1. INTRODUCTION

- **PROJECT TITLE:**

  Pattern Sense: Classifying Fabric Patterns Using Deep Learning

- **TEAM MEMBERS:**

1.	K. Sree Vidya Lakshmi	22HM1A0569
2.	M. Madhulatha	23HM5A0511
3.	P. Arif Hussain	23HM5A0513
4.	P. Suhaib Khan	23HM5A0512

2. PROJECT OVERVIEW

▪ PURPOSE:

The main purpose of this project is to build a smart machine learning system that can accurately recognize different fabric patterns from images. By using transfer learning, the project shows how powerful deep learning can be for image classification tasks. It also includes a simple web application made with Flask, where users can upload fabric images and get instant predictions. This project connects machine learning with real-world use, making fabric pattern identification quick, easy, and accessible. It aims to help designers, manufacturers, and retailers by saving time and improving accuracy in classifying fabrics. Pattern Sense: Classifying Fabric Patterns using Deep Learning is a project designed to automate the process of identifying and categorizing fabric patterns using advanced deep learning techniques. The system can be used in various industries such as fashion, textiles, and interior design to streamline pattern recognition tasks.

- **Scenario 1:** Fashion Industry
  In the fashion industry, designers and manufacturers often deal with a wide range of fabric patterns. Pattern Sense can automatically classify different patterns like stripes, polka dots, floral prints, and geometric designs, saving time and effort in manual categorization.
- **Scenario 2:** Textile Quality Control
  Textile companies can use Pattern Sense for quality control purposes. By analyzing fabric patterns, the system can detect any irregularities or defects in the patterns, ensuring that only high-quality fabrics are sent for production or distribution.
- **Scenario 3:** Interior Design
  Interior designers frequently work with fabric patterns for furniture, curtains, and upholstery. Pattern Sense can assist designers in quickly identifying and selecting suitable fabric patterns that match their design concepts, leading to more efficient project workflows.

▪ GOALS:

- ➢ To develop a machine learning model that can accurately identify different fabric patterns from images.
- ➢ To use pre-trained deep learning models to achieve better performance with less training time and data.
- ➢ To learn techniques like resizing, normalization, and data augmentation to prepare images for model training.
- ➢ To measure how well the model works using accuracy scores, loss curves, and confusion matrices.
- ➢ To create a user-friendly web app where users can upload fabric images and get instant predictions.

- To connect the trained model with a web interface to demonstrate practical use of AI.
- To strengthen your ability to design, build, and troubleshoot a complete machine learning pipeline.
- To understand the full process of web application development.

FEATURES:

- **Fabric Pattern Classifier**: A deep learning model that can classify multiple types of fabric patterns with high accuracy.
- **Image Preprocessing Pipeline**: Automated resizing, normalization, and augmentation of input images to improve model performance.
- **Transfer Learning Integration**: Uses powerful pre-trained CNN architectures like ResNet or VGG for efficient training and better results.
- **Real-time Predictions**: Instant classification of uploaded fabric images through the web app.
- **Interactive Web Application**: A clean and simple Flask-based web interface for easy user interaction.
- **Performance Visualization**: Plots of training and validation accuracy/loss to track model learning over time.
- **Model Evaluation Tools**: Generates confusion matrices and classification reports to understand strengths and weaknesses.
- **Error Handling**: Displays user-friendly error messages when unsupported or corrupted images are uploaded.

