Artificial Intelligence and Machine Learning Project Documentation

1. Introduction

- **Project Title:** Pattern Sense: Classifying Fabric Patterns using Deep Learning
- **Team Members:**

  Team Leader: Ch Sambasiva Rao- Model Development & Backend Integration
  Team member: Prameela Grandi – Dataset Preparation & Testing

2. Project Overview

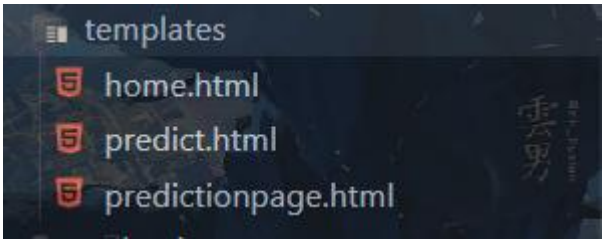
Purpose: The project aims to automate the classification of fabric patterns using a Convolutional Neural Network (CNN). This deep learning model assists designers, manufacturers, and retailers by categorizing textile images into predefined pattern categories.

Features:

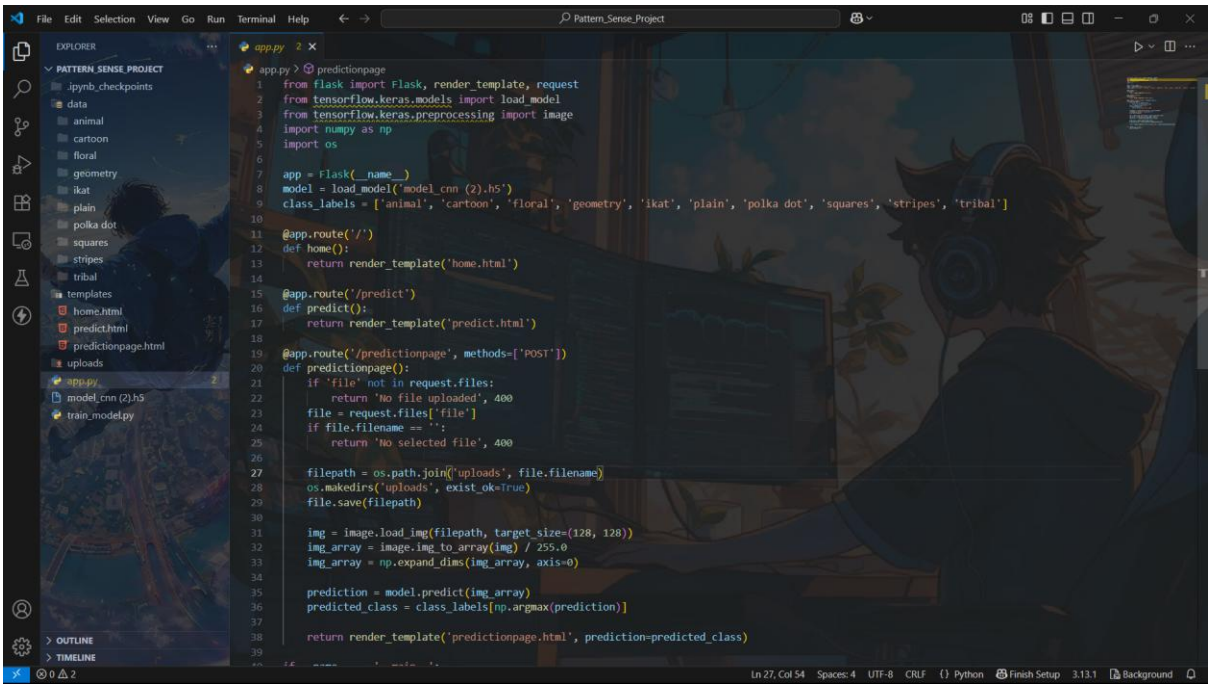
- Real-time image upload and prediction via a Flask web interface
- Trained CNN model with fine-tuning for high accuracy
- Easy preview of uploaded images
- Extendable with new classes and images

