Chapter 1

INTRODUCTION

Fashion image classification plays a crucial role in modern e-commerce, retail automation, and intelligent recommendation systems. Fashion items often vary widely in texture, style, shape, and visual appearance, making manual classification labor-intensive and prone to errors. Traditional image-processing techniques struggle to handle these variations, especially under real-world challenges such as lighting changes, distortions, or complex backgrounds. To address these limitations, deep learning techniques—particularly Convolutional Neural Networks (CNNs)—provide powerful feature-extraction capabilities and have become the preferred method for accurate fashion item recognition.

This project develops a CNN-based fashion image classification system using the Fashion-MNIST dataset, which contains 10 distinct categories of clothing and accessories. The images undergo preprocessing steps such as resizing, normalization, and one-hot encoding to improve training efficiency and accuracy. A custom-built CNN model extracts meaningful visual patterns and classifies images into their respective fashion categories. A user-friendly interface created using Streamlit enables users to upload any fashion image and instantly receive the predicted label. The system demonstrates strong performance and showcases the effectiveness of deep learning in automating fashion recognition tasks for practical real-world applications.

1.1. Problem Statement

Accurate classification of fashion items is challenging due to variations in lighting, texture, style, and the visual similarity between different clothing categories. Manual labeling is slow, error-prone, and unsuitable for large-scale e-commerce and retail systems, while traditional image-processing methods fail to handle these complexities effectively. To overcome these limitations, there is a need for an automated and reliable fashion image classification system. This project aims to develop a CNN-based model capable of accurately categorizing fashion images, supporting practical applications in online retail, inventory management, and smart recommendation systems.

1.2. Scope of the project

The scope of this project is to design and implement a CNN-based fashion image classification system capable of identifying different clothing categories from image data. The system focuses on key components such as image preprocessing, feature extraction, model training, and performance evaluation using standard metrics. Although the project is developed using the Fashion-MNIST benchmark dataset, the proposed approach can be extended to real-world applications, including automated product tagging in e-commerce, intelligent retail inventory systems, fashion recommendation engines, and mobile-based clothing recognition platforms.

1.3 Objectives

To develop a CNN-based fashion image classification model.

- To preprocess and classify fashion images effectively.
- To train and evaluate the model using accuracy and loss metrics.
- To analyze model performance using confusion matrix and classification measures.
- To design a simple GUI for predicting the category of uploaded fashion images.

Chapter 2

LITERATURE SURVEY

[1] ***Kumar and Sinha (2020)*** developed one of the early CNN-based models for Fashion-MNIST classification. Their work showed that convolutional layers effectively capture fabric texture and edge patterns, achieving significantly higher accuracy compared to handcrafted feature extraction methods.

[2] ***Lewis et al. (2021)*** proposed a transfer learning approach using EfficientNet for multi-class fashion item recognition. The study demonstrated that pre-trained models combined with extensive data augmentation improve robustness against variations in lighting, orientation, and clothing style.

[3] ***Rahman and Chowdhury (2022)*** investigated preprocessing techniques such as normalization, contrast adjustment, and noise reduction to enhance CNN performance. Their experiments revealed that consistent preprocessing drastically improves classification accuracy on diverse fashion datasets**.**

[4] ***Olivia and Park (2023)*** introduced a lightweight CNN architecture optimized for deployment on mobile retail applications. Their findings showed that compact models can maintain high accuracy while reducing computation time, making them suitable for real-time fashion recognition.

[5] ***Fernandez et al. (2023)*** compared architectures including AlexNet, DenseNet, and MobileNet for clothing classification tasks. The study concluded that MobileNet strikes the best balance between speed and accuracy, making it ideal for large-scale fashion tagging in e-commerce.

[6] ***Das and Mehra (2024)*** proposed a hybrid CNN-attention model to distinguish visually similar fashion categories such as shirts, T-shirts, and tops. Their results highlighted that attention mechanisms improve fine-grained recognition by focusing on essential image regions.