3. ARCHITECTURE

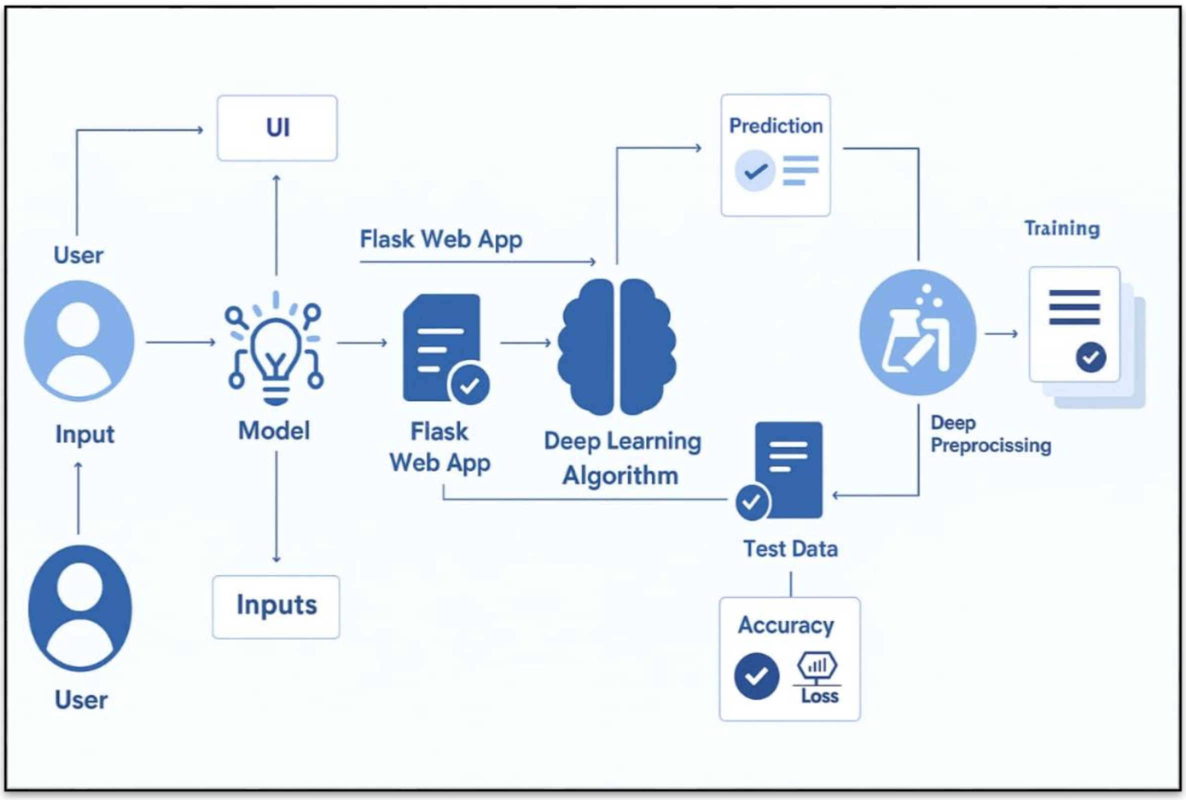


Fig: Technical Architecture

4. SETUP INSTRUCTIONS

▪ PREREQUISITES

To complete this Pattern Sense project, you must require the following:

- **Software Requirements:**
  - Google Colab
  - Visual Studio Code (VS Code) or any Python-supported IDE.
  - Python 3.11 (recommended for compatibility of all packages).

- **Python Packages Installation:**
  - Open anaconda prompt as administrator (or) VS Code terminal (or) Command Prompt
  - pip install numpy
  - pip install pandas
  - pip install tensorflow==2.14.0
  - pip install keras==2.14.0
  - pip install Flask
  - pip install pillow

▪ INSTALLATION

◈ Set Up a Python Virtual Environment (Recommended)

python -m venv venv and venv\Scripts\activate

◈ Install Required Packages

pip install flask tensorflow keras numpy pandas pillow

◈ Download or Prepare the Fabric Images Dataset

- Download the dataset using the link: https://www.kaggle.com/datasets/nguyngiabol/dress-pattern-dataset
- Place your training images and test images in different folders.

