Image Classification Using Convolutional Neural Network

INDEX

Image Classification Using CNN

1. Introduction

Fashion image classification plays an important role in e-commerce platforms, enabling automatic tagging, recommendation systems, and visual search. This project implements a Convolutional Neural Network (CNN) to classify fashion images into different clothing categories and deploys the trained model using a Streamlit web application.

2. Problem Statement

Manual classification of fashion products is time-consuming and error-prone.
The goal is to automatically classify fashion images into their respective categories using deep learning techniques.

3. Objectives

- To build a CNN-based image classification model
- To train the model on a real-world fashion dataset
- To save the trained model and class labels
- To develop a user-friendly web interface using Streamlit
- To predict fashion categories from uploaded images

4. Dataset Description

The dataset used is the Myntra Fashion Dataset, consisting of:

- Fashion product images (`.jpg`)
- Metadata stored in `styles.csv`

Dataset: https://www.kaggle.com/datasets/paramaggarwal/fashion-product-images-small

5. Tools & Technologies Used

Component	Technology
Programming Language	Python 3.10
Deep Learning	TensorFlow/Keras
Image Processing	PIL, OpenCV
Data Handling	NumPy, Pandas
Model Training	CNN
Label Encoding	Scikit-learn
Web Interface	Streamlit

6. System Architecture

Image Dataset → Preprocessing → CNN Model → Model Training

↓

Saved Model + Labels

↓

Streamlit Application

↓

Prediction Output

7. Model Design & Implementation

The CNN model is built using `Sequential` API:

- Conv2D (32 filters)
- MaxPooling
- Conv2D (64 filters)
- MaxPooling
- Conv2D (128 filters)
- MaxPooling
- Flatten
- Dense (128 units)
- Output Dense layer (Softmax)

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(len(label_encoder.classes_), activation='softmax'))

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

8. Training Process

Data Preparation

- Image resizing to 150×150
- Normalization using `rescale=1./255`
- Data augmentation:
- Horizontal flip
- Zoom
- Shear

Data Splitting

- 80% Training
- 20% Validation
- 20% Testing (from training data)

Training

```
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=validation_generator
)
```

9. Model Saving & Label Encoding

```
# Save the model
model.save("model.h5")

# Save the label encoder classes
np.save("label.npy", label_encoder.classes_)
```

10. Application Development (Streamlit)

The web application allows users to:

- Upload an image
- View the uploaded image
- Receive predicted fashion category

Features:

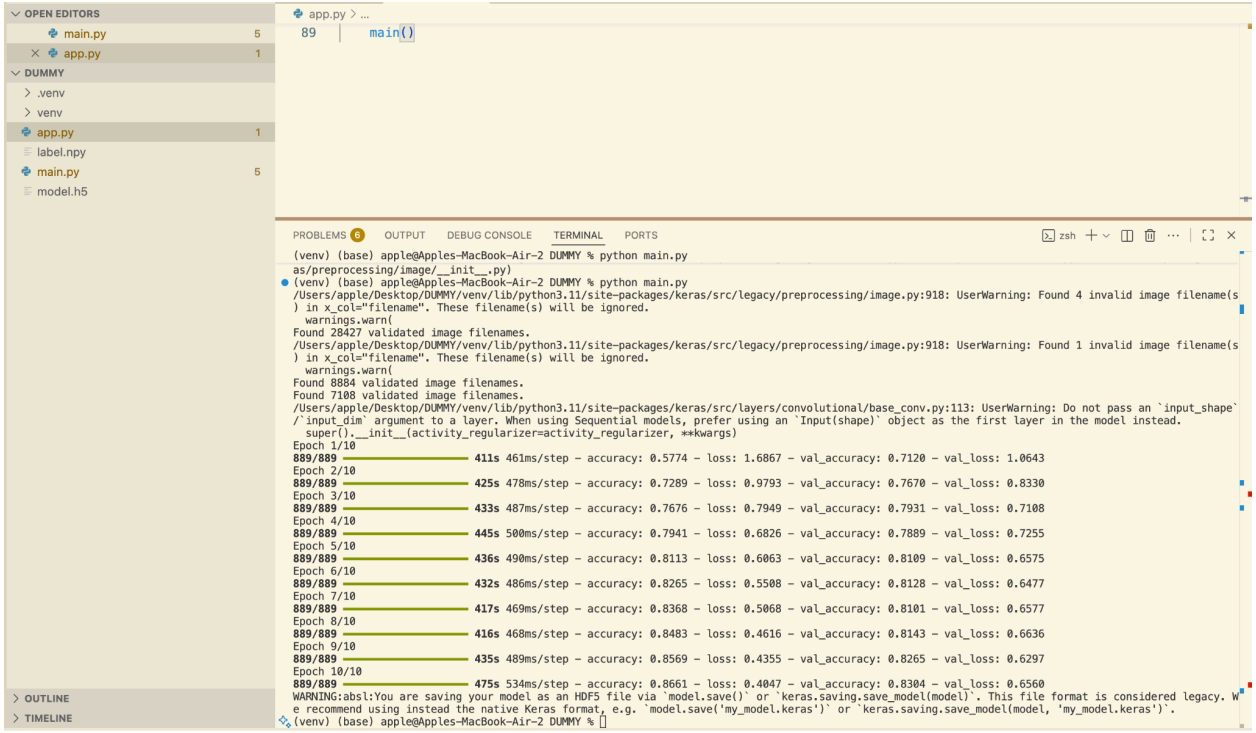
- Custom background image
- Styled UI
- Real-time prediction