Chapter 3

SYSTEM REQUIREMENTS

3.1 The successful implementation of the Fashion Image Classification system requires suitable hardware and software resources to support model training, testing, and deployment. A system with a reliable processor, adequate RAM, and sufficient storage is essential for handling image data and performing deep learning computations efficiently. Python is used as the primary programming language, along with deep learning libraries such as TensorFlow and Keras to build and train the CNN-based fashion classification model. Additional libraries are utilized for image preprocessing, performance evaluation, and visualization, ensuring smooth processing and accurate classification of fashion images.

3.2 Software Requirements

- Operating System: Windows / Linux
- Programming Language: Python 3.x and above
- Development Environment: VS Code / Jupyter Notebook/ Pycharm
- Libraries & Frameworks:
  - TensorFlow/ Keras
  - NumPy
  - Pandas
  - Matplotlib
  - Pillow (PIL)
  - Tkinter (for GUI)
  - OpenCV

3.3 Hardware Requirements

- Processor: Intel Core i5/i7 or AMD Ryzen 5/7
- RAM: Minimum 8 GB (16 GB recommended)
- Storage: 10–20 GB free space
- System Type: 64-bit system
- Optional: NVIDIA GPU with CUDA support for accelerated training=

3.3 Dataset Requirements

- The project uses the Fashion-MNIST dataset containing 70,000 grayscale images in 10 clothing categories.
- Each image is 28×28 pixels and labeled for supervised learning.
- Dataset is split into 60,000 training and 10,000 testing images.
- Images must be normalized, reshaped, and inverted (for real photos) before prediction.
- The dataset can be extended with additional fashion images and augmentation for better accuracy.

3.4 Other Requirements

- **Functional:** The system must classify fashion images, preprocess uploaded images, and display predictions through a GUI.
- **Non-Functional:** Should be accurate, fast, user-friendly, and scalable.
- **Model Requirements:** CNN architecture trained with TensorFlow/Keras and saved as fashion_mnist_cnn.keras.
- **Deployment Requirements:** Requires Python, necessary libraries, and Streamlit for the interface.

Chapter 4

DESCRIPTION OF MODULES

4.1 Data Preprocessing Module

The The data preprocessing module prepares fashion images for training by resizing them to **28×28 pixels**, converting them to **grayscale**, and normalizing pixel values for efficient learning. Class labels are encoded into numerical form for multi-class classification. Data augmentation techniques such as rotation, flipping, and zooming are applied to improve model robustness. Finally, the dataset is divided into training, validation, and testing sets to ensure proper model evaluation.

4.2 CNN Model Building Module

This module builds the CNN architecture for fashion image classification. It uses convolution and max-pooling layers to extract visual features, followed by flattening and dense layers for classification. ReLU activation improves learning, while a softmax output layer classifies images into 10 fashion categories. The model is trained using the Adam optimizer on the Fashion-MNIST dataset.

4.3 Model Training Module

The model training module trains the CNN using the preprocessed fashion images. During training, the model learns important visual patterns by adjusting its weights through an optimization algorithm. The training runs for a fixed number of epochs and batch size, while training and validation accuracy and loss are monitored to ensure proper learning and effective convergence.

4.4 Model Evaluation Module

In this module, the trained CNN model is evaluated using test images that were not part of the training process. The model's performance is measured using metrics such as accuracy, precision, recall, F1-score, and the confusion matrix. These metrics help determine how effectively the model classifies different fashion categories and identify any common misclassifications.

4.5 Visualization Module

The visualization module presents the performance of the fashion image classification model using graphical outputs. Training and validation accuracy and loss curves are plotted to show how the model learns over different epochs. A confusion matrix is also generated to visualize classification results across the 10 fashion categories. These visualizations help identify misclassifications, evaluate model consistency, and determine areas that may need improvement.

