Final Report

Project Title: Classifying Fabric Patterns Using Deep Learning

Date: 28 June 2025

Team ID: LTVIP2025TMID46247

Team Members:

|  |  |
| --- | --- |
| Kovvuri Adilakshmi | Model Development |
| Palla Lakshmitha | Data Handling & Preprocessing |
| Korukonda Chiranjeevi Mani Vinayak | Streamlit Frontend |
| Agraharapu Anji Kumar | Testing & Deployment |

# 1. INTRODUCTION

## 1.1 Project Overview

This project involves building a deep learning model to classify different types of fabric patterns such as cotton, denim, nylon, polyester, and silk. It uses a Convolutional Neural Network (CNN) implemented with TensorFlow and is deployed using Streamlit for real-time classification.

## 1.2 Purpose

To automate the process of fabric classification to improve efficiency, reduce errors in textile industries, and support quality control operations.

# 2. IDEATION PHASE

## 2.1 Problem Statement

Manual fabric classification in textile industries is inefficient, inconsistent, and time-consuming. An AI-based classifier can improve accuracy and operational speed.

## 2.2 Empathy Map Canvas

Thinks: Needs a reliable method to identify fabric types accurately.  
Feels: Frustrated with human errors in manual classification.  
Says: 'We need to automate this process.'  
Does: Manually checks fabrics and labels them.  
Pains: Human errors, time delays.  
Gains: Accurate, quick classification using a trained CNN model.

## 2.3 Brainstorming

Ideas included building a CNN model, using Streamlit for UI, expanding dataset diversity, implementing Docker deployment, and adding confidence scores to predictions.

# 3. REQUIREMENT ANALYSIS

## 3.1 Customer Journey Map

Customer: Quality control officer.  
Need: Classify fabrics reliably.  
Touchpoint: Upload interface.  
Action: Upload image.  
Result: Gets classified result.

## 3.2 Solution Requirement

Functional Requirements:  
- Upload image  
- Predict fabric type  
- Show confidence score  
- Train model with image data  
  
Non-functional Requirements:  
- Usable UI  
- Reliable prediction  
- Available offline  
- Fast response

## 3.3 Data Flow Diagram

To be inserted manually by the user (diagram placeholder).

## 3.4 Technology Stack

Frontend: Streamlit  
Backend: TensorFlow, Keras, Python  
File Storage: Local filesystem  
Model: CNN  
Infrastructure: Local system

# 4. PROJECT DESIGN

## 4.1 Problem Solution Fit

The CNN solution solves the problem of inaccurate and slow manual classification.

## 4.2 Proposed Solution

A trained CNN deployed via Streamlit to classify uploaded fabric images in real-time.

## 4.3 Solution Architecture

Architecture diagram to be inserted manually by the user.

# 5. PROJECT PLANNING & SCHEDULING

## 5.1 Project Planning

Phases: Data collection → Model training → Evaluation → Deployment via Streamlit → Testing → Documentation.

# 6. FUNCTIONAL AND PERFORMANCE TESTING

## 6.1 Performance Testing

Tested using unseen validation images. Achieved over 90% classification accuracy with fast inference times.

# 7. RESULTS

## 7.1 Output Screenshots

Screenshots of the Streamlit app and classified results to be added manually.

# 8. ADVANTAGES & DISADVANTAGES

Advantages:  
- Fast and accurate classification  
- Simple UI  
- No database required  
  
Disadvantages:  
- Limited to trained categories  
- Accuracy depends on image quality

# 9. CONCLUSION

This project demonstrates how deep learning can automate fabric classification, improving speed and accuracy in textile manufacturing.

# 10. FUTURE SCOPE

Enhance dataset, deploy on cloud, integrate with inventory systems, add user authentication.

# 11. APPENDIX

Source Code:

import os

import keras

from keras.models import load\_model

import streamlit as st

import tensorflow as tf

import numpy as np

# Set page title

st.header(' Fabric Classification Model')

# Make sure upload directory exists

UPLOAD\_DIR = 'upload'

os.makedirs(UPLOAD\_DIR, exist\_ok=True)

# Corrected variable name: 'fabric\_names' instead of 'fabric\_name'

fabric\_names = ['cotton', 'denim', 'nylon', 'polyester', 'silk']

# Load model safely

MODEL\_PATH = 'Fabric\_C\_Model.keras'

try:

    model = load\_model(MODEL\_PATH)

except Exception as e:

    st.error(f"Failed to load model: {e}")

    st.stop()

# Function to classify uploaded image

def classify\_pictures(image\_path):

    input\_image = tf.keras.utils.load\_img(image\_path, target\_size=(180, 180))

    input\_image\_array = tf.keras.utils.img\_to\_array(input\_image)

    input\_image\_exp\_dim = tf.expand\_dims(input\_image\_array, 0)  # Shape: (1, 180, 180, 3)

    predictions = model.predict(input\_image\_exp\_dim)

    result = tf.nn.softmax(predictions[0])

    outcome = f"🌼 The image most likely belongs to \*\*{fabric\_names[np.argmax(result)]}\*\* with a confidence of \*\*{np.max(result) \* 100:.2f}%\*\*."

    return outcome

# Streamlit uploader

uploaded\_file = st.file\_uploader('📁 Upload an image', type=['jpg', 'jpeg', 'png'])

if uploaded\_file is not None:

    # Save the uploaded file

    saved\_path = os.path.join(UPLOAD\_DIR, uploaded\_file.name)

    with open(saved\_path, 'wb') as f:

        f.write(uploaded\_file.getbuffer())

    # Display uploaded image

    st.image(uploaded\_file, width=200)

    # Classify and display result

    prediction = classify\_pictures(saved\_path)

    st.markdown(prediction)

GitHub Link :- **https://github.com/Pallalakshmitha123/Fabric-Classification**

Project Demo Link: **http://localhost:8501/**