**Introduction**

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

**Team Members:**

* **Team ID:** LTVIP2025TMID42814
* **Team Size:** 4
* **Team Leader:** Narisetty Harshitha
* **Team Member:** Suraboyina Pujya Balaji
* **Team Member:** Pathan Jansaida
* **Team Member:** KasaDhanalakshmi

**Project Overview**

**Purpose:**  
The purpose of *Pattern Sense* is to automate the identification and categorization of fabric patterns using advanced deep learning techniques. The system streamlines pattern recognition tasks in in dustries such as fashion, textiles, and interior design.

**Features:**

* **Automated Pattern Classification:** Accurately classify different fabric patterns, including stripes, polka dots, floral prints, and geometric designs.
* **Quality Control:** Detect irregularities or defects in fabric patterns, ensuring high-quality production.
* **Efficient Pattern Selection:** Quickly identify and select suitable patterns to match design concepts.
* **Industry Applications:** Designed for fashion, textiles, and interior design workflows.

**Skills Required:**

* Python for implementing the model
* Data preprocessing techniques for preparing datasets
* TensorFlow for deep learning development
* Deep learning model design and fine-tuning

**Scenarios:**

**Scenario 1 – Fashion Industry:**

* Automate pattern categorization to save time and effort.
* Enhance design and manufacturing by quickly identifying suitable fabrics.

**Scenario 2 – Textile Quality Control:**

* Detect defects and irregularities in patterns.
* Improve accuracy and reduce manual inspection time.

**Scenario 3 – Interior Design:**

* Efficiently select fabric patterns matching design themes.
* Streamline project workflows.

**Technical Requirements:**

* **Programming Language:** Python
* **Deep Learning Framework:** TensorFlow
* **Data Preprocessing:** Image processing and augmentation

**Potential Benefits:**

* Increased efficiency by reducing manual work.
* Improved accuracy in pattern classification and defect detection.
* Enhanced quality control in production and distribution.

**Architecture**

The *Pattern Sense* system architecture consists of the following components:

1. **Data Preprocessing Module:**  
   Load, clean, and augment the fabric pattern dataset.
2. **Model Training Module:**  
   Build and train deep learning models using TensorFlow and Keras.
3. **Pattern Classification Module:**  
   Predict pattern classes for new fabric samples.
4. **Defect Detection Module:**  
   Identify anomalies or defects in patterns.
5. **User Interface (optional future enhancement):**  
   Interface for uploading images and viewing predictions.

**Installation and Setup using VS Code and Python**

**Prerequisites**

- Python installed on your system (preferably the latest version)

- VS Code installed on your system

- pip (Python package manager) installed on your system

**Step 1: Install Required Libraries**

Open your terminal or command prompt and install the required libraries using pip:

pip install tensorflow numpy pandas matplotlib scikit-learn

**Step 2: Create a New Project in VS Code**

1. Open VS Code.

2. Create a new folder for your project and navigate to it in the terminal/command prompt.

3. Create a new Python file (e.g., pattern\_sense.py) in your project folder.

**Step 3: Set Up Your Project Structure**

Create the following folders in your project directory:

- data: For storing your fabric pattern dataset.

- models: For storing your trained deep learning models.

- utils: For storing utility functions.

Your project structure should look like this:

pattern\_sense/

|--- data/

|--- models/

|--- utils/

|--- pattern\_sense.py

**Step 4: Implement Your Deep Learning Model**

In your pattern\_sense.py file, implement your deep learning model using TensorFlow and Keras. Here's a simple example:

import tensorflow as tf

from tensorflow import keras

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

import numpy as np

Load your dataset

Replace this with your actual dataset loading code

(X\_train, y\_train), (X\_test, y\_test) = keras.datasets.mnist.load\_data()

Normalize pixel values

X\_train = X\_train.astype('float32') / 255

X\_test = X\_test.astype('float32') / 255

Split data into training and validation sets

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train, y\_train, test\_size=0.2, random\_state=42)

**Define your deep learning model**

model = keras.Sequential([

keras.layers.Flatten(input\_shape=(28, 28)),

keras.layers.Dense(128, activation='relu'),

keras.layers.Dropout(0.2),

keras.layers.Dense(10, activation='softmax')

])

Compile your model

model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

Train your model

model.fit(X\_train, y\_train, epochs=5, validation\_data=(X\_val, y\_val))

Evaluate your model

test\_loss, test\_acc = model.evaluate(X\_test, y\_test)

print(f'Test accuracy: {test\_acc:.2f}')

Step 5: Run Your Project

Run your pattern\_sense.py file using VS Code or your terminal/command prompt:

python pattern\_sense.py

This will train your deep learning model and evaluate its performance on the test dataset.

