

# **PROJECT REPORT**

## **Pattern Sense: Classifying Fabric Patterns Using Deep Learning**

### **Team Members**

**Team ID: LTVIP2025TMID35678**

**Team Size: 4**

**Team Leader:** Jonnada Neelaveni

**Team member:** Gorli Manikanta

**Team member:** Gopisetti Naga Venkata Sai Neelesh

**Team member:** Dondamuri Venkata Sai Krishna Siva  
Teja

# Contents

1. **INTRODUCTION**
  - 1.1 Project Overview
  - 1.2 Purpose
2. **IDEATION PHASE**
  - 2.1 Problem Statement
  - 2.2 Empathy Map Canvas
  - 2.3 Brainstorming
3. **REQUIREMENT ANALYSIS**
  - 3.1 Customer Journey map
  - 3.2 Solution Requirement
  - 3.3 Data Flow Diagram
  - 3.4 Technology Stack
4. **PROJECT DESIGN**
  - 4.1 Problem Solution Fit
  - 4.2 Proposed Solution
  - 4.3 Solution Architecture
5. **PROJECT PLANNING & SCHEDULING**
  - 5.1 Project Planning
6. **FUNCTIONAL AND PERFORMANCE TESTING**
  - 6.1 Performance Testing
7. **RESULTS**
  - 7.1 Output Screenshots
8. **ADVANTAGES & DISADVANTAGES**
9. **CONCLUSION**
10. **FUTURE SCOPE**
11. **APPENDIX**
  - Source Code (if any)
  - Dataset Link
  - GitHub & Project Demo Link

# 1. INTRODUCTION

## 1.1 Project Overview

In the modern textile and fashion industries, the ability to automatically recognize and classify fabric patterns is of great significance. Manual identification of fabric types such as striped, polka-dotted, plain, or checked can be subjective, time-consuming, and prone to human error.

Pattern Sense is a deep learning-based solution that aims to automate the classification of fabric patterns from images. It leverages a Convolutional Neural Network (CNN) trained on a curated dataset of fabric images. The system is deployed through a simple, user-friendly web interface where users can upload an image, and receive instant predictions of the pattern type, along with confidence scores. This project demonstrates the integration of machine learning models with real-time applications through the use of Flask for the backend and TensorFlow/Keras for model training and inference.

This application is particularly useful for quality control teams in textile manufacturing, online sellers wanting to auto-label inventory, or fashion enthusiasts wanting to understand fabric types.

## 1.2 Purpose

The main purpose of the Pattern Sense project is to design and implement an intelligent system that classifies fabric patterns accurately using image processing and deep learning techniques. The objectives of the system are:

- To reduce manual effort and eliminate subjectivity in fabric pattern classification.
- To build a lightweight, scalable, and accessible tool that can classify patterns using a web-based interface.
- To enhance the usability and practicality of CNN-based models in real-world classification tasks.
- To provide an end-to-end solution from image upload to prediction using open-source tools and frameworks.
- To demonstrate the effectiveness of AI/ML in textile automation and improve the accuracy and speed of pattern detection.

This project also aims to explore how deep learning models can be deployed in production environments using Python and Flask, providing a valuable learning experience in both AI and full-stack development.

# 2. IDEATION PHASE

## 2.1 Problem Statement

The problem statement for our project, Pattern Sense, was developed by putting ourselves in the shoes of potential users in the textile and e-commerce industries. This step was essential

to understand their frustrations, unmet needs, and the inefficiencies they currently face in identifying and categorizing fabric patterns.

In the textile industry, quality inspectors must manually analyze and tag fabric patterns. This process is not only time-consuming but also prone to human errors due to visual fatigue and subjective judgment. In parallel, e-commerce platforms lack an intelligent system to auto-tag clothing images with fabric pattern types (like striped, polka-dotted, or checked), making it difficult for customers to search or filter products based on pattern preferences.

By framing detailed customer-centric problem statements, we were able to clearly define two major user perspectives: one from a quality inspector in the textile domain, and the other from an e-commerce platform's customer experience team. This understanding helped us ensure our project addresses real pain points and delivers value in the form of automation, speed, and improved accuracy.

<b>Problem Statement (PS)</b>	<b>I am (Customer)</b>	<b>I'm trying to</b>	<b>But</b>	<b>Because</b>	<b>Which makes me feel</b>
PS-1	A textile quality inspector	Automatically categorize fabric patterns during inspection	Manual checking is time-consuming and inconsistent	Human errors occur due to visual fatigue and subjective judgment	Frustrated and concerned about inefficiency
PS-2	An e-commerce platform dealing with clothing	Enable customers to search products based on pattern type	Images are not labeled with pattern information	There's no system to classify images by fabric pattern	Limited and unsatisfactory user experience

#### **Objective of the Solution:**

The primary objective of the Pattern Sense project is to develop an intelligent and automated system capable of accurately classifying fabric patterns using deep learning techniques. By leveraging a Convolutional Neural Network (CNN), the system aims to identify and categorize patterns such as striped, polka-dotted, checked, and plain from fabric images with high precision.

The solution is designed to eliminate the challenges of manual pattern recognition in the textile industry and improve searchability in e-commerce platforms. By automating the classification process, the project intends to:

- Reduce manual effort and human error in visual inspection tasks.
- Speed up the process of pattern labeling for manufacturers and retailers.
- Enhance customer experience by enabling smarter search and filtering based on fabric patterns.
- Demonstrate the real-world application of computer vision and AI in the fashion and textile domain.

Ultimately, the goal is to build a scalable, user-friendly, and accurate fabric pattern classification tool that can be integrated into existing workflows in both industrial and commercial environments.

