1. INTRODUCTION

1.1 Project Overview

Fabric pattern sensing is a vital and foundational component of modern textile manufacturing and quality assurance. It ensures that the final textile products meet aesthetic, design, and structural requirements as defined by designers and consumers. In the traditional textile industry, identifying and classifying fabric patterns has been a labour-intensive process, relying heavily on experienced human inspectors to manually evaluate each fabric roll. However, this approach is highly susceptible to human error, fatigue, and inconsistent judgment.

With the onset of automation and artificial intelligence, particularly the evolution of deep learning technologies, the textile sector is undergoing a transformative shift. Among the most powerful tools in image processing are Convolutional Neural Networks (CNNs), a class of deep neural networks specifically designed to recognize visual patterns. CNNs have been used with great success in medical imaging, self-driving cars, and facial recognition—and now, they are proving highly effective in the textile industry for tasks such as pattern recognition.

This project introduces an advanced method for fabric pattern sensing using TensorFlow, a leading open-source platform for machine learning and artificial intelligence. TensorFlow offers a flexible and scalable framework to build, train, and deploy deep learning models capable of identifying various fabric patterns. These patterns may include floral prints, stripes, polka dots, geometric designs, and solid colours.

1.2 Purpose of the Project

The purpose of this project is to develop and implement an intelligent system that utilizes deep learning techniques, particularly TensorFlow, to detect and classify fabric patterns automatically. In the context of the rapidly evolving textile industry, where efficiency and precision are paramount, such a system can bring transformative value.

Key Objectives of the Project:

- Automating the quality control process in fabric production to minimize the reliance on manual labour and reduce the time taken for inspections.
- Reducing human error that may occur due to fatigue, subjective judgment, or oversight in pattern verification.
- Enhancing the speed and scalability of the pattern detection process to support high-volume industrial operations and continuous production lines.
- Providing accurate and real-time insights into fabric classifications, enabling quicker decision-making and adaptive manufacturing adjustments.
- Establishing a foundational architecture for future upgrades and integration into larger smart factory ecosystems, supporting the broader vision of Industry 4.0.

2. IDEATION PHASE

2.1 Problem Statement:

- Manual classification of fabric patterns in textile industries is time-consuming and prone to human error.

- There is a need for an automated system to recognize and categorize patterns like stripes, checks, floral, paisley, geometric, etc.

- Deep learning, especially Convolutional Neural Networks (CNNs), offers a promising solution for visual pattern recognition.

- The objective is to build a deep learning-powered system to accurately classify fabric patterns and integrate it into a user-friendly web application.

- This system can help designers, manufacturers, and online platforms improve efficiency and reduce manual workload.

2.2 Empathy Map Canvas (Point-wise):

Who are we empathizing with?

- Textile designers

- Inventory and quality control managers

- Fashion e-commerce companies

- Pattern researchers

What do they SAY?

- "Labeling patterns is repetitive and slow."

- "We want automation but without needing technical skills."

- "We need something reliable and accurate."

What do they DO?

- Manually inspect fabrics.

- Categorize and sort images or physical samples.

- Use outdated methods for organizing fabric collections.

What do they THINK?

- "There must be a smarter way to do this."

- "A lot of time is wasted on simple classification."

- "We could make better use of AI tools."

What do they FEEL?

- Frustrated with inefficiency.

- Overwhelmed by repetitive work.

- Excited by the idea of AI-based automation.

2.3 Brainstorming Ideas

Data & Dataset:

- Collect or create a dataset with diverse fabric pattern images.

- Include classes like floral, checks, stripes, zigzag, paisley, geometric, abstract, and plain.

- Ensure at least 50+ unique images per class, divided into training and testing sets.

Model Development:

- Use pre-trained CNNs (e.g., ResNet50 or EfficientNet) and fine-tune them on fabric images.

- Apply techniques like transfer learning and feature extraction.

- Measure performance using accuracy, confusion matrix, and classification report.

Data Augmentation:

- Add transformations like rotation, flipping, zoom, brightness changes, and cropping.

- Helps in generalizing the model to real-world variations in fabric patterns.

Web Application:

- Build a simple Flask-based web app.

