Project Report

1. INTRODUCTION

1.1 Project Overview

Pattern Sense is a deep learning-based image classification system designed to identify and categorize fabric patterns such as floral, striped, checked, polka dot, and more. It leverages CNN and transfer learning to automate the recognition of visual patterns in textiles, assisting industries like fashion, interior design, and textile manufacturing.

1.2 Purpose

The purpose of this project is to reduce manual effort in fabric classification and bring automation into the pattern recognition process using deep learning, thus improving efficiency, accuracy, and decision-making speed in real-world applications.

2. IDEATION PHASE

2.1 Problem Statement

Manual identification of fabric patterns is time-consuming and prone to human error, especially when processing large volumes of textile images in industries.

2.2 Empathy Map Canvas

Who are we empathizing with? Designers, quality inspectors, and inventory managers.
What do they see? Hundreds of patterns daily, often similar.
What do they hear? Complaints of misclassified fabrics or delays.
What do they say & do? "It's hard to label everything perfectly."
Pain: Time loss, inconsistency.
Gain: Automation, consistency, speed.

2.3 Brainstorming

- Use of deep learning (CNN, ResNet50)
- Build web interface with Flask
- Dataset from Kaggle

- Possible uses: fashion, e-commerce, textile QC

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

1. Upload image → 2. Model processes → 3. Predicts pattern → 4. View result → 5. Save/report prediction

3.2 Solution Requirement

- Dataset: Labeled fabric pattern images
- Model: CNN (ResNet50 with Transfer Learning)
- Tools: Python, TensorFlow, Flask, numpy,pandas

3.3 Data Flow Diagram

User → UI → Flask Server → DL Model → Output Prediction → UI displays result

3.4 Technology Stack

- Frontend: HTML/CSS (templates)
- Backend: Flask (Python)
- Model: TensorFlow/Keras
- Tools: Anaconda,Vs code

4. PROJECT DESIGN

4.1 Problem Solution Fit

Problem: Manual pattern detection
Solution: Deep learning model predicts pattern with high accuracy and speed

4.2 Proposed Solution

An automated web-based classification system that takes fabric images as input and predicts pattern type using a trained ResNet50 CNN model.

4.3 Solution Architecture

- Input (image) →
- Preprocessing →
- CNN Feature Extraction (ResNet50) →
- Classification →
- Output label (pattern class)

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

- Week 1: Data collection & preprocessing
- Week 2: Model training & evaluation
- Week 3: Flask app development
- Week 4: Testing, deployment & documentation