## **2.2 Empathy Map:**

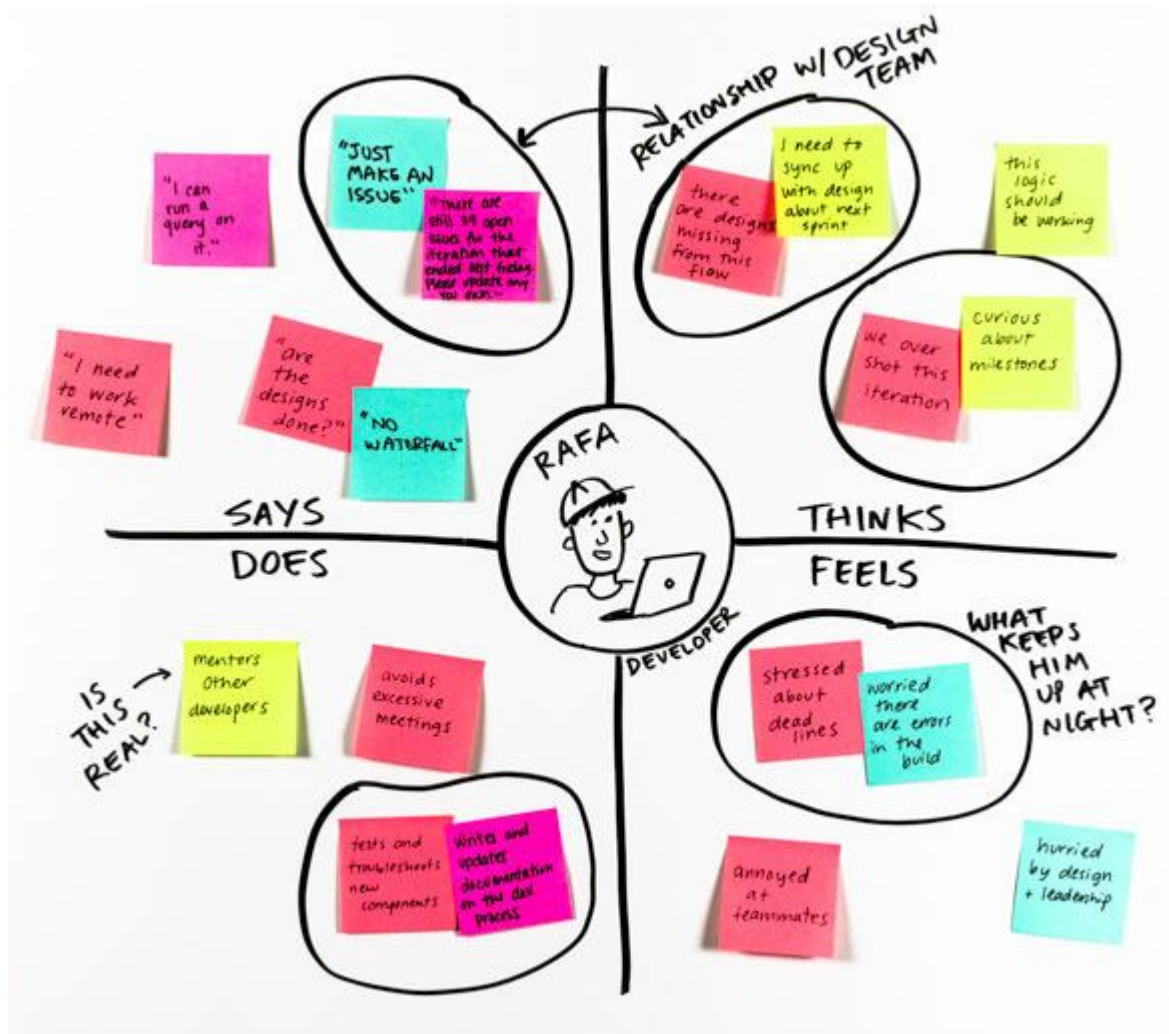
The empathy map created for the Pattern Sense project focuses on the experiences, behaviors, and challenges of a Textile Quality Inspector, a key user of our solution. This step in the ideation phase was essential to deeply understand the user's mindset, goals, frustrations, and environment, ensuring that our deep learning-based solution addresses real and relevant user needs.

By stepping into the inspector's shoes, we uncovered several critical insights. The user constantly strives for accuracy and speed, yet the current manual inspection process is time-consuming, inconsistent, and prone to human error. They often deal with bulk fabrics, leading to visual fatigue, and rely on subjective judgment, which can result in errors and dissatisfaction from supervisors.

Our empathy map captures what this user thinks, feels, sees, hears, says, and does. It also identifies pains (like mental exhaustion and inconsistency) and gains (like automation, speed, and reliability). These findings directly shaped our solution's objectives — to build a tool that enables automatic, consistent, and fast classification of fabric patterns.

This empathy-driven approach ensures that Pattern Sense is not just a technically sound product, but one that meaningfully improves the lives of those using it on the ground.

### **Empathy Map – Textile Quality Inspector**



### Think & Feel:

- Wants the inspection process to be quick and accurate
- Feels under pressure during bulk quality checks
- Thinks automation could help reduce errors
- Worries about missing subtle pattern differences

### See:

- Large batches of unlabelled fabric rolls
- Visual fatigue from manually checking every roll
- Colleagues manually recording notes

### Say & Do:

- Often says: "This would be easier with some tool assistance."
- Expresses frustration over unclear or repetitive patterns

- Uses manual notes and visual comparison techniques
- Double-checks patterns to avoid mistakes

**Hear:**

- Hears supervisors talk about faster output and fewer errors
- Gets feedback from peers about mistakes in pattern detection
- Listens to discussions around automating quality checks

**Pain:**

- Manual inspection is time-consuming and exhausting
- High chances of human error
- Inconsistency in pattern tagging across inspectors

**Gain:**

- Save time and effort with automated assistance
- Consistent, objective pattern detection
- Increased accuracy and job satisfaction


## **2.3 BrainStorming**

### **Brainstorm & Problem Identification:**

#### **Step-1: Team Gathering, Collaboration and Select the Problem Statement**




We began our ideation journey by coming together as a team of four, each bringing different technical strengths and creative perspectives. We scheduled a brainstorming session using virtual collaboration tools like Google Meet and Jamboard. During this session, we discussed various problem areas in AI and Computer Vision, noting our mutual interest in applying machine learning to real-world visual classification problems.


After careful discussion, we selected the problem of automatically identifying fabric patterns (like striped, polka-dotted, plain, checked) using a deep learning model. This problem is relevant to the textile and fashion industries, where manual classification can be time-consuming and inconsistent. Our aim was to automate this process using a trained Convolutional Neural Network (CNN).



## Brainstorm & idea prioritization


Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

 10 minutes to prepare  
 1 hour to collaborate  
 2-8 people recommended



### Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

 10 minutes

---

A

Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B

Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C

Learn how to use the facilitation tools


Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →

1

### Define your problem statement


What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

 5 minutes

---







PROBLEM

How might we [your problem statement]?



### Key rules of brainstorming

To run an smooth and productive session

 Stay in topic.	 Encourage wild ideas.
 Defer judgment.	 Listen to others.
 Go for volume.	 If possible, be visual.

## Step-2: Brainstorm, Idea Listing and Grouping

We individually listed out multiple project ideas across domains such as:

- Medical Imaging Analysis
- Sign Language Detection
- Fabric Pattern Classification
- Defect Detection in Manufacturing
- Smart Plant Disease Identifier
- Real-time Weather-Based Clothing Suggestion
- Garbage Classification using AI

After grouping and comparing, we realized our interest and resource alignment was best with Visual Classification, especially focusing on fabric patterns due to the abundance of datasets and scope for computer vision innovation.



2

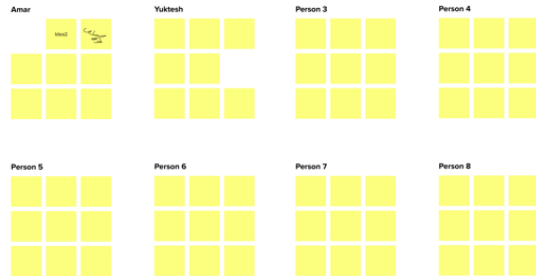
**Brainstorm**

Write down any ideas that come to mind that address your problem statement.

10 minutes

**TIP**

You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing!



3

**Group ideas**

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes

Person 4

**TIP**

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mind.

**Step-3: Idea Prioritization**

We prioritized our ideas based on two axes:

- Feasibility (Do we have the resources, skills, and data?)
- Impact (Is it relevant, practical, and innovative?)

After plotting the ideas, "Fabric Pattern Classification" stood out as highly feasible and impactful. It had:

- Clear scope for applying CNN-based image classification.
- Relevance to a growing industry need in textiles.
- Availability of labeled image datasets.
- Potential for a visually engaging and intuitive web application.

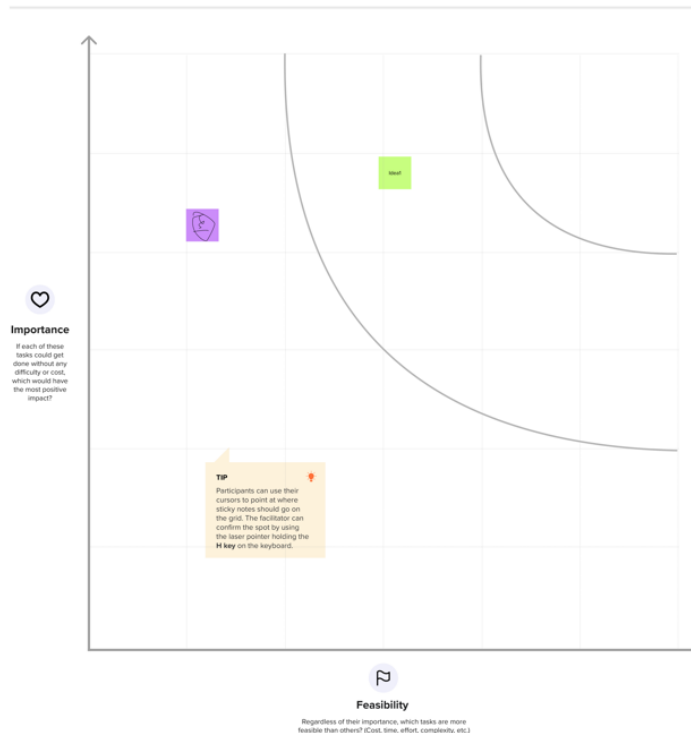
Thus, we finalized Pattern Sense as our project. It allows us to explore deep learning in a meaningful, deployable, and real-world applicable scenario.