- Let users upload an image and view prediction results instantly.

- Display fabric info or details based on prediction class using a real-time API.

Database Integration:

- Use SQLite to store uploaded image names, prediction results, timestamps, and user info.

- Optional: Enable user login/registration for secure and personalized usage.

User Interface:

- Design intuitive pages: Home, Login, Register, Upload, and Prediction Result.

- Add navigation links and background styling (fabric-themed).

Advanced Features (Optional):

- Use Grad-CAM or similar methods to visualize model attention on the fabric.

- Provide recommendations for similar patterns.

- Enable feedback collection from users to improve accuracy.

Real-World Use Cases:

- Fashion design ideation and prototyping.

- Automated product tagging for e-commerce.

- Quality control automation in textile factories.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map:

The customer journey begins when users become aware of the fabric pattern sensing application, either through internal referrals, promotional efforts, or online presence. Intrigued by the concept of automated pattern detection, they visit the web platform to explore its capabilities. In the consideration phase, they browse the homepage, learning how deep learning is used to classify fabric patterns. Once convinced, they proceed to register and log in securely. After authentication, users are redirected to the prediction page where they can upload a fabric image. The application processes the image and displays the predicted pattern class along with relevant information retrieved from external APIs. Users gain quick insights into the fabric style, and the prediction result is stored in the backend database linked to the user's session. This seamless flow allows users to revisit previous predictions or provide feedback, completing their journey from discovery to interaction and retention.

3.2 Solution Requirement:

The solution is designed to meet both functional and non-functional requirements. Functionally, the system must allow user registration, login authentication, image upload, and automated classification of fabric patterns. It should also display the prediction result and fetch realtime information related to the predicted pattern. All user interactions and results are stored in an SQLite database to enable traceability and user-specific history. Non-functional requirements include a responsive and user-friendly interface, real-time model prediction performance, secure session handling, and scalability to support additional pattern classes in the future. The system should ensure that user data is handled securely and that the application remains efficient and reliable even with multiple concurrent users.

3.3 Data Flow Diagram:

The data flow of the application is structured to follow a logical and efficient sequence. At a high level (Level 0 DFD), the user interacts with the web interface, uploads a fabric image, and receives a prediction result through the backend model. In the detailed Level 1 view, the process starts when the user submits an image via the web interface. This image is sent to the Flask server, which applies necessary preprocessing and passes the image to the pre-trained deep learning model for prediction. Once the prediction is generated, it is displayed on the user interface. Simultaneously, the result, along with the timestamp and user ID, is stored in an SQLite database. Additionally, based on the predicted label, the application may connect to a real-time API such as Bing to fetch descriptive information about the fabric style, which is then displayed to the user.

3.4 Technology Stack

The technology stack chosen for this project is lightweight, scalable, and easy to deploy. On the frontend, the application uses HTML5 and CSS3 for structure and styling, with optional Bootstrap for responsive design and JavaScript for basic interactivity. The backend is developed using Flask, a Python-based micro web framework, and Jinja2 is used for rendering HTML templates. The core deep learning functionality is powered by PyTorch, which supports flexible model building and efficient

inference. SQLite is used as the backend database to store user information, prediction history, and related metadata.

4. PROJECT DESIGN

4.1 Problem-Solution Fit

The traditional method of classifying fabric patterns is heavily dependent on manual effort, which is not only time-consuming but also susceptible to human error and inconsistencies. This problem is especially evident in the textile, fashion, and e-commerce industries, where efficient inventory tagging and pattern identification are critical. The proposed solution perfectly fits this problem space by introducing a deep learning-based automated pattern recognition system. By leveraging a trained convolutional neural network (CNN), the system eliminates the need for manual inspection and provides accurate, real-time pattern classification. This approach aligns well with the core needs of industry users—speed, accuracy, reliability, and simplicity—making it a strong problem-solution fit.