◈ Train the Model

Run your training script to train the model with your dataset: python train.py

Save the trained model as model_cnn.h5

◈ Start the Flask Web Application

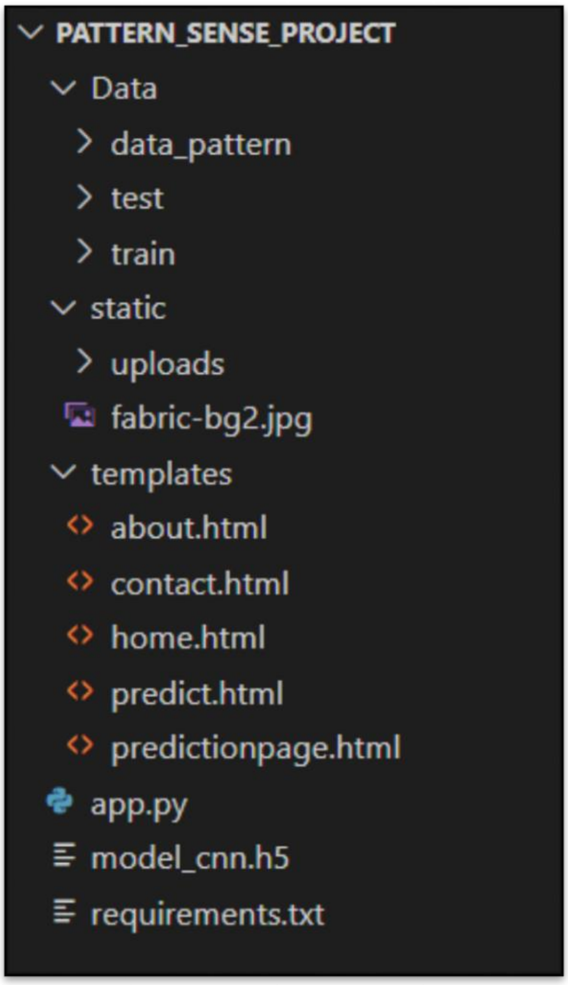
Launch your Flask app to serve predictions: python app.py

◈ Access the Web Interface

- Open your web browser and go to: http://127.0.0.1:5000/

- Upload fabric images to see the model's predictions.

5. FOLDER STRUCTURE



```
PATTERN_SENSE_PROJECT
  Data
    data_pattern
    test
    train
  static
    uploads
    fabric-bg2.jpg
  templates
    about.html
    contact.html
    home.html
    predict.html
    predictionpage.html
  app.py
  model_cnn.h5
  requirements.txt
```

6. RUNNING THE APPLICATION

- To start the frontend and backend servers, we need to give the commands as:

  - **FRONTEND:**

    The project uses Flask's **Jinja2 HTML templates** for the user interface:

    ~ homepage.html – This is the main home page for the application.

    ~ about.html – This is the webpage for the "ABOUT" section.

    ~ contact.html – This is the webpage for the "CONTACT" section.

    ~ classify.html – This is the webpage for uploading the pictures for the image classification.

    ~ predictionpage.html – This is the webpage for the prediction of the images.

  - **BACKEND:**

    - Open the Anaconda prompt from the start menu.
    - Navigate to the folder where your Python script is.
    - Now type the "python my_app.py" command.
    - Navigate to the localhost where you can view your web page.
    - It will be like: http://127.0.0.1:5000
    - You can input the image i.e., the fabric image.
    - When you click predict, it will give the 'Fabric pattern name'.

7. API DOCUMENTATION

Pattern Sense is a web application that classifies fabric patterns using a Convolutional Neural Network (CNN) model. Users can upload fabric images and get predicted pattern categories.

- **Base URL**
  http://127.0.0.1:5000/

- **Endpoints**

1. Homepage

- **URL:** /
- **Method:** GET
- **Description:** Displays the main homepage with project overview and navigation to other sections.

2. About Page

- **URL:** /about
- **Method:** GET
- **Description:** Describes the project's purpose, approach using CNNs, and its impact on textile design.

3. Contact Page

- **URL:** /contact
- **Method:** GET
- **Description:** Shows contact information and a location map.

4. Predict Page

- **URL:** /predict
- **Method:** GET
- **Description:** Displays the upload interface for image classification.

5. Predict Image Pattern

- **URL:** /predict

- **Method:** POST
- **Description:** Accepts an image file, processes it through the trained CNN model, and returns the predicted fabric pattern.

Example Request

◆ **Endpoint:**

POST /predict

◆ **Full URL (locally):**

http://127.0.0.1:5000/predict

◆ **Request Type:**

multipart/form-data

◆ **Request Body:**

```
Content-Type: multipart/form-data
image_file: <fabric_image.jpg>
```

Using **JavaScript Fetch**:

```
const formData = new FormData();
formData.append("image_file", fileInput.files[0]);
fetch("/predict", {
  method: "POST",
  body: formData
})
.then(res => res.json())
.then(data => console.log(data));
```

Example Response

- If the model predicts successfully:

```
{
  "prediction": "Striped",
  "confidence": "82.64%"
}
```

- If an error occurs (e.g., no file uploaded):

```
{
  "error": "No file selected"
}
```

- Or if the file format is invalid:

```
{
  "error": "Invalid image format"
}
```

8. AUTHENTICATION

- **CURRENT STATUS:**
  - ➢ No authentication enforced by default.
  - ➢ All routes, including /predict and /classify, are publicly accessible.
  - ➢ The application is intended for demo/testing use with a simple frontend and backend integration.
  - ➢ User login, registration, and session tracking features are not yet implemented.

