**Fashion MNIST Image Classification Report**

1. **Introduction**

This report documents the implementation of a Fashion MNIST image classification project. It covers data preprocessing, feature engineering, model selection, optimization, deployment strategy, and API usage.

**2. Data Preprocessing & Feature Engineering**

The dataset consists of Fashion MNIST images in CSV format, with separate training and testing splits. The following preprocessing steps were performed:

1. **Exploratory Data Analysis (EDA**) – Understanding class distribution and identifying any missing or inconsistent data.
2. **Normalization** – Scaling pixel values to the range [0, 1] to enhance model performance.
3. **Conversion** – Transformed grayscale images to RGB format (3 channels) for compatibility with pre-trained models like MobileNetV2.
4. **Resizing** – Changed image dimensions to 32×32 pixels.
5. **Data Augmentation** – Applied transformations such as rotation, flipping, zooming, and shifting to increase dataset variability and improve generalization.
6. **Dataset Splitting** – Ensuring a balanced split of training, validation, and test sets for robust evaluation.

**3. Model Selection & Optimization Approach**

For classification, we selected **MobileNetV2**, a lightweight and efficient **Convolutional Neural Network (CNN)**. The model was fine-tuned using **TensorFlow/Keras** with the following techniques:

1. **Pre-trained Weights** – Leveraging MobileNetV2’s ImageNet-trained weights to accelerate convergence.
2. **Early Stopping** – Monitoring validation loss to prevent overfitting and stop training at the optimal point.
3. **Learning Rate Scheduling** – Dynamically adjusting the learning rate to improve stability.
4. **Optimizer** – **Adam** optimizer used for faster and adaptive learning.
5. **Evaluation Metrics** – Accuracy, precision, recall, and confusion matrix were used to assess model performance.

**4. Deployment Strategy & API Usage Guide**

**Here I did Deployment with FastAPI**

The trained MobileNetV2 model was deployed using **FastAPI** to serve real-time predictions. The steps involved:

1. **Building a FastAPI Application** – Creating a /predict endpoint to accept image input and return class predictions.
2. **Model Loading** – The trained model is loaded using TensorFlow/Keras for inference.
3. **Basic Authentication** – Secured the API with authentication mechanisms.
4. **Testing the API** – UsingCURL to send image requests and verify responses.
5. **Containerization with Docker** – Created a Docker container for easy deployment.

**5. Conclusion**

This project successfully implemented Fashion MNIST classification using MobileNetV2. Through efficient preprocessing, model fine-tuning, and API deployment.