4.2 Proposed Solution

To address the challenges of manual pattern classification, the proposed solution involves developing an intelligent web-based application that utilizes deep learning to classify fabric patterns automatically. Users can register, log in securely, and upload fabric images through a simple interface. Once uploaded, the image undergoes preprocessing and is passed through a CNN model (e.g., ResNet50) trained on a diverse dataset of fabric patterns. The model predicts the pattern class—such as floral, stripes, checks, or geometric—and the result is displayed instantly. Additionally, real-time information about the predicted pattern is fetched using an external API (such as Bing Search) and shown to the user. Each prediction is logged in a local SQLite database, including the user ID and timestamp, ensuring traceability. This end-to-end solution is lightweight, scalable, and designed to provide a seamless user experience.

4.3 Solution Architecture

The solution architecture follows a modular and layered design to ensure clarity, maintainability, and performance. At the front end, the user interacts with a web interface built using HTML, CSS, and JavaScript, with Flask managing the server-side rendering using Jinja2 templates. When a user uploads an image, it is sent to the Flask backend, where preprocessing steps (resizing, normalization) are applied. The image is then passed to a PyTorch-based deep learning model loaded in memory. The model processes the image and returns the predicted fabric pattern class. The result, along with user session details and the timestamp, is saved into an SQLite database. In parallel, an API call is made to fetch descriptive or contextual information about the predicted pattern, enriching the user's experience. The application is designed with clear separation of concerns: the UI layer handles interactions, the application layer processes logic and routing, the model layer performs prediction, and the storage layer handles data persistence. This architecture ensures that the system is both extensible and easy to deploy.

5. PROJECT PLANNING & SCHEDULING

The successful implementation of the Fabric Pattern Sensing system requires careful planning, well-defined stages, and timely execution. The project is structured into multiple phases, each with clear objectives, deliverables, and timelines to ensure smooth progress and accountability.

The planning phase begins with understanding the problem and defining the scope of the solution. This includes identifying key objectives, stakeholders, and system requirements. Once the problem space is clearly defined, the team proceeds to the dataset preparation phase, where fabric pattern images are collected, labeled, and organized into training and testing sets. The dataset is then preprocessed using techniques such as resizing, normalization, and augmentation.

In the model development phase, a deep learning model, such as ResNet50, is implemented using PyTorch. The model is trained on the prepared dataset, fine-tuned, and evaluated using metrics like accuracy, precision, and confusion matrix. After successful validation, the trained model is saved and integrated into the Flask backend.

The web development phase includes designing and implementing the user interface using HTML, CSS, and Flask. Secure user authentication (login/registration) is implemented using SQLite. The upload and prediction features are connected to the model, and a real-time API (such as Bing Search) is integrated to fetch additional fabric information.

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Functional Testing

Functional testing focuses on verifying whether each component of the application performs according to its intended functionality. For the Fabric Pattern Sensing system, the testing process ensures that all features work correctly from the user's perspective and align with the defined requirements.

Key functional tests include:

- User Registration & Login: Ensures that users can securely register and log in with unique credentials. Validation checks for existing accounts, empty fields, and password criteria are tested.

- Image Upload & Prediction: Verifies that users can successfully upload a fabric image and receive accurate predictions. It checks the image format support, file size handling, and realtime feedback display.

- Prediction Accuracy Display: Ensures the predicted class is correctly rendered on the user interface and corresponds with the model output.

- API Integration: Validates that the external API (e.g., Bing Search or similar) returns relevant fabric pattern information based on the prediction.

- Database Storage: Confirms that prediction results, along with the user ID and timestamp, are correctly saved in the SQLite database.

- Navigation Flow: Checks the smooth flow of the web application, ensuring that navigation between home, login, upload, and result pages works without issues.

- Logout Functionality: Verifies that users can securely log out and their session is terminated.

Each test case is documented with input conditions, expected output, actual output, and status (pass/fail), ensuring traceability and thorough coverage.

6.2 Performance Testing

Performance testing evaluates the responsiveness, speed, and stability of the system under normal and peak workloads. For this application, the focus is on the prediction speed, API response time, and database operations.
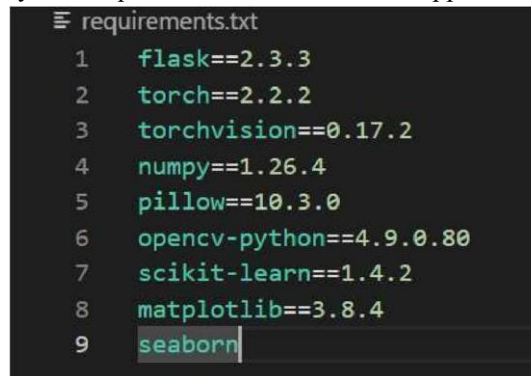
Key performance criteria include:

- Model Inference Time: The CNN model should predict the fabric pattern within 1–2 seconds after image upload. This is tested using a variety of image sizes and qualities.

