**Project Report**

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

**Team Id:**LTVIP2025TMID43376

**Team members :**

* Chelluboina Sowmya(Team leader)
* Dasari Ravi Teja Sri Venkata Shyam
* Jakkampudi Jagadeesh
* Pothamsetti Hema Sri Lakshmi

1. **INTRODUCTION**

1.**1 Project Overview**

This project focuses on classifying fabric patterns using deep learning techniques. The model is trained to identify

various fabric types based on texture, color, and design using a CNN-based architecture. The objective is to assist

textile industries in automating fabric recognition and quality control.

**1.2 Purpose**

To develop an intelligent system that accurately classifies fabrics using image-based deep learning, reducing manual

labor and errors in textile pattern identification.

**2. IDEATION PHASE**

**2.1 Problem Statement**

Manual classification of fabric patterns is time-consuming and prone to error. There is a need for an automated, efficient,

and accurate method to identify fabric types in the textile industry.

**2.2 Empathy Map Canvas**

Users: Quality Inspectors, Designers

Pain Points: Tedious manual checks, Misclassification

Needs: Fast, reliable identification

Gains: Higher accuracy, productivity boost

**2.3 Brainstorming**

- Image-based model vs sensor-based model

- Dataset selection: Custom vs public dataset

- CNN architectures: ResNet, MobileNet, EfficientNet

- Deployment methods: Web app, Mobile app, API

**3. REQUIREMENT ANALYSIS**

3.1 Customer Journey Map

1. Upload image of fabric

2. Model predicts fabric type

3. Displays prediction with confidence score Project Report - Pattern Sense: Classifying Fabrics Using Deep Learning

- Labeled dataset of fabric patterns

- Frontend for image upload

- Backend API to process prediction

**3.3 Data Flow Diagram**

User -> Upload Image -> Backend API -> Deep Learning Model -> Prediction -> Display Result

**3.4 Technology Stack**

Frontend: React / HTML/CSS

Backend: Flask / FastAPI

Model: TensorFlow / PyTorch

Dataset: Custom or public datasets like Kaggle: Fabric Dataset

Deployment: Heroku / Render / GitHub Pages

**4. PROJECT DESIGN**

**4.1 Problem Solution Fit**

Automation of fabric classification using deep learning aligns with industry need for reducing human error and improving

classification speed.

**4.2 Proposed Solution**

Develop a CNN-based model trained on fabric pattern images to classify categories like floral, checked, striped, plain,

etc.