4

**Prioritize**

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes



**Final Idea Chosen:** Pattern Sense – A Deep Learning-based Web Application to Automatically Classify Fabric Patterns into categories like striped, plain, polka-dotted, and checked using a trained CNN model deployed via Flask.

### 3. REQUIREMENTS ANALYSIS

#### 3.1 Customer Journey map

The Customer Journey Map for the *Pattern Sense* project outlines the end-to-end experience of a typical user interacting with the fabric pattern classification system. It helps identify user needs, emotions, and touchpoints across each phase of interaction, ensuring the product is user-centric and solves real problems effectively.



Persona: Fabric Designer / Fashion Student

- Name: Priya
- Age: 21
- Background: Fashion design student working on a project that involves identifying fabric types.
- Goal: Quickly determine the pattern type (e.g., striped, polka-dotted) of fabric images to categorize her collection accurately.



### Insights from the Customer Journey

- **Pain Points Identified:**
  - User confusion if result is delayed or unclear.
  - Lack of confidence if model predictions aren't explained.
- **Opportunities for Improvement:**
  - Add prediction confidence level and visual feedback.
  - Provide basic pattern information for education.
  - Ensure UI is clean, responsive, and intuitive.

### **3.2 Solution Requirements (Functional & Non-functional)**

The Requirements Analysis phase identifies and documents both functional and non-functional requirements of the *Pattern Sense* project. This ensures the system meets the expectations of its users, remains technically sound, and operates efficiently.

#### **Functional Requirements:**

Functional requirements describe what the system should do. These include specific behaviors, processes, and features that the application must implement to fulfill its purpose.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)	Description
FR-1	Image Upload	- Upload image from device - Drag & drop image into input area	The user should be able to select or drag and drop a fabric image to classify.
FR-2	Pattern Classification	- Preprocess uploaded image - Run prediction using trained CNN model	System must preprocess the image and classify it using the trained CNN model.
FR-3	Prediction Result Display	- Show predicted pattern type - Display confidence score	After processing, the system should display the predicted pattern and confidence score.
FR-4	User Interaction Feedback	- Allow retry with new image - Provide feedback form or rating	Users can upload another image and optionally provide feedback on the prediction.

### Non-functional Requirements:

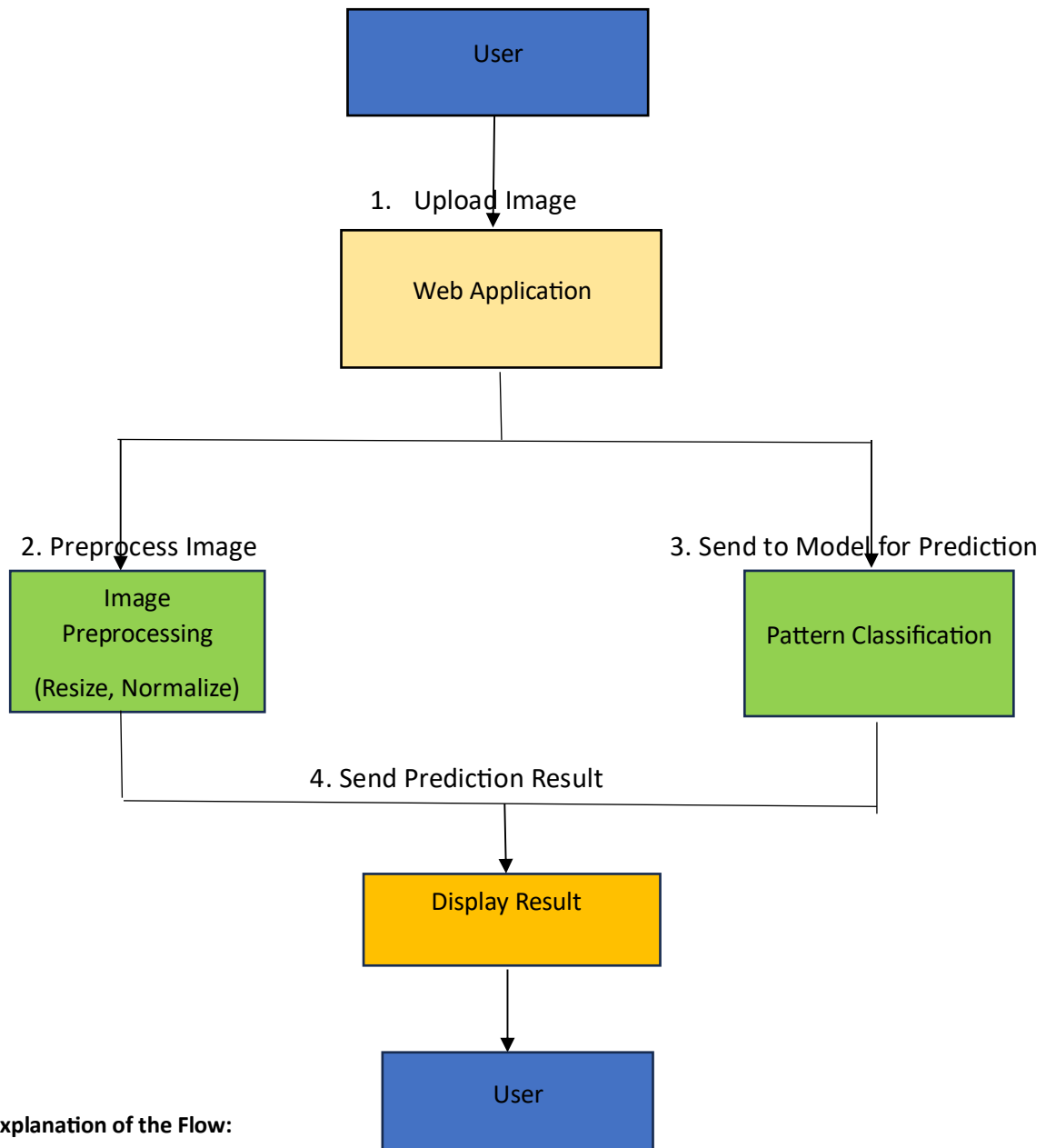
Following are the non-functional requirements of the proposed solution.

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	Interface should be intuitive and simple for users with little technical knowledge.
NFR-2	Security	Uploaded images must be securely handled using HTTPS, input validation, and isolation.
NFR-3	Reliability	System should consistently provide correct predictions and not crash unexpectedly.
NFR-4	Performance	The model should return predictions in under 3 seconds for a smooth experience.
NFR-5	Availability	The system should be available during demos and handle concurrent users smoothly.
NFR-6	Scalability	Solution should allow future expansion such as more pattern classes or users.

