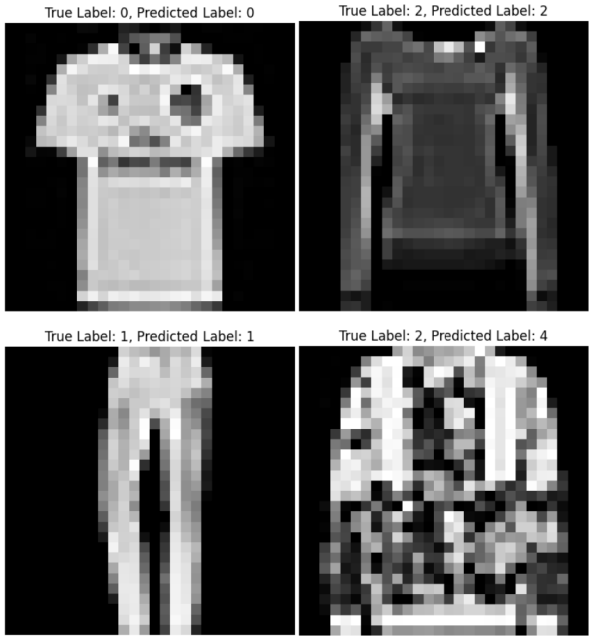
Balanced Metrics: Categories like Dress, Sandal, and Sneaker showed a good balance between precision and recall, leading to high f1-scores, which measure the accuracy of the model's predictions.

Model Testing - Prediction results

The CNN model demonstrates high accuracy in classifying fashion items, with a few misclassifications

