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:

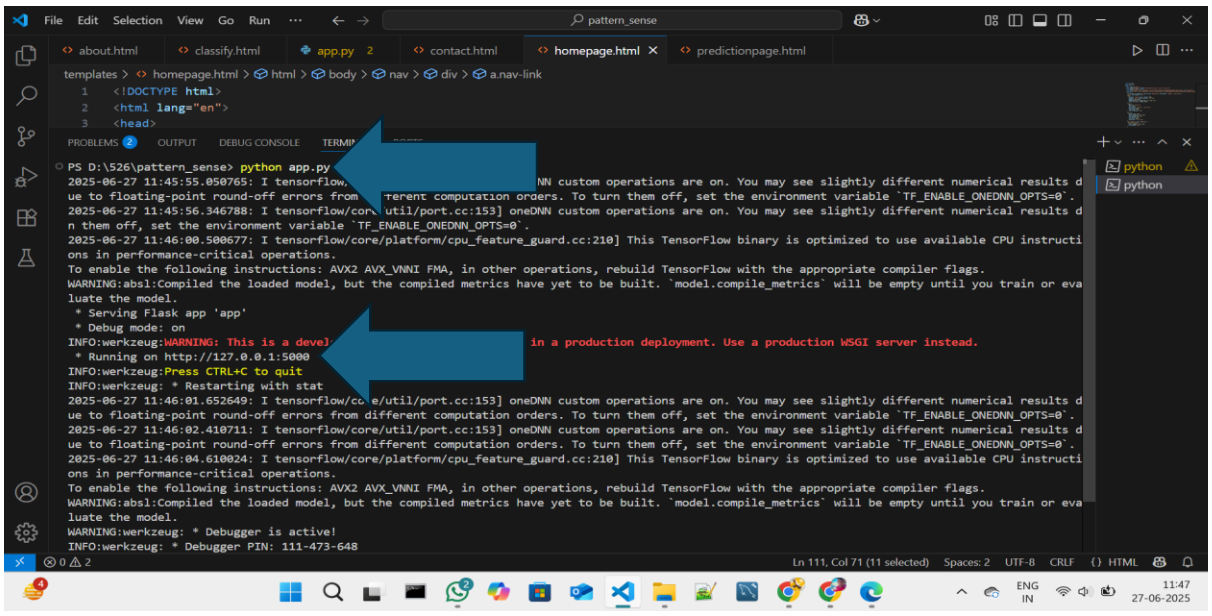
- **Automating the quality control process** in fabric production to minimize the reliance on manual labour and reduce the time taken for inspections.
- **Reducing human error** that may occur due to fatigue, subjective judgment, or oversight in pattern verification.
- **Enhancing the speed and scalability** 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.

7. RESULTS

7.1 Output Screenshots

The complete execution of the Smart Sorting application is shown in the images step by step as shown below.

Step 1: Run the app.py code and you will get a link in terminal as https://127.0.0.1.5000 to access web page and to do the other process.

