# Instruction Document to Run CNN Model for Fashion Product Classification

## 1. Environment Setup

Ensure you have the required environment and dependencies installed.  
  
1. Install Python: Python 3.8+ is recommended.  
2. Install Required Libraries:  
 pip install tensorflow matplotlib scikit-learn numpy pandas  
  
3. Enable GPU (Optional but Recommended):  
 - Use Google Colab, AWS, or a local machine with GPU support.  
 - For Colab, enable GPU: Runtime > Change Runtime Type > Hardware Accelerator > GPU.

## 2. Dataset Preparation

1. Organize Dataset:  
 Place your dataset in the following directory structure:  
   
 dataset/  
 ├── Class1/  
 │ ├── image1.jpg  
 │ ├── image2.jpg  
 ├── Class2/  
 │ ├── image1.jpg  
 │ ├── image2.jpg  
  
2. Upload Dataset:  
 If using Google Colab, upload your dataset as a .zip file and extract it:  
   
 from zipfile import ZipFile  
 zip\_path = "/content/dataset.zip"  
 extract\_path = "/content/dataset"  
   
 with ZipFile(zip\_path, 'r') as zip\_ref:  
 zip\_ref.extractall(extract\_path)  
   
 Alternatively, mount your Google Drive:  
 from google.colab import drive  
 drive.mount('/content/drive')  
 dataset\_dir = "/content/drive/My Drive/path\_to\_dataset"

## 3. Running the Script

Copy and paste the following script into a Python file (e.g., train\_cnn.py) or a Jupyter Notebook.

import tensorflow as tf  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
import matplotlib.pyplot as plt  
  
# Step 1: Load Dataset  
dataset\_dir = "path\_to\_your\_dataset" # Update this path  
  
datagen = ImageDataGenerator(  
 rescale=1.0 / 255, # Normalize pixel values  
 validation\_split=0.2 # Split data into training and validation  
)  
  
train\_generator = datagen.flow\_from\_directory(  
 dataset\_dir,  
 target\_size=(128, 128),  
 batch\_size=32,  
 class\_mode="categorical",  
 subset="training"  
)  
  
val\_generator = datagen.flow\_from\_directory(  
 dataset\_dir,  
 target\_size=(128, 128),  
 batch\_size=32,  
 class\_mode="categorical",  
 subset="validation"  
)  
  
# Step 2: Define the CNN Model  
model = Sequential([  
 Conv2D(32, (3, 3), activation='relu', input\_shape=(128, 128, 3)),  
 MaxPooling2D((2, 2)),  
 Dropout(0.25),  
 Conv2D(64, (3, 3), activation='relu'),  
 MaxPooling2D((2, 2)),  
 Dropout(0.25),  
 Conv2D(128, (3, 3), activation='relu'),  
 MaxPooling2D((2, 2)),  
 Dropout(0.25),  
 Flatten(),  
 Dense(128, activation='relu'),  
 Dropout(0.5),  
 Dense(train\_generator.num\_classes, activation='softmax') # Number of classes  
])  
  
# Step 3: Compile the Model  
model.compile(  
 optimizer='adam',  
 loss='categorical\_crossentropy',  
 metrics=['accuracy']  
)  
  
# Step 4: Train the Model  
history = model.fit(  
 train\_generator,  
 validation\_data=val\_generator,  
 epochs=10,  
 steps\_per\_epoch=len(train\_generator),  
 validation\_steps=len(val\_generator)  
)  
  
# Step 5: Save the Model  
model.save("fashion\_cnn\_model.h5")  
print("Model saved as 'fashion\_cnn\_model.h5'.")  
  
# Step 6: Evaluate the Model  
val\_loss, val\_acc = model.evaluate(val\_generator)  
print(f"Validation Accuracy: {val\_acc:.2f}")  
  
# Step 7: Visualize Training Progress  
plt.figure(figsize=(12, 4))  
plt.subplot(1, 2, 1)  
plt.plot(history.history['accuracy'], label='Training Accuracy')  
plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')  
plt.legend()  
plt.title('Training and Validation Accuracy')  
  
plt.subplot(1, 2, 2)  
plt.plot(history.history['loss'], label='Training Loss')  
plt.plot(history.history['val\_loss'], label='Validation Loss')  
plt.legend()  
plt.title('Training and Validation Loss')  
plt.show()

## 4. Evaluate and Test the Model

1. Load the Saved Model:  
 from tensorflow.keras.models import load\_model  
 model = load\_model("fashion\_cnn\_model.h5")  
  
2. Test with New Images:  
 from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array  
 import numpy as np  
  
 def predict\_image(image\_path):  
 img = load\_img(image\_path, target\_size=(128, 128))  
 img\_array = img\_to\_array(img) / 255.0  
 img\_array = np.expand\_dims(img\_array, axis=0)  
  
 prediction = model.predict(img\_array)  
 predicted\_class = np.argmax(prediction, axis=1)[0]  
 class\_labels = list(train\_generator.class\_indices.keys())  
 print(f"Predicted Class: {class\_labels[predicted\_class]}")  
  
 predict\_image("path\_to\_test\_image.jpg")

## 5. Tips for Better Results

1. Optimize Hyperparameters:  
 - Increase epochs (e.g., 20–50) for better accuracy.  
 - Tune learning rate, batch size, and dropout values.  
  
2. Use Transfer Learning:  
 - Replace the CNN model with a pre-trained model (e.g., VGG16, ResNet) for better performance.  
  
3. Monitor Training:  
 - Watch for overfitting: If validation accuracy plateaus or drops, use techniques like:  
 - Early stopping.  
 - Data augmentation.