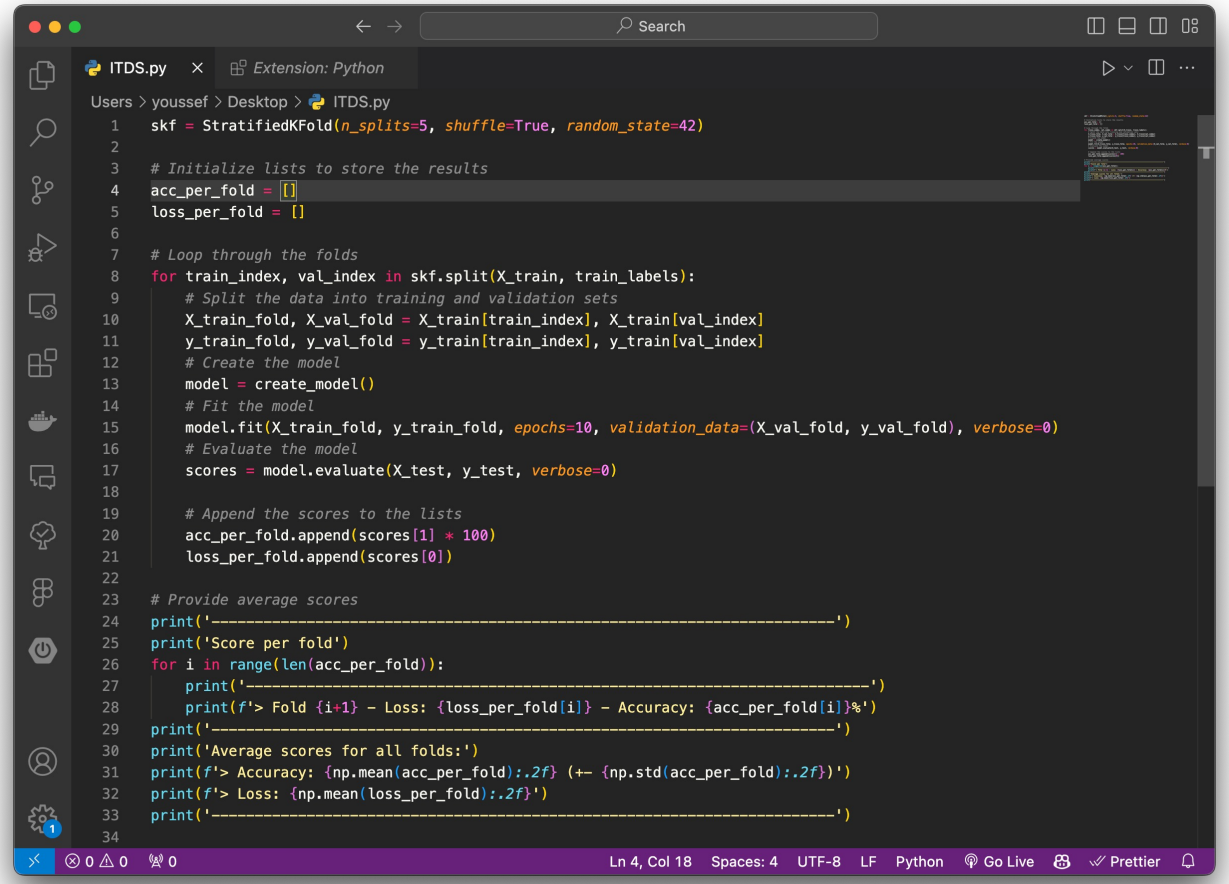


Original test Images



True vs. Predicted Labels

Model Testing – Cross Validation



The following are the result of the cross validation of out CNN

```
------------------------------------------------------------------------
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 0.4180976450443268 - Accuracy: 88.99999856948853%
------------------------------------------------------------------------
> Fold 2 - Loss: 0.41137993335723877 - Accuracy: 88.88000249862671%
------------------------------------------------------------------------
> Fold 3 - Loss: 0.5078486800193787 - Accuracy: 86.83000206947327%
------------------------------------------------------------------------
> Fold 4 - Loss: 0.45873013138771057 - Accuracy: 88.30000162124634%
------------------------------------------------------------------------
> Fold 5 - Loss: 0.44572535157203674 - Accuracy: 88.26000094413757%
------------------------------------------------------------------------
Average scores for all folds:
> Accuracy: 88.25 (+- 0.77)
> Loss: 0.45
------------------------------------------------------------------------
```