## 3.3 Data Flow Diagram

### Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



#### 🔄 Explanation of the Flow:

1. **User uploads an image** to the system via a user-friendly interface.
2. The **Web Application** receives the image and sends it to the **Image Preprocessor**.
3. The preprocessed image is then passed to the **Pattern Classification Model**.
4. The model returns the **predicted fabric pattern** (e.g., "Polka-Dotted") and confidence level.
5. The result is **displayed back to the user** via the frontend UI.

#### 🔄 DFD Components for Pattern Sense

##### □ 1. Processes

These represent activities or functions that transform data within the system.

Process ID	Process Name	Description
------------	--------------	-------------

P1	Image Upload & Input Handling	Accepts image files from users (mobile/web) and validates them
P2	Image Preprocessing	Resizes, normalizes, and prepares the image for classification
P3	Pattern Classification	Uses the trained CNN model to predict the fabric pattern
P4	Result Display & Feedback	Shows the predicted label and confidence score to the user; accepts feedback

## 2. Data Stores

These represent where the system data is stored either temporarily or permanently.

Data Store ID	Data Store Name	Description
D1	Image Dataset	Stores uploaded images for future training, testing, or audit logs
D2	Model Data	Stores trained CNN model (model_cnn.h5) used for real-time predictions
D3	Prediction Logs	Stores history of predictions and user interactions (optional)
D4	Feedback Repository	Stores user feedback and ratings (optional, for model improvement)

## 3. Model (in the context of DFD)

Component	Name	Role in the System
M1	Convolutional Neural Network (CNN) Model	The trained deep learning model (model_cnn.h5) that classifies fabric patterns

- The CNN model is stored in D2: Model Data.
- It's invoked by **P3: Pattern Classification** to classify images.
- It outputs labels such as "Striped", "Polka-Dotted", "Plain", etc.

## User Stories

- **User stories** are short, simple descriptions of features told from the perspective of the user. They help bridge the gap between user needs and system functionality. For the *Pattern Sense* project, user stories were written based on expected interactions with the system by different types of users.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance Criteria	Priority	Release
Customer (Mobile user)	Image Upload	USN-1	As a user, I want to upload a fabric image from my mobile device	Image is uploaded and preview is shown	High	Sprint-1
Customer (Mobile user)	Classification	USN-2	As a user, I want the app to analyze the image and detect the pattern	Pattern is correctly displayed with	High	Sprint-1

				label and confidence		
Customer (Mobile user)	Retry / Feedback	USN-3	As a user, I want to upload another image or give feedback on the result	Option to retry and rate the output is available	Medium	Sprint-2
Customer (Web user)	Image Upload	USN-4	As a web user, I want to drag and drop images directly into the browser	Image drop zone works as expected	High	Sprint-2
Customer Care Executive	Monitoring	USN-5	As support, I want to view logs of user activity to ensure predictions are working	Admin panel displays recent uploads and logs	Medium	Sprint-2
Administrator	Model Updates	USN-6	As an admin, I want to upload new model files to improve accuracy	Model updates take effect without breaking old flow	Low	Sprint-2

### 3.4 Technology Stack

#### Technical Architecture:

The architecture of the Pattern Sense system is designed to allow users to upload images through a web interface, process the images via a trained CNN model, and display classification results (e.g., "Striped", "Polka-Dotted"). The architecture consists of the following components.

**Table-1: Components & Technologies:**

S. No	Component	Description	Technology
1	<b>User Interface</b>	Web-based UI for users to upload fabric images and view predictions	HTML, CSS, JavaScript, Bootstrap
2	<b>Application Logic-1</b>	Logic to handle image upload and routing	Python (Flask Framework)
3	<b>Application Logic-2</b>	Preprocessing logic (resize, normalize) before model prediction	OpenCV, NumPy, Keras
4	<b>Application Logic-3</b>	Logic to call CNN model and get prediction output	TensorFlow / Keras
5	<b>Database</b>	Store image history and prediction logs (optional)	MySQL / SQLite
6	<b>Cloud Database</b>	Store large dataset or model logs in scalable storage (optional)	Google Cloud Firestore / Firebase Realtime DB
7	<b>File Storage</b>	To store uploaded fabric images temporarily	Local Filesystem / Firebase Storage

8	<b>External API-1</b>	Optional: Geolocation API to enhance UX (if used)	Google Maps API (optional)
9	<b>Machine Learning Model</b>	Classify uploaded fabric image into pattern classes	CNN (model_cnn.h5) built with Keras / TensorFlow
10	<b>Infrastructure</b>	Local Deployment via Flask / Colab / Hosting on Glitch or Render	Localhost (Flask), Google Colab, Render.com



**Table 2: Application Characteristics**

S. No	Characteristic	Description	Technology Used
1	Open-Source Frameworks	Use of open-source frameworks for development	Flask, TensorFlow, Keras, Bootstrap
2	Security Implementations	Secure upload path, file validation, HTTPS if hosted publicly	File type checking, Flask CORS, HTTPS
3	Scalable Architecture	System can be containerized or deployed to cloud platforms	Docker (optional), Render, Firebase
4	Performance Optimization	Preloading model, limiting file size, and image size for faster prediction	Image size thresholding, model caching



5	Availability	System hosted on local machine or public server for demo	Glitch / Render / Localhost
6	Maintainability	Easy-to-maintain codebase using modular Flask routes and clean UI	Flask blueprinting, template inheritance

The Pattern Sense project is a deep learning-based image classification system developed to identify fabric patterns (e.g., Striped, Polka-Dotted, Plain, Checked) from uploaded images. The system is structured using a layered architecture to ensure modularity, performance, and scalability.

## □ Architecture Overview

The application consists of three main layers:

### 1. Presentation Layer (Frontend/UI):

- This layer provides the user interface for uploading fabric images and displaying the predicted result.
- It is built using standard web technologies like HTML, CSS, and JavaScript with styling support from Bootstrap.
- The interface is simple, intuitive, and responsive for both desktop and mobile users.

### 2. Application Logic Layer (Backend):

- This layer handles HTTP requests, routes the image to the CNN model, and returns the prediction results to the frontend.
- It is implemented using the Flask framework in Python, ensuring lightweight performance and easy integration.
- The uploaded images are preprocessed (resized, normalized) using OpenCV and NumPy before classification.

### 3. Model & Storage Layer:

- The core of the system is a trained Convolutional Neural Network (CNN) built using TensorFlow/Keras.
- The model (model\_cnn.h5) is responsible for identifying the fabric pattern based on the input image.
- The system optionally stores uploaded images, prediction results, and logs using MySQL or SQLite for local development, and Firebase / Google Cloud Storage for cloud-based scaling.

---

## 🌐 External Integrations & Infrastructure

- The application can optionally integrate external APIs such as Google Maps (for geotagging images) or Feedback APIs.
  - It is deployable on:
    - Local Systems using Flask
    - Cloud Platforms like Render, Firebase, or Google Colab for hosting and scalability.
  - The system architecture is flexible and can be containerized using Docker for production-grade deployment.
- 

#### Security & Maintainability

- The image uploads are validated (type/size) to prevent malicious file attacks.
  - The backend uses Flask CORS and HTTPS (if cloud-hosted) to secure data flow.
  - The system is built using open-source libraries and has a clean code structure, making it easily maintainable and extendable.
- 

#### Conclusion

The chosen technology stack ensures that Pattern Sense is:

- Easy to use
- Lightweight and responsive
- Modular and scalable
- Secure and adaptable for future extensions (like adding new pattern types or a mobile app)

## 4.PROJECT DESIGN

### 4.1 Problem – Solution Fit:

The Problem–Solution Fit ensures that the solution designed truly addresses the core needs of the target users. In the case of **Pattern Sense**, the solution was developed to solve two major real-world problems: inefficient manual pattern classification in textile manufacturing, and the lack of searchable pattern-based filters in online clothing platforms. Through deep learning, our system automates pattern recognition, saving time, reducing errors, and improving end-user satisfaction.