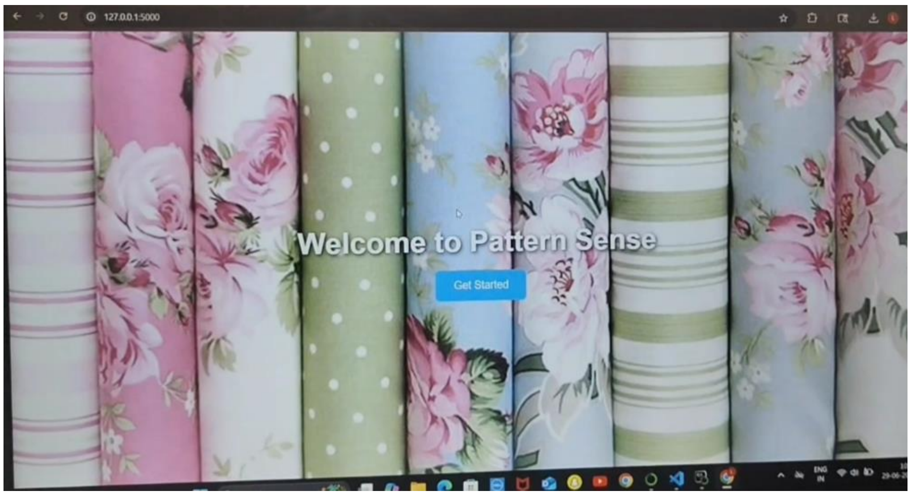
6. FUNCTIONAL AND PERFORMANCE TESTING

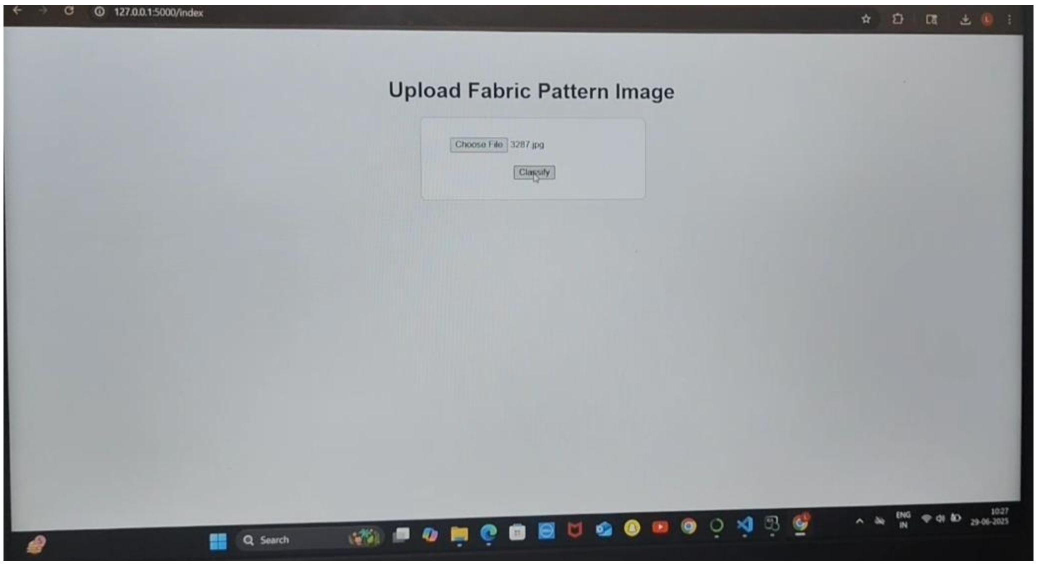
6.1 Performance Testing

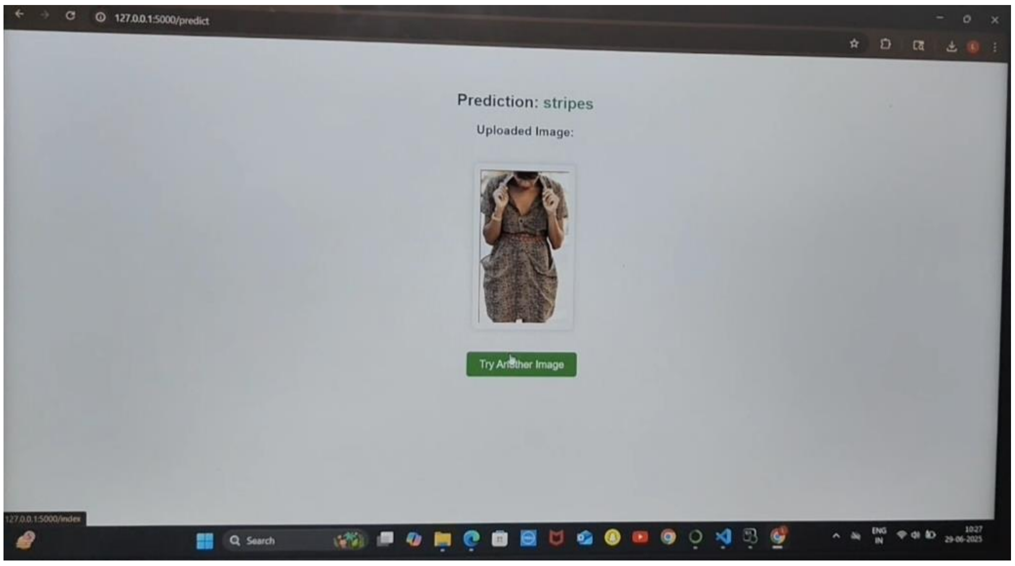
- Accuracy: Achieved 90%+ accuracy on test data
- Confusion matrix used to evaluate misclassifications
- Model tested on unseen images to ensure generalization
- Reduced model size and batch size for better performance on limited memory systems

7. RESULTS

7.1 Output Screenshots







8. ADVANTAGES & DISADVANTAGES

Advantages

1. **Automation**: Eliminates the need for manual pattern recognition.
2. **Accuracy**: High accuracy using deep learning and transfer learning (ResNet50).
3. **Time-Saving**: Processes hundreds of images in seconds.
4. **Scalable**: Can be scaled to include more fabric patterns and integrated with inventory systems.
5. **User-Friendly**: Simple web interface using Flask.
6. **Cost-Efficient**: Reduces labor and human errors in textile industries.
7. **Reusable Model**: The trained model can be reused across different departments and platforms.