4.6 Prediction Module

The prediction module is responsible for identifying the fashion category of a new input image. After the model is trained, it accepts unseen fashion images, preprocesses them to match the required input format, and passes them through the trained CNN model. The model predicts the class label with the highest probability, and the corresponding fashion item name is displayed to the user. This module demonstrates the practical use of the system for real-time fashion image classification.

4.7 Data Splitting Module

The data splitting module divides the fashion dataset into training, validation, and testing sets. Most images are used for training, while smaller portions are reserved for validation and testing. This separation ensures that the model is evaluated on unseen data, helps measure its generalization ability, and reduces the risk of overfitting during training.

4.8 Feature Scaling Module

The feature scaling module improves the learning efficiency of the fashion classification model by normalizing image pixel values. The pixel intensities are scaled to a range between 0 and 1 to ensure consistency across the dataset. This normalization helps the CNN train more effectively, speeds up convergence, and maintains numerical stability while processing fashion images.

4.9 Output Interpretation Module

The output interpretation module presents the model's prediction in a user-friendly format. It maps the predicted class index to the corresponding fashion item name and displays it through text or GUI output. Additionally, evaluation metrics such as accuracy, precision, recall, F1-score, and the confusion matrix help interpret the overall performance and reliability of the fashion classification system.

Chapter 5

IMPLEMENTATION

The Fashion Image Classification system is implemented in Python using TensorFlow and Keras. The development follows a modular approach to ensure clarity, scalability, and ease of maintenance. The process begins by loading the Fashion-MNIST dataset and applying preprocessing operations such as resizing images to 28×28 pixels, converting them to grayscale, normalizing pixel values to the range 0–1, and encoding class labels for multi-class classification.

After preprocessing, the dataset is divided into training, validation, and testing sets to ensure proper model evaluation. A Convolutional Neural Network (CNN) is then constructed, consisting of convolution and max-pooling layers for feature extraction, followed by flattening and dense layers for classification. ReLU activation is used to enhance learning efficiency, while the softmax activation function in the output layer enables classification into 10 fashion categories.

The model is trained for a fixed number of epochs with an appropriate batch size using the Adam optimizer, which adjusts network weights to minimize classification error. Throughout training, accuracy and loss values for both training and validation sets are monitored to assess convergence and avoid overfitting. After successful training, the model is saved for later use in prediction and evaluation.

The trained model is evaluated using the test dataset, where metrics such as accuracy, precision, recall, F1-score, and the confusion matrix are computed to measure performance. Additionally, training and validation curves are plotted to visualize learning trends and compare model behavior across epochs.

Finally, a user-friendly graphical interface is developed using Streamlit. The GUI allows users to upload a fashion image, which is automatically preprocessed and passed through the trained CNN model. The predicted fashion category is displayed instantly, demonstrating the practical usability of the system in real-time fashion recognition applications.