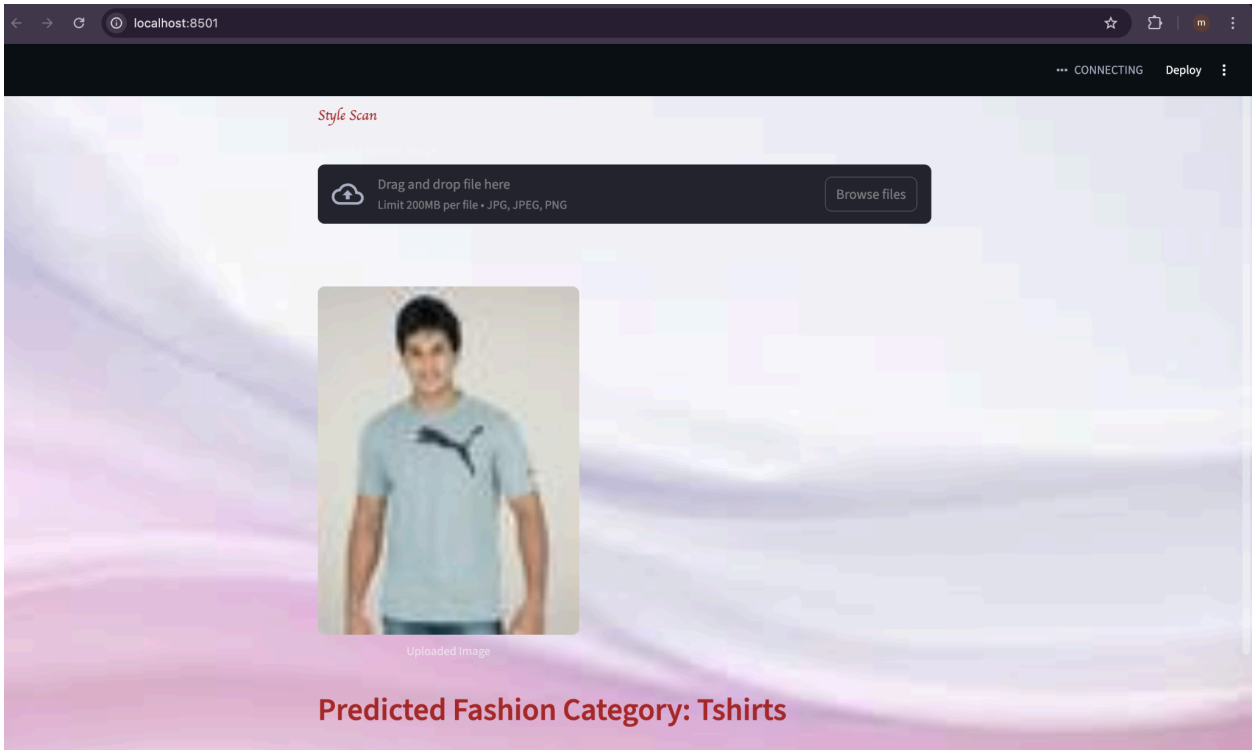
11. Prediction Workflow

1. User uploads an image
2. Image is resized to **150×150**
3. Pixel values normalized
4. Image passed to CNN model
5. Softmax output generated
6. Hghest probability class selected
7. Label decoded using LabelEncoder

12. Results & Observations

- Model successfully predicts fashion categories
- Accurate classification for clear clothing images
- Works well for common apparel types
- Performance depends on image quality





13. Challenges Faced

- TensorFlow & Keras version incompatibility
- .h5 model deserialization issues
- Python 3.11 compatibility problems
- Environment dependency conflicts

14. Best Practices Followed

- Data augmentation to reduce overfitting
- Label encoding consistency
- Separate training and inference scripts
- Modular preprocessing functions
- Clean UI with Streamlit

15. Conclusion

This project successfully demonstrates how **convolutional neural networks(CNNs)** can be used to classify fashion images effectively. By integrating TensorFlow with Streamlit, the system provides an intuitive and interactive user experience. The project highlights the importance of proper environment management and model format compatibility.