FASHION MNIST CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK (CNN)

1. OVERVIEW

This Jupyter Notebook presents a case study on classifying fashion items from the Fashion MNIST dataset using a Convolutional Neural Network (CNN). The goal is to predict the correct category of fashion items from grayscale images.

Key Objectives:

- Load and preprocess the Fashion MNIST dataset
- Visualize sample images from different classes
- Build, train, and evaluate a CNN model
- Analyze model performance using metrics like accuracy, precision, and recall

2. DATASET DESCRIPTION

Source: The dataset is available on Kaggle and is a popular benchmark for image classification tasks.

Dataset Characteristics:

- Total Samples: 70,000
- Training Samples: 60,000
- Testing Samples: 10,000
- Image Dimensions: 28x28 pixels (grayscale)
- Pixel Values: 0 (white) to 255 (black)

Classes (10 Categories): Label 0 - T-shirt/top Label 1 - Trouser Label 2 - Pullover Label 3 - Dress Label 4 - Coat Label 5 - Sandal Label 6 - Shirt Label 7 - Sneaker Label 8 - Bag Label 9 - Ankle boot

3. METHODOLOGY

Step 1: Data Loading & Preprocessing

- Libraries Used: pandas, numpy, matplotlib, seaborn, tensorflow, keras
- Data Loading: fashion_train_df = pd.read_csv("fashion-mnist_train.csv"), fashion_test_df = pd.read_csv("fashion-mnist_test.csv")
- Data Conversion: Converted to float32 arrays for training
- Normalization: Scaled pixel values to [0, 1] by dividing by 255

Step 2: Data Visualization

- Random Image Display: plt.imshow(training[i, 1:].reshape(28, 28))
- Grid Visualization: Displayed 225 random images in a 15x15 grid

Step 3: Model Architecture (CNN) The CNN model consists of:

- Convolutional Layer (Conv2D): 32 filters, kernel size (3,3), ReLU activation
- Max Pooling Layer (MaxPooling2D): Pool size (2,2)
- Flatten Layer: Converts 2D feature maps into 1D vectors
- Dense Layers (Fully Connected): 32 neurons with ReLU activation
- Output Layer: 10 neurons (one per class) with sigmoid activation

Step 4: Model Training

- Optimizer: Adam
- Loss Function: sparse_categorical_crossentropy
- Metrics: accuracy
- Training Parameters:
  - Epochs: 50
  - Batch Size: 512
  - Validation Split: 20%

Step 5: Model Evaluation

- Test Accuracy: 86.7%
- Confusion Matrix: Used to visualize true vs. predicted labels
- Classification Report:
  - Best Performing Classes: Trouser (Class 1): 97% precision & recall, Bag (Class 8): 97% precision & recall
  - Worst Performing Class: Shirt (Class 6): 66% precision, 59% recall

4. RESULTS & CONCLUSION

Key Findings:

- The model achieved 86.7% accuracy on the test set
- Best Predicted Classes: Bags, trousers, and ankle boots
- Challenging Classes: Shirts (often confused with T-shirts and pullovers)

Future Improvements:

- Advanced Architectures: Use deeper CNNs (ResNet, VGG) or transfer learning
- Data Augmentation: Rotations, flips, and brightness adjustments to improve generalization
- Hyperparameter Tuning: Adjust learning rate, batch size, and optimizer
- Multi-Label Classification: Extend to color and texture recognition (e.g., using the DeepFashion Dataset)

5. CODE STRUCTURE

Step 1 - Data Loading Description: Loading dataset from CSV files Key Functions Used: pd.read_csv()

Step 2 - Visualization Description: Displaying sample images Key Functions Used: plt.imshow(), subplots()

Step 3 - Preprocessing Description: Data preparation for training Key Functions Used: train_test_split(), reshape()

Step 4 - Model Building Description: Creating CNN architecture Key Functions Used: Sequential(), Conv2D(), MaxPooling2D(), Dense()

Step 5 - Training (model compilation and fitting) Description: Compiling and training the model Key Functions Used: model.compile(), model.fit()

Step 6 - Evaluation Description: Assessing model performance Key Functions Used: model.evaluate(), confusion_matrix(), classification_report()

6. HOW TO RUN

Dependencies: pip install pandas numpy matplotlib seaborn tensorflow keras scikit-learn Execution: Run cells sequentially in Jupyter Notebook or Google Colab. Modify hyperparameters (epochs, batch size) for experimentation.

7. REFERENCES

- Fashion MNIST Dataset: https://www.kaggle.com/datasets/zalando-research/fashionmnist
- Keras Documentation: https://keras.io/
- DeepFashion Dataset: https://mmlab.ie.cuhk.edu.hk/projects/DeepFashion.html