- **FUTURE SCOPE**

1. **User Registration & Login System**
   - o Implement secure registration and login using Flask-Login or Flask-Security.
   - o Store hashed passwords securely using bcrypt or werkzeug.security.
2. **Session-Based Authentication**
   - o Restrict access to /predict and /classify for logged-in users only.
   - o Add session management using Flask sessions and login tracking.
3. **Role-Based Access Control**
   - o Introduce roles like "Admin" and "User" to manage different levels of access.
   - o Admins can upload new training data or view model statistics.
4. **Token-Based Authentication (API)**
   - o Use JWT tokens for secure, stateless API access (especially if used in mobile or frontend apps).
   - o Integrate OAuth2 if you expand to enterprise or multi-user support.
5. **Password Reset and Email Verification**
   - o Allow users to reset passwords and verify emails during registration for added security.

- **RECOMMENDED FUTURE FEATURES**

1. User Authentication & Roles
- ➢ Add user registration, login, logout.
- ➢ Role-based access: Admin (upload/train), User (classify/view).
- ➢ JWT or session-based protection for /predict.

2. Dataset Upload & Management

- Allow admins to upload new images and organize them into pattern categories.
- Create a dashboard to view, add, and delete training data.

3. Model Retraining Interface

- Add a section where the model can be retrained with new data via UI.
- Show training progress and final accuracy using charts (e.g., accuracy/loss graph).

4. Model Performance Dashboard

- Display real-time metrics: accuracy, confusion matrix, recent predictions.
- Use charts (e.g., with Chart.js or Recharts) to visualize data insights.

5. Feedback & Correction System

- Let users confirm if classification was correct.
- Use feedback to improve future model performance (active learning).

6. Multiple Model Support

- Compare different models (e.g., ResNet vs. custom CNN).
- Let users choose which model to use for classification.

7. Multilingual UI Support

- Add language toggle (Telugu, Hindi, etc.) for regional accessibility.

8. Responsive Design / Mobile App

- Make the UI mobile-friendly.
- Optionally, build a lightweight Android app using Flutter or React Native.

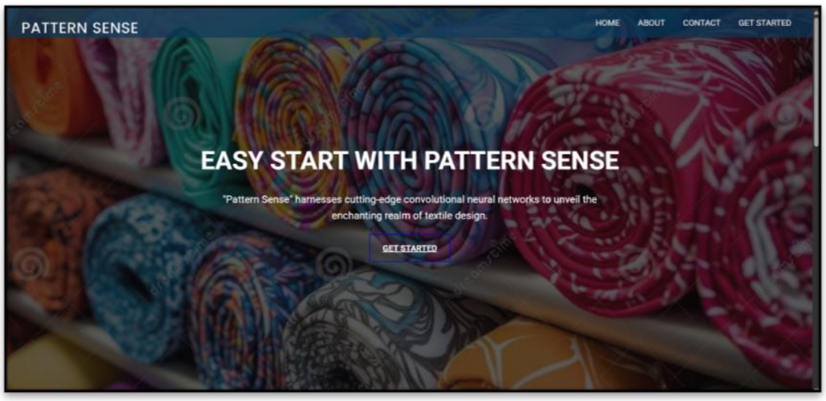
9. Secure File Handling

- Sanitize uploads, limit file types/sizes.
- Store files in cloud storage (e.g., AWS S3 or DigitalOcean Spaces).

10. Cloud Deployment & Scalability

- Deploy on platforms like Render, DigitalOcean, or AWS.
- Add GPU support for faster model inference if needed.