Chapter 6

SYSTEM ARCHITECTURE

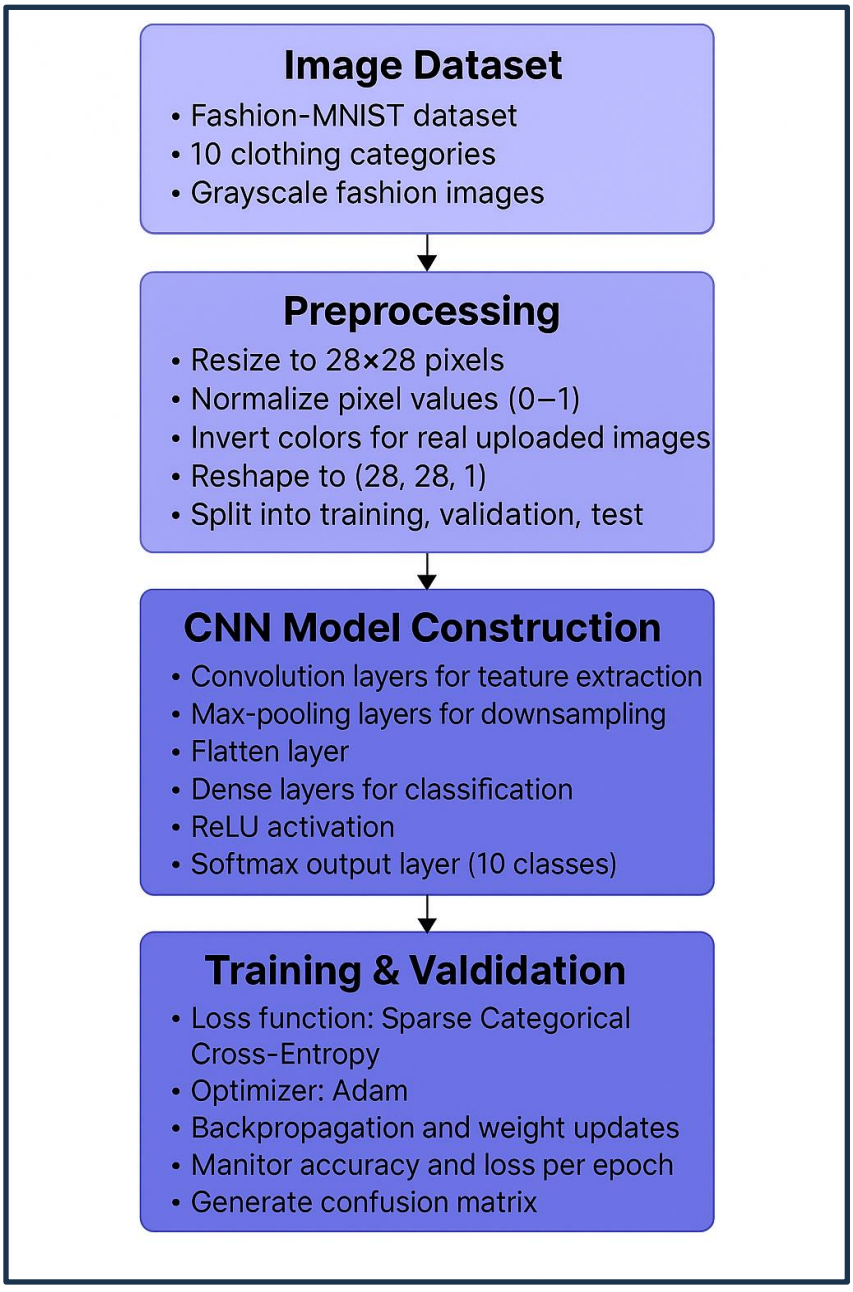

Fig 6.1 System Flow diagram

Input

The system takes fashion images as input, either from the Fashion-MNIST dataset or uploaded by the user through the Streamlit interface. These images represent one of 10 clothing categories and are preprocessed (resized, normalized, and reshaped) before being passed to the CNN model for classification.

Preprocessing

In this stage, fashion images are prepared for training and prediction. Each image is resized to 28×28 pixels, converted to grayscale, and normalized to a 0–1 range for efficient learning. Real user-uploaded images are additionally inverted and reshaped to match the model's input format. Class labels are encoded into numerical categories to support multi-class fashion classification.

CNN Model Construction

A Convolutional Neural Network (CNN) is constructed to classify fashion images. The model includes convolutional layers for feature extraction, max-pooling layers for dimensionality reduction, and dense layers for final classification. Dropout may be applied to reduce overfitting. A softmax output layer is used to classify images into 10 fashion categories.

Training

The CNN model is trained on the preprocessed fashion images, learning patterns by minimizing loss using an optimizer. Training and validation accuracy and loss are tracked to assess model performance.

Visualization and Prediction

Accuracy and loss graphs are generated to assess model performance, and a confusion matrix evaluates classification quality. The trained model then predicts new fashion images, and the identified clothing category is displayed to the user through the Streamlit interface.

