**Ideation Phase**

**Brainstorm & Idea Prioritization Template**

|  |  |
| --- | --- |
| Date | 28 June 2025 |
| Team ID | LTVIP2025TMID46247 |
| Project Name | Classifying Fabric Patterns Using Deep Learning |
| Maximum Marks | 4 Marks |

**Brainstorm & Idea Prioritization Template:**

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Reference: <https://www.mural.co/templates/brainstorm-and-idea-prioritization>

**Step-1: Team Gathering, Collaboration and Select the Problem Statement**

Manual classification of fabrics in the textile industry is time-consuming, error-prone, and subjective. The aim is to automate this using deep learning for improved speed, accuracy, and consistency.

**Step-2: Brainstorm, Idea Listing and Grouping**

\*\*Ideas Generated:\*\*  
1. Train CNN using TensorFlow/Keras  
2. Build an image classification UI with Streamlit  
3. Use data augmentation techniques  
4. Show prediction confidence score  
5. Simple image upload interface  
6. Normalize and preprocess images  
7. Deploy using Docker/cloud  
8. Add more diverse dataset images  
9. Use database to store classification history  
10. Add authentication to limit usage  
  
\*\*Grouped by Themes:\*\*  
- Model & Dataset: 1, 3, 6, 8  
- User Interface: 2, 4, 5  
- Scalability & Security: 7, 9, 10

**Step-3: Idea Prioritization**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

|  |  |  |  |
| --- | --- | --- | --- |
| Idea | Impact | Feasibility | Priority |
| Train CNN with TensorFlow | High | High | High |
| Real-time Streamlit image classification | High | High | High |
| Apply data augmentation | Medium | High | High |
| Display softmax-based prediction confidence | Medium | High | Medium |
| Provide easy upload UI | High | High | High |
| Expand dataset | High | Medium | Medium |
| Cloud/Docker deployment | High | Medium | Medium |
| Integrate database to store results | Medium | Medium | Low |
| Add user authentication | Medium | Medium | Low |

**Ideation Phase**

**Define the Problem Statements**

|  |  |
| --- | --- |
| Date | 28 June 2025 |
| Team ID | LTVIP2025TMID46247 |
| Project Name | Classifying Fabric Patterns Using Deep Learning |
| Maximum Marks | 2 Marks |

**Customer Problem Statement Template:**

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love.

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you’ll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

Graphical user interface, text, application, email

Description automatically generated

Reference: <https://miro.com/templates/customer-problem-statement/>

**Example:**

Chart, treemap chart

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Statement (PS)** | **I am (Customer)** | **I’m trying to** | **But** | **Because** | **Which makes me feel** |
| PS-1 |  |  |  |  |  |
| PS-2 |  |  |  |  |  |

**Ideation Phase**

Empathize & Discover

Date  
28 June 2025

Team ID  
  
LTVIP2025TMID46247

Project Name  
Classifying Fabric Patterns Using Deep Learning

Maximum Marks  
4 Marks

**Empathy Map Canvas:**

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user’s behaviours and attitudes.  
  
It is a useful tool to help teams better understand their users.  
  
Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user’s perspective along with their goals and challenges.

**User Persona**

Primary Users:  
- Fabric manufacturers  
- Quality control technicians  
- Textile product sellers

**Says**

- “Manually checking fabrics is slow and tiring.”  
- “I need a tool that can quickly tell me what type of fabric this is.”  
- “We can't afford to mislabel items, especially for exports.”

**Thinks**

- “Is there a way to make fabric inspection more reliable?”  
- “Will AI be accurate enough to trust with classification?”  
- “If it works, it could save us a lot of time.”

**Sees**

- Manual labor doing repetitive visual checks  
- Occasional misclassification due to fatigue or inexperience  
- Peers starting to use tech tools in textile operations

**Hears**

- “AI is being used for visual tasks like defect detection.”  
- “There are models that can classify images with high accuracy.”  
- “Competitors are adopting automation for inspection.”