06 Comparison with other Classifiers

SVC Classifier

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60000 entries, 0 to 59999
Columns: 785 entries, label to pixel784
dtypes: int64(785)
memory usage: 359.3 MB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Columns: 785 entries, label to pixel784
dtypes: int64(785)
memory usage: 59.9 MB
None
Evaluation Metrics for SVC Classifier:
--------------------------------------
Accuracy: 0.8921
Balanced Accuracy: 0.8920999999999999
F1 Score: 0.8915003945780919
Log Loss: 0.29818199856446
```

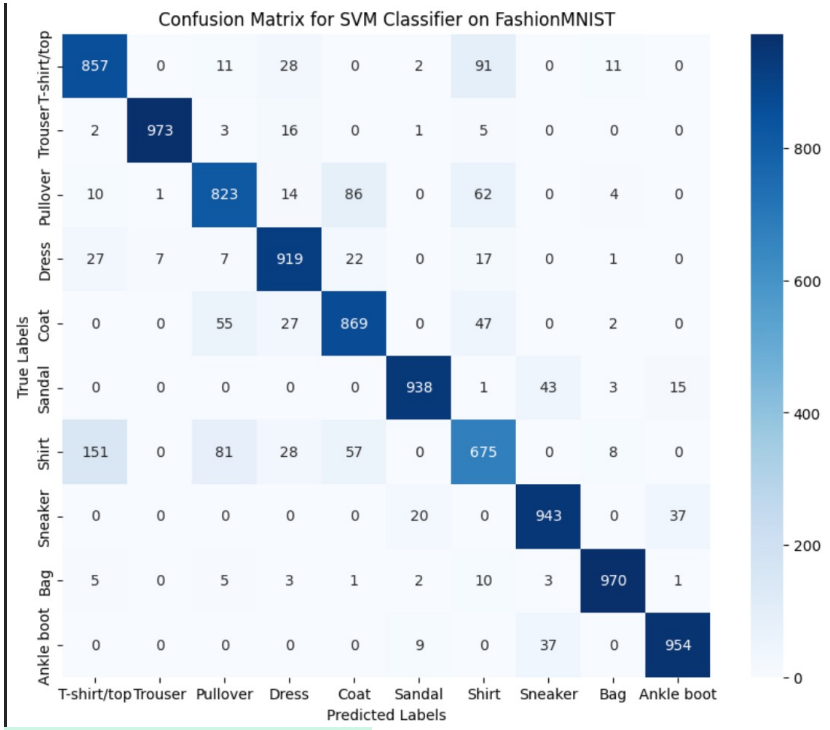
Training Metrics

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

 T-shirt/top       0.81      0.86      0.84      1000
     Trouser       0.99      0.97      0.98      1000
    Pullover       0.84      0.82      0.83      1000
       Dress       0.89      0.92      0.90      1000
        Coat       0.84      0.87      0.85      1000
      Sandal       0.97      0.94      0.95      1000
       Shirt       0.74      0.68      0.71      1000
     Sneaker       0.92      0.94      0.93      1000
         Bag       0.97      0.97      0.97      1000
  Ankle boot       0.95      0.95      0.95      1000

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```

Classification report



Confusion Matrix

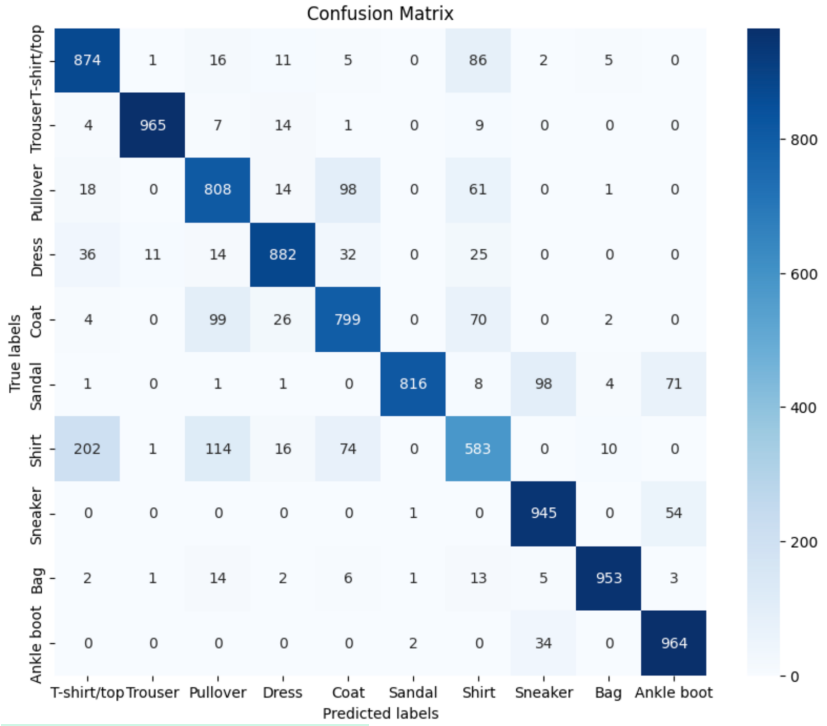
K-Neighbors Classifier

	precision	recall	f1-score	support
T-shirt/top	0.77	0.87	0.82	1000
Trouser	0.99	0.96	0.98	1000
Pullover	0.75	0.81	0.78	1000
Dress	0.91	0.88	0.90	1000
Coat	0.79	0.80	0.79	1000
Sandal	1.00	0.82	0.90	1000
Shirt	0.68	0.58	0.63	1000
Sneaker	0.87	0.94	0.91	1000
Bag	0.98	0.95	0.97	1000
Ankle boot	0.88	0.96	0.92	1000
accuracy			0.86	10000
macro avg	0.86	0.86	0.86	10000
weighted avg	0.86	0.86	0.86	10000

Classification report

