

Fabric Classification Using Opencv2

Aman Adusumilli (22261A6601)

&

B Saaketh (22261A6603)

Under the guidance of

Dr. M. Rama Bai

Head of the Department, ET

Department of Emerging Technologies

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

Autonomous | (Affiliated to Jawaharlal Nehru Technological University Hyderabad) GANDIPET,
HYDERABAD – 500075 TELANGANA, INDIA

ABSTRACT

This project focuses on using OpenCV2 to classify different types of fabric, aiming to enhance the efficiency of automated visual inspection in the textile industry. Traditional methods of fabric inspection are manual, time-consuming, and prone to human error. By automating this process, the system provides a faster, more consistent, and scalable solution for fabric identification and quality control.

The system leverages a combination of image processing techniques such as pre-processing, texture analysis, and feature extraction to analyze fabric images. OpenCV2 is used to process these images by enhancing clarity, removing noise, and standardizing input formats. Key visual features like texture patterns and color distributions are extracted using techniques such as Local Binary Patterns (LBP), GLCM, and color histograms. These features are then used in pattern recognition algorithms to classify fabrics into distinct categories with high accuracy.

Early results demonstrate that the proposed system is both scalable and effective in classifying various fabric types. It ensures reliable performance across different imaging conditions and fabric styles, making it highly applicable for real-world industrial use. This approach not only improves quality control in textile manufacturing but also lays the foundation for further enhancement using advanced machine learning models in the future.

1.INTRODUCTION

The textile industry relies heavily on the accurate identification and classification of various fabric types for quality control, inventory management, and automation purposes. Traditionally, this classification has been carried out manually by skilled professionals, leading to inconsistencies, higher labor costs, and reduced scalability. In recent years, advancements in computer vision and machine learning have made it possible to automate such processes, enhancing efficiency and consistency.

This project investigates a computer vision-based solution to classify fabrics using the OpenCV2 library in Python. OpenCV is a powerful open-source toolkit that provides a wide range of tools for image analysis, including filtering, edge detection, texture analysis, and feature extraction. Our project focuses on leveraging these tools to process fabric images and extract meaningful descriptors that help distinguish between different fabric types such as cotton, silk, wool, denim, and nylon.

By implementing both rule-based and machine learning approaches, we evaluate the feasibility of using statistical texture metrics like entropy, standard deviation, and edge density to classify the fabrics. We also explore the use of supervised learning models, trained on labeled datasets, to improve the prediction accuracy. The overall goal is to develop a robust, scalable, and explainable fabric classification pipeline that can assist in quality control applications in the textile industry.

1.1 Problem Definition

Manual inspection of fabric types is labor-intensive and often inconsistent due to human subjectivity and fatigue. In mass production environments, the need for a consistent, fast, and scalable solution for fabric identification is critical. The challenge lies in automating the visual recognition of textures and patterns to classify fabrics accurately using minimal hardware and accessible software tools. This project aims to address these challenges by creating a fabric classification system using OpenCV2, which employs texture-based feature extraction and pattern recognition for reliable classification.

1.2 Existing Applications

While automated fabric classification using computer vision is still an emerging field, there are a few notable applications and research-backed use cases where similar technology is being tested or gradually implemented. Despite its potential, adoption has been limited due to high setup costs, variability in fabric appearance under different lighting, and a lack of standardized datasets.

1.3 Proposed Web Application

To enhance accessibility and usability of the fabric classification system, a web-based application is proposed as the next step. This platform will serve as an interactive interface for users to upload fabric images and receive instant classification results using the trained model or rule-based logic. The proposed system would eliminate the need for local setup and bring the tool into the hands of industry professionals, textile workers, or even everyday consumers.

1.4 Requirements Specification

To achieve the project's goals, the following software and hardware requirements have been identified:

1.4.1 Software Requirements

IDE: Visual Studio Code 1.77 or later

Framework: Python

Libraries: Streamlit, OpenCV

1.4.2 Target Devices

RAM: 4 GB or more

Storage: 1 GB or more

Connectivity: Wi-Fi or cellular network

2. LITERATURE SURVEY

[1] Automated Fabric Defect Classification in Textile Manufacturing Using Advanced Optical and Deep Learning Techniques

Published in 2025, this work combined an optimized optical environment (dark field + backlight) with deep learning models such as YOLOv8, YOLOv11, RefineDet, DETR, and Faster R-CNN for real-time fabric defect detection. YOLOv11 achieved high accuracy (mAP 81%) and fast detection speed, enabling real-time application suitability. Despite these strengths, the approach depends heavily on well-annotated data and struggles with lighting sensitivity and motion blur.