**Pains**

- Time wasted on repetitive manual work  
- Inconsistent accuracy from human checks  
- Difficulty distinguishing similar-looking fabrics like nylon and polyester

**Gains**

- Consistent and fast classification  
- Fewer human errors and reduced workload  
- A scalable system for quality control and e-commerce use

Reference:  
https://www.mural.co/templates/empathy-map-canvas

**Project Development Phase**

**Model Performance Test**

|  |  |
| --- | --- |
| Date | 28 June 2025 |
| Team ID | LTVIP2025TMID46247 |
| Project Name | Classifying Fabric Patterns Using Deep Learning |
| Maximum Marks | 10 Marks |

**Model Performance Testing:**

Project team shall fill the following information in model performance testing template.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Parameter** | **Values** | **Screenshot** |
|  | Metrics | Confusion Matrix - [Insert Confusion Matrix Image or Description Here], Accuracy Score - 92.5%, Classification Report - [Insert Classification Report Screenshot Here] |  |
|  | Tune the Model | Hyperparameter Tuning - Adam optimizer, dropout=0.2, epochs=15 Validation Method - 20% validation split using tf.image\_dataset\_from\_directory |  |

**Project Development Phase**

**Model Performance Test**

|  |  |
| --- | --- |
| Date | 28 June 2025 |
| Team ID | LTVIP2025TMID46247 |
| Project Name | Classifying Fabric Patterns Using Deep Learning |
| Maximum Marks |  |

**Model Performance Testing:**

Project team shall fill the following information in model performance testing template.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Parameter** | **Values** | **Screenshot** |
|  | Model Summary CNN with 3 Conv2D layers, MaxPooling, Dropout(0.2), Dense layers. 128 hidden units. | **-** |  |
|  | Accuracy | Training Accuracy - 98% Validation Accuracy - 92.5% |  |
| 3. | Fine Tuning Result Validation Accuracy - 92.5% | Validation Accuracy - |  |

**Project Design Phase**

**Problem – Solution Fit Template**

|  |  |
| --- | --- |
| Date | 28 June 2025 |
| Team ID | LTVIP2025TMID46247 |
| Project Name | Classifying Fabric Patterns Using Deep Learning |
| Maximum Marks | 2 Marks |

**Problem – Solution Fit Template:**

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer’s problem. It helps entrepreneurs, marketers and corporate innovators identify behavioral patterns and recognize what would work and why

**Purpose:**

* Solve complex problems in a way that fits the state of your customers.
* Succeed faster and increase your solution adoption by tapping into existing mediums and channels of behavior.
* Sharpen your communication and marketing strategy with the right triggers and messaging.
* Increase touch-points with your company by finding the right problem-behavior fit and building trust by solving frequent annoyances, or urgent or costly problems.
* **Understand the existing situation in order to improve it for your target group.**

**Template:**

Calendar

Description automatically generated

References:

1. <https://www.ideahackers.network/problem-solution-fit-canvas/>
2. <https://medium.com/@epicantus/problem-solution-fit-canvas-aa3dd59cb4fe>

**Problem – Solution Fit for Fabric Classification**

Problem:  
Manual classification of fabric types is time-consuming, error-prone, and requires domain expertise. This becomes a bottleneck for textile manufacturers, quality assurance teams, and retailers handling large volumes of fabric inventory. Misclassification can lead to production delays, increased costs, and poor customer satisfaction.

Solution:  
Our deep learning-based fabric classification model automates the process of recognizing five major fabric types—cotton, denim, nylon, polyester, and silk—from images. This significantly reduces dependency on manual inspection, improves consistency, and accelerates the sorting and quality control process. Fit:  
The solution integrates into the visual inspection process already used in textile and manufacturing units. By using existing camera setups or uploading images to a simple Streamlit interface, users can receive accurate predictions instantly. This bridges the gap between operational needs and AI-powered automation with minimal change to existing workflows.

**Project Design Phase**

**Proposed Solution Template**

|  |  |
| --- | --- |
| Date | 28 June 2025 |
| Team ID | LTVIP2025TMID46247 |
| Project Name | Classifying Fabric Patterns Using Deep Learning |
| Maximum Marks | 2 Marks |

**Proposed Solution Template:**

Project team shall fill the following information in the proposed solution template.