3. Architecture

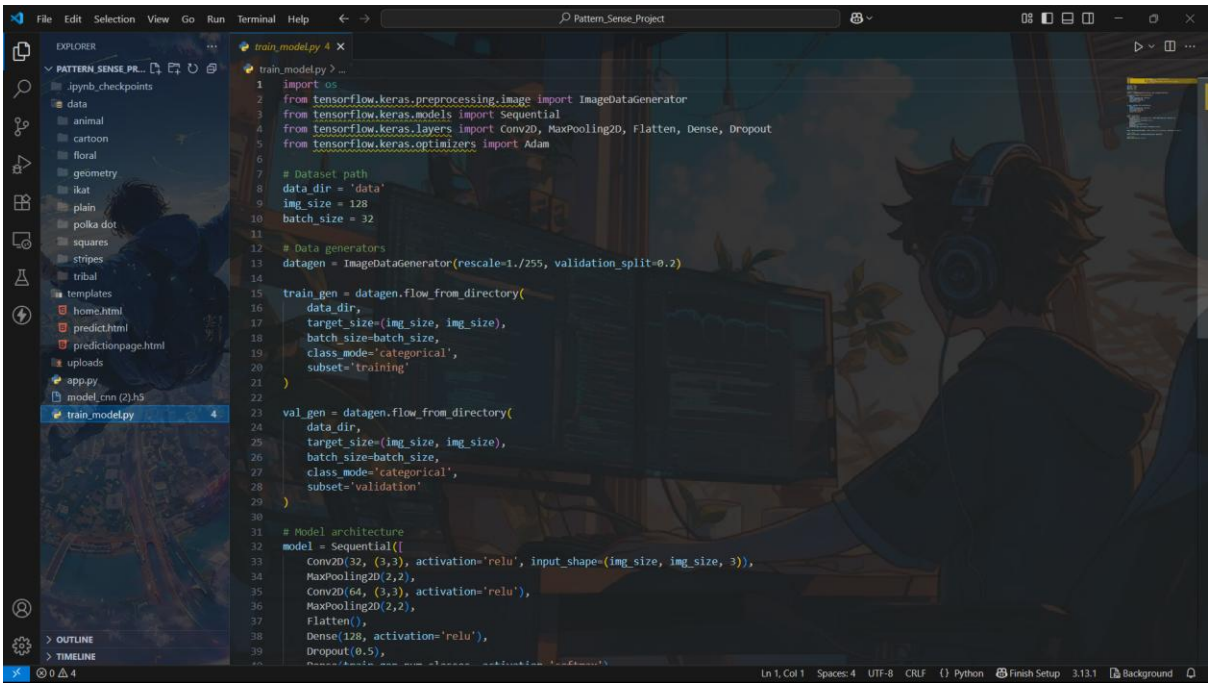
Frontend:

- Built using HTML5 and CSS3 with Jinja2 templating through Flask
- Pages: home.html, predict.html, predictionpage.html
- Image preview and result display post prediction
- 

Backend:

- Python Flask application
- app.py handles routing and model prediction
- TensorFlow CNN model loads and predicts uploaded images
- 

```python
from flask import Flask, render_template, request
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import os

app = Flask(__name__)
model = load_model('model_cnn (2).h5')
class_labels = ['animal', 'cartoon', 'floral', 'geometry', 'ikat', 'plain', 'polka dot', 'squares', 'stripes', 'tribal']

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/predict')
def predict():
    return render_template('predict.html')

@app.route('/predictionpage', methods=['POST'])
def predictionpage():
    if 'file' not in request.files:
        return 'No file uploaded', 400
    file = request.files['file']
    if file.filename == '':
        return 'No selected file', 400

    filepath = os.path.join('uploads', file.filename)
    os.makedirs('uploads', exist_ok=True)
    file.save(filepath)

    img = image.load_img(filepath, target_size=(128, 128))
    img_array = image.img_to_array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    prediction = model.predict(img_array)
    predicted_class = class_labels[np.argmax(prediction)]

    return render_template('predictionpage.html', prediction=predicted_class)
```

- 

```python
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam

# Dataset path
data_dir = 'data'
img_size = 128
batch_size = 32

# Data generators
datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

train_gen = datagen.flow_from_directory(
    data_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

val_gen = datagen.flow_from_directory(
    data_dir,
    target_size=(img_size, img_size),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)

# Model architecture
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(img_size, img_size, 3)),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
```

Database:

No persistent database is used for this version. The dataset is stored in the local file system (dataset/), structured by class labels.

