**Image Classification Project Report**

**Part 1: Dataset Overview**

For this project, we used the **Fashion MNIST** dataset, a collection of grayscale images representing 10 different clothing categories. Each image is 28x28 pixels, and the dataset is commonly used for benchmarking machine learning models.

* **Training Set:** 60,000 images
* **Test Set:** 10,000 images
* **Categories:** T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot

**Part 2: Data Preprocessing & Feature Engineering**

To prepare the data for model training, we performed the following steps:

1. **Rescaling:** Pixel values were normalized to the range [0, 1] to improve model performance.
2. **Reshaping:** Since Fashion MNIST images are grayscale, they were reshaped to include a channel dimension (28x28x1).
3. **Data Augmentation:** We applied transformations such as rotation, zoom, and horizontal flips to improve model generalization.

**Part 3: Model Architecture and Training Process**

We selected **MobileNetV2** as our base model due to its efficiency and strong performance on image classification tasks. The key steps in our model development were:

1. **Loading Pre-trained Model:** We loaded **MobileNetV2** as a feature extractor with include\_top=False to exclude the original classification layer.
2. **Custom Model Head:**
   * Flatten Layer
   * Dense Layer with 128 neurons (ReLU activation)
   * Output Layer with 10 neurons (Softmax activation)
3. **Compilation:** The model was compiled with the following configurations:
   * Optimizer: Adam (learning rate 0.001)
   * Loss Function: Sparse Categorical Crossentropy
   * Metrics: Accuracy
4. **Training:**
   * Batch Size: 32
   * Epochs: 10

**Part 4: Deployment Strategy and API Usage Guide**

We deployed the model using **FastAPI** and containerized the application using **Docker**.

**Steps for Deployment:**

1. **Creating a FastAPI Application:**
   * Defined API routes using FastAPI.
   * Used Swagger UI for easy testing and documentation.
2. **Dockerization:**
   * Created a Dockerfile with the necessary configurations:
     + Base Image: python:3.9-slim
     + Installed dependencies via requirements.txt
     + Exposed port 8000 for API access
3. **Building the Docker Image:**

docker build -t image-classification-api .

1. **Running the Container:**

docker run -p 8000:8000 image-classification-api

1. **Accessing the API:**
   * Navigate to http://localhost:8000/docs to access the Swagger UI.

**Part 5: Sample API Response**

When a sample image (sample\_image.png) was submitted to the API, the following response was generated:

{

"predicted\_class": 3

}

**Part 6: GitHub Repository and Code Documentation**

All code is organized and well-commented in the project repository. The repository includes:

* **app.py**: Contains FastAPI application code.
* **model.py**: Model architecture and training logic.
* **Dockerfile**: Configuration for building the Docker image.
* **requirements.txt**: List of necessary dependencies.
* **README.md**: Detailed instructions for running the project.

**Conclusion**

This project demonstrates effective image classification using Fashion MNIST, showcasing data preprocessing, model training, deployment via FastAPI with Swagger UI, and containerization with Docker. The clear structure and documentation ensure that the project is easily reproducible and accessible for future improvements.