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
|  | Problem Statement (Problem to be solved) | Identifying and classifying different fabric types (cotton, denim, nylon, polyester, silk) from images is a manual and time-consuming task in industries like fashion, textiles, and e-commerce. Automating this process using computer vision can save time and reduce human error. |
|  | Idea / Solution description | The project proposes a deep learning solution using Convolutional Neural Networks (CNNs) trained on a dataset of fabric images. The model is integrated into a Streamlit-based web interface that allows users to upload images and receive real-time fabric classification results. |
|  | Novelty / Uniqueness | While image classification models are common, this project uniquely targets fabric pattern recognition using a lightweight architecture suitable for real-time use. It includes data augmentation, live demo via Streamlit, and can be deployed for industrial or educational applications. |
|  | Social Impact / Customer Satisfaction | The solution benefits industries by improving efficiency and reducing costs in quality control, inventory management, and online retail. It also supports visually impaired users in identifying fabric types. Educational institutions can use it as a learning tool for AI in textiles. |
|  | Business Model (Revenue Model) | The model can be offered as a subscription-based API to fabric suppliers and garment manufacturers. It could also be integrated into mobile apps for consumers or sold as a white-label SaaS product to e-commerce platforms for automated tagging and filtering. |
|  | Scalability of the Solution | The solution can be scaled by training on larger, more diverse datasets and deployed to cloud platforms like AWS or Streamlit Cloud. It can also be extended to classify patterns, colors, or defects in fabrics and integrated with existing textile ERP systems. |

**Project Design Phase**

**Solution Architecture**

|  |  |
| --- | --- |
| Date | 28 June 2025 |
| Team ID | LTVIP2025TMID46247 |
| Project Name | Classifying Fabric Patterns Using Deep Learning |
| Maximum Marks | 4 Marks |

**Solution Architecture:**  
This project utilizes a Convolutional Neural Network (CNN) for fabric classification, trained using TensorFlow. The model is wrapped with a Streamlit frontend to enable users to upload fabric images and receive classification results in real time.  
  
Key Components:  
1. \*\*Image Dataset\*\*: A collection of labeled fabric images (cotton, denim, nylon, polyester, silk) used to train the model.  
2. \*\*Deep Learning Model\*\*: Built with TensorFlow/Keras, includes layers for convolution, pooling, dropout, and dense classification.  
3. \*\*Frontend Interface\*\*: Implemented using Streamlit to provide a simple upload and prediction system for users.  
4. \*\*Deployment\*\*: The model is saved and served through the Streamlit application locally or via a cloud service like Streamlit Cloud.  
  
Data Flow:  
- Input: User uploads an image.  
- Preprocessing: Image resized and normalized.  
- Prediction: Image passed through the trained CNN.  
- Output: Predicted fabric type with confidence score.

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

* Find the best tech solution to solve existing business problems.
* Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
* Define features, development phases, and solution requirements.
* Provide specifications according to which the solution is defined, managed, and delivered.

**Example - Solution Architecture Diagram:**



*Figure 1: Architecture and data flow of the voice patient diary sample application*

**Reference:** [**https://aws.amazon.com/blogs/industries/voice-applications-in-clinical-research-powered-by-ai-on-aws-part-1-architecture-and-design-considerations/**](https://aws.amazon.com/blogs/industries/voice-applications-in-clinical-research-powered-by-ai-on-aws-part-1-architecture-and-design-considerations/)

**Planning Logic: Fabric Pattern Classification Using Deep Learning**

A Sprint is a fixed period during which a team completes a set of tasks.  
An Epic is a large task divided into Stories.  
A Story is a manageable task, estimated using Story Points (Fibonacci series).

**Sprint 1: (5 Days)**

Data Collection

Gather fabric images from various sources 2

Organize data into labeled folders 1

Data Preprocessing

Resize and normalize images 2

Split data into training and validation sets 3

**Sprint 2 (5 Days)**

Model Building

Build CNN model using TensorFlow 5

Train and evaluate the model 3

Deployment

Create Streamlit frontend for image upload 3

Integrate model with Streamlit app 5

Total Story Points

Sprint 1 = 8

Sprint 2 = 16

Total Story Points = 24

Number of Sprints = 2

Velocity = 24/2 = 12.0 (Story Points per Sprint)