4. Setup Instructions

Prerequisites:

- Python 3.8+
- TensorFlow 2.x
- Flask
- Pillow
- NumPy

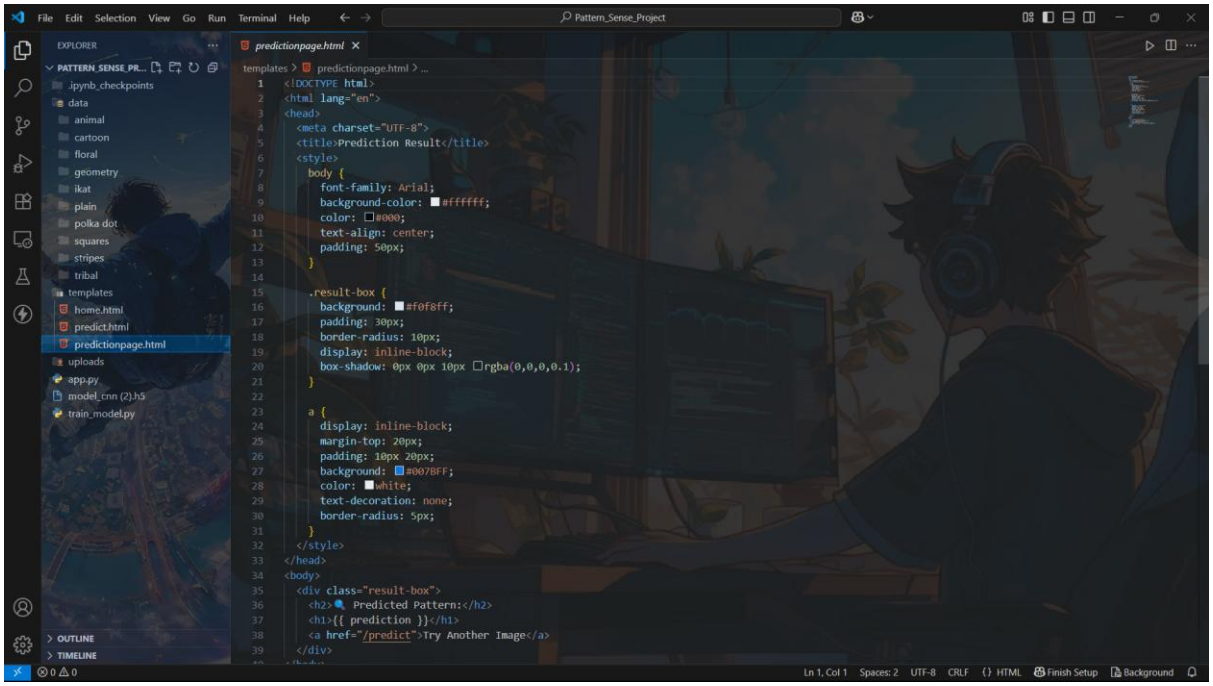
Installation:

```
# Clone the repository
git clone https://github.com/yourusername/Pattern_Sense_Project.git
cd Pattern_Sense_Project

# Create virtual environment
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt
```

5. Folder Structure

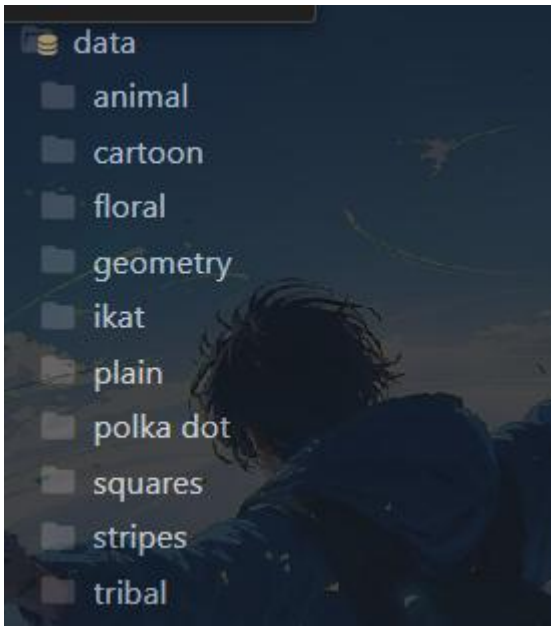
Client (templates/):