[2] Development of a Fabric Classification System Using Drapability and Tactile Characteristics

This project focused on classifying fabrics based on their drapability and tactile features. The methodology involved using fuzzy c-means clustering and artificial neural networks (ANN) for classification. The system achieved a commendable accuracy of 83.5%, offering a more objective approach to fabric classification. However, it faced challenges due to inconsistencies in subjective evaluation, which could affect its practical reliability.

[3] The Improved Method in Fabric Image Classification Using Convolutional Neural Network

This project proposes a lightweight CNN combining VGG, Inception, and ResNet with GAP and CLAHE preprocessing for fabric image classification. Trained on the FID dataset, it achieved 99.55% accuracy with reduced computational complexity. The model performs well on small and imbalanced datasets with high efficiency. However, it's limited to wool fabrics and not fully validated for real-time use.

[4] Deer Hunting Optimization with Deep Learning-Driven Automated Fabric Defect Detection and Classification

In this 2024 study, the authors implemented MobileNetV3 for feature extraction and used Deer Hunting Optimization (DHO) for hyperparameter tuning, followed by BiLSTM for classification. The system achieved a high test accuracy of 96.54% with low computation time, outperforming models like YOLOv5 and Faster R-CNN. However, it remains sensitive to texture and lighting variations and requires large, well-labeled datasets for effective training.

[5] Automatic fabric defect detection in textile images using a labview based multiclass classification approach

This 2024 study applied YOLOv4 for defect detection and EfficientNet for classification. The approach included preprocessing steps like Gaussian blur and sharpening, along with k-means clustering. It achieved an accuracy of 98.34%

and low detection latency (0.051s/image), showing robustness against noise and distortions.

[6] Woven Fabric Pattern Recognition and Classification Based on Deep Convolutional Neural Networks

This work utilized ResNet-50, a deep convolutional neural network, in combination with data augmentation and transfer learning, to recognize and classify woven fabric patterns. Although the system performed well for woven fabrics, it struggled with nonwoven materials due to limitations in the dataset, both in quantity and diversity.

[7] Patterned Fabric Defect Detection and Classification (FDDC) Techniques: A Review

This comprehensive review evaluated various defect detection techniques for patterned fabrics, including wavelet analysis, Fourier transforms, and Local Binary Patterns (LBP). It provided valuable insights into the available methodologies for identifying fabric defects.

[8] Application of Computer Vision in the Automatic Identification and Classification of Woven Fabric Weave Patterns

This study applied Fuzzy C-Means (FCM) clustering and a two-stage Back Propagation Neural Network (BPNN) to classify woven fabric patterns. The approach improved classification accuracy compared to earlier methods. Nevertheless, it was sensitive to varying optical conditions and limited to basic weave types.

Table 2.1 Comparative study of existing models

S.No	Author(S)	Title	Year	Methodology	Merits	Demerits
1	Hieu Dau Sy, Phuc Dang Thi, Huy Vo Gia, Khang Le Nguyen An	Automated Fabric Defect Classification in Textile Manufacturing Using Advanced Optical and Deep Learning Techniques	2025	Combined an optimized optical setup (dark field + back light) with deep learning models like YOLOv8, YOLOv11, RefineDet, DETR, and Faster R-CNN for real-time defect detection	YOLOv11 achieved 81% mAP with 22.1 ms/image speed, enabling real-time detection and cost-effective hardware use for small/blurry defects.	Requires diverse, well-annotated training data; performance of RefineDet and DETR drops on challenging defect types; Faster R-CNN is too slow for real-time use; system performance sensitive to lighting and motion blur
2	Somin Lee, Yoojung Han, Changsang Yun	Development of a Fabric Classification System Using Drapability and Tactile Characteristics	2024	Used fuzzy c-means clustering & ANN for classification.	Objective, 83.5% accuracy.	Subjective evaluation inconsistencies.