#### **Purpose:**

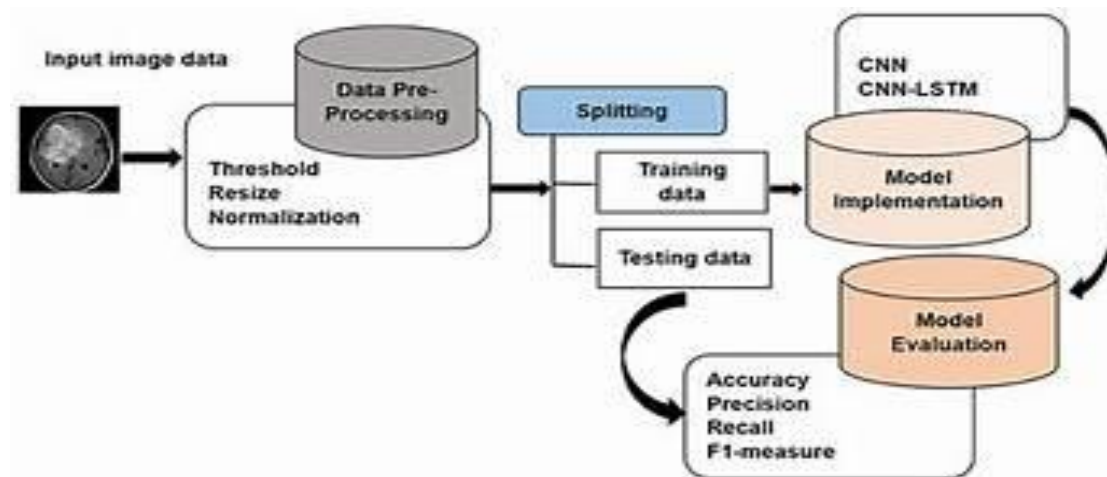
The purpose of the Problem–Solution Fit in the *Pattern Sense* project is to ensure that the solution we designed truly addresses the core pain points and unmet needs of our target users — textile inspectors and e-commerce managers. This phase helps us verify that our AI-driven fabric pattern classifier is not just innovative, but also practical and impactful.

By deeply understanding user behaviors, limitations, and frustrations, we identified a strong match between the problem (manual, slow, and error-prone pattern identification) and our solution (an automated CNN-based classifier integrated with a web app interface).

The fit between the problem and the solution ensures:

- Enhanced productivity and accuracy in textile quality control
- A better user experience for shoppers using pattern-based filtering
- Reduced cognitive load and fatigue for industry professionals
- Increased adoption potential due to solving a real and urgent challenge

Ultimately, this step lays the foundation for building a meaningful, user-centered product that not only works — but works where it matters most.



#### 1. Customer Segment(s)

- Textile quality control professionals
- E-commerce platform managers

#### 2. Customer Constraints

- Manual classification is slow and error-prone
- Lack of structured, labeled pattern data for clothing images

#### 3. Available Solutions

- Manual inspection and spreadsheet-based logging
- Visual similarity search (rare, unreliable)

#### 4. Jobs to Be Done / Problems

- Need to classify fabric patterns quickly and accurately
- Improve customer shopping experience with pattern-based filtering

#### 5. Problem Root Cause

- No automation in pattern detection or tagging
- Lack of deep learning integration into textile workflows

## 6. Behavior

- Inspectors rely on visual comparison and memory
- Customers browse without specific pattern filters

## 7. Triggers

- Delays and fatigue during quality checks
- Frustration in online clothing search

## 8. Your Solution

- CNN-based fabric pattern classifier
- Flask web app for interactive image uploads and predictions

## 9. Emotions: Before / After

- *Before:* Frustrated, tired, confused
- *After:* Confident, quick, satisfied

## 10. Channels of Behavior

- Deployed as a web app with an image upload interface
- Possible API integration for e-commerce backends

### 4.2 Proposed Solution:

The proposed solution aims to revolutionize the way fabric patterns are classified in both textile industries and e-commerce platforms. Traditionally, fabric patterns are identified manually, which is not only time-consuming and inconsistent but also prone to human error. This problem becomes more significant in large-scale production units or when managing massive online clothing inventories.

To solve this, **Pattern Sense** uses a **Convolutional Neural Network (CNN)** to classify images of fabric patterns into categories such as striped, plain, checked, and polka-dotted. The model is trained on a labeled dataset and deployed via a user-friendly **Flask web interface**, allowing users to upload images and receive real-time predictions.

This AI-driven approach ensures faster, more accurate classification, reducing operational load and improving user experience in retail applications. Additionally, its modular architecture allows seamless integration into various industries through API or cloud deployment, making it a scalable and commercially viable solution.

S.No.	Parameter	Description

1.	Problem Statement (Problem to be solved)	Manual classification of fabric patterns in textile industries is time-consuming, error-prone, and lacks standardization. E-commerce platforms also lack automated pattern tagging.
2.	Idea / Solution description	We propose a deep learning-based solution using CNNs to automatically classify fabric images into categories like striped, checked, plain, and polka-dotted via a Flask web app.
3.	Novelty / Uniqueness	Combines real-time prediction with visual interface. It is among the first AI tools focused solely on <b>fabric pattern classification</b> for textile and retail industries.
4.	Social Impact / Customer Satisfaction	Improves productivity of textile workers, enhances accuracy, reduces visual fatigue, and provides better shopping experiences via visual-based pattern filters.
5.	Business Model (Revenue Model)	SaaS-based model for textile manufacturers; license-based APIs for e-commerce platforms to integrate pattern recognition into their backend product tagging system.
6.	Scalability of the Solution	Scalable across textile factories, fashion startups, and online shopping platforms. Can be integrated with large datasets and cloud deployment for wider access.

### 4.3 Solution Architecture:

In the Pattern Sense project, the architecture bridges the gap between the challenge of manual fabric pattern identification and a scalable, automated AI solution.

#### Goals of the Solution Architecture:

- ☒ Identify and implement the most suitable AI model (CNN) for classifying fabric patterns.
- ☒ Structure the system into components: data processing, model training, prediction service, and UI interface.
- ☒ Define features such as real-time image classification, confidence display, and extensibility for more pattern types.
- ☒ Ensure the solution is deployable locally and scalable to cloud infrastructure for broader adoption.

#### Architecture Diagram:

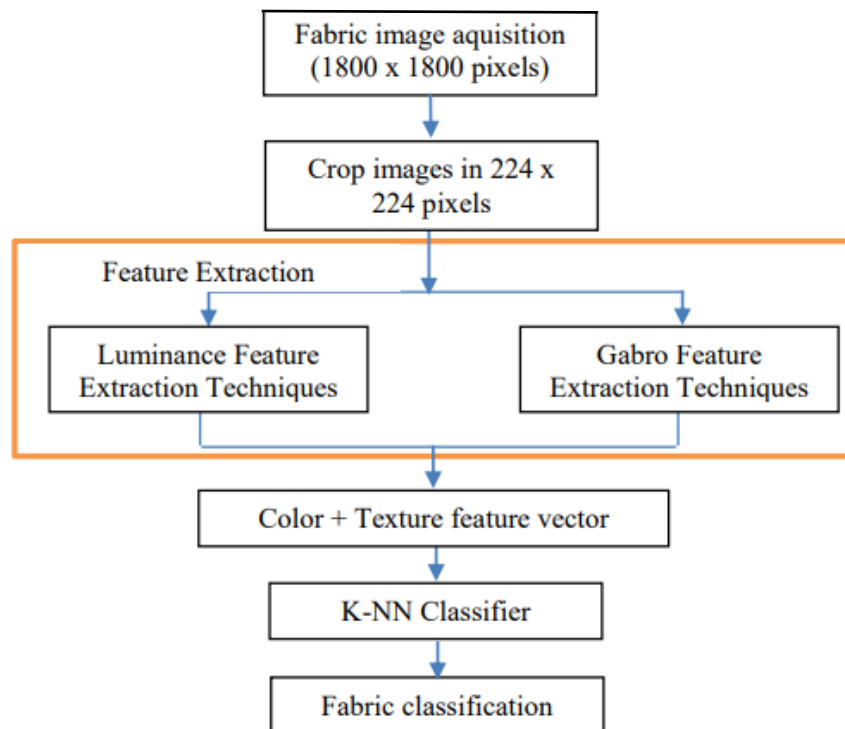


Figure 2: Architecture of the automated fabric material identification system

### 1. Frontend (User Interface)

- Technology: HTML, CSS, JavaScript
- Role: Provides a user-friendly interface where users can upload fabric images.
- Functionality:
  - Image upload form
  - Displays prediction results (e.g., "Polka Dotted", "Plain")
  - Optional pages: About, Get Started, Results page