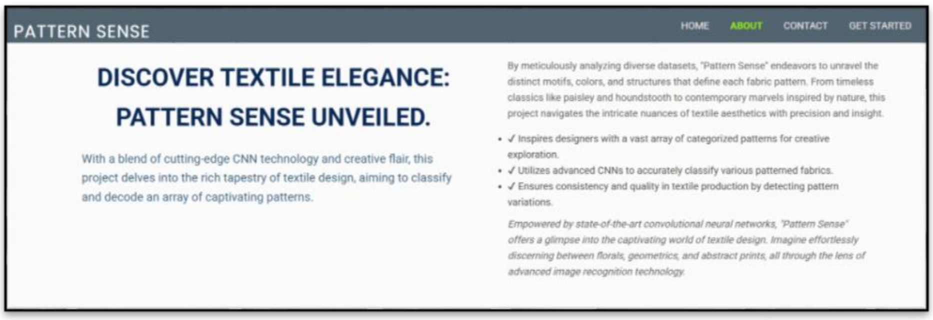
9. USER INTERFACE



EASY START WITH PATTERN SENSE

"Pattern Sense" harnesses cutting-edge convolutional neural networks to unveil the enchanting realm of textile design.

Fig: Home page



DISCOVER TEXTILE ELEGANCE: PATTERN SENSE UNVEILED.

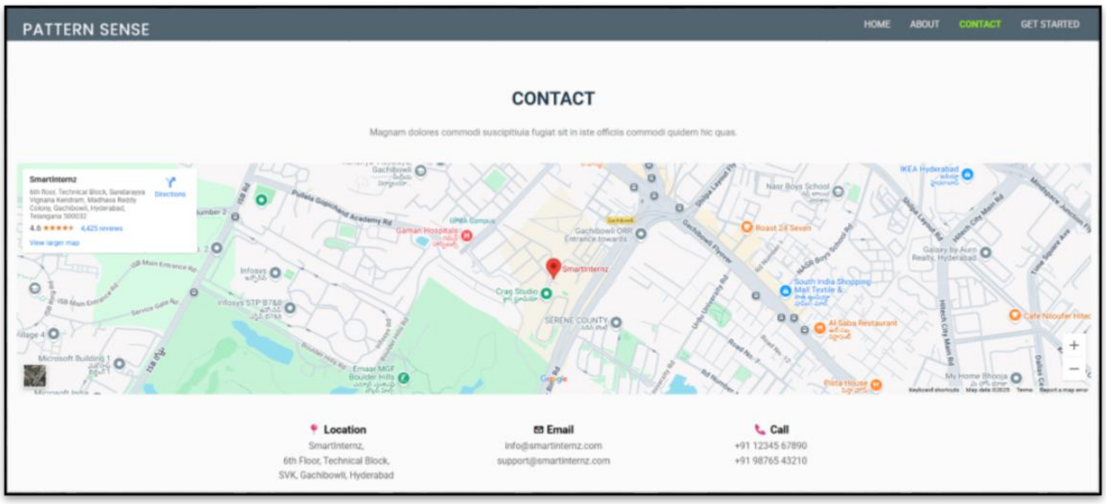
With a blend of cutting-edge CNN technology and creative flair, this project delves into the rich tapestry of textile design, aiming to classify and decode an array of captivating patterns.

By meticulously analyzing diverse datasets, "Pattern Sense" endeavors to unravel the distinct motifs, colors, and structures that define each fabric pattern. From timeless classics like paisley and houndstooth to contemporary marvels inspired by nature, this project navigates the intricate nuances of textile aesthetics with precision and insight.

- ✓ Inspires designers with a vast array of categorized patterns for creative exploration.
- ✓ Utilizes advanced CNNs to accurately classify various patterned fabrics.
- ✓ Ensures consistency and quality in textile production by detecting pattern variations.

Empowered by state-of-the-art convolutional neural networks, "Pattern Sense" offers a glimpse into the captivating world of textile design. Imagine effortlessly discerning between florals, geometrics, and abstract prints, all through the lens of advanced image recognition technology.

Fig: About page



CONTACT

Magnam dolores commodi suscipitiula fugiat sit in iste officiis commodi quidem hic quas.

Smartinternz
6th Floor, Technical Block, Sundarayya Vignana Kendram, Madhava Reddy Colony, Gachibowli, Hyderabad, Telangana 500032