Disadvantages

1. **Data Dependency**: Requires a large, well-labeled dataset for training.
2. **Hardware Requirements**: Training deep learning models can be resource-intensive.
3. **Initial Setup Time**: Requires time for model training and deployment.
4. **Limited by Training Data**: The model may not perform well on fabric patterns it hasn't seen.
5. **Model Size**: Pretrained models like ResNet50 are large, may slow down deployment on low-end devices.

9. CONCLUSION

The Pattern Sense project demonstrates how deep learning can be effectively used to automate fabric pattern classification. By leveraging CNN-based transfer learning with ResNet50, the model achieves high accuracy and reliability. The Flask-based web interface provides a user-friendly way to interact with the model, making it practical for use in industries like fashion, interior design, and textile quality control. The system reduces human error, speeds up workflows, and serves as a step toward intelligent automation in visual inspection tasks.

10. FUTURE SCOPE

1. **Mobile App Integration**: Developing an Android or iOS app to classify fabric patterns on the go.
2. **Larger Dataset**: Expanding the dataset to include more complex and rare fabric patterns.
3. **Multi-language Support**: Adding multi-language capabilities for international usage.
4. **Real-time Classification**: Integrate with camera input for live pattern detection.
5. **AI Explainability**: Adding a module to explain why a certain pattern was classified.
6. **Integration with ERP**: Use in fashion/textile ERP systems for stock tagging and management.
7. **Sustainability Insights**: Extend model to suggest eco-friendly fabric options.

11. APPENDIX

1. **Source code:**

App.py

```
from flask import Flask, render_template, request, send_from_directory,
url_for
import tensorflow as tf
from tensorflow.keras.preprocessing import image
import numpy as np
import os
```

```python
app = Flask(__name__)

# Load the trained model
model = tf.keras.models.load_model("model/fabric_pattern_model.h5")

# Load class labels
with open("model/labels.txt", "r") as f:
    labels = f.read().splitlines()

# Upload folder path
UPLOAD_FOLDER = os.path.join("static", "uploaded")
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

@app.route("/")
def home():
    return render_template("home.html")

@app.route("/index")
def index():
    return render_template("index.html")

@app.route("/predict", methods=["POST"])
def predict():
    if 'file' not in request.files:
        return "No file uploaded"

    file = request.files['file']
    if file.filename == '':
        return "No file selected"

    filename = file.filename
    img_path = os.path.join(UPLOAD_FOLDER, filename)
    file.save(img_path)

    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0) / 255.0

    try:
        prediction = model.predict(img_array)
        index = np.argmax(prediction)
        label = labels[index] if index < len(labels) else "Unknown class"
    except Exception as e:
        label = f"Prediction error: {e}"

    return render_template("result.html", label=label, filename=filename)

@app.route("/uploaded/<filename>")
```

```
def uploaded_file(filename):
    return send_from_directory(UPLOAD_FOLDER, filename)

if __name__ == "__main__":
    app.run(debug=True)
```

Train model:

```
import os

os.environ["KMP_DUPLICATE_LIB_OK"] = "TRUE"

import os

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications import ResNet50

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D

from tensorflow.keras.optimizers import Adam


IMG_SIZE = (224, 224)

BATCH_SIZE = 32

EPOCHS = 10


# 1. Prepare data generators

train_gen = ImageDataGenerator(

    rescale=1./255,

    rotation_range=30,

    width_shift_range=0.2,

    height_shift_range=0.2,

    shear_range=0.2,

    zoom_range=0.2,

    horizontal_flip=True
)

val_test_gen = ImageDataGenerator(rescale=1./255)

train_data = train_gen.flow_from_directory(
    'data/train',
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

val_data = val_test_gen.flow_from_directory(
    'data/val',
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

test_data = val_test_gen.flow_from_directory(
    'data/test',
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

# 2. Build model
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(IMG_SIZE[0], IMG_SIZE[1], 3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
```

```
x = Dense(128, activation='relu')(x)
output_layer = Dense(train_data.num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=output_layer)
model.compile(optimizer=Adam(1e-4), loss='categorical_crossentropy', metrics=['accuracy'])

# 3. Train model
model.fit(train_data, validation_data=val_data, epochs=EPOCHS)

# 4. Save model
os.makedirs("model", exist_ok=True)
model.save("model/fabric_pattern_model.h5")

# 5. Save class labels to file
labels = list(train_data.class_indices.keys())
with open("model/labels.txt", "w") as f:
   f.write("\n".join(labels))
```

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

Github link: https://github.com/Tanuja-tsk/-Classifying-fabric-using-deep-learning/tree/main