3	Ruihao Liu, Zhenzhong Yu, Qigao Fan, Qiang Sun, Zhongsheng Jiang	The Improved Method in Fabric Image Classification Using Convolutional Neural Network	2024	Proposed a custom lightweight CNN combining VGG, Inception, and ResNet features with GAP instead of fully connected layers; used CLAHE preprocessing on HSV images and trained using the Fabric Image Dataset (FID)	Achieved 99.55% accuracy; lower computational complexity compared to deeper CNNs; performs well on small and imbalanced datasets	Limited evaluation on real-time applications; model tested only on wool fabrics; potential overfitting risk on small class-specific datasets
4	Mohammed Maray, Ghadah Aldehim, Abdulrahman Alzahrani, Faiz Alotaibi, Safa Alsafari, Elham AbdullahAlghamdi, Manar Ahmed Hamza	Deer Hunting Optimization with Deep Learning-Driven Automated Fabric Defect Detection and Classification	2024	Used improved MobileNetV3 for feature extraction, Deer Hunting Optimization (DHO) for hyperparameter tuning, and BiLSTM for classification	Achieved high accuracy (96.54% test accuracy), superior to YOLOv5, ResNet, and Faster R-CNN; low computation time (1.75s)	Performance may drop on unseen or diverse fabric patterns; sensitive to variations in texture and lighting; requires large, well-labeled datasets for training
5	T. Meeradevi, S. Sasikala	Automatic fabric defect detection in textile images using a labview based multiclass classification approach	2024	Utilized YOLOv4 for object detection and EfficientNet for classification. Integrated pre-processing (e.g., Gaussian blur and sharpening) and k-means clustering for feature enhancement	Achieved 98.34% accuracy with 0.051 s/image speed; robust to noise and effective on real-time video.	Model performance degrades in presence of heavy occlusions or very small-scale defects; requires consistent camera calibration and lighting setup; computational load may increase with high-resolution inputs
6	Muhammad Ather Iqbal Hussain, Babar Khan, Zhijie Wang, Shenyi Ding	Woven Fabric Pattern Recognition and Classification Based on Deep Convolutional Neural Networks	2020	Applied ResNet-50 with data augmentation & transfer learning.	Limited database, struggles with nonwoven fabrics	Limited database, struggles with nonwoven fabrics.

7	D.I Oni, John Ojo, Seyi Alabi, Adeyayo Adebayo	Patterned Fabric Defect Detection and Classification (FDDC) Techniques: A Review	2018	Reviewed defect detection methods (wavelet, Fourier, LBP).	Comprehen sive analysis.	Issues with complex textures & high- resolution defects.
8	Chung-Feng Jeffrey Kuo, Chung-Yang Shih, Cheng- En Ho, Kai- Ching Peng	Application of Computer Vision in the Automatic Identification and Classification of Woven Fabric Weave Patterns	2010	Used FCM clustering & two-stage BPNN.	Better classificati on accuracy.	Sensitive to optical conditions, limited to basic weaves.

3. FABRIC CLASSIFICATION USING OPENCV2

3.1 Data Collection

The foundation of the fabric classification system lies in building a comprehensive and diverse dataset. The data collection process involves gathering a wide range of fabric images that represent different textures, weave types, color schemes, and material categories such as cotton, silk, denim, wool, nylon, and velvet. These images should be captured under consistent lighting and from various angles to account for visual diversity and real-world variability. Wherever possible, publicly available datasets may be supplemented with custom image captures using a high-resolution camera setup to ensure that the model is exposed to both standard and complex fabric surfaces. Each image in the dataset should be accurately labeled with its corresponding fabric type to facilitate effective training and validation of supervised models.

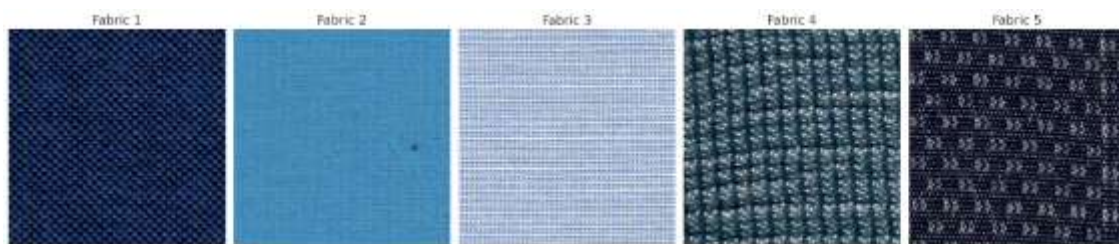


Figure 3.1 Fabric Samples

Figure 3.1 showcases five distinct fabric samples (Fabric 1 to Fabric 5), each illustrating varying patterns, textures, and weave structures. These samples represent the visual diversity in fabric types that the classification model is designed to recognize.

3.2 Image Processing

Preprocessing is a critical step in fabric classification to ensure that the input images are uniform and free from variations that can negatively impact the model's performance. It also helps in emphasizing the texture-based features required for accurate classification.

Image Resizing: All fabric images were resized to a standard dimension of 256×256 pixels. This ensures consistency across the dataset and reduces computational load while preserving enough detail for texture analysis.

Grayscale Conversion: Images were converted from RGB to grayscale using OpenCV's `cv2.cvtColor()` method. This simplifies analysis by focusing purely on texture and pattern, which are crucial for fabric recognition, while reducing the number of input channels.

Noise Reduction: Gaussian blur or median filters were optionally applied to minimize high-frequency noise without erasing texture detail. This helps to avoid false edge detections or LBP mismatches.

Data Augmentation (for training set): To improve model generalization, augmented versions of images were created using:

- Rotation ($\pm 15^\circ$)
- Horizontal/Vertical flipping
- Random cropping or zooming

Pixel Normalization: All pixel values were normalized between 0 and 1 (or scaled to 0–255 if needed for certain OpenCV operations). This allows consistent mathematical interpretation during feature extraction.