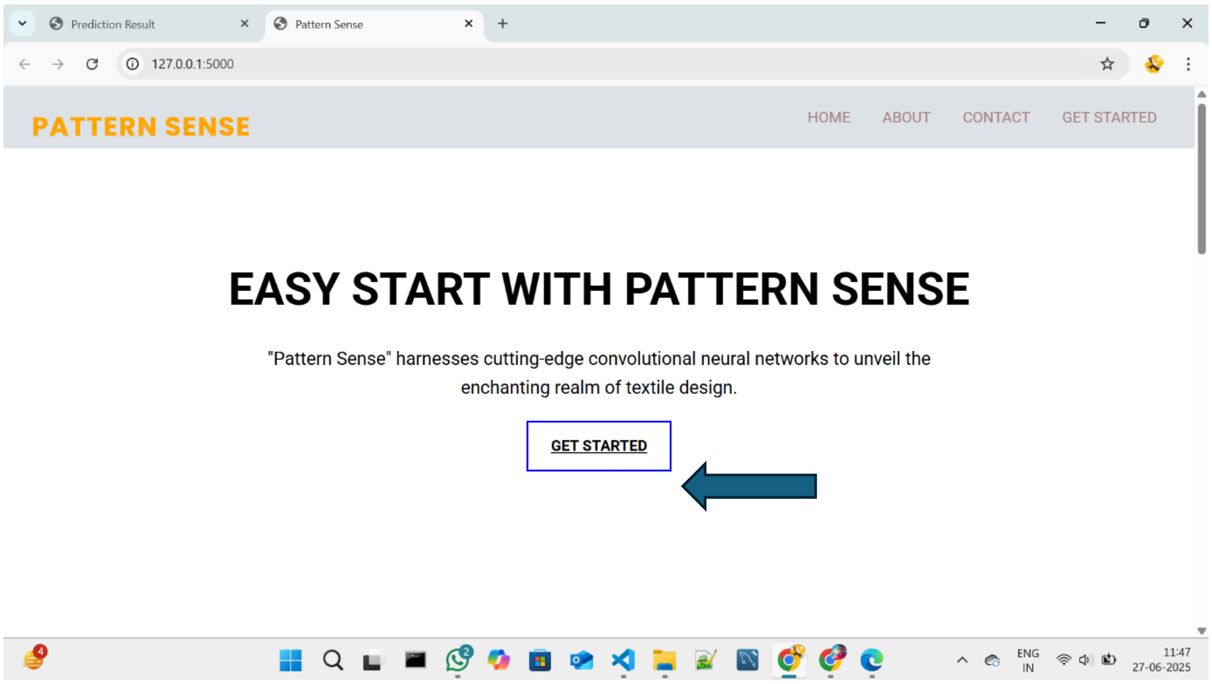


```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
PS D:\526\pattern_sense> python app.py
2025-06-27 11:45:55.050765: I tensorflow[illegible]NN custom operations are on. You may see slightly different numerical results d
ue to floating-point round-off errors from [illegible]ferent computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-06-27 11:45:56.346788: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results d
n them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-06-27 11:46:00.500677: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructi
ons in performance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI 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 eva
luate the model.
 * Serving Flask app 'app'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a devel[illegible] 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
2025-06-27 11:46:01.652649: I tensorflow/co[illegible]e/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results d
ue to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-06-27 11:46:02.410711: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results d
ue to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-06-27 11:46:04.610024: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructi
ons in performance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI 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 eva
luate the model.
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 111-473-648
```

Fig 7.1.1: Code running in Terminal

Step 2: Click on that link a web page of fabric patterns will be open in the web browser.