Project Planning Document

📅 Date: 28 June 2025

👥 Team ID: LTVIP2025TMID46247

📌 Project Name: Classifying Fabric Patterns Using Deep Learning

🎯 Maximum Marks: 5 Marks

**📋 Product Backlog, Sprint Schedule, and Estimation (4 Marks)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
| Sprint-1 | Dataset Handling | USF-1 | Load all fabric images into datasets | 3 | High | Member 1 |
| Sprint-1 | Data Visualization | USF-2 | View sample fabric images with labels | 2 | Medium | Member 2 |
| Sprint-1 | Preprocessing Pipeline | USF-3 | Normalize, shuffle, and cache datasets | 2 | High | Member 3 |
| Sprint-2 | Data Augmentation | USF-4 | Apply augmentation (flip, rotate, zoom) | 2 | Medium | Member 3 |
| Sprint-2 | Model Architecture | USF-5 | Create CNN model for fabric classification | 3 | High | Member 1 |
| Sprint-2 | Model Training | USF-6 | Train the model and evaluate accuracy | 3 | High | Member 2 |
| Sprint-3 | Prediction Function | USF-7 | Predict fabric type from new images | 2 | High | Member 4 |
| Sprint-3 | Model Deployment (Streamlit) | USF-8 | Create Streamlit UI for classification | 3 | High | Member 4 |
| Sprint-3 | Save and Load Model | USF-9 | Save trained model for deployment | 1 | High | Member 1 |

**📈 Project Tracker, Velocity & Burndown Chart (4 Marks)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint | Total Story Points | Duration | Sprint Start Date | Sprint End Date (Planned) | Story Points Completed | Sprint Release Date |
| Sprint-1 | 7 | 6 Days | 01 July 2025 | 06 July 2025 | 7 | 06 July 2025 |
| Sprint-2 | 8 | 6 Days | 07 July 2025 | 12 July 2025 | 8 | 12 July 2025 |
| Sprint-3 | 8 | 6 Days | 13 July 2025 | 18 July 2025 |  |  |

**📊 Velocity**

If each sprint is 6 days long and the team completes around 8 story points on average:

\*\*Avg Velocity = 8 story points / 6 days = 1.33 points/day\*\*

**📉 Burndown Chart**

Use the following tools to visualize work left vs. time:

🔗 https://www.visual-paradigm.com/scrum/scrum-burndown-chart/

🔗 https://www.atlassian.com/agile/tutorials/burndown-charts

**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 28 June 2025 |
| Team ID | LTVIP2025TMID46247 |
| Project Name | Classifying Fabric Patterns Using Deep Learning |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

**Example: Order processing during pandemics for offline mode**

**Reference:** [**https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/**](https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/)



Guidelines:

Include all the processes (As an application logic / Technology Block)

Provide infrastructural demarcation (Local / Cloud)

Indicate external interfaces (third party API’s etc.)

Indicate Data Storage components / services

Indicate interface to machine learning models (if applicable)

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | How user interacts with application e.g.  Web UI, Mobile App, Chatbot etc. | HTML, CSS, JavaScript / Angular Js / React Js etc. |
|  | Application Logic-1 | Logic for a process in the application | Java / Python |
|  | Application Logic-2 | Logic for a process in the application | IBM Watson STT service |
|  | Application Logic-3 | Logic for a process in the application | IBM Watson Assistant |
|  | Database | Data Type, Configurations etc. | MySQL, NoSQL, etc. |
|  | Cloud Database | Database Service on Cloud | IBM DB2, IBM Cloudant etc. |
|  | File Storage | File storage requirements | IBM Block Storage or Other Storage Service or Local Filesystem |
|  | External API-1 | Purpose of External API used in the application | IBM Weather API, etc. |
|  | External API-2 | Purpose of External API used in the application | Aadhar API, etc. |
|  | Machine Learning Model | Purpose of Machine Learning Model | Object Recognition Model, etc. |
|  | Infrastructure (Server / Cloud) | Application Deployment on Local System / Cloud  Local Server Configuration:  Cloud Server Configuration : | Local, Cloud Foundry, Kubernetes, etc. |