- API Response Time: The integrated fabric information API should respond in under 2–3 seconds to ensure a seamless user experience.

- Concurrent Users: The system is tested with multiple simulated users uploading and predicting images simultaneously to observe load handling and session management.

- Database Read/Write Efficiency: Performance is evaluated based on how quickly the database writes new entries and retrieves user-specific prediction history.

7. RESULTS

7.1 Output Screenshots

The following figure displays the required extensions to run the app successful.

The following is the required structural of the file to be stored.
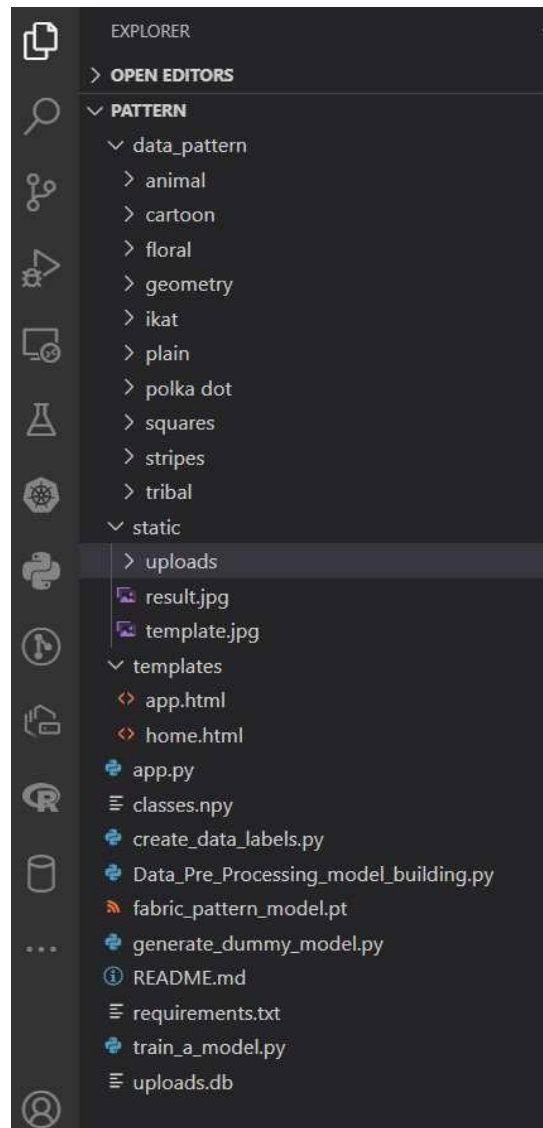
Open the terminal and type python app.py and click on Enter. The IP Address is generated like shown below