Location
Smartinternz,
6th Floor, Technical Block,
SVK, Gachibowli, Hyderabad

Email
info@smartinternz.com
support@smartinternz.com

Call
+91 12345 67890
+91 98765 43210

Fig: Contact page

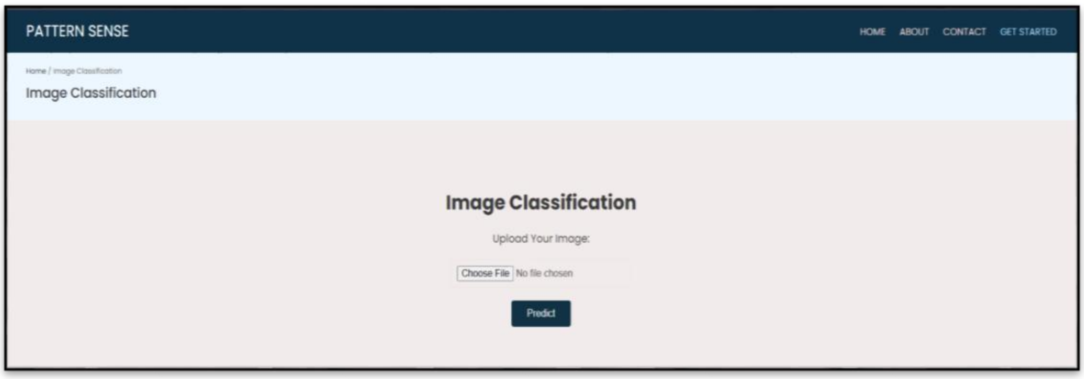


Fig: Predict page

10.TESTING

MANUAL TESTING

- **Objective:**
  The objective of manual testing was to verify the functionality and usability of the Pattern Sense web application by simulating end-user behavior without using automation tools.

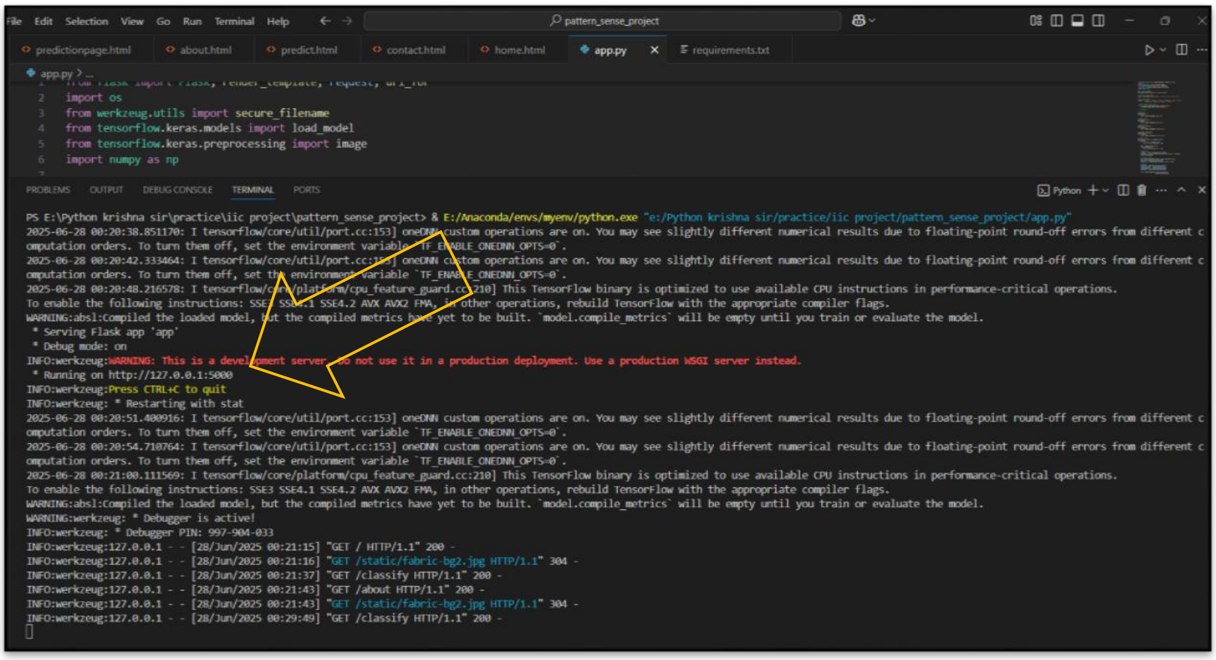
- **Scope:**
  Manual testing was performed on the Flask-based web application, particularly focusing on:
  - Image upload and classification functionality
  - Page navigation and responsiveness
  - Visual layout and design elements
  - Error handling for unsupported file types or empty submissions
  - Backend communication with the trained CNN model

- **Testing Approach:**
  The testing followed a black-box approach, where the internal code was not examined. Instead, the focus was on testing the input-output behavior and user interface interactions.