**Step-6:running the applicartion**

By following these steps, you can set up your Pattern Sense project using VS Code and Python

Command to start the Flask Server:

Python app.py

After running the above,the application will start locally and can be accessed in your brower at:

<http://127.0.0.1:5000/>

The frontend is rendered from the /templates/ directory.

The backend handles the uploaded image and returns the prediction result using the trained model.

**Step-7:API Documentation**

The Flask backend of this butterfly classification project exposes two core endpoints that power the web interface shown in the application.

1. **/ – Homepage Endpoint**

* Method: GET
* Purpose: Loads the main interface where users can upload butterfly images.
* Parameters: None
* Response:  
  Renders index.html, which contains:
  + A file input (Choose File)
  + A submit button labeled **"**Pattren Sense**"**
  + Project title and team member details displayed on screen

1. **/predict – Prediction Endpoint**

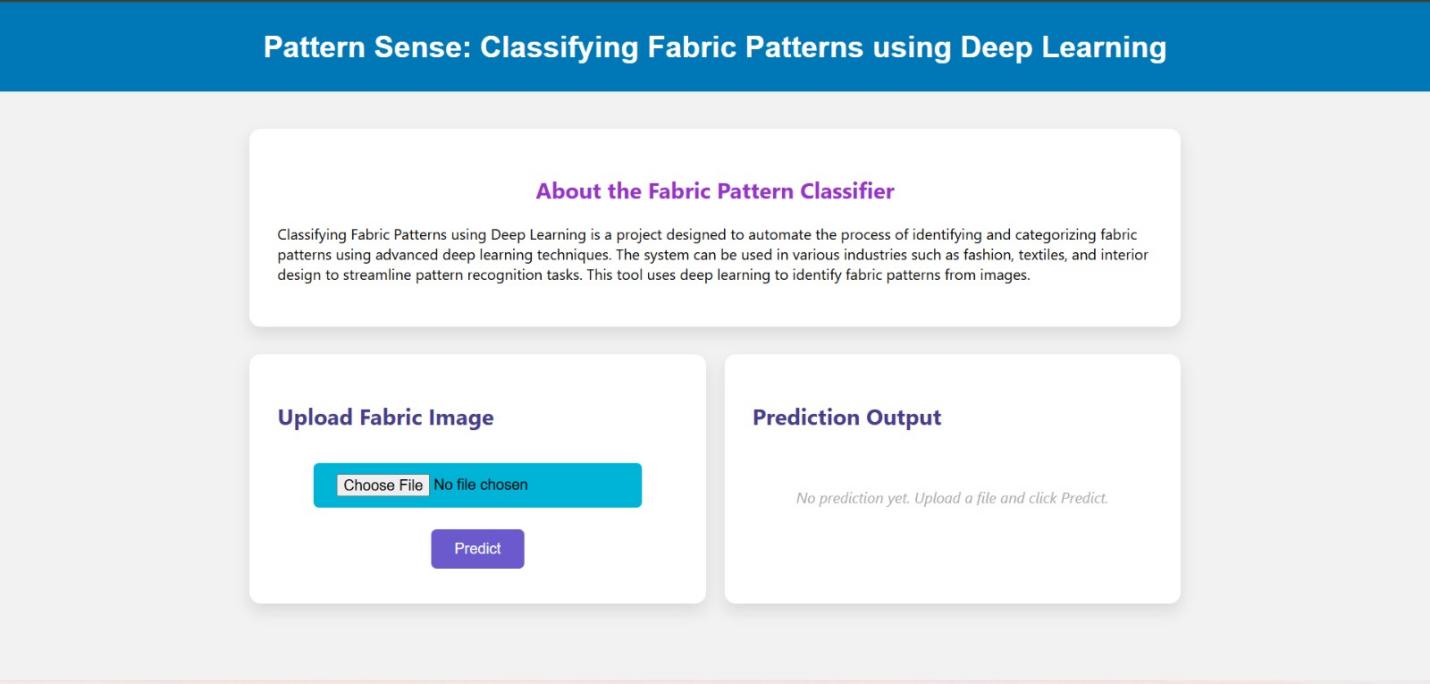
* Method: POST
* Purpose: Receives the uploaded image, performs preprocessing, runs the trained VGG16 model, and predicts the Pattern Sense.
* Parameters:
  + image: Image file uploaded by the user (from the HTML form)
* Response:  
  Renders output.html displaying the predicted species.