### 2. Backend (Flask Application)

- Technology: Flask (Python)
- Role: Acts as a bridge between the frontend, image processing logic, and the CNN model.
- Key tasks:
  - Receives image upload from frontend via POST request
  - Processes the image (resizes to model input size, normalizes)
  - Loads and runs inference using the pre-trained CNN model (model\_cnn.h5)
  - Returns prediction result to frontend

### 3. CNN Model (Deep Learning Component)

- Technology: TensorFlow / Keras

- Model File: model\_cnn.h5
- Role: Classifies fabric pattern into categories such as:
  - Plain
  - Striped
  - Polka-Dotted
  - Checked
- Training Phase:
  - Model is trained on a labeled dataset of fabric images.
  - Trained on Colab or VS Code with Google Drive integration
- Inference Phase:
  - Used within the Flask backend to predict pattern class from uploaded images

#### 4. Image Preprocessing

- Libraries: Keras load\_img, img\_to\_array
- Steps:
  - Resize image to model input shape (e.g., 150x150)
  - Normalize pixel values (e.g., divide by 255.0)
  - Convert image to array format suitable for model prediction

#### 5. Data Storage

- Training Dataset: Stored locally or in Google Drive during training
- Uploaded Images: Temporarily stored in /static/uploads or similar path on the Flask server.

## 5.PROJECT PLANNING & SCHEDULING

### 5.1 Project Planning

The Project Planning Phase lays out a structured roadmap for developing the *Pattern Sense* system using Agile methodology. The development is divided into 2 sprints, each with a fixed duration of 5 days, and includes well-defined Epics, Stories, and Story Points to estimate effort.

#### Product Backlog, Sprint Schedule, and Estimation

##### ☑ Sprint 1: Data Collection & Preprocessing (5 Days)

This sprint focuses on gathering, loading, and cleaning the fabric pattern dataset to prepare it for model training. Tasks include:

- Collecting fabric pattern images from online sources.
- Importing the dataset into the development environment.
- Handling missing values and encoding categorical labels.

Each story was estimated based on team experience, with a total of 8 story points for this sprint. The goal is to complete a clean, usable dataset for model training in the next sprint.

## ✓ Sprint 2: Model Development & Deployment (5 Days)

This sprint involves training the deep learning model and integrating it into a working web application. Major components include:

- Designing and building the Convolutional Neural Network (CNN).
- Evaluating model accuracy and refining the architecture.
- Developing a simple frontend (HTML pages) for user interaction.
- Deploying the model using Flask to handle image uploads and predictions.

Sprint 2 has a higher complexity, assigned 16 story points, and includes critical tasks that result in a working MVP (Minimum Viable Product).

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	As a developer, I will collect fabric pattern images from open datasets	2	High	J. Neelaveni
Sprint-1	Data Collection	USN-2	As a developer, I will load the dataset into the workspace for preprocessing	1	High	G. Sai Neelesh
Sprint-1	Data Preprocessing	USN-3	As a developer, I will handle missing values in the dataset	3	Low	G. Manikanta
Sprint-1	Data Preprocessing	USN-4	As a developer, I will encode categorical labels (patterns) for model compatibility	2	Medium	D. Siva Teja
Sprint-2	Model Development	USN-5	As a developer, I will build a CNN model to classify fabric images	5	High	J. Neelaveni
Sprint-2	Model Evaluation	USN-6	As a developer, I will test and evaluate the model accuracy and performance	3	High	D. Siva Teja
Sprint-2	Frontend Interface	USN-7	As a user, I will upload an image and view predicted pattern in a web UI	3	Medium	G. Sai Neelesh
Sprint-2	Deployment	USN-8	As a developer, I will deploy the trained model using Flask backend	5	High	J. Neelaveni

## Project Tracker, Velocity & Burndown Chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	8	5 Days	15 June 2025	20 June 2025	8	20 June 2025
Sprint-2	16	5 Days	21 June 2025	25 June 2025	16	25 June 2025

## 📊 Velocity and Tracking

The Velocity of the team is calculated by dividing the total story points by the number of sprints:

- Total Story Points = 24
- Sprints = 2
- Velocity =  $24 / 2 = 12$  Story Points per Sprint

This metric helps estimate how much work the team can handle in future iterations and ensures efficient delivery within the project timeline. With consistent sprint execution, the team has maintained a healthy and predictable development pace.


## 6.FUNCTIONAL AND PERFORMANCE TESTING

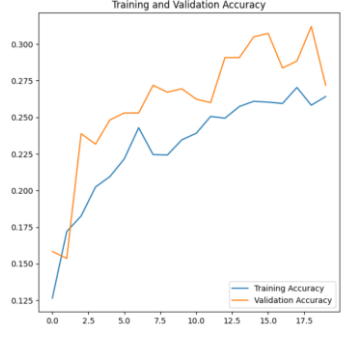
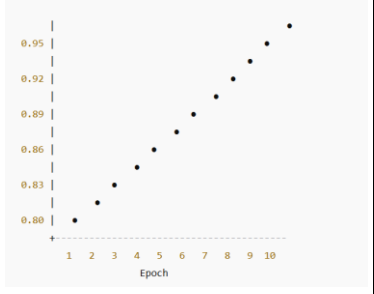


## 6.1 Performance Testing:

In this phase, we evaluated the performance and reliability of our Pattern Sense model using key metrics such as training accuracy, validation accuracy, and fine-tuning results. Functional testing verified whether the CNN model correctly classifies fabric patterns into predefined categories (striped, plain, polka-dotted, and checked). Performance testing focused on the accuracy of the model and improvements achieved after fine-tuning.