3.3 Feature Extraction

In this phase, OpenCV2 and other image processing libraries are used to extract meaningful visual features that differentiate one fabric type from another. Texture analysis techniques, such as Local Binary Patterns (LBP) and Gray-Level Co-occurrence Matrix (GLCM), help quantify surface patterns and textures. Edge detection algorithms like Canny or Sobel filters are used to identify and highlight weave patterns or fiber outlines. Additionally, color features are captured using color histograms in HSV or LAB color spaces, enabling the system to distinguish between fabrics based on their dominant color distributions. These extracted features form a multi-dimensional representation of each fabric sample, serving as the input to the classification model.

3.4 Algorithm Implementation

The final step involves implementing and optimizing machine learning algorithms to classify the fabric types based on the extracted features. Depending on the availability of labeled data and complexity of the classification task, various approaches can be employed. Unsupervised methods such as Fuzzy C-Means (FCM) clustering can group visually similar fabrics without requiring labeled data. Alternatively, supervised models like Artificial Neural Networks (ANN) or Convolutional Neural Networks (CNN) can be trained to learn complex patterns and achieve high classification accuracy. The model undergoes training, validation, and tuning processes using metrics like accuracy, precision, recall, and F1-score to ensure robust and generalizable performance. Techniques such as dropout, learning rate adjustment, and data augmentation may be applied to optimize the model for both performance and stability.

3.5 User Interface

The user interface (UI) is a key component of the proposed fabric classification system. It enables users to interact with the backend model in a seamless, intuitive manner, allowing them to upload images and receive instant classification

results. For this project, the UI was designed using Streamlit, a Python-based framework ideal for building data-centric web applications with minimal effort.

StreamLit

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers. Data scientists or machine learning engineers are not web developers and they're not interested in spending weeks learning to use these frameworks to build web apps. Instead, they want a tool that is easier to learn and to use, as long as it can display data and collect needed parameters for modeling. Streamlit allows you to create a stunning-looking application with only a few lines of code.

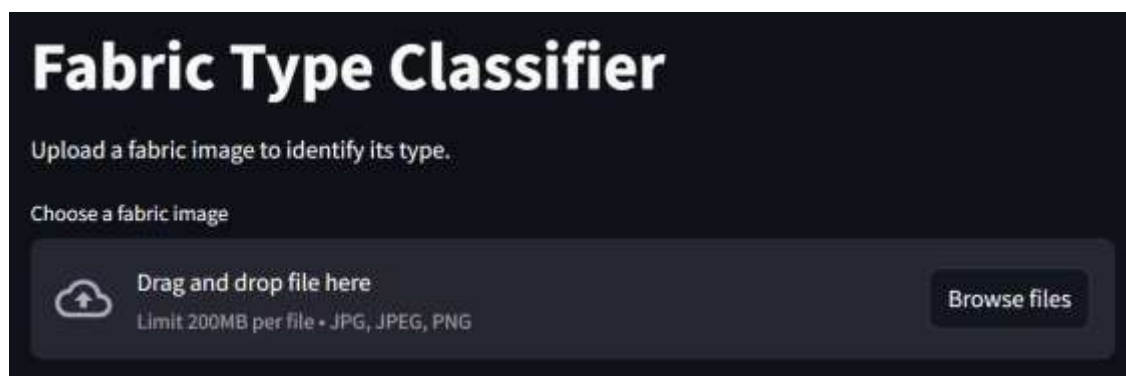


Figure 3.2 User Interface

Figure 3.2 shows the User Interface for the project, it was made using the Streamlit python library.

3.6 Image Acquisition

The first step in the fabric classification pipeline involves collecting high-quality images of various fabric types. This process requires a **high-resolution digital camera** to capture fine texture details, weave patterns, and subtle color differences that are crucial for accurate classification. The setup must be in a **controlled lighting environment**—preferably using diffused white light to eliminate shadows and reflections, ensuring uniformity across all samples. Consistency in image scale and orientation is also critical to maintain standardization for subsequent processing.

3.7 Pre-Processing

Once images are captured, pre-processing techniques are applied to enhance their quality and make them suitable for feature extraction. This step involves several tasks:

- **Noise Reduction:** Filtering methods such as Gaussian blur or median filters are used to remove image noise that may interfere with pattern detection.
- **Normalization:** Pixel values are normalized to bring uniformity across samples, reducing the influence of varying lighting conditions.
- **Contrast Enhancement:** Techniques like histogram equalization or CLAHE (Contrast Limited Adaptive Histogram Equalization) are used to highlight subtle features in fabric textures.

- **Cropping and Resizing:** Images are cropped to remove irrelevant borders and resized to a standard dimension for consistent input to the classifier.