# **Step-8:Authentication**

This project does not include authentication or authorization features, as it is designed for open, single-user access. The focus of the system is on classifying Pattern Sense from uploaded images without requiring user login or account management.

**Step-9:user interface**

The user interface is built using Flask and HTML. It allows users to upload Pattern Sense and view predictions. The interface includes a title, file upload button, prediction button, and displays the result along with team information and a thank-you message.



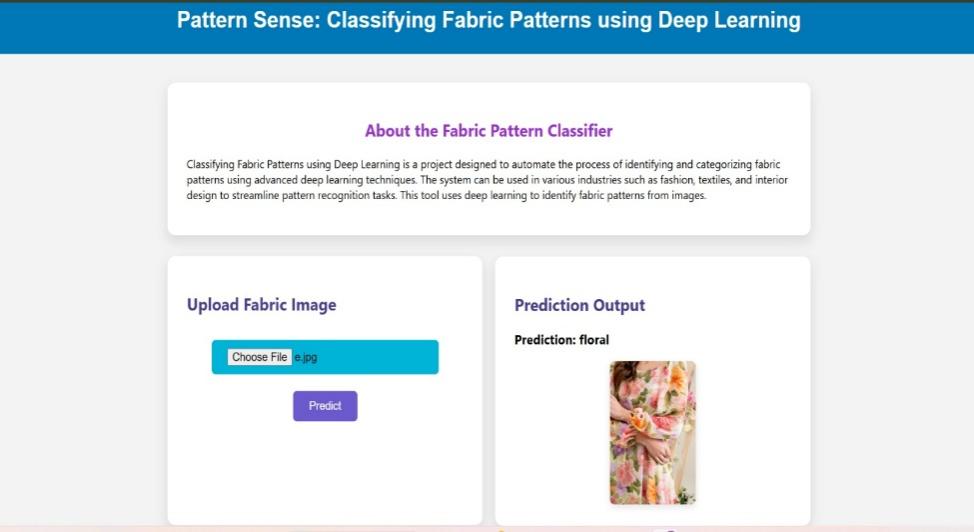
**Step-10:testing**

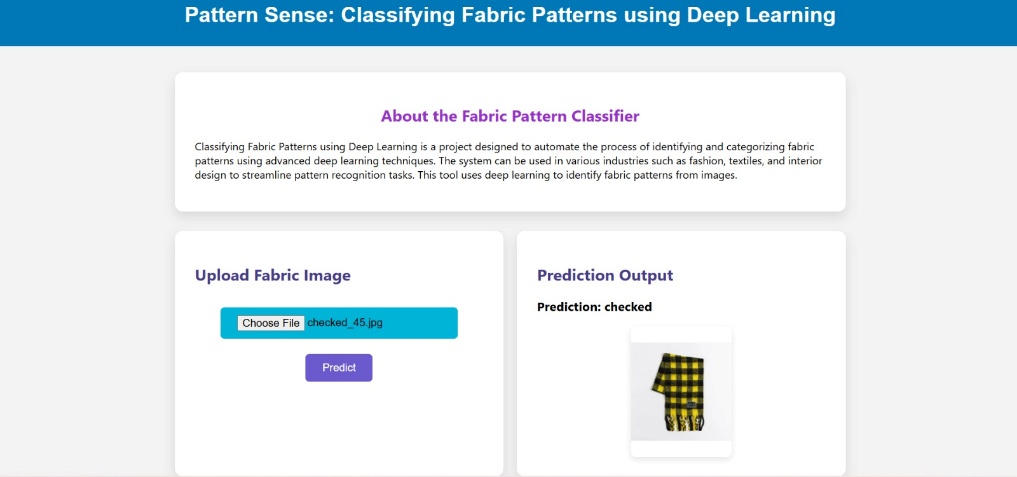
The testing for this project was focused on verifying the accuracy and functionality of the Pattern Sense model and its integration with the Flask web interface.