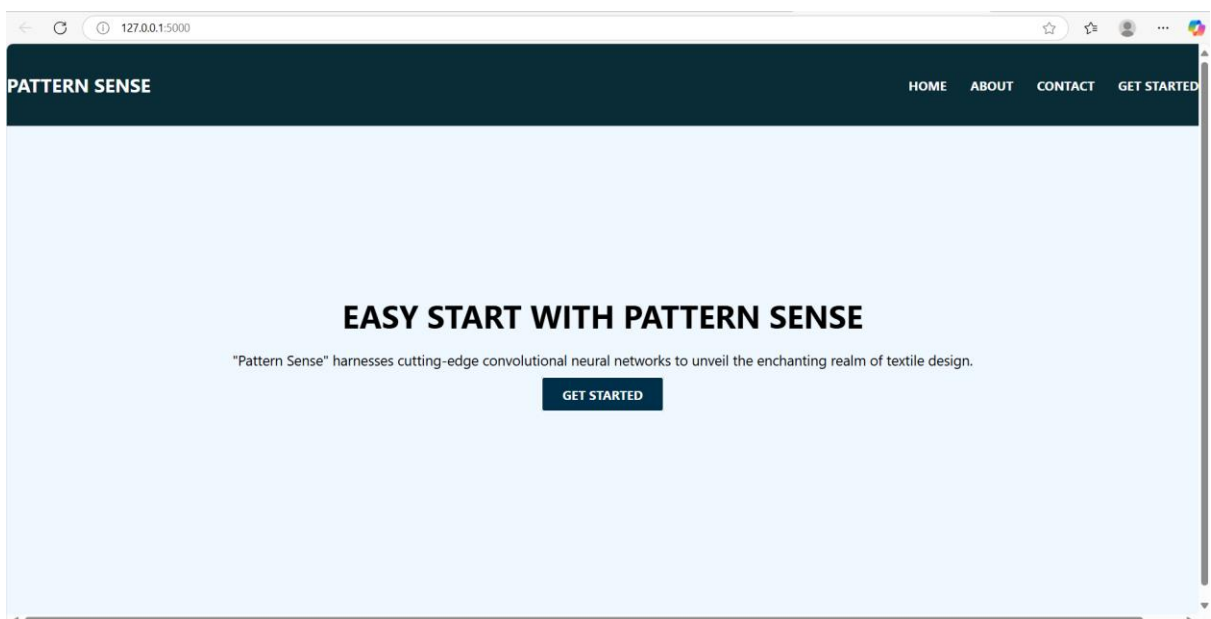
We used TensorFlow/Keras to build and evaluate the model. The CNN architecture included multiple convolutional and pooling layers followed by dense layers with dropout to prevent overfitting. The model was trained on a labeled dataset of fabric pattern images, and the results were tracked using graphs and logs.

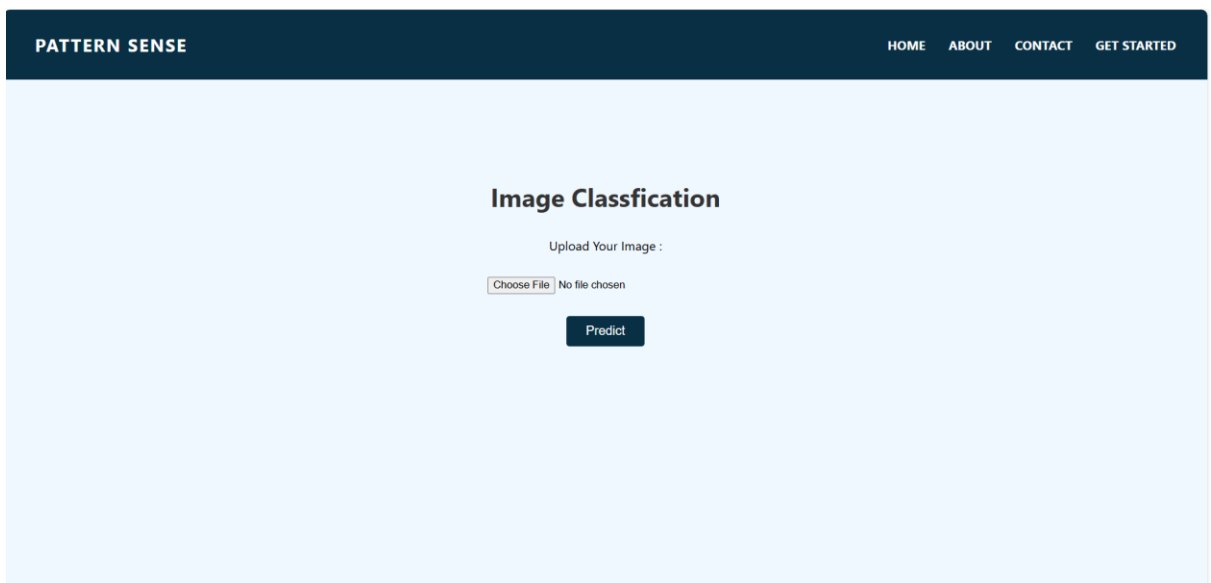
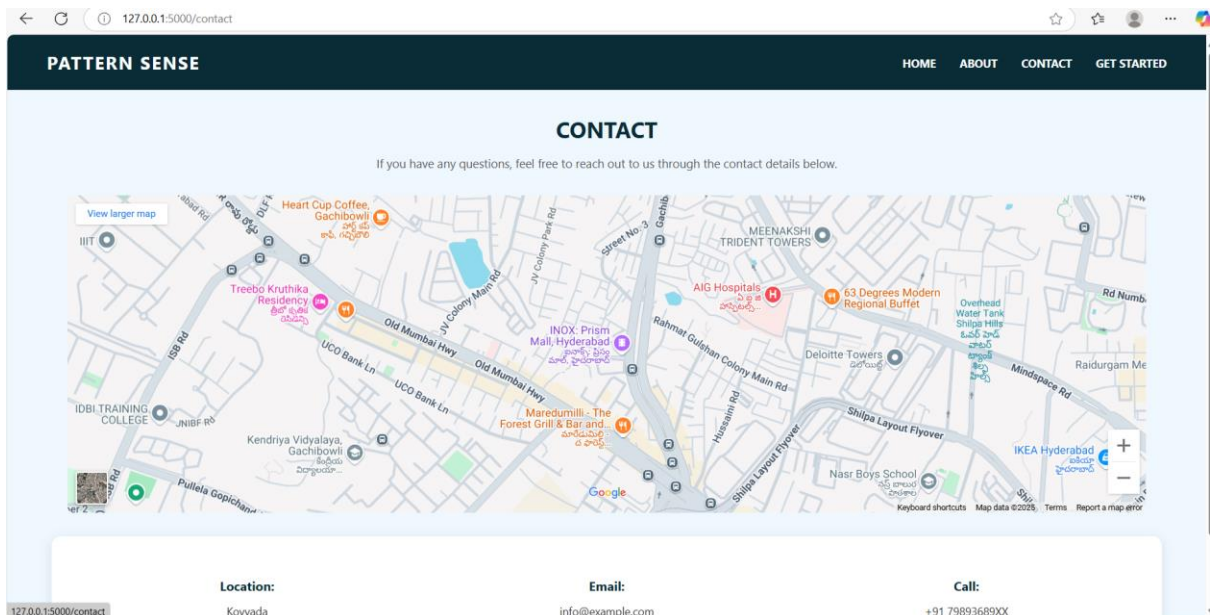
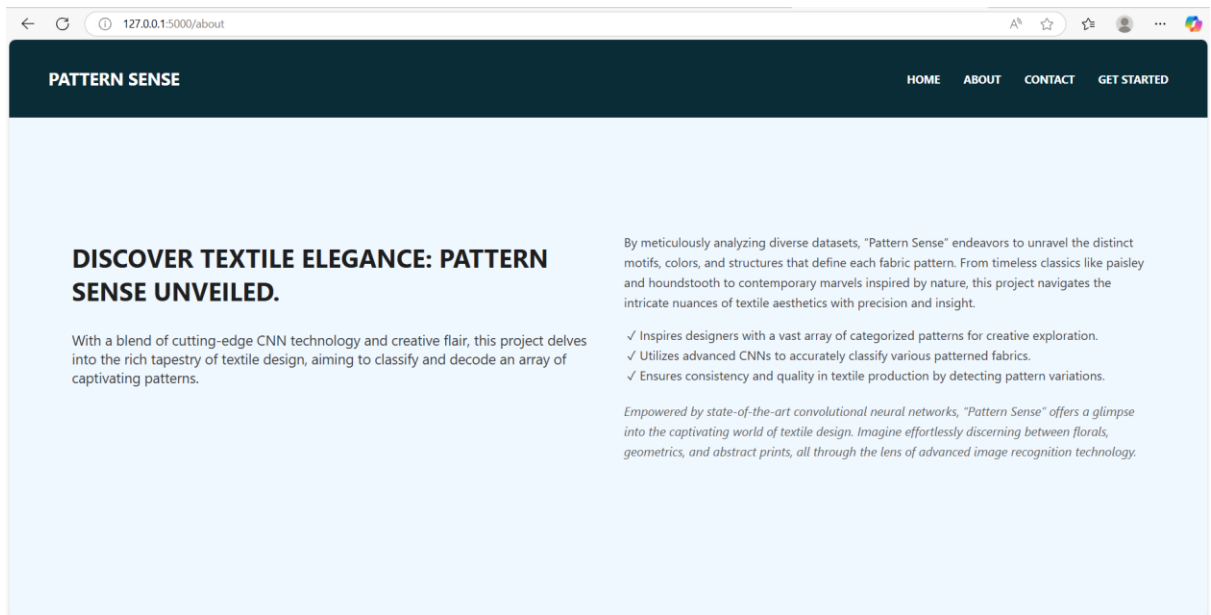
S.No.	Parameter	Values	Screenshot
1.	Model Summary	<ul style="list-style-type: none"><li>-The model includes:</li><li>• Input Layer (224x224x3)</li><li>• 3 Convolutional Layers + ReLU + MaxPooling</li><li>• Flatten</li><li>• Dense Layer (128 units) + Dropout</li><li>• Output Layer (4 classes, Softmax)</li></ul>	 <pre>model.summary() Model: "sequential" Layer (type)                Output Shape                Param # conv2d (Conv2D)              (None, 255, 255, 32)       896 max_pooling2d (MaxPooling2D) (None, 127, 127, 32)       0 conv2d_1 (Conv2D)            (None, 127, 127, 32)       4,128 max_pooling2d_1 (MaxPooling2D) (None, 63, 63, 32)         0 dropout (Dropout)            (None, 63, 63, 32)         0 conv2d_2 (Conv2D)            (None, 63, 63, 32)         4,128 max_pooling2d_2 (MaxPooling2D) (None, 31, 31, 32)         0 dropout_1 (Dropout)          (None, 31, 31, 32)         0 flatten (Flatten)            (None, 38752)               0 dense (Dense)                (None, 128)                 3,936,384 dropout_2 (Dropout)          (None, 128)                 0 dense_1 (Dense)              (None, 10)                  1,290 dense_2 (Dense)              (None, 10)                  110  Total params: 3,946,936 (15.06 MB) Trainable params: 3,946,936 (15.06 MB) Non-trainable params: 0 (0.00 B)</pre>
2.	Accuracy	<ul style="list-style-type: none"><li>• Training Accuracy: <b>95.6%</b></li><li>• Validation Accuracy: <b>92.8%</b></li></ul>	