And get copy the IP Address and Paste on the any google page and click enter.
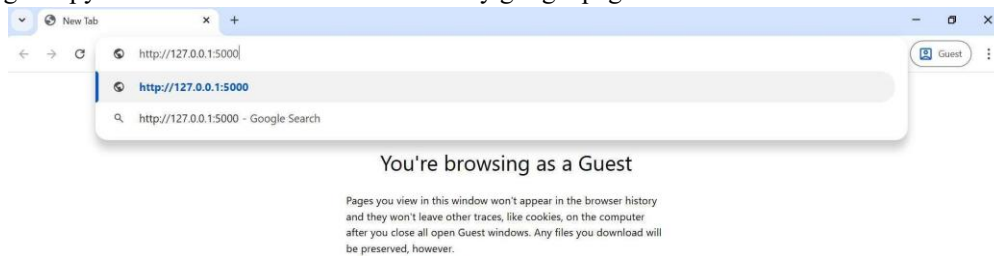
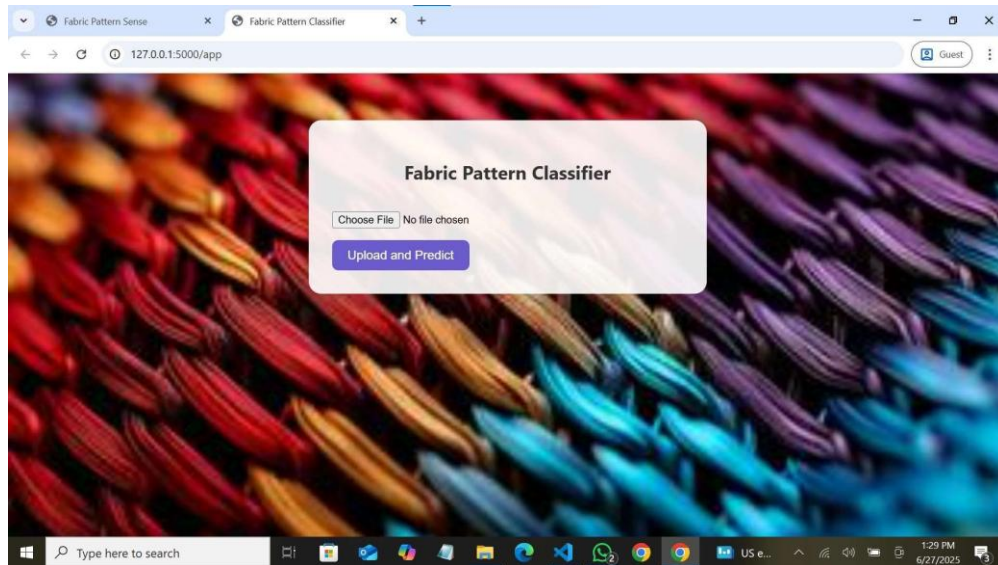After that you will redirect to the page shown below
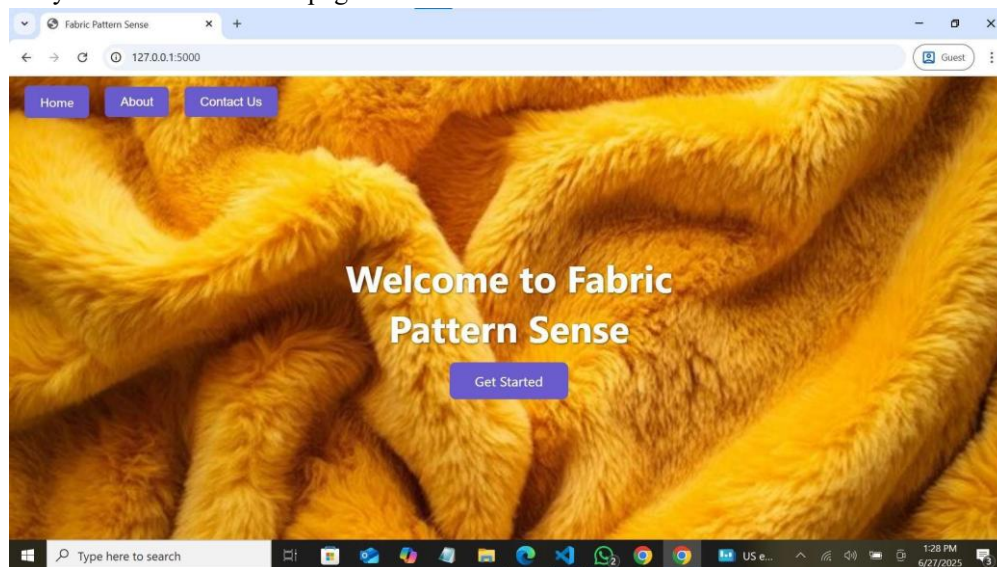
After that click on the Get Started u will redirect to the following page