**Table-1 : Components & Technologies:**

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | List the open-source frameworks used | Technology of Opensource framework |
|  | Security Implementations | List all the security / access controls implemented, use of firewalls etc. | e.g. SHA-256, Encryptions, IAM Controls, OWASP etc. |
|  | Scalable Architecture | Justify the scalability of architecture (3 – tier, Micro-services) | Technology used |
|  | Availability | Justify the availability of application (e.g. use of load balancers, distributed servers etc.) | Technology used |
|  | Performance | Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN’s) etc. | Technology used |

**References:**

[**https://c4model.com/**](https://c4model.com/)

[**https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/**](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/)

[**https://www.ibm.com/cloud/architecture**](https://www.ibm.com/cloud/architecture)

[**https://aws.amazon.com/architecture**](https://aws.amazon.com/architecture)

[**https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d**](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)

**Project Design Phase-II**

Data Flow Diagram & User Stories

📅 Date: 28 June 2025

👥 Team ID: LTVIP2025TMID46247

📌 Project Name: Classifying Fabric Patterns Using Deep Learning

🎯 Maximum Marks: 4 Marks

**🔄 Data Flow Diagrams**

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. It shows how data enters and leaves the system, what changes the information, and where data is stored.

This project uses DFDs to illustrate how images are processed through the system to classify fabric patterns.

✅ DFD Level 0 (Context Diagram):

• User → Uploads image  
• System → Loads & processes image → Predicts fabric type → Returns result to user

✅ DFD Level 1:

1. Image Input - User uploads image via Streamlit  
2. Preprocessing - Resize, normalize, expand dimensions  
3. Prediction Engine - Model performs prediction using CNN  
4. Result Output - Class and confidence shown on screen

**✅ User Stories**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User Type | Functional Requirement (Epic) | User Story Number | User Story / Task | Acceptance Criteria | Priority | Release |
| End User (Web) | Upload Fabric Image | USF-1 | As a user, I can upload a fabric image to be classified | Image uploads successfully | High | Sprint-1 |
| End User (Web) | View Prediction | USF-2 | As a user, I want to see predicted fabric type with confidence score | Fabric class and confidence % are displayed | High | Sprint-1 |
| Developer | Load Dataset | USF-3 | As a developer, I want to load and preprocess fabric datasets | Images are loaded and dataset is split for training/validation | High | Sprint-1 |
| Developer | Augmentation | USF-4 | As a developer, I want to apply augmentations for model robustness | Data is augmented with flips, zoom, etc. | Medium | Sprint-2 |
| Developer | Model Creation & Training | USF-5 | As a developer, I want to create and train a CNN for fabric classification | Model reaches acceptable validation accuracy | High | Sprint-2 |
| Developer | Save Model | USF-6 | As a developer, I want to save the trained model | Model saved in .keras format | High | Sprint-2 |
| Developer | Load Trained Model | USF-7 | As a developer, I want to load and use a pre-trained model for inference | Model loads and predicts new images | High | Sprint-3 |
| End User (Web) | Use Streamlit UI | USF-8 | As a user, I want a UI to interact and see results from the fabric classifier | Streamlit app displays upload UI and prediction results | High | Sprint-3 |

**Project Design Phase-II**

**Solution Requirements (Functional & Non-functional)**

|  |  |
| --- | --- |
| Date | 28 June 2025 |
| Team ID | LTVIP2025TMID46247 |
| Project Name | Classifying Fabric Patterns Using Deep Learning |
| Maximum Marks | 4 Marks |

**Functional Requirements:**

Following are the functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | Image Upload & Classification | Upload image via Streamlit UI |
|  |  | Display classification result with confidence |
| FR-2 | Model Training | Train CNN on local fabric dataset |
|  |  | Use data augmentation to improve generalization |
| FR-3 | Fabric Category Management | Support multiple fabric categories |
| FR-4 | Model Deployment | Host Streamlit interface for usage |
|  |  | Allow local or containerized deployment |

**Non-functional Requirements:**

Following are the non-functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | Usability | Simple UI for non-technical users using Streamlit |
| NFR-2 | Security | Local use only; no external authentication required |
| NFR-3 | Reliability | Consistent predictions under similar input conditions |
| NFR-4 | Performance | Fast classification after model training |
| NFR-5 | Availability | Available locally on demand |
| NFR-6 | Scalability | Model can be retrained with new data as needed |

**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 28 June 2025 |
| Team ID | LTVIP2025TMID46247 |
| Project Name | Classifying Fabric Patterns Using Deep Learning |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

**Example: Order processing during pandemics for offline mode**

**Reference:** [**https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/**](https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/)



Guidelines:

Include all the processes (As an application logic / Technology Block)

Provide infrastructural demarcation (Local / Cloud)

Indicate external interfaces (third party API’s etc.)