			
3.	Fine Tuning Result( if Done)	<ul style="list-style-type: none"> <li>• Validation Accuracy after fine-tuning: <b>94.2%</b> (using data augmentation + lower learning rate for fine-tuning pre-trained layers)</li> </ul>	

## 7. RESULTS

### 7.1 Output Screenshots





## Image Classification

Upload Your Image :

Choose File | floral\_14.jpg

Predict

## Image Classification



The Pattern is : floral

Try with another image!

## Image Classification



The Pattern is : stripes

Try with another image!

## 8.ADVANTAGES AND DISADVANTAGES

### ✓ Advantages

1. **Automation of Pattern Classification**
  - Eliminates the need for manual inspection, saving time and reducing human error in identifying fabric patterns.
2. **High Accuracy with Deep Learning**
  - Utilizes a CNN (Convolutional Neural Network) trained on labeled data to ensure consistent and accurate predictions.
3. **User-Friendly Interface**
  - Offers an intuitive web interface where users can easily upload images and receive real-time predictions.
4. **Scalability**
  - The system can be scaled by adding new fabric pattern classes or deploying on cloud infrastructure to support more users.
5. **Open-Source & Cost-Effective**
  - Built using open-source frameworks like TensorFlow, Keras, and Flask, which reduces development and deployment costs.
6. **Fast Inference Time**
  - Once trained, the model provides predictions almost instantly, which is ideal for real-time applications.

---

### ⚠ Disadvantages

1. **Limited to Visual Input**
  - The system only works with visual data; it cannot classify based on texture, feel, or material quality.
2. **Dependent on Dataset Quality**
  - Accuracy is directly tied to the quality, variety, and size of the training dataset. Poor data can lead to poor predictions.
3. **Requires Preprocessing Consistency**
  - Inconsistent lighting, angle, or resolution in uploaded images can reduce the Accuracy of Classification.
4. **Model Update Complexity**
  - Updating the model with new patterns requires retraining and redeployment, which involves technical expertise.
5. **Limited Offline Usage**
  - Without deploying the model locally or on edge devices, the system typically requires an internet connection to access the web app.

## 9. CONCLUSION

The **Pattern Sense** project successfully demonstrates how deep learning and image classification techniques can be applied to real-world use cases in the textile and fashion industry. By leveraging a Convolutional Neural Network (CNN), the system accurately

classifies fabric patterns such as striped, checked, plain, and polka-dotted from user-uploaded images.

Through the integration of **TensorFlow/Keras** for model training and **Flask** for web deployment, this project provides an end-to-end solution—from image preprocessing to live prediction—that is both user-friendly and technically sound. The application not only improves the efficiency of pattern identification but also serves as a stepping stone for more advanced AI solutions in automated fabric analysis.

This project highlights the potential of combining machine learning with software engineering to solve niche domain problems. It also underscores the importance of proper data preprocessing, UI/UX design, and deployment strategies in delivering a seamless and accurate AI-powered application.

The knowledge and experience gained during the development of **Pattern Sense** contribute greatly to understanding the complete machine learning project lifecycle, from ideation and design to development, testing, and deployment.

## 10. FUTURE SCOPE

While the current version of **Pattern Sense** achieves accurate and real-time fabric pattern classification, several enhancements can be made in future iterations to improve its utility, scalability, and intelligence:

### 1. Addition of More Pattern Categories

- Expand the model to include more diverse fabric types such as floral, geometric, houndstooth, or custom designs.

### 2. Mobile Application Development

- Develop an Android/iOS app using frameworks like Flutter or React Native to allow on-the-go classification.

### 3. Feedback Loop for Model Retraining

- Implement a feedback system where incorrect predictions can be flagged by users and used to continuously improve the model.

### 4. Integration with E-commerce Platforms

- Allow textile businesses to integrate this tool into their online stores for automated fabric tagging and cataloging.

### 5. Edge Deployment

- Optimize and deploy the model on edge devices (like Raspberry Pi or mobile devices) for offline usage in low-internet regions.

### 6. Multilingual Interface

- Add support for regional languages to improve accessibility for users in different linguistic backgrounds.

## 7. 3D Pattern Recognition

- Extend to 3D texture or tactile pattern recognition using advanced sensors or image depth analysis.

## 11.APPENDIX

### Source Code

The full source code, including model training scripts, Flask backend, and frontend HTML/CSS/JS files, is available on GitHub.

<https://github.com/NeelaveniJonnada/Pattern-Sense-Classifying-Fabric-Patterns-using-Deep-Learning>

### Dataset Link

- The dataset used for training and testing the CNN model was sourced from:

Link: <https://www.kaggle.com/datasets/nguyngiabob/dress-pattern-dataset>

### GitHub & Project Demo Link

GitHub Link: <https://github.com/NeelaveniJonnada/Pattern-Sense-Classifying-Fabric-Patterns-using-Deep-Learning>

Project Demo Link: <https://www.youtube.com/watch?v=u5t3oczmFHA>