```
Evaluation Metrics for KNeighbors Classifier:
--------------------------------------
Accuracy: 0.8589
Balanced Accuracy: 0.8589
F1 Score: 0.8580386825779729
Log Loss: 1.749503487525264
```

Training Metrics



Confusion Matrix

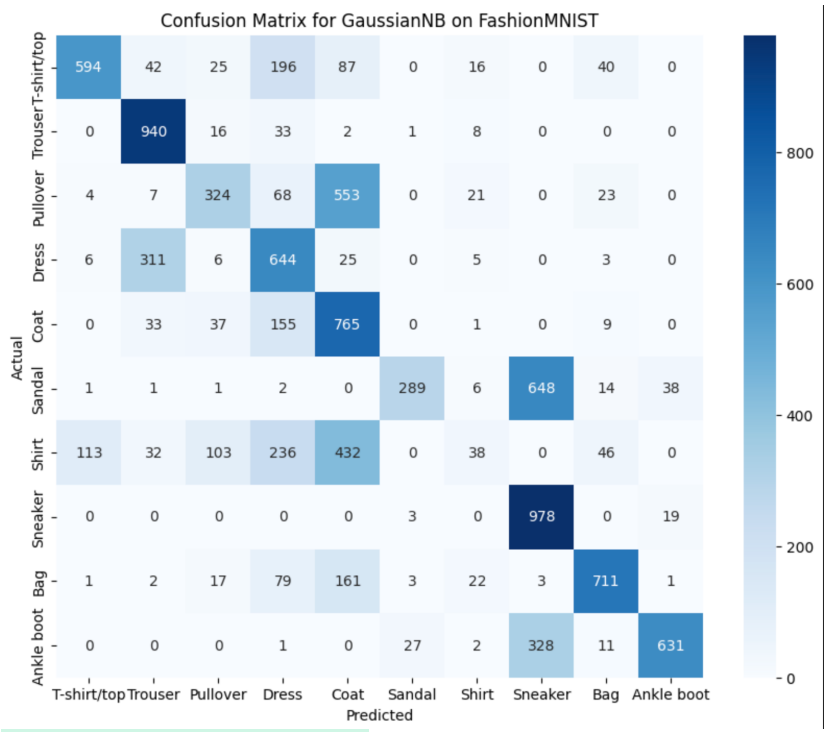
Gaussian Naive Bayes Classifier

	precision	recall	f1-score	support
T-shirt/top	0.83	0.59	0.69	1000
Trouser	0.69	0.94	0.79	1000
Pullover	0.61	0.32	0.42	1000
Dress	0.46	0.64	0.53	1000
Coat	0.38	0.77	0.51	1000
Sandal	0.89	0.29	0.44	1000
Shirt	0.32	0.04	0.07	1000
Sneaker	0.50	0.98	0.66	1000
Bag	0.83	0.71	0.77	1000
Ankle boot	0.92	0.63	0.75	1000
accuracy			0.59	10000
macro avg	0.64	0.59	0.56	10000
weighted avg	0.64	0.59	0.56	10000

Classification report