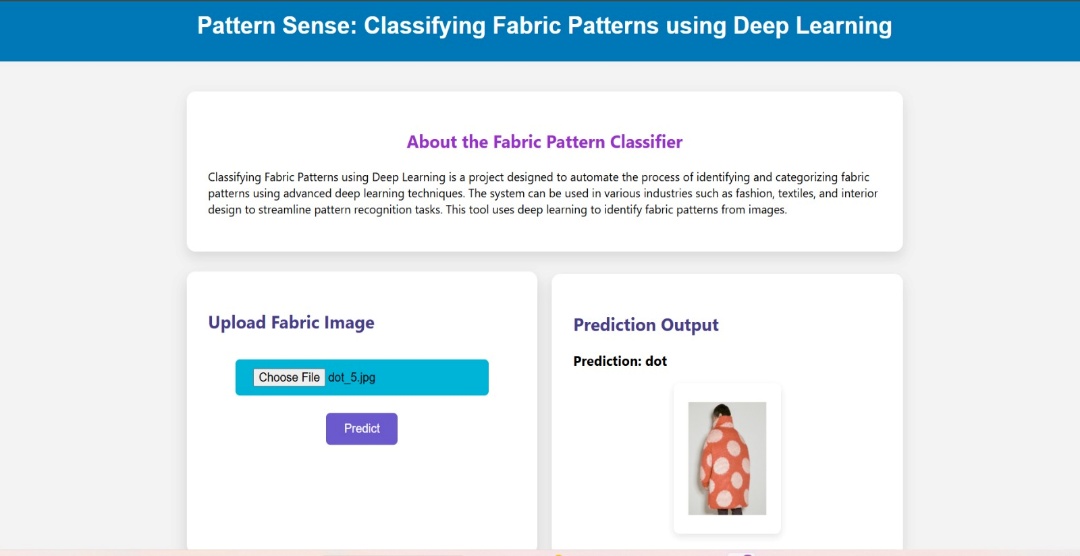
**Testing Strategy:**

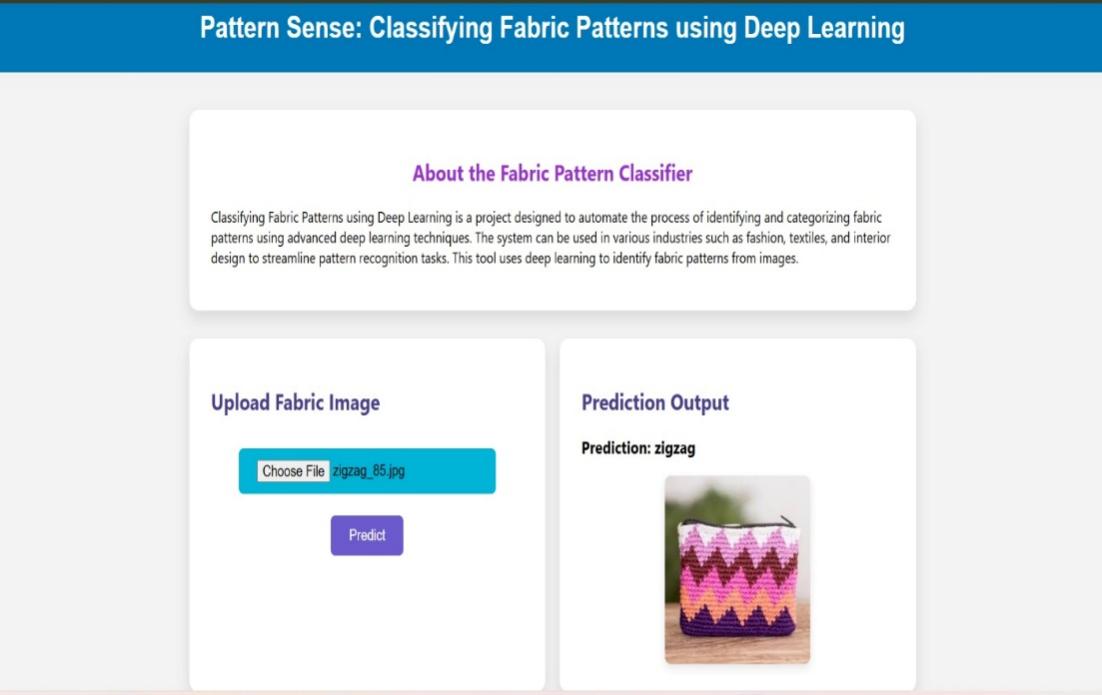
* **Model Testing**:
  + The dataset was split into training and testing sets.
  + The trained model was evaluated using the test images to check prediction accuracy.
  + Sample images were manually tested by uploading through the UI and verifying the predicted pattern sense.
* **Interface Testing**:
  + The Flask web interface was tested by uploading various pattern images.
  + Verified whether the UI correctly accepted files and displayed the predicted output.
  + Checked behavior for invalid inputs (e.g., no file selected, wrong file type).