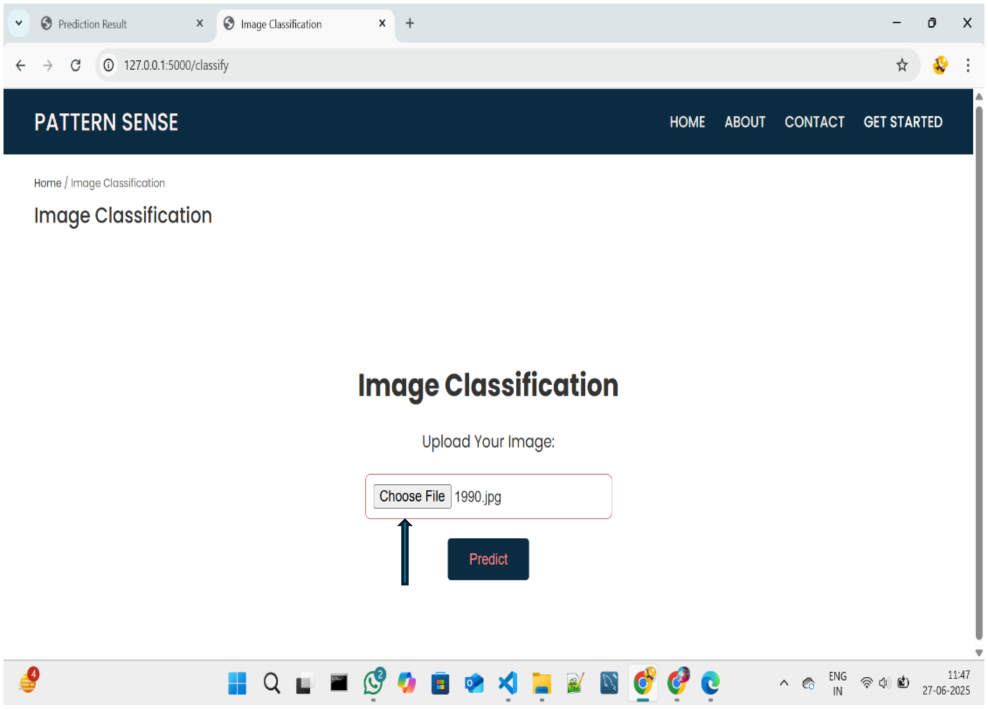
EASY START WITH PATTERN SENSE

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

GET STARTED

Fig 7.1.2: Fabric Pattern sense Home Page

Step 3: Click on GET STARTED or PREDICT option to open the prediction page.



.

Fig 7.1.3: Prediction page in pattern sense

Step 4: Click on choose file option to choose the images that need to predict

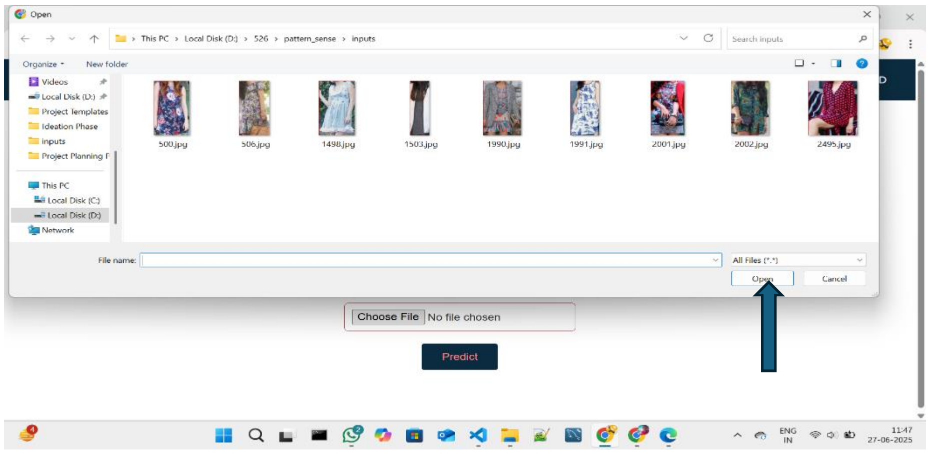


Fig 7.1.4: Window to choose image for prediction

Select any image for prediction and click on Open.

Step 5: Click on Predict to predict the quality of selected image.

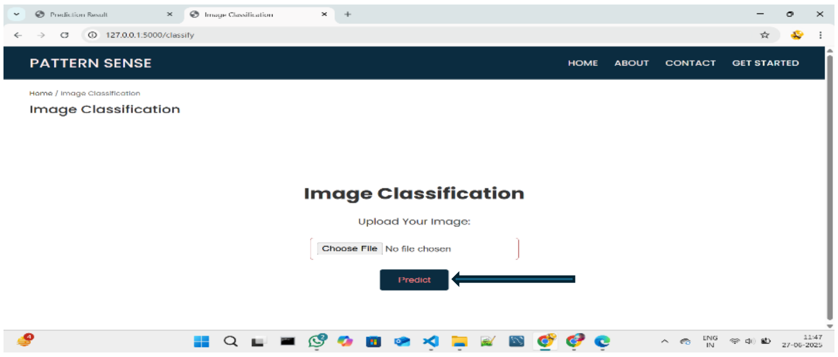
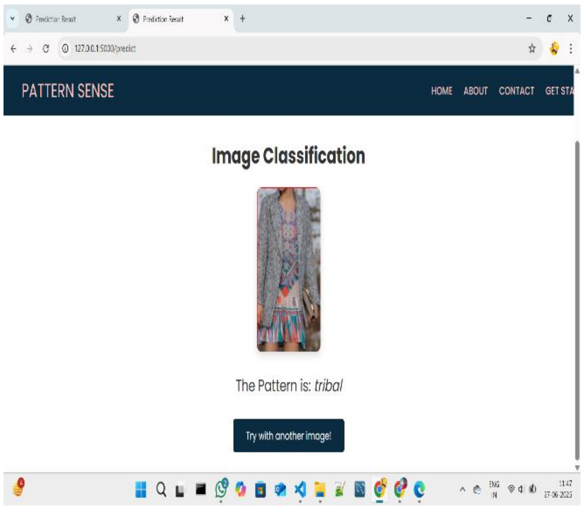
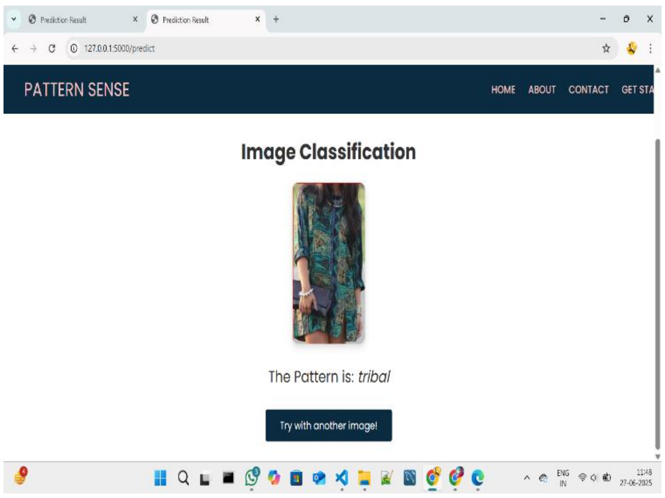