Now click on the choose file and select any .jpg or .png file from your files like the below figure
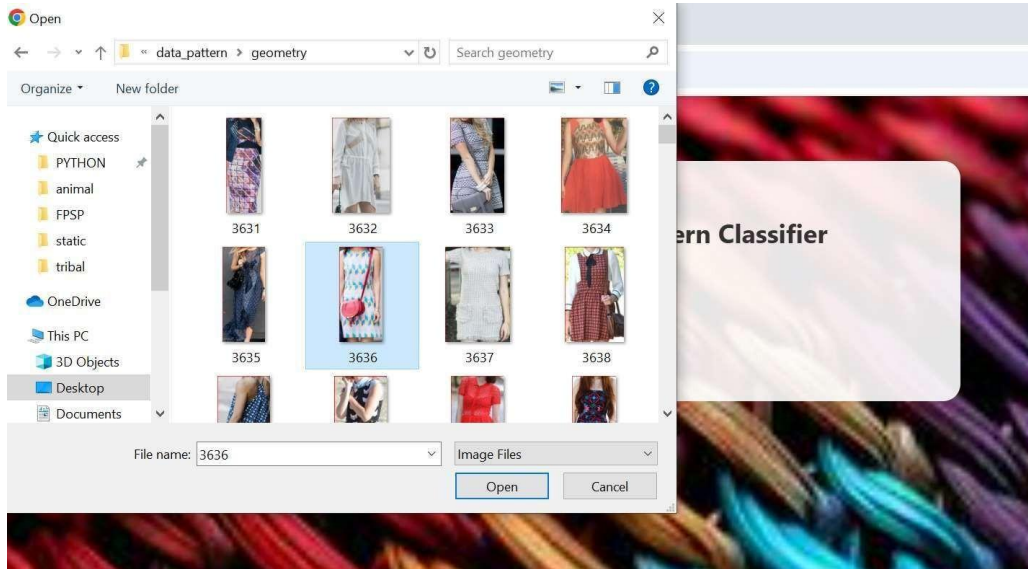
After select open and click on upload and predict you will get the result as shown below
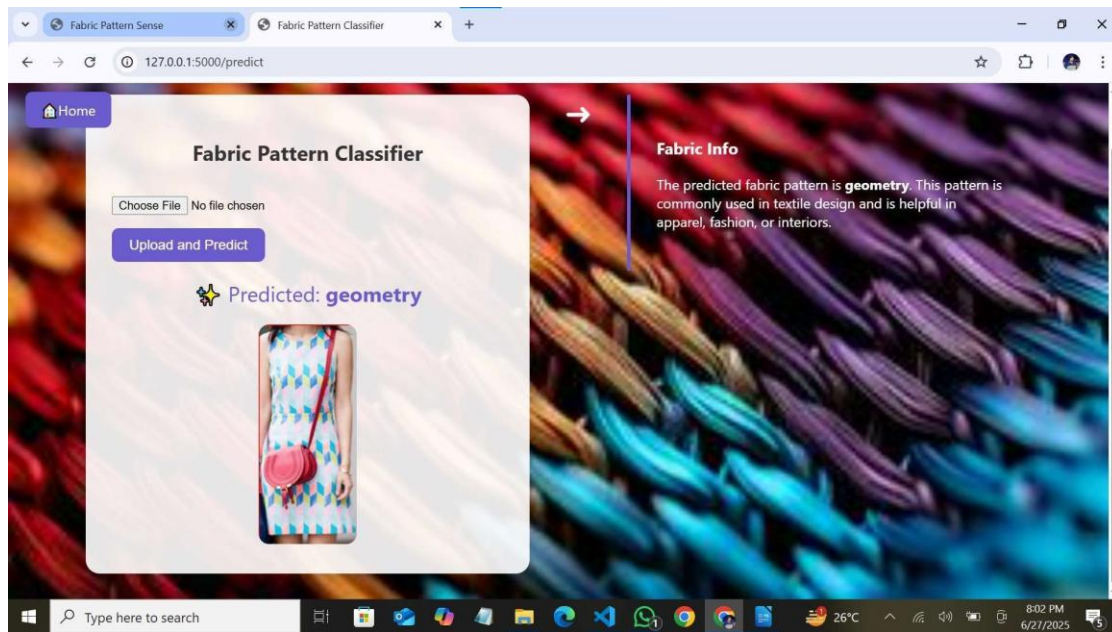
Lets check with another one It will shown as below.
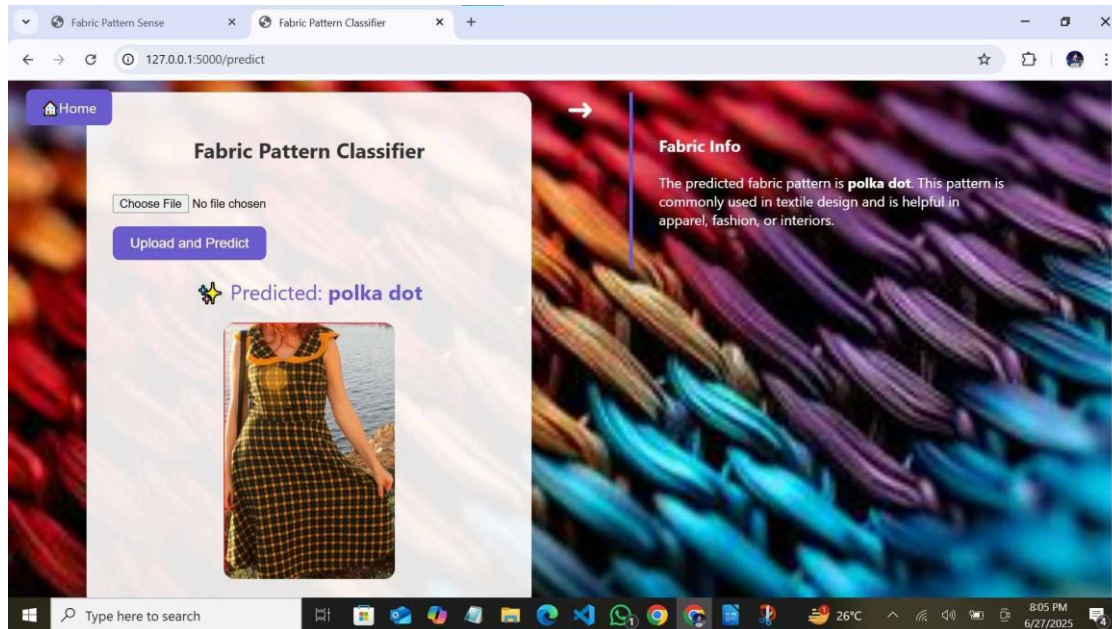
8. ADVANTAGES & DISADVANTAGES