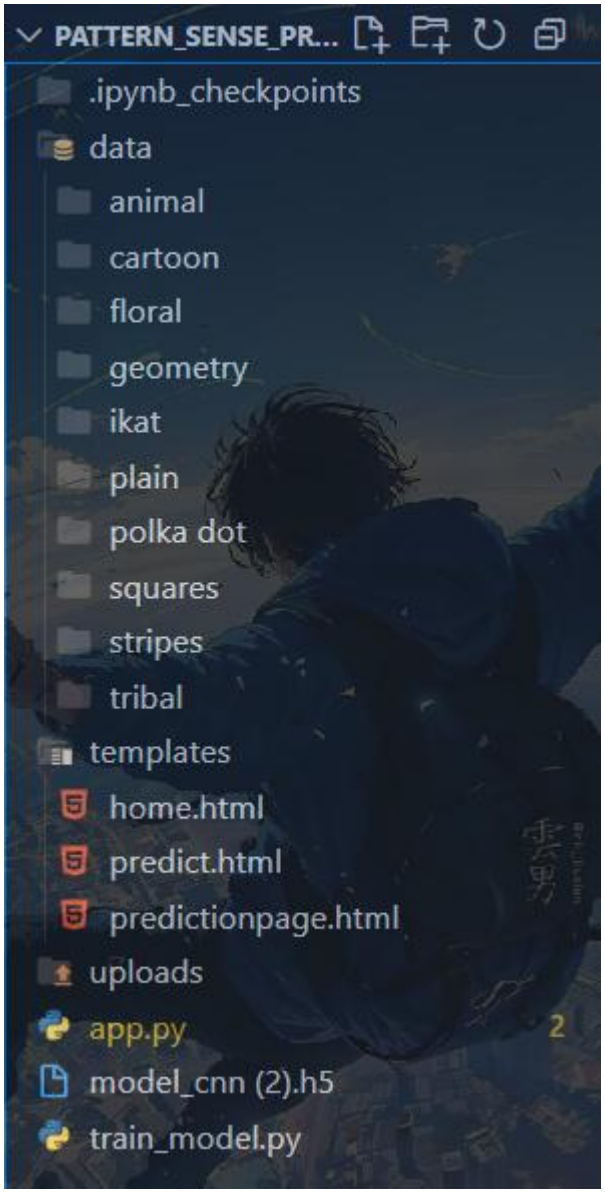
- home.html: Landing page with intro and start button
- predict.html: Upload UI for image selection
- predictionpage.html: Displays the predicted fabric pattern

- 

Server:

- app.py: Flask app with model loading and route handling
- train_model.py: CNN training and saving script
- model/: Contains trained model_cnn (2).h5
- dataset/: Training images sorted by class folders





6. Running the Application

Frontend & Backend (Flask serves both):

```
# Activate virtual environment
source venv/bin/activate

# Run Flask app
python app.py


# Access locally at
http://127.0.0.1:5000
```

7. API Documentation

Endpoints:

- GET / – Loads homepage
- GET /predict – Loads upload form
- POST /predictionpage – Accepts image file and returns predicted class

Request Example:

- POST /predictionpage with file: image.jpg

Response Example:

Predicted Class: floral

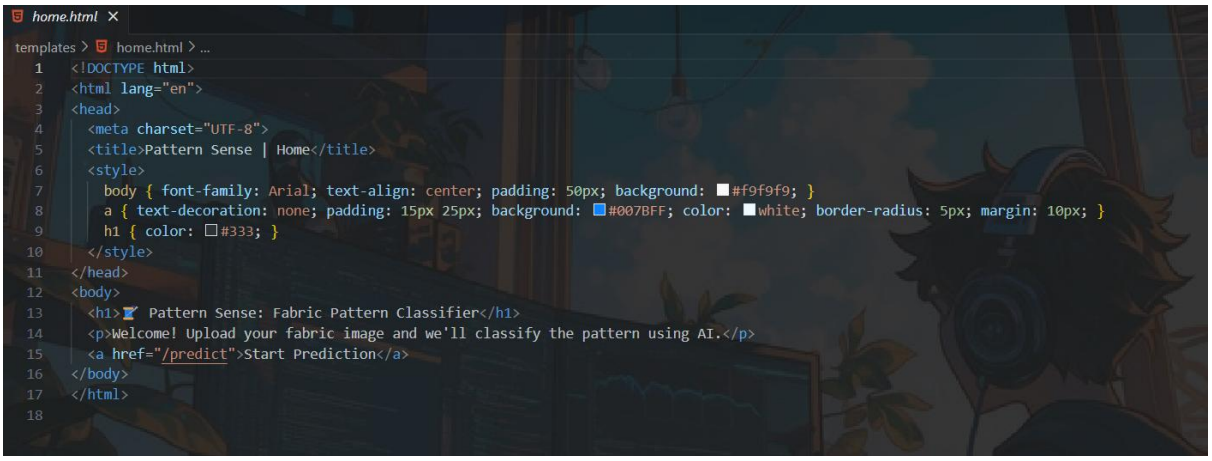
8. Authentication

- No authentication is implemented in this version. All routes are open.
- Future versions may include login and session management.