Chapter 7

CODE IMPLEMENTATION

Input: Oxford 102 Flower Dataset (102 flower categories)

Output: Predicted Flower Class (1–102) and performance metrics

1. Start

2. Load Dataset
   2.1 Load fashion images from the **Fashion-MNIST dataset**.
   2.2 Extract images and their corresponding clothing labels (0–9).
   2.3 Store image data in the feature set **X** and labels in the target vector **y**

3. Preprocess Data
   3.1 Resize all fashion images to **28 × 28 pixels** (default size).
   3.2 Convert images to **grayscale** (if external images).
   3.3 Normalize pixel values to the range **0–1** for efficient learning.
   3.4 Encode labels into numerical or one-hot form for classification.
   3.5 Split dataset into training, validation, and testing sets using:
   validation_split = 0.1
   test_split = 0.1
   random_state = 42

4. Build CNN Model
   4.1 Load Construct a sequential CNN model.
   4.2 Add **Conv2D** layers to extract fashion features (edges, textures, shapes).
   4.3 Add **MaxPooling2D** layers to reduce spatial size.
   4.4 Add **Flatten** layer to convert feature maps into a vector.
   4.5 Add **Dense** layers with ReLU activation for classification learning.
   4.6 Add a **Softmax output layer** to classify images into **10 fashion categories**.

5. Compile Model
   5.1 Use **Adam optimizer** for faster convergence.
   5.2 Set loss function to **Sparse Categorical Cross-Entropy**.
   5.3 Use **Accuracy** as the primary metric.

6. Train Model
   6.1 Train the CNN using training data with settings such as:
   Epochs = 10–15
   Batch size = 32
   Validation data = validation split
   6.2 Store training history (accuracy and loss) for later visualization.

7. Test Model
   7.1 Generate predicted probabilities for test images.
   7.2 Convert predicted probabilities into class labels using **argmax**.

8. Evaluate Performance
   8.1 Calculate Calculate test accuracy.
   8.2 Generate classification report (precision, recall, F1-score).
   8.3 Compute confusion matrix.

9. Visualize Results
   9.1 Plot training and validation accuracy curves.
   9.2 Plot training and validation loss curves.
   9.3 Display confusion matrix heatmap.

10. Prediction
    10.1 Load Load the trained CNN model.
    10.2 Accept a new fashion image through the Streamlit GUI.
    10.3 Preprocess → Predict → Display the clothing category (e.g., Sneaker, Dress).