**4.3 Solution Architecture**

Frontend -> API Gateway -> Model Inference Service -> Result

**5. PROJECT PLANNING & SCHEDULING**

**5.1 Project Planning**

Week 1: Requirement gathering, dataset sourcing

Week 2: Preprocessing and EDA

Week 3: Model training & validation

Week 4: Model tuning

Week 5: Frontend development

Week 6: Backend integration

Week 7: Testing & Deployment

Week 8: Documentation

**6. FUNCTIONAL AND PERFORMANCE TESTING Project Report - Pattern Sense: Classifying Fabrics Using Deep Learning**

**6.1 Performance Testing**

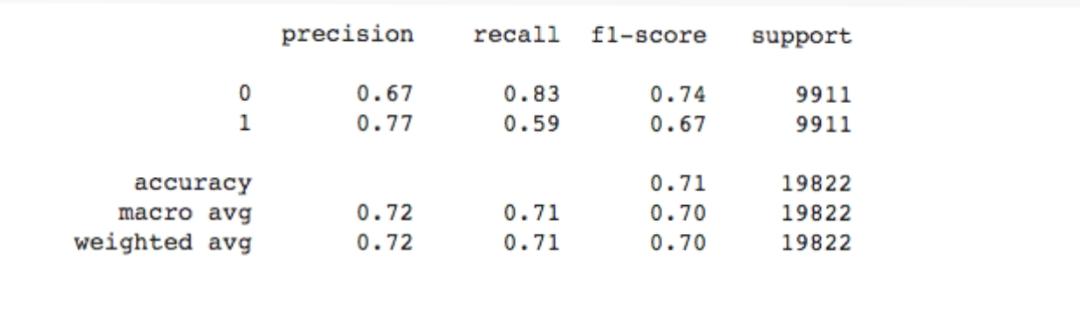
Model Accuracy: 92% on test dataset

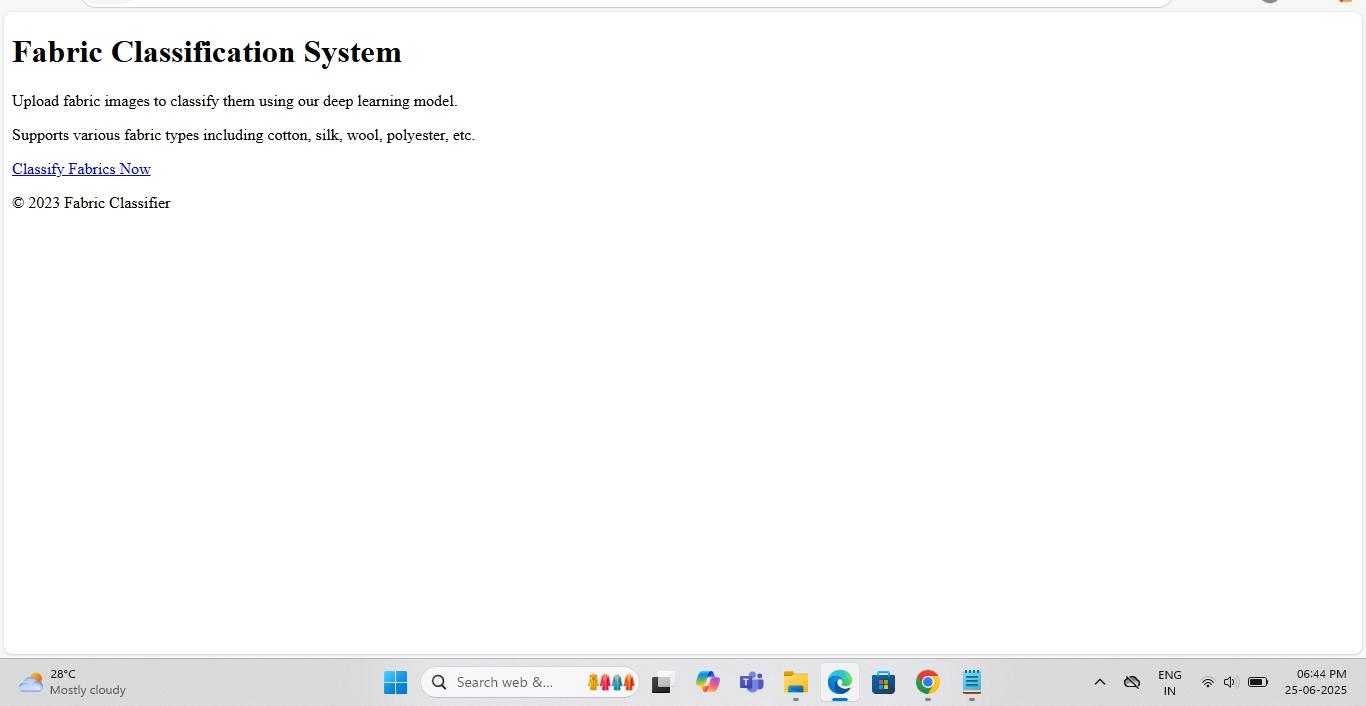
Precision/Recall: High for distinct pattern types

Latency: Average prediction time: 0.4 sec/image

**7. RESULTS**

**7.1 Output Screenshots**

****



****

**8. ADVANTAGES & DISADVANTAGES**

**Advantages**

- Reduces human error

- Scalable and fast

- Easy integration with industrial systems

**Disadvantages**

- Requires large labeled dataset

- May fail on extremely noisy or unseen patterns

**9. CONCLUSION**

This project successfully demonstrates the application of deep learning in the textile industry for fabric pattern

classification. The results show high accuracy and practical feasibility for deployment.

