Pattern Sense: Classifying Fabric Patterns Using Deep Learning

# Introduction

Pattern Sense is an AI-powered solution leveraging Convolutional Neural Networks (CNNs) to classify fabric patterns from images. It addresses manual classification challenges in industries like fashion, textiles, and interior design.

# Use Case Scenarios

Scenario 1: Fashion Industry

Fashion professionals often deal with numerous pattern types (e.g., stripes, polka dots, florals). Pattern Sense automates their classification, speeding up cataloging and reducing manual effort.

Scenario 2: Textile Quality Control

Textile firms can utilize the system for defect detection in patterns, ensuring consistent quality before production or distribution.

Scenario 3: Interior Design

Interior designers can efficiently match fabric patterns to their decor concepts, enhancing workflow speed and visual matching.

# Problem Statement

Manual classification is time-consuming and error-prone. Businesses need scalable, consistent, and fast AI systems to handle complex pattern identification automatically.

# Solution Overview

CNN-Based Model:

- Uses MobileNet, ResNet, or custom CNN

- Handles diverse classes: floral, geometric, striped, abstract, ethnic

- Data augmentation enhances generalization

Web Interface:

- Flask-based UI for image upload

- Shows prediction + confidence score in under 2 seconds

Architecture:

User → Web UI → Flask Backend → Trained CNN → Prediction Output → UI

# Technology Stack

Frontend: HTML, CSS

Backend: Flask, Python

Deep Learning: TensorFlow, Keras

Deployment: GitHub, Docker (optional), Streamlit

Dataset: Kaggle, Custom Labeled Dataset

# Performance

Accuracy: 87.2%

Precision: 86.5%

Recall: 88.0%

F1 Score: 87.2%

Inference Time: 1.89 sec

# Advantages & Limitations

Advantages:

- Speeds up fabric classification

- Reduces human error

- Easily scalable for new patterns

Limitations:

- Depends on image clarity

- Requires retraining for unseen patterns

- Sensitive to lighting/occlusions

# Project Plan & Timeline

Phase: Data Collection → Outcome: Clean, labeled dataset

Phase: Model Training → Outcome: CNN with augmentation + tuning

Phase: Web UI Development → Outcome: Image upload, prediction display

Phase: Integration → Outcome: Full end-to-end pipeline

Phase: Testing & Deployment → Outcome: Flask app tested and deployed

# Future Enhancements

- Mobile real-time prediction

- Fabric-type recognition (cotton, silk, etc.)

- TensorFlow Lite edge deployment

- REST API integration for third-party systems

# Resources