3.8 Feature Extraction

This is the core stage where meaningful features are derived from the fabric images using OpenCV2 and associated libraries. The goal is to capture the unique visual properties of each fabric type:

- **Texture Analysis:** Use Gray Level Co-occurrence Matrix (GLCM), Local Binary Patterns (LBP), or Gabor filters to extract texture features.
- **Color Analysis:** Convert images to various color spaces (e.g., HSV, LAB) and compute color histograms or dominant color profiles.
- **Weave Pattern Detection:** Edge detection and pattern recognition methods (e.g., Hough Transform or morphological filters) are applied to identify repeating patterns indicative of the fabric weave.

3.9 Classification Model

In this stage, the system uses machine learning algorithms to categorize the fabrics based on the extracted features. Common approaches include:

- **Supervised Learning:** Classifiers like Support Vector Machines (SVM), Random Forests, or Convolutional Neural Networks (CNN) are trained on labeled fabric images.
- **Unsupervised Learning:** Clustering methods such as K-means or Fuzzy C-Means can be used when labels are unavailable, grouping fabrics based on visual similarity.
- **Hybrid Models:** Combining traditional feature extraction with deep learning architectures (e.g., feeding texture features into a CNN for better accuracy) can further improve performance.

3.10 Evaluation & Optimization

To ensure the reliability and scalability of the system, extensive testing and fine-tuning are carried out:

- **Dataset Diversity:** The model is tested across a wide range of fabrics—cotton, denim, silk, nylon, wool, etc. captured under varying conditions to assess robustness.
- **Performance Metrics:** Accuracy, precision, recall, F1-score, and confusion matrices are used to evaluate the classifier's effectiveness.
- **Model Optimization:** Hyperparameter tuning, cross-validation, and techniques like dropout or regularization are employed to improve model generalization and prevent overfitting.
- **Feedback Loop:** Misclassifications are analysed to enhance feature extraction and model parameters iteratively.

4.DESIGN

The design phase of the project involves a comprehensive plan for the application's architecture, components, and user interface. This step ensures that all necessary elements are identified and properly integrated to achieve the desired functionality.