9. User Interface

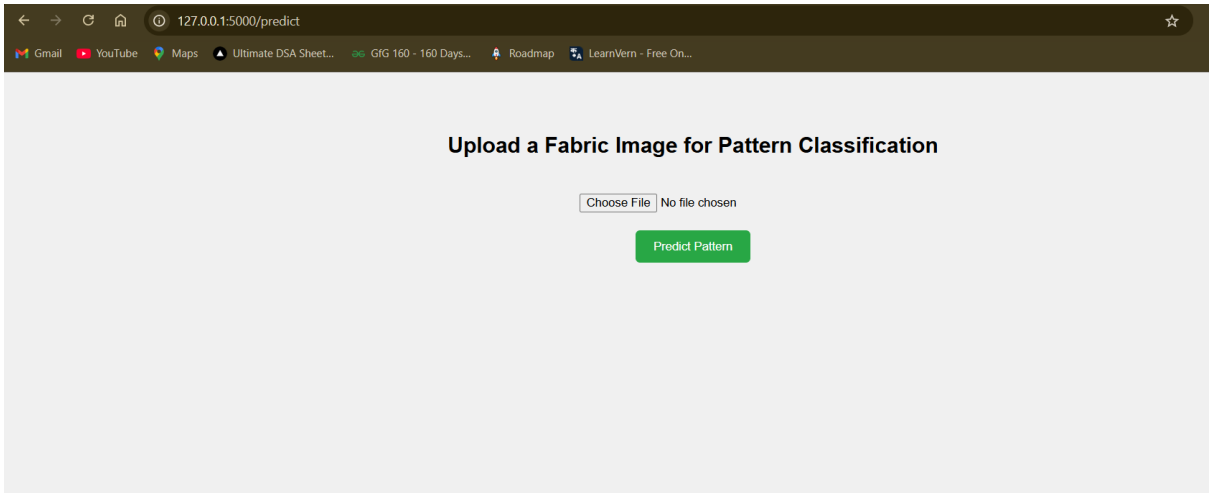
- Clean and minimal HTML interface
- Image upload button with preview
- Prediction result displayed clearly on a separate page

Screenshots:

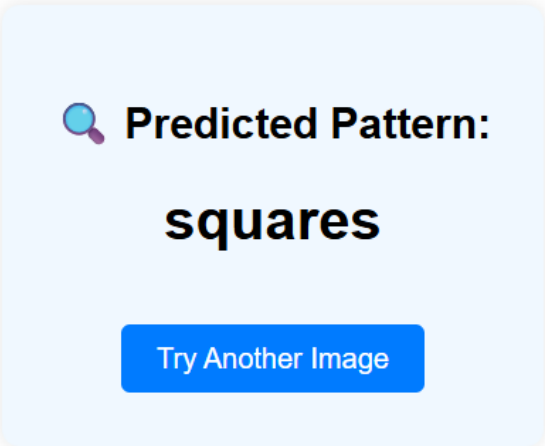
- Home page
- 
home.html ×
templates > home.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Pattern Sense | Home</title>
  <style>
    body { font-family: Arial; text-align: center; padding: 50px; background: #f9f9f9; }
    a { text-decoration: none; padding: 15px 25px; background: #007BFF; color: white; border-radius: 5px; margin: 10px; }
    h1 { color: #333; }
  </style>
</head>
<body>
  <h1>🧵 Pattern Sense: Fabric Pattern Classifier</h1>
  <p>Welcome! Upload your fabric image and we'll classify the pattern using AI.</p>
  <a href="/predict">Start Prediction</a>
</body>
</html>
```

- Upload interface

- 

- Result page with prediction



-

10. Testing

- Manual testing of different image classes
- Tested multiple edge cases: empty uploads, incorrect file types
- TensorFlow accuracy monitoring during training

11. Screenshots or Demo

• https://drive.google.com/file/d/1-StquWipy3ODTNxAXpfUbrTjzx2WLWMi/view?usp=sharing

12. Known Issues

- No error messaging for non-image uploads
- Prediction may fail for images outside trained classes

13. Future Enhancements

- Deploy the app on cloud (Heroku/AWS)
- Add authentication for different users
- Expand dataset with more fabric classes
- Integrate Grad-CAM for model explanation
- Build a mobile app version