**Project Documentation**

# **Project Title**

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

# **Team Members**

Team ID : LTVIP2025TMID37339

Team Leader : Chandrakantha Lokeswari

Team member : Guda Mamatha

Team member : Hima Varsha Palle

Team member : Dileep Tunga

Team member : V Sunny

**Phase 1: Brainstorming & Ideation**

**Objective:**

Generate a viable and innovative idea to classify fabric patterns using deep learning.

**Key Points:**

- **Problem Statement:** Manual sorting of fabric patterns is inefficient, subjective, and unscalable.

- **Idea:** Build an AI-powered image classification system that can detect and label fabric patterns like floral, striped, geometric, Ethnic style etc.

- **Inspiration**: Challenges in textile/fashion industries, high demand for automation.

**- Technologies Considered:** CNN, TensorFlow, OpenCV, Flask/Streamlit, Transfer Learning (ResNet50/MobileNet).

- **Final Direction:** Train a CNN model and deploy it via a simple web interface for easy use.

# **Phase 2: Requirement Analysis**

**Objective:**

Identify functional and technical requirements of the solution.

**Functional Requirements:**

- Upload an image of fabric through a web interface.

- Predict and display the pattern class (e.g., Ethnic style, geometric).

- Display a confidence score.

**Non-Functional Requirements:**

- Real-time prediction (under 2 seconds).

- Accuracy above 85%.

- Responsive and secure deployment.

**Technology Stack:**

- Python, Flask, TensorFlow/Keras, OpenCV.

- **Dataset:** Kaggle clothing pattern dataset + custom augmentation.

**Data Flow:**

User → Web UI → Flask Backend → CNN Model → Predicted Output

# **Phase 3: Project Design**

**Objective:**

Plan the system architecture and UI components.

**Key Points:**

**- Architecture:**

User uploads image via frontend.

Flask backend handles preprocessing and inference.

CNN predicts the pattern and returns output to user.

**- Model Design:**

Transfer learning with MobileNet/ResNet50.

**Image preprocessing:** resize, normalize.

**Metrics:** Accuracy, Precision, Recall, F1-score.

- UI/UX Design:

Simple upload button.

Clear output with label and confidence score.

# **Phase 4: Project Planning (Agile)**

**Objective:**

Divide project into iterative sprints for timely execution.

**Sprints:**

- Sprint 1: Dataset cleaning and augmentation.

- Sprint 2: Model training and evaluation.

- Sprint 3: Web UI development (Flask).

- Sprint 4: Integration, testing, documentation.

**Project Tools:** GitHub, Jupyter Notebook, VS Code

# **Phase 5: Project Development**

**Objective:**

Develop the model and integrate it with the interface.

**Technology Stack:**

- Frontend: HTML/CSS (minimal Flask templating)

- Backend: Flask (Python), TensorFlow/Keras

- Model: Transfer Learning (ResNet/MobileNet), trained on augmented dataset

- **Deployment:** Local server / Streamlit / Docker (optional)

# **Phase 6: Functional & Performance Testing**

**Objective:**

Ensure system accuracy and speed meet expectations.

**Performance Results:**

- Accuracy: 87.2%

- Precision: 86.5%

- Recall: 88.0%

- F1-Score: 87.2%

- Inference Time: ~1.89 sec per image

- UI Responsiveness: Under 2 sec for full prediction cycle

**Test Types:**

- Unit Testing (model input/output)

- Integration Testing (UI + backend)

- User Testing (for usability)

# **Final Submission**

**Deliverables:**

- Trained Model (.h5/.pb)

- Flask Application Source Code

- Documentation (this file)

- Dataset Source Reference

- Demo video

# **Github Link**

https://github.com/LokEswariChandrakantha/Pattern-Sense-Classifying-Fabric-Pattern-using-Deep-Learning.git

# **Conclusion**

This project demonstrates the power of deep learning in textile pattern recognition. It achieves reliable performance, reduces manual effort, and provides a scalable solution for fabric classification in fashion-tech environments.

# **Future Scope**

- Mobile app integration

- Multi-label classification (e.g., texture + pattern)

- Real-time camera-based predictions

- REST API for third-party integration

- Edge deployment using TensorFlow Lite