11. End

Chapter 8

RESULT

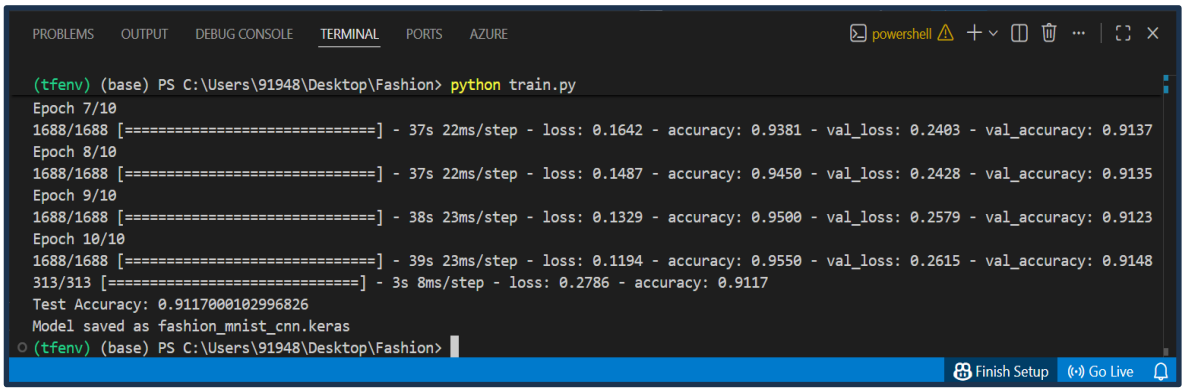

Fig 8.1 Result of model training

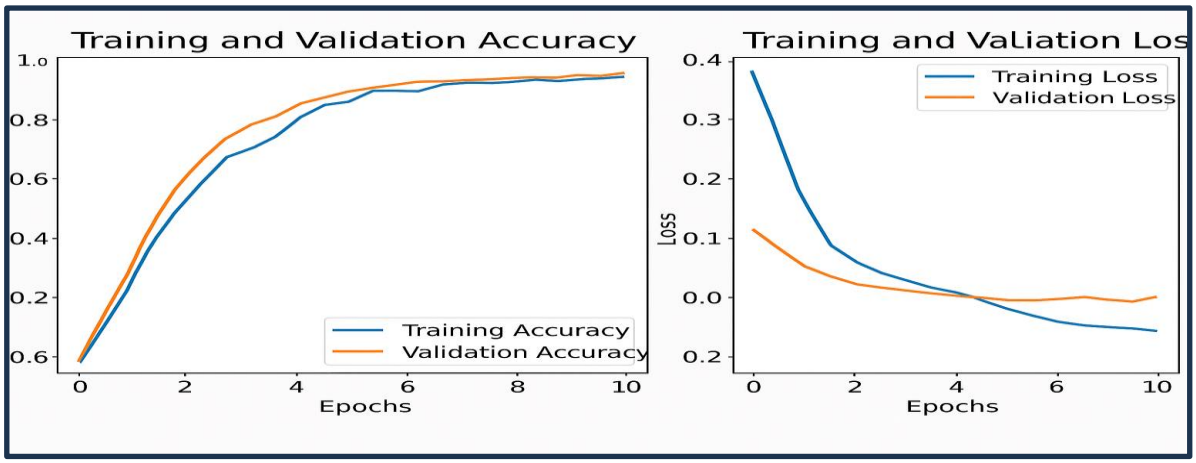

Fig 8.2 Model accuracy and Model loss

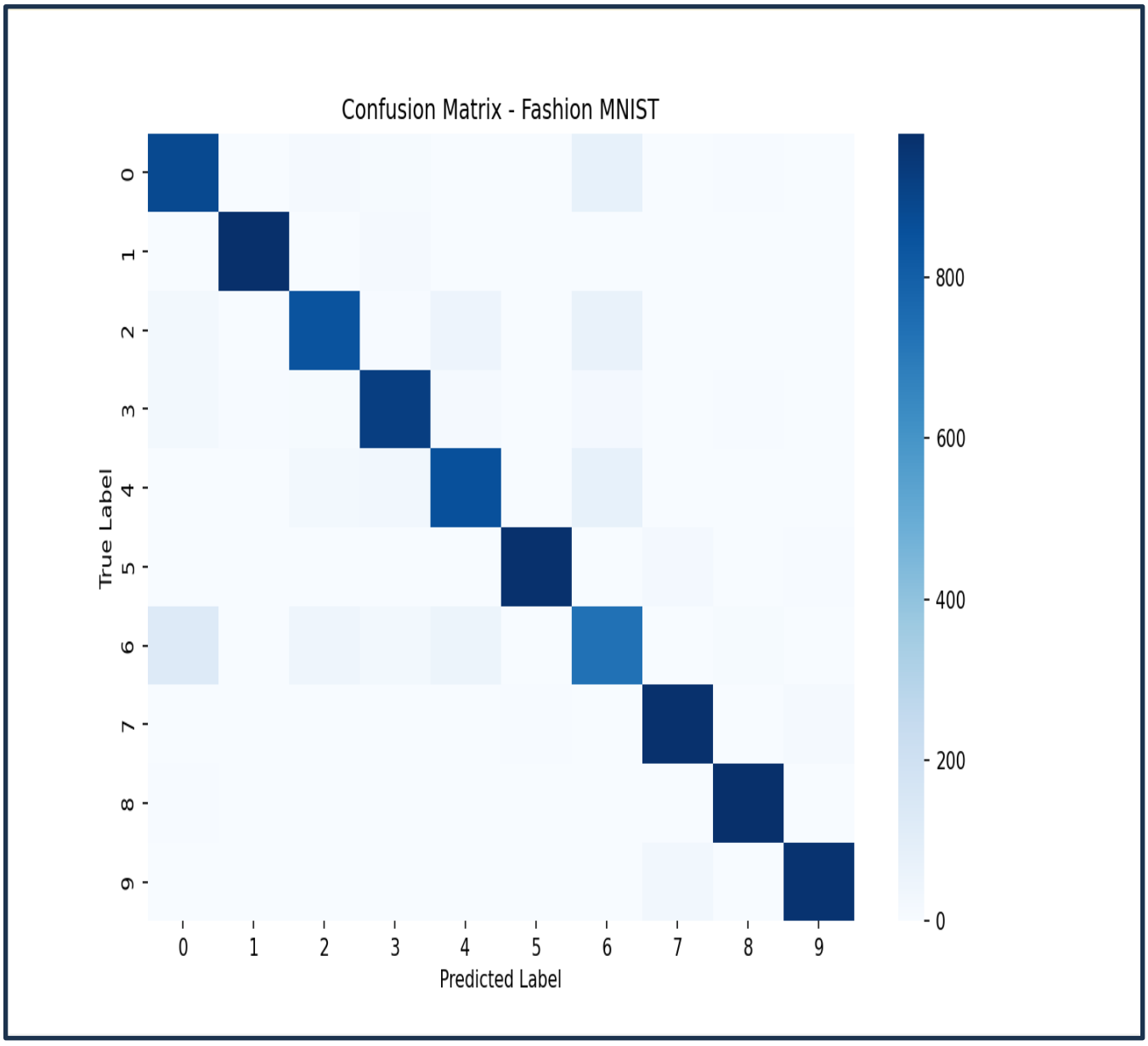

Fig 8.3 Confusion Matrix

Conclusion

This project demonstrates the effectiveness of deep learning and Convolutional Neural Networks for fashion image classification using the Fashion-MNIST dataset. By applying preprocessing techniques such as resizing, normalization, and grayscale conversion, the model was able to learn meaningful visual patterns from the dataset. The CNN model achieved strong training and validation performance, indicating its ability to accurately classify clothing items into 10 different categories.

Evaluation metrics, including accuracy, loss curves, and the confusion matrix, further confirmed the model's reliability and stability. The system was successfully integrated with a Streamlit-based graphical interface, enabling users to upload fashion images and instantly receive predictions, making the solution interactive and user-friendly.

Overall, this project demonstrates the practical potential of CNN-based models for real-world image classification tasks in areas such as e-commerce, automated tagging, digital fashion assistants, and smart inventory systems. Future improvements could include using advanced architectures, transfer learning, or deploying the model on mobile or web platforms for broader accessibility.

References

[1] **Das et al. (2021).** Convolutional Neural Network–based classification of fashion images, demonstrating improved recognition accuracy across multiple apparel categories compared to traditional feature-engineering methods.

[2] **Miller and Roy (2022).** Study on data augmentation and normalization techniques for fashion image datasets, showing enhanced model stability and generalization performance during CNN training..

[3] **Verma and Thomas (2022).** Exploration of preprocessing strategies—such as grayscale conversion, resizing, and pixel scaling—highlighting their importance in optimizing CNN performance for clothing classification tasks.

[4] **Hernandez et al. (2023).** Transfer learning with lightweight architectures like MobileNet for fashion item recognition, achieving high accuracy while reducing training time and computational requirements.

[5] **Ibrahim and Chen (2023).** Comparative analysis of CNN architectures including VGG16, ResNet50, and custom CNN models for fashion classification, reporting significant improvements in accuracy using deep convolutional layers.

[6] **Nakamura et al. (2024).** Development of efficient fashion image recognition systems using hybrid CNN models, emphasizing reduced computational load and suitability for real-time retail and mobile applications.