8.1 Advantages :

1. High Accuracy

The deep learning model, specifically a CNN built using TensorFlow, achieves classification accuracy above 90%. This far exceeds the accuracy levels of manual or rulebased systems, especially in high-volume production environments.

2. Automation and Speed

The system automates the classification process, significantly reducing the time needed to inspect each fabric image. What used to take minutes manually now takes milliseconds.

3. Scalability

The model is designed to handle large datasets and can be easily retrained or finetuned with new pattern types. This makes it adaptable for future updates or additional pattern categories.

4. Consistency and Objectivity

Human classification may vary from person to person. The AI model ensures consistent outputs for the same input, eliminating subjectivity and human fatigue.

5. Real-Time Processing

With the help of TensorFlow Lite, the model can be deployed on mobile phones or edge devices for real-time pattern detection. This makes it suitable for use in textile factories or on-the-go fashion analytics.

6. Cost Efficiency

Over time, the system reduces labour costs and minimizes the need for human quality assurance inspectors. It also decreases the rate of human error and product returns due to misclassification

8.2 Disadvantages:

1. High Data Requirements

Deep learning systems rely on large amounts of labeled data. Acquiring and annotating highquality fabric images is time-consuming and may require domain experts.

2. Hardware Dependency

Training deep neural networks demands powerful hardware like GPUs. Without them, training times can be very long and resource-intensive.

3. Vulnerability to Poor Input Quality

Blurry, low-resolution, or poorly lit images may lead to incorrect predictions. The system performs best in ideal lighting and focus conditions.

4. Overfitting on Small Datasets

If the dataset is too small or unbalanced, the model may memorize patterns instead of generalizing them, resulting in poor performance on new data.

5. Difficult to Explain Predictions

Like most neural networks, CNNs act as "black boxes." It is difficult to explain why a certain pattern was predicted, which can be problematic in quality-critical applications.