- **Test Scenarios Covered:**
  - Uploading a valid image and checking the predicted fabric pattern
  - Uploading a non-image file to verify error messages
  - Clicking on navigation links like Home, About, Contact, and Get Started
  - Verifying smooth scrolling between homepage sections
  - Checking background image load and responsiveness
  - Observing server error handling for missing files or undefined routes

- **Results**:
  - Most functionalities performed as expected.
  - Some template errors were identified and fixed.
  - All critical paths for image classification and user navigation were successfully validated through manual inputs.

11.SCREENSHOTS AND DEMO



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

```
PS E:\Python krishna sir\practice\iic project\pattern_sense_project> & E:/Anaconda/envs/myenv/python.exe "e:/Python krishna sir/practice/iic project/pattern_sense_project/app.py"
2025-06-28 00:20:38.851170: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different c
omputation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-06-28 00:20:42.333464: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different c
omputation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-06-28 00:20:48.216578: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
 * Serving Flask app 'app'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
2025-06-28 00:20:51.400916: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different c
omputation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-06-28 00:20:54.710764: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different c
omputation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-06-28 00:21:00.111569: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 997-904-033
INFO:werkzeug:127.0.0.1 - - [28/Jun/2025 00:21:15] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [28/Jun/2025 00:21:16] "GET /static/fabric-bg2.jpg HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [28/Jun/2025 00:21:37] "GET /classify HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [28/Jun/2025 00:21:43] "GET /about HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [28/Jun/2025 00:21:43] "GET /static/fabric-bg2.jpg HTTP/1.1" 304 -
INFO:werkzeug:127.0.0.1 - - [28/Jun/2025 00:29:49] "GET /classify HTTP/1.1" 200 -
```

Fig: Link generated

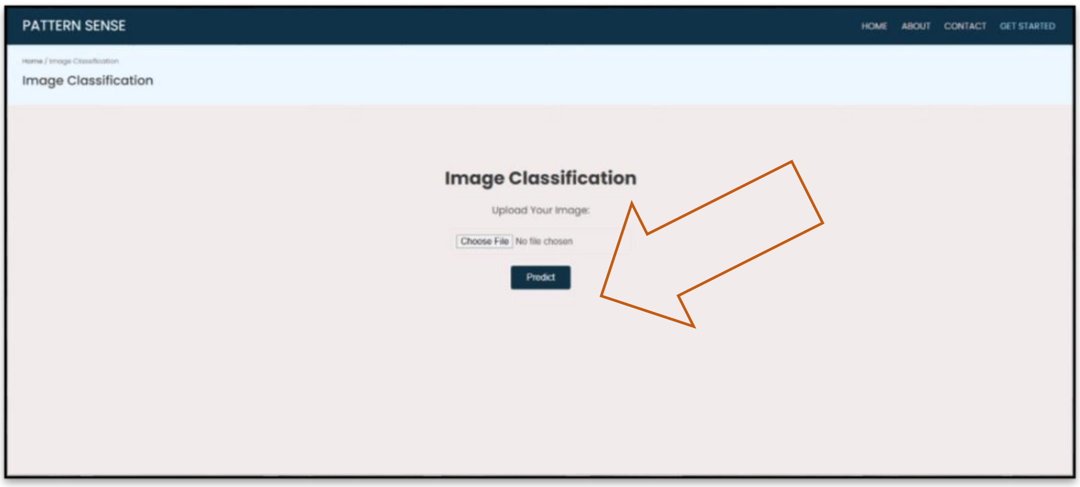
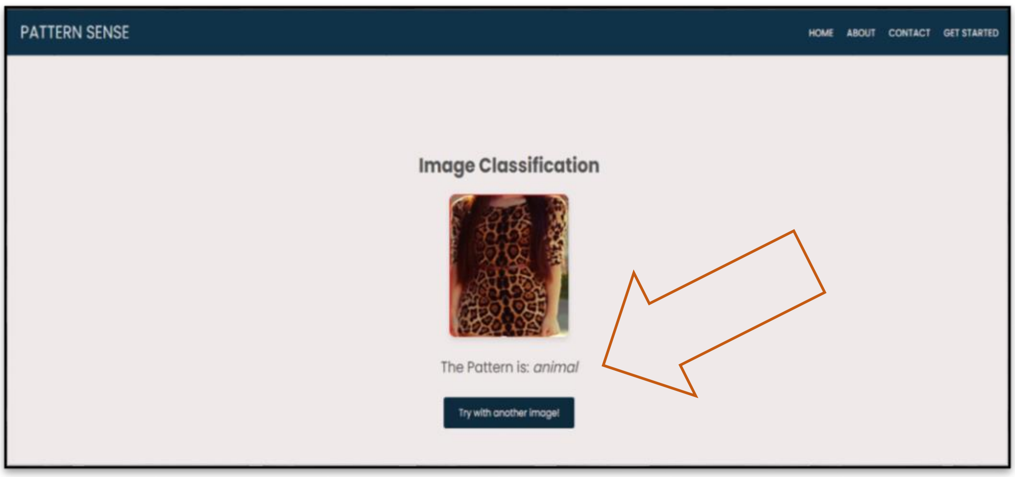