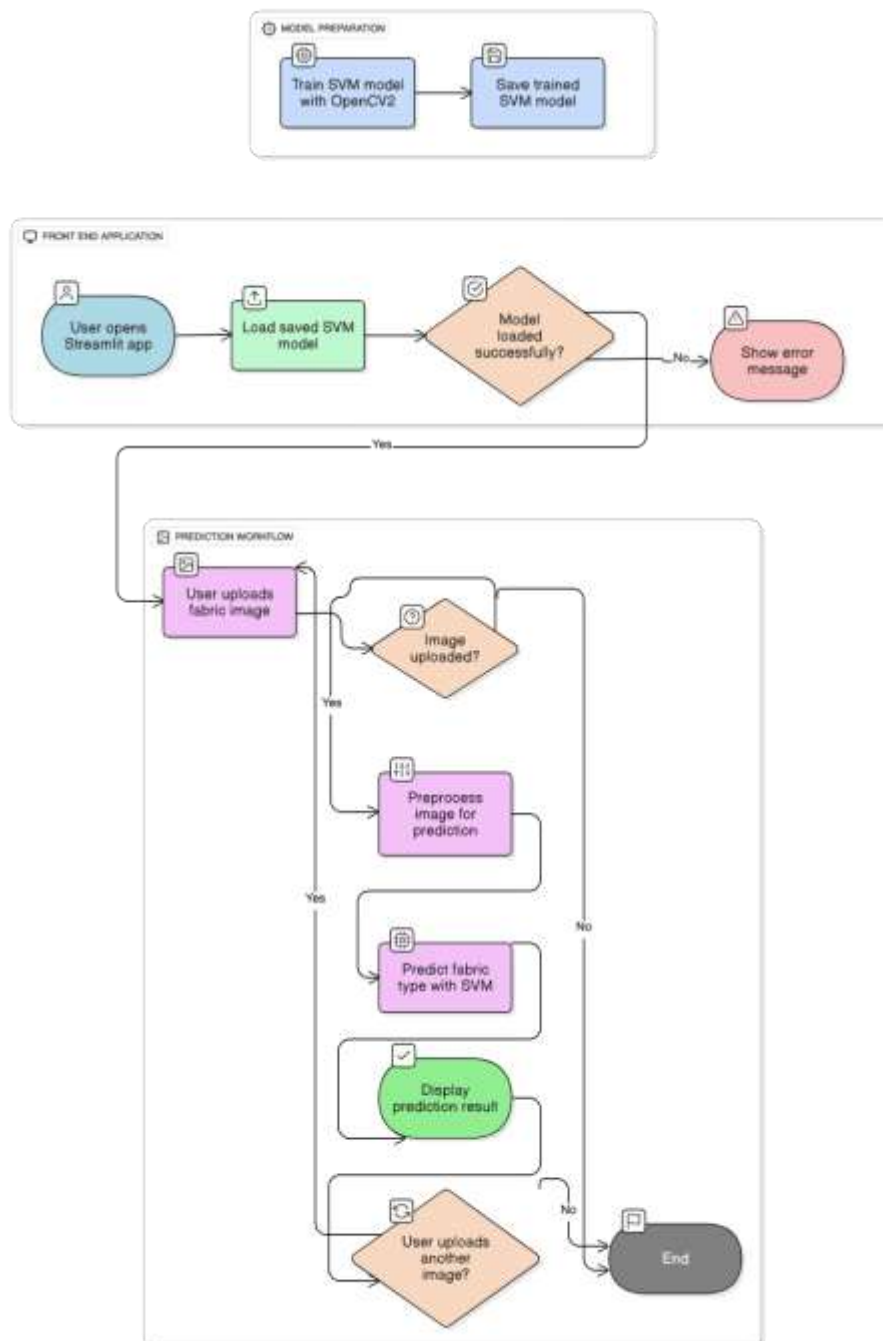


Figure 4.1 Architecture Diagram for Fabric Classification

Figure 4.1 The above UML activity diagram illustrates the complete workflow of the fabric classification system using OpenCV2 and a trained SVM model. It is divided into three key sections: Model Preparation, Front-End Application Initialization, and Prediction Workflow.

Model Preparation

This section outlines the backend training process. It begins with training the Support Vector Machine (SVM) model using OpenCV2 and extracted image features. Once the model is trained, it is saved for later use in the prediction phase.

Front-End Application

In this part of the workflow, the user initiates the application through the Streamlit interface. The system then attempts to load the previously trained SVM model. If successful, it proceeds to the prediction workflow; otherwise, it displays an appropriate error message.

Prediction Workflow

This is the core interactive loop for classification. The user uploads a fabric image, which the system checks for validity. Once an image is confirmed, it undergoes preprocessing for feature extraction. The preprocessed data is then fed into the SVM model to predict the fabric type. The result, including the fabric class and confidence score, is displayed to the user. The workflow then prompts the user to either upload another image or end the session.

5. TESTING AND DEBUGGING

Testing and debugging were essential to ensure the accuracy and stability of the Fabric Classification Using OpenCV2 system. Unit tests were performed on core components such as preprocessing, feature extraction, and the classifier to validate functionality. Integration testing confirmed smooth operation across the complete pipeline—from image upload to result display. Usability testing via the Streamlit interface ensured clarity and responsiveness. Debugging tools like Python's PDB and logging libraries were used to track and fix issues. Continuous testing and refinement ensured the system remained reliable and user-friendly across various fabric types and scenarios.

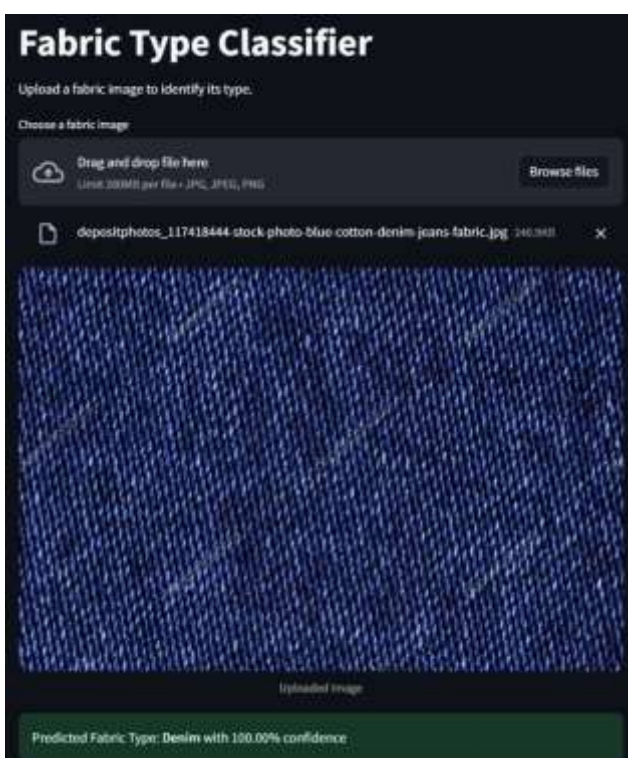


Figure 5.1 (Testing Case I) shows a sample image of denim fabric uploaded to the system. The model accurately predicts the fabric type as Denim with a high confidence level of 100.00%, indicating strong alignment between the extracted texture features and the trained model's understanding of denim.