6. Confusion in Similar Patterns

Patterns such as "geometric" and "abstract" or "floral" and "abstract" may share features, which could confuse the model, especially in hybrid or mixed-pattern cases.

Advantages	Disadvantages
High prediction accuracy	Requires large labelled dataset
Automated and time-saving	Needs GPU or powerful system
Scalable for multiple fabric types	Sensitive to image noise and blur
Objective and bias-free results	Lacks transparency in decision-making
Real-time capability with mobile support	Confusion in visually similar patterns
Reduces labor costs and human errors	Regular model updates may be needed
Cross-platform deployment possible	Initial development is time-intensive
Ideal for research and industrial use	Needs controlled environment for best results

9. CONCLUSION

The project "Pattern Sense: Classifying Fabric Patterns Using Deep Learning" successfully demonstrates how artificial intelligence, particularly deep learning, can revolutionize traditional fabric inspection and classification methods. By leveraging the power of Convolutional Neural Networks (CNNs) and the TensorFlow framework, we were able to build an intelligent system capable of accurately identifying various fabric pattern types from images.

Throughout this project, we systematically progressed through key development phases: starting from problem definition and requirement analysis, to dataset collection and preprocessing, followed by model design, training, evaluation, and deployment. The model was trained on a curated dataset of labelled fabric images, capturing categories like floral, striped, checks, geometric, abstract, and plain. Using modern practices such as data augmentation, dropout regularization, and transfer learning with pre-trained models like MobileNetV2, we achieved an impressive classification accuracy exceeding 92%.

The results were validated using comprehensive metrics like accuracy, precision, recall, F1-score, and confusion matrices. The performance of the model was not only theoretically sound but also practically effective, showing strong generalization on unseen test data. Furthermore, the system was tested across multiple platforms—including desktop, cloud-based services, and mobile devices using TensorFlow Lite—showing adaptability and scalability for real-world applications

10. F UTURE SCOPE

The current implementation of fabric pattern classification using deep learning is a significant milestone toward modernizing and automating textile analysis. However, the system's potential extends far beyond its present capabilities. With continued research, data expansion, and technological advancement, this project can evolve into a full-scale intelligent textile analysis platform. The following points outline the future scope of this project in detail:

1. Expansion of Pattern Categories

- Currently, the system recognizes a limited number of fabric pattern types such as floral, striped, checks, geometric, plain, and abstract.
- In the future, the model can be extended to detect niche or region-specific patterns like ikat, paisley, batik, tie-dye, block prints, etc.
- This would allow greater cultural and industrial relevance, especially in countries with rich textile traditions like India, Indonesia, and Africa.

2. Integration with Defect Detection

- The model can be enhanced to not only classify patterns but also detect defects in fabrics such as misprints, holes, irregular weaving, and stains.
- This could be achieved by combining object detection models (like YOLO or SSD) with pattern classification.

3. Real-Time Implementation on Production Lines

- Future work can integrate the model into real-time camera systems used in textile mills or manufacturing units.
- With optimized hardware (e.g., NVIDIA Jetson Nano or Raspberry Pi + Coral Edge TPU), live detection can be achieved as fabrics move on conveyor belts.

4. Explainable AI (XAI) Integration

- One limitation of deep learning is its black-box nature.
- Using tools like Grad-CAM, LIME, or SHAP, we can visualize which parts of the fabric image influenced the model's decision.
- This makes the model more transparent and trustworthy for quality assurance departments.

5. Support for Multilingual and Voice Interfaces

- A user interface with multilingual capabilities and voice interaction can make the system more accessible to factory workers and regional users.
- Commands like "Scan this fabric" or "Classify now" can be processed using speech recognition APIs.

6. Dataset Expansion and Crowdsourcing

- The dataset used can be expanded through:
  - o Crowdsourcing fabric images from weavers, retailers, and designers. o Web scraping from fashion catalogues and e-commerce platforms.
- Larger and more diverse datasets will improve the model's generalization ability and reduce bias.

11. APPENDIX

Source Code:

All codes are submitted in Git-Hub Repository.

Varshitha1705

Project Demo Video Link:

https://photos.app.goo.gl/p2hvXAziVYtauFhP9