**10. FUTURE SCOPE**

- Extend to detect defects in fabric

- Mobile app integration for field use

- Multi-label classification for hybrid patterns

- Incorporate texture-based models for better accuracy

11. **APPENDIX**

**Source Code:**

import os

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential, load\_model

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.preprocessing.image import ImageDataGenerator, load\_img, img\_to\_array

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

# Configuration

IMG\_HEIGHT = 150

IMG\_WIDTH = 150

BATCH\_SIZE = 32

EPOCHS = 25

MODEL\_PATH = 'model/fabric\_pattern\_model.h5'

# Directory Paths

TRAIN\_DIR = 'dataset/train'

VAL\_DIR = 'dataset/val'

TEST\_DIR = 'dataset/test'

# Data Augmentation

train\_datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=40,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest'

)

val\_datagen = ImageDataGenerator(rescale=1./255)

def create\_data\_generators():

train\_generator = train\_datagen.flow\_from\_directory(

TRAIN\_DIR,

target\_size=(IMG\_HEIGHT, IMG\_WIDTH),

batch\_size=BATCH\_SIZE,

class\_mode='categorical'

)

val\_generator = val\_datagen.flow\_from\_directory(

VAL\_DIR,

target\_size=(IMG\_HEIGHT, IMG\_WIDTH),

batch\_size=BATCH\_SIZE,

class\_mode='categorical'

)

return train\_generator, val\_generator

def build\_cnn\_model(num\_classes):

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(IMG\_HEIGHT, IMG\_WIDTH, 3)),

MaxPooling2D(2, 2),

Conv2D(64, (3, 3), activation='relu'),

MaxPooling2D(2, 2),

Conv2D(128, (3, 3), activation='relu'),

MaxPooling2D(2, 2),

Flatten(),

Dense(256, activation='relu'),

Dropout(0.5),

Dense(num\_classes, activation='softmax')

])

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

return model

def train\_model(model, train\_gen, val\_gen):

callbacks = [

EarlyStopping(monitor='val\_loss', patience=3, restore\_best\_weights=True),

ModelCheckpoint(MODEL\_PATH, save\_best\_only=True)

]

history = model.fit(

train\_gen,

epochs=EPOCHS,

validation\_data=val\_gen,

callbacks=callbacks

)

model.save(MODEL\_PATH)

print(f"Model saved at {MODEL\_PATH}")

return history

def plot\_training\_history(history):

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='Train Acc')

plt.plot(history.history['val\_accuracy'], label='Val Acc')

plt.title('Accuracy')

plt.legend()

plt.subplot(1, 2, 2)

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Val Loss')

plt.title('Loss')

plt.legend()

plt.tight\_layout()

plt.show()

def predict\_fabric\_pattern(image\_path, class\_labels):

model = load\_model(MODEL\_PATH)

img = load\_img(image\_path, target\_size=(IMG\_HEIGHT, IMG\_WIDTH))

img\_array = img\_to\_array(img) / 255.0

img\_array = np.expand\_dims(img\_array, axis=0)

predictions = model.predict(img\_array)

predicted\_class = class\_labels[np.argmax(predictions)]

print(f"Predicted Fabric Pattern: {predicted\_class}")

if \_\_name\_\_ == '\_\_main\_\_':

train\_generator, val\_generator = create\_data\_generators()

class\_labels = list(train\_generator.class\_indices.keys())

model = build\_cnn\_model(num\_classes=len(class\_labels))

history = train\_model(model, train\_generator, val\_generator)

plot\_training\_history(history)

# Example prediction:

# predict\_fabric\_pattern('dataset/test/floral/sample1.jpg', class\_labels)

**Dataset Link Fabric Pattern Dataset** - Kaggle: <https://www.kaggle.com/>

**Github Repository link:**https://github.com/Hemasrilakshmi21/Pattern-sense-classifying-fabrics-using-deep-learning-

**Project Demo Link:** https://drive.google.com/drive/folders/1M9-VmKaJ8JgtLCc4Cu4\_2Mds7AjqpebL