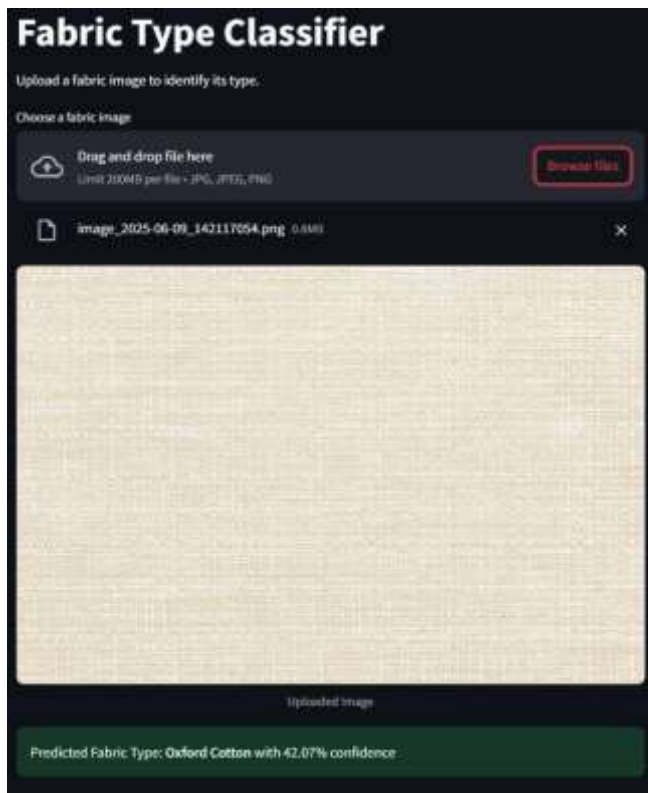


Figure 5.2 Testing case II

Figure 5.2 (Testing Case II) presents a lighter, woven fabric image uploaded for testing. The model predicts the fabric type as Oxford Cotton with a lower confidence of 42.07%, suggesting a probable match but also reflecting limitations in certainty, possibly due to similarity with other fabric types or lack of similar examples in the training data.

6. CONCLUSION AND FUTURE SCOPE

Conclusion:

This project presents a comprehensive and automated fabric classification system developed using OpenCV2 and machine learning techniques. The system is designed to address the complexities associated with identifying various types of fabrics based on their visual and structural characteristics. By integrating advanced image processing techniques, including high-resolution image acquisition, robust pre-processing pipelines, and detailed feature extraction—the system captures the critical information embedded in fabric textures, colors, and weave patterns. These extracted features are then analyzed and classified using carefully chosen algorithms such as neural networks or clustering models, optimized to handle both simple and intricate fabric types.

One of the key strengths of the system lies in its adaptability and reliability across diverse imaging conditions. The controlled acquisition setup, combined with normalization and contrast enhancement methods, ensures consistent input

quality regardless of lighting variations or environmental disturbances. Furthermore, the use of OpenCV2 provides a flexible and efficient platform for image analysis, enabling real-time processing capabilities and scalability for larger datasets or commercial applications.

Through continuous testing and optimization, the system demonstrates high classification accuracy and robustness, making it a viable tool for industries such as textiles, fashion, quality control, and e-commerce. The incorporation of feedback loops and evaluation metrics like precision, recall, and F1-score allows for ongoing improvement and refinement of the model. Ultimately, this project showcases how a fusion of computer vision and machine learning can offer a practical, intelligent solution for fabric identification—simplifying manual inspection processes, reducing error rates, and accelerating fabric analysis in both research and industrial settings.

Future Scope:

The fabric classification system developed using OpenCV2 and machine learning shows potential for practical use in textile manufacturing, quality control, and retail. It aims to provide accurate, automated identification of fabrics based on texture, color, and weave, making the process faster and more consistent than manual methods.

The workflow includes image acquisition under controlled lighting, followed by pre-processing steps like resizing and noise reduction. Features such as texture patterns and color distributions are extracted using techniques like LBP, GLCM, and color histograms, and classified using models like SVM, CNN, or clustering algorithms.

To ensure real-time performance, the system may be optimized through model tuning, compression, and GPU support. It is designed to reduce human error, handle varying conditions, and serve as a scalable solution for fabric inspection. Future improvements may include deeper learning integration and expanded deployment options.

REFERENCES

- [1] Sy, H. D., Thi, P. D., Gia, H. V., & An, K. L. N. (2025). Automated fabric defect classification in textile manufacturing using advanced optical and deep learning techniques. Utilized YOLOv8, YOLOv11, RefineDet, DETR, and Faster R-CNN with optimized optical setup.
- [2] Lee, S., Han, Y., & Yun, C. (2024). Development of a fabric classification system using drapability and tactile characteristics.
- [3] Liu, R., Yu, Z., Fan, Q., Sun, Q., & Jiang, Z. (2024). The improved method in fabric image classification using convolutional neural network.
- [4] Maray, M., Aldeib, G., Alzahrani, A., Alotaibi, F., Alsafari, S., Abdullah, E., Alghamdi, A., & Hamza, M. A. (2024). Deer hunting optimization with deep learning-driven automated fabric defect detection and classification. Used MobileNetV3, DHO for tuning, and BiLSTM for classification.