Fig 7.1.5: Image selected in prediction page

Step 6: After clicked on the predict button the model predicts the image quality and displays the quality of image.
The below images are the some of samples tested for prediction.





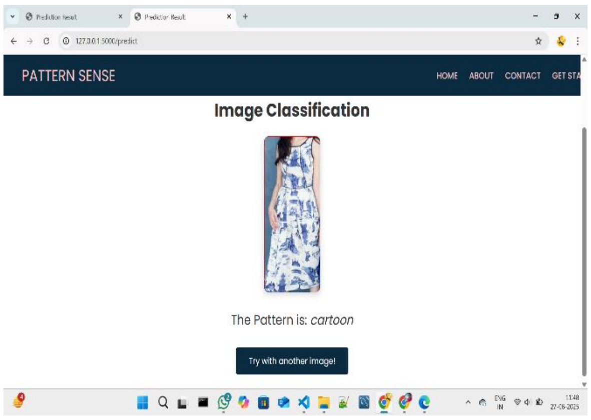
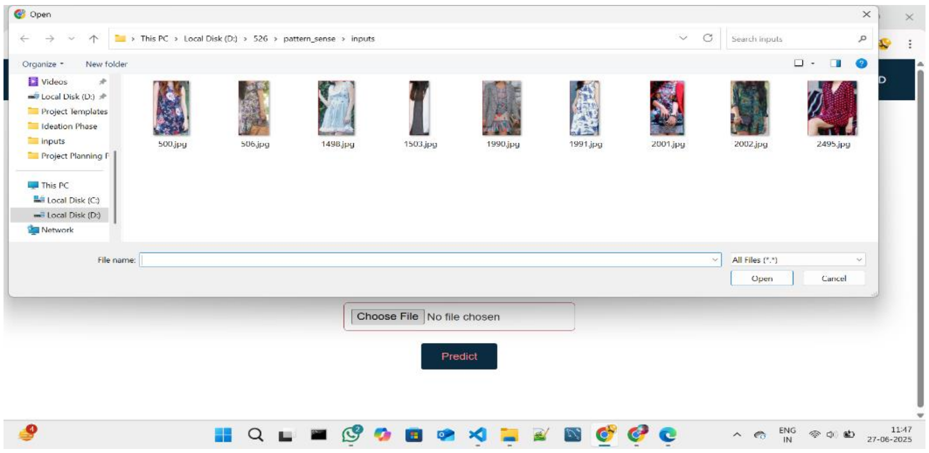
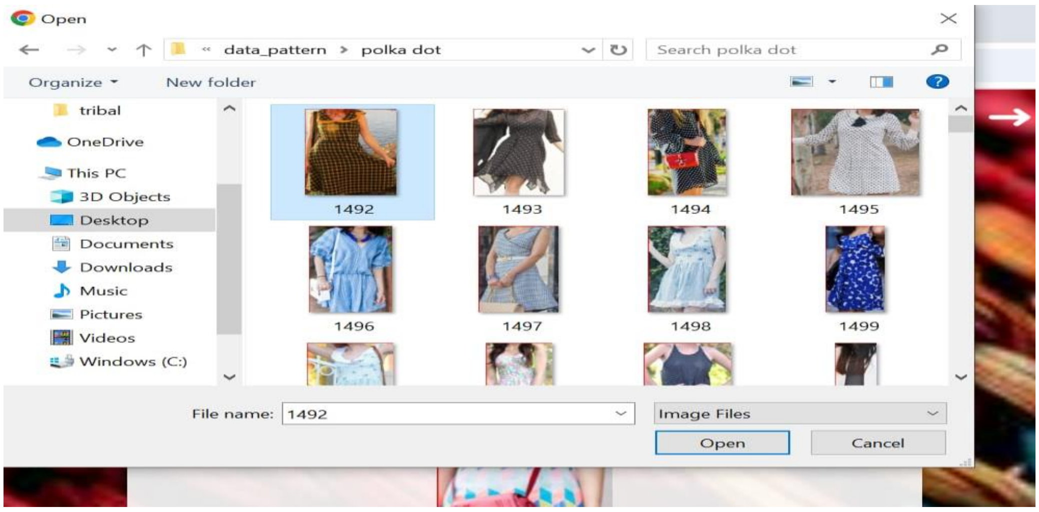