```
Evaluation Metrics for Gaussian Naive Bayes Classifier:
--------------------------------------
Accuracy: 0.5914
Balanced Accuracy: 0.5914
F1 Score: 0.5627387972343205
Log Loss: 14.219901360032864
```

Training Metrics



Confusion Matrix

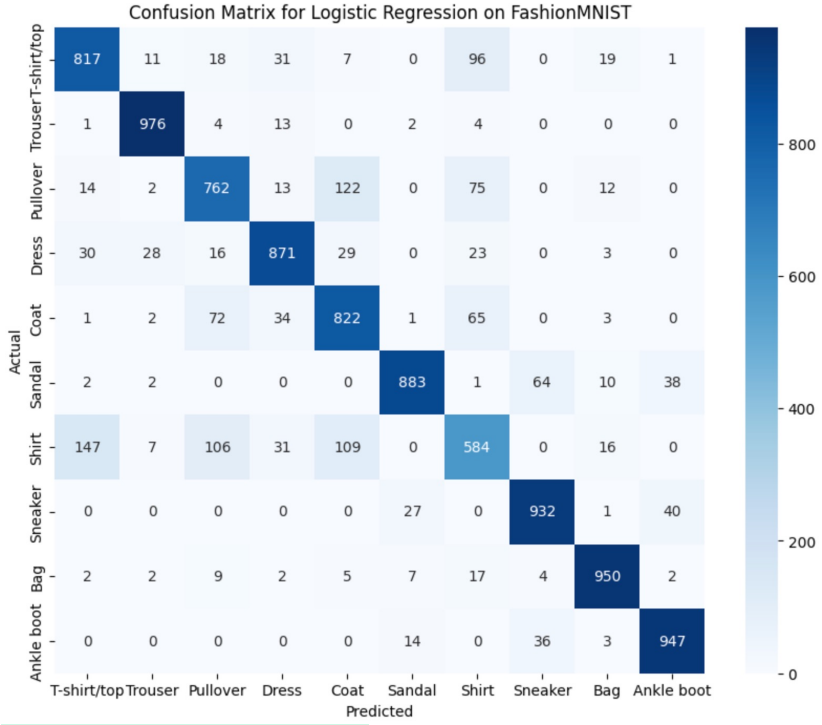
Logistic Regression

	precision	recall	f1-score	support
T-shirt/top	0.81	0.82	0.81	1000
Trouser	0.95	0.98	0.96	1000
Pullover	0.77	0.76	0.77	1000
Dress	0.88	0.87	0.87	1000
Coat	0.75	0.82	0.79	1000
Sandal	0.95	0.88	0.91	1000
Shirt	0.68	0.58	0.63	1000
Sneaker	0.90	0.93	0.92	1000
Bag	0.93	0.95	0.94	1000
Ankle boot	0.92	0.95	0.93	1000
accuracy			0.85	10000
macro avg	0.85	0.85	0.85	10000
weighted avg	0.85	0.85	0.85	10000

Classification report

```
Evaluation Metrics for Logistic Regression Classifier:
-------------------------------------
Accuracy: 0.8544
Balanced Accuracy:  0.8544
F1 Score:  0.8529011356168574
Log Loss:  0.44334444146632257
```

Training Metrics

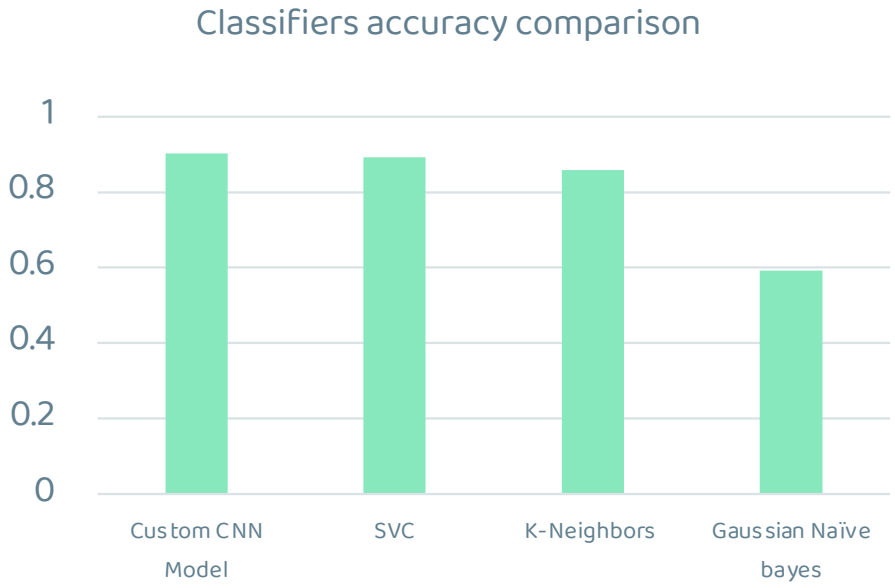


Confusion Matrix

07
Conclusion

After comparing our CNN classifier with other pretrained ones like SVC, K-Neighbors, Gaussian Naïve bayes, and the logistic regression ones
We found that ours is more accurate than them

Classifier	Accuracy
Custom CNN Model	0.9030
SVC	0.8921
K-Neighbors	0.8589
Gaussian Naïve bayes	0.5914
Logistic regression	0.8544



Thanks!