Indicate Data Storage components / services

Indicate interface to machine learning models (if applicable)

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
| 1 | User Interface | Web interface for uploading fabric images and displaying results | Streamlit |
| 2 | Application Logic-1 | Image preprocessing and prediction pipeline | Python (TensorFlow, NumPy) |
| 3 | Application Logic-2 | Model training with data augmentation | TensorFlow / Keras |
| 4 | Application Logic-3 | Data augmentation logic | Keras Preprocessing |
| 5 | Database | No structured DB; uses directory-based organization | N/A |
| 6 | Cloud Database | Not applicable | N/A |
| 7 | File Storage | Stores uploaded images and trained model files | Local File System |
| 8 | External API-1 | Not used | N/A |
| 9 | External API-2 | Not used | N/A |
| 10 | Machine Learning Model | CNN for fabric pattern classification | TensorFlow / Keras |
| 11 | Infrastructure (Server / Cloud) | Local deployment of Streamlit app | Local Machine |

**Table-1 :**

**Components & Technologies:**

**Table-2:**

**Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
| 1 | Open-Source Frameworks | Uses open-source frameworks for AI and UI | TensorFlow, Streamlit, NumPy |
| 2 | Security Implementations | Local-only usage; no web exposure | N/A |
| 3 | Scalable Architecture | Model can be retrained with larger datasets | TensorFlow |
| 4 | Availability | App is available on local system at all times | Localhost |
| 5 | Performance | Lightweight model and fast classification on local machine | TensorFlow, Streamlit |

**References:**

[**https://c4model.com/**](https://c4model.com/)

[**https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/**](https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/)

[**https://www.ibm.com/cloud/architecture**](https://www.ibm.com/cloud/architecture)

[**https://aws.amazon.com/architecture**](https://aws.amazon.com/architecture)

[**https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d**](https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d)

**Final Report**

Project Title: Classifying Fabric Patterns Using Deep Learning

Date: 31 January 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: [**Streamlit**](http://localhost:8501/)

**Fabric Pattern Classification Using Deep Learning - Project Report**

**1. Introduction**

**Project Title**: Fabric Pattern Classification Using Deep Learning

Team Members & Roles:

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

**2. Project Overview**

Purpose:  
This application classifies fabric types (cotton, denim, nylon, polyester, silk) from images using deep learning. It automates fabric identification for industries like textiles and e-commerce.

Features:

• Upload image and receive classification result  
• TensorFlow CNN model  
• Streamlit web interface  
• Real-time prediction

**3. Architecture**

Frontend: Streamlit app for image upload and results.  
Backend: TensorFlow/Keras CNN model served via Streamlit.  
Database: Not required for this version.

**4. Setup Instructions**

Prerequisites:  
• Python 3.x  
• pip  
• TensorFlow, Streamlit, NumPy  
Installation Steps:  
• Clone repo  
• Install requirements  
• Run Streamlit app

**5. Folder Structure**

/fabric-classifier  
├── app.py # Streamlit app  
├── train\_model.py # Model training script  
├── Fabric\_C\_Model.keras # Trained model  
├── upload/ # Image uploads  
├── Pictures/ # Dataset

**6. Running the Application**

Use the command: streamlit run app.py

**7. API Documentation**

Internal prediction function:  
• classify\_pictures(image\_path)  
Returns predicted fabric class and confidence.

**8. Authentication**

Not applicable for local demo version.

**9. User Interface**

Built using Streamlit:  
• File uploader  
• Image display  
• Prediction output with confidence score

**10. Testing**

Manual testing via UI and validation accuracy during training.

**11. Screenshots or Demo**

Add screenshots or record demo after deployment.

**12. Known Issues**

• Model accuracy can improve with more data.  
• Uploading large images may slow down predictions.

**13. Future Enhancements**

• Add user authentication  
• Expand to more fabric types  
• Deploy on cloud  
• Add Grad-CAM visualization