Fig 7.1.6: Prediction output for the several inputs with accuracy

After the Images Predicted the images will be stored in the uploads folder as fresh and rotten as shown in the figure given below.





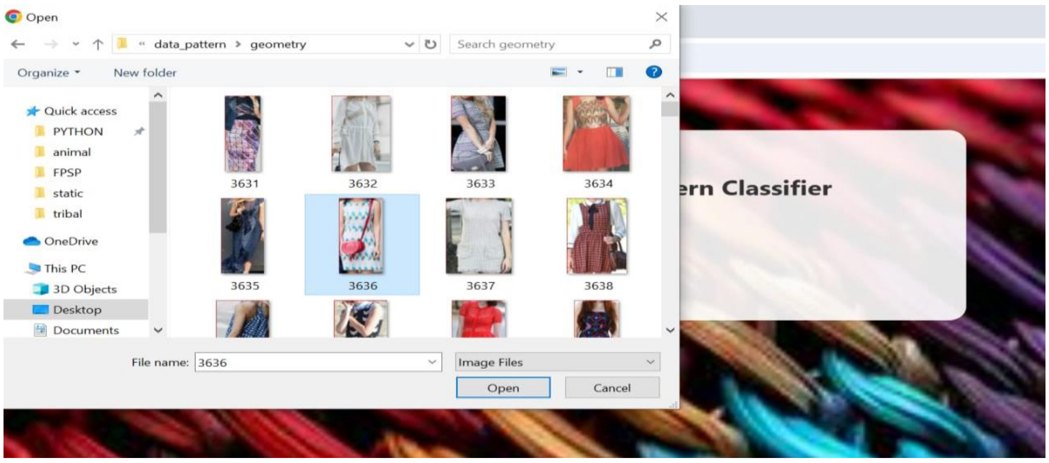


Fig 7.1.7: Folder Structure to store predicted images

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 rule-based 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 fine-tuned 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 high-quality 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. FUTURE 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:
  - **Crowdsourcing fabric images** from weavers, retailers, and designers.
  - **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.

Git-Hub Repository Link:

https://github.com/JamunaSomineni/Pattern-Sense

Dataset Link: **https://www.kaggle.com/datasets/nguyngiabol/dress-pattern-dataset**

or
Use the dataset_downloader code in Git-Hub Repository

Project Demo Link:

https://drive.google.com/file/d/1UrcIGw5ZFwvoOvn5i39NIdt2OKl8JVVk/view?usp=sharing