- [5] T. Meeradevi , S. Sasikala (2024). Automatic fabric defect detection in textile images using a labview based multiclass classification approach. Employed object detection, classification, and preprocessing techniques for robust defect identification.
- [6] Hussain, M. A. I., Khan, B., Wang, Z., & Ding, S. (2020). Woven fabric pattern recognition and classification based on deep convolutional neural networks.
- [7] Oni, D. I., Ojo, J., Alabi, S., & Adebayo, A. (2018). Patterned fabric defect detection and classification (FDDC) techniques: A review.
- [8] Kuo, C. F. J., Shih, C. Y., Ho, C. E., & Peng, K. C. (2010). Application of computer vision in the automatic identification and classification of woven fabric weave patterns.

APPENDIX

Code to create the machine learning model for fabric classification using OpenCV2, Local Binary Patterns (LBP), and Support Vector Machine (SVM):

```
import cv2
import numpy as np
import os
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import shuffle
import joblib

# Function to extract LBP features
def lbp_feature(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    lbp = np.zeros_like(gray)
    for i in range(1, gray.shape[0] - 1):
        for j in range(1, gray.shape[1] - 1):
            center = gray[i, j]
            binary = ""
            binary += '1' if gray[i-1, j-1] > center else '0'
            binary += '1' if gray[i-1, j] > center else '0'
```

```
    binary += '1' if gray[i-1, j+1] > center else '0'
    binary += '1' if gray[i, j+1] > center else '0'
    binary += '1' if gray[i+1, j+1] > center else '0'
    binary += '1' if gray[i+1, j] > center else '0'
    binary += '1' if gray[i+1, j-1] > center else '0'
    binary += '1' if gray[i, j-1] > center else '0'
    lbp[i, j] = int(binary, 2)

hist, _ = np.histogram(lbp.ravel(), bins=256, range=(0, 256))
hist = hist.astype("float")
hist /= (hist.sum() + 1e-7)

return hist

# Load dataset
def load_dataset(dataset_path):
    data = []
    labels = []
    for folder in os.listdir(dataset_path):
        folder_path = os.path.join(dataset_path, folder)
        if not os.path.isdir(folder_path):
            continue
        for file in os.listdir(folder_path):
            file_path = os.path.join(folder_path, file)
            image = cv2.imread(file_path)
            if image is None:
                continue
            image = cv2.resize(image, (256, 256))
            features = lbp_feature(image)
            data.append(features)
            labels.append(folder)
    return np.array(data), np.array(labels)

# Main training pipeline
dataset_path = "FabricDataset" # Replace with your dataset folder
X, y = load_dataset(dataset_path)
X, y = shuffle(X, y, random_state=42)

# Encode labels
le = LabelEncoder()
y_encoded = le.fit_transform(y)
```

```
# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

# Train model
model = SVC(kernel='linear', probability=True)
model.fit(X_train, y_train)

# Evaluate
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=le.classes_))

# Save model and label encoder
joblib.dump(model, "fabric_classifier_model.pkl")
joblib.dump(le, "label_encoder.pkl")
```

Code for the streamlit and front-end part of the project:

```
import streamlit as st
import cv2
import numpy as np
import joblib
from skimage.feature import local_binary_pattern
from PIL import Image

# Load the trained model
model = joblib.load("fabric_classifier_model.pkl")

# LBP parameters
radius = 3
n_points = 8 * radius
METHOD = 'uniform'

def extract_lbp_features(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    lbp = local_binary_pattern(gray, n_points, radius, METHOD)
    (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, n_points + 3), range=(0, n_points + 2))
    hist = hist.astype("float")
```

```
hist /= (hist.sum() + 1e-6)
```

```
return hist.reshape(1, -1)
```

```
st.title("Fabric Type Classifier")
```

```
st.write("Upload a fabric image to identify its type.")
```

```
uploaded_file = st.file_uploader("Choose a fabric image", type=["jpg", "jpeg", "png"])
```

```
if uploaded_file is not None:
```

```
    image = Image.open(uploaded_file)
```

```
    st.image(image, caption='Uploaded Image', use_container_width=True)
```

```
    # Convert to OpenCV format
```

```
    image_cv = cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR)
```

```
    # Extract features and predict
```

```
    features = extract_lbp_features(image_cv)
```

```
    prediction = model.predict(features)[0]
```

```
    probability = model.predict_proba(features).max() * 100
```

```
    st.success(f"Predicted Fabric Type: **{prediction}** with {probability:.2f}% confidence")
```