Fig: Giving input to the model



The Pattern is: *animal*

Fig: Output generated

12.KNOWN ISSUES

- **No Authentication System Implemented**
  - **Description:** The application does not require any login or authentication to access core features like image classification or uploading data.
  - **Impact:** This means anyone can access and use the system without restrictions, which is not secure for sensitive or commercial use cases.
  - **Future Plan**: A basic login system with role-based access control (admin/user) is recommended to enhance security and manage usage.

- **Limited Model Accuracy (~59%)**
  - **Description:** The deep learning model currently achieves about 59% accuracy, which may not be reliable enough for real-world textile classification tasks.
  - **Impact**: This limits the application's ability to make precise pattern predictions and could lead to misclassifications.
  - **Next Steps**: Model performance can be improved with better dataset balancing, fine-tuning, data augmentation, and exploring different architectures

- **Functional Issue: Static Image Not Displaying as Background**
  - **Description**: The homepage background image (fabric-bg2.jpg) placed in the static folder failed to display in the browser.
  - **Impact**: The UI looked incomplete or plain, reducing visual appeal and the user's first impression of the app.
  - **Cause**: Although the image existed in the correct location, the background CSS was pointing to an incorrect or mismatched URL due to templating issues or ID mismatches.
  - **Resolution**: Verified the path using {{ url_for('static', filename='fabric-bg2.jpg') }} and confirmed the image was accessible directly via browser. Also ensured that the id="home" matched the anchor tag href="#home" to enable proper loading and scrolling.

- **Data Handling Issue: Lack of Input Validation for Uploaded Images**

- **Description**: The image upload form did not restrict users from uploading unsupported or corrupted image formats.
- **Impact**: Uploading invalid files caused errors during prediction, such as failure in load_img() or unexpected crashes.
- **Cause**: Missing server-side checks for allowed file types and sizes in the Flask app.
- **Resolution**: Introduced validation logic using Python to only accept .jpg, .jpeg, or .png files. Enhanced error handling to gracefully notify the user when an invalid file is selected.

13. FUTURE ENHANCEMENTS

1. Accuracy & Performance Improvements

- Fine-tune the pretrained model: Unfreeze deeper layers of ResNet or try EfficientNet to capture complex fabric patterns better.
- Use advanced augmentations: Libraries like Albumentations offer more realistic augmentations (e.g. motion blur, random contrast).
- Add attention mechanisms: Use Squeeze-and-Excitation (SE) blocks or CBAM to help the model focus on key areas in fabric patterns.
- Ensemble models: Combine predictions from multiple CNNs for better generalization.

2. Dataset Enhancements

- Add more classes or subcategories: E.g., traditional patterns, printed vs woven, seasonal patterns, etc.
- Balance dataset: Use SMOTE or augment underrepresented classes to improve learning consistency.

3. Web App Integration

- Deploy as web app with Flask or Streamlit: Allow users to upload fabric images and get predictions instantly.
- Add batch prediction feature: Allow users to upload multiple images at once.
- Show prediction confidence graphically: Display top 3 predictions with probabilities in a bar chart.

4. Real-time Applications

- Build a mobile version: Using TensorFlow Lite for on-device prediction.
- Add image segmentation: Identify patterned regions in a larger textile image using UNet or Mask R-CNN.

5. Explainability

- Use Grad-CAM or LIME: Visualize which parts of the fabric image influenced the prediction.
- Log model metrics and versions: Use MLflow or Weights & Biases to monitor training and experiment history.