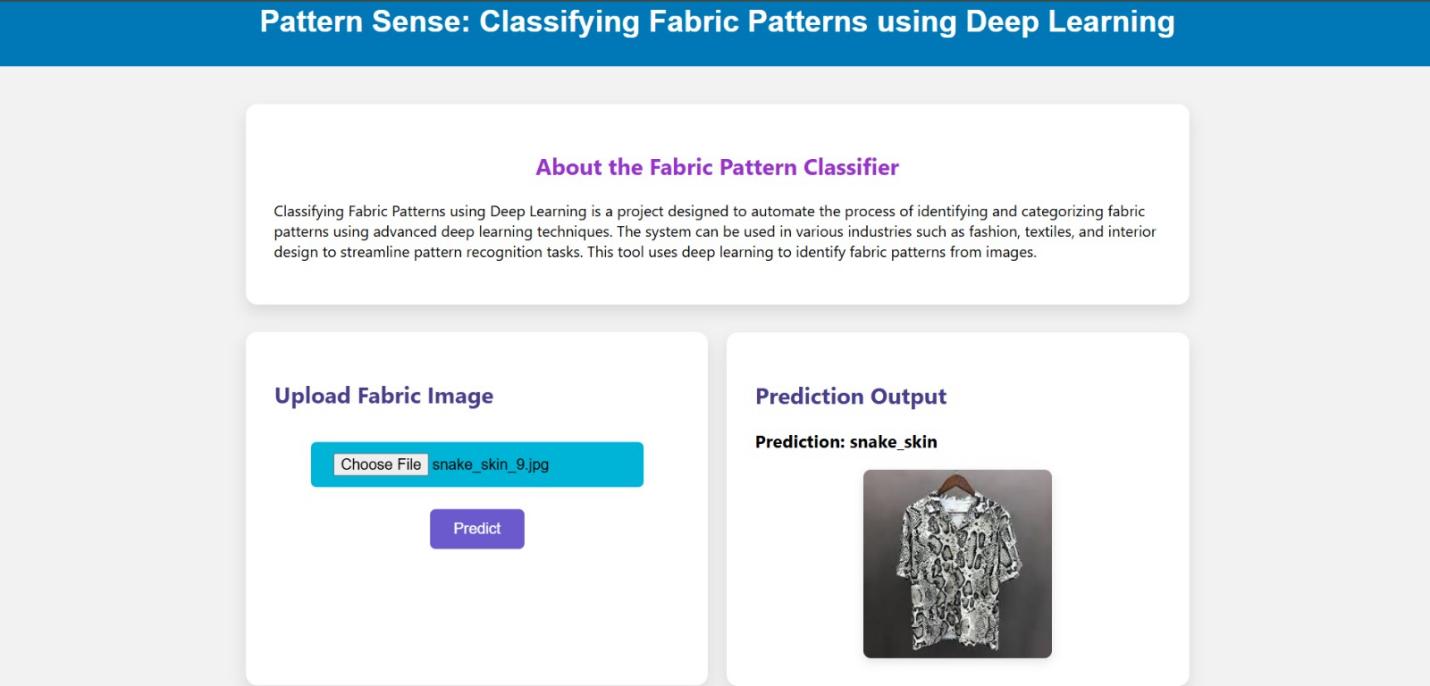
# Step-11: Screenshots











**Step-12. Known Issues**

While the fabric pattern classification system performs well in most scenarios, the following limitations and known issues were observed during testing:

* **Visually Similar Pattern Confusion**:  
  Certain fabric patterns with subtle differences (e.g., stripes vs. pinstripes, abstract vs. floral) may be misclassified due to overlapping visual features.
* **Imbalanced Dataset Impact**:  
  The model may show bias towards pattern classes with more training examples, leading to lower accuracy for underrepresented categories.
* **Lighting and Wrinkle Artifacts**:  
  Images with uneven lighting, shadows, or folds in fabric can negatively impact prediction accuracy.
* **Low-Resolution Image Limitations**:  
  Small or blurry fabric images reduce the ability of the model to detect intricate patterns reliably.

**Step-13:Future Enhancements**

* **Deploy as a Web or Mobile Application** to make the tool accessible for designers, textile manufacturers, and end-users.
* **Adopt More Advanced Architectures** such as EfficientNet or Vision Transformers to enhance classification accuracy.
* **Expand Dataset with Diverse Fabric Types** to improve model generalization across various pattern styles and materials.
* **Allow Batch Upload and Processing** for classifying multiple fabric samples in one go.
* **Include Pattern Metadata and Use Cases** to provide users with additional context, such as design origin, cultural relevance, or typical usage in fashion or interiors.
* **Introduce Real-Time Camera Input** for on-the-fly fabric classification during shopping or production inspection.