West Visayas State University
(Formerly Iloilo Normal School)
COLLEGE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY
Luna St., La Paz, Iloilo City 5000
Iloilo, Philippines
* Trunkline: (063) (033) 320-0870 to 78 loc. 1403 * Telefax No.: (033) 320-0879
* Website: www.wvsu.edu.ph * Email Address: cict@wvsu.edu.ph

Deep Neural Network–based Fabric Defect Detection Using CNN Architecture

A Final Project
Presented to the Faculty of the
College of Information and Communications Technology
West Visayas State University
La Paz, Iloilo City

In Partial Fulfillment
of the Requirements for the Subject
CCS 248: Artificial Neural Networks

Submitted to
Mr. John Cristopher Mateo

Submitted by
Alorro, Kurt Allen R.
Escrupulo, MaAsherah Francine Faith S.
Maravilla, Christine Joy D.

DECEMBER 2025

I. Problem

Fabric defects, such as stains, tears, holes, and surface irregularities, remain a significant challenge in textile manufacturing. These defects are often subtle, irregular in shape, or concealed within complex textures and patterns, making consistent detection difficult. Variations in fabric color, weave, material, and lighting conditions further complicate inspection. Traditionally, defect detection relies on **manual inspection**, which is slow, labor-intensive, and prone to human error—subtle defects can be missed, and inspection quality can vary depending on the inspector's experience and fatigue. To overcome these limitations, some facilities have implemented **automated inspection systems**, but these systems also have shortcomings. Many fail to accurately detect all types of defects, require extensive labeled datasets, or struggle to generalize across different fabrics and production conditions. Consequently, defective fabrics can still pass through production, resulting in wasted materials, increased costs, and reduced product quality. These challenges underscore the need for **efficient, reliable defect detection**, capable of accurately identifying and classifying multiple types of fabric defects.

II. Proposed Solution

To overcome the challenges associated with manual inspection, this project employs a **Convolutional Neural Network (CNN)**, a type of Artificial Neural Network (ANN) particularly effective for image-based classification tasks. The proposed system leverages deep learning to automatically identify and classify fabric defects into three classes: **Damaged**, **Normal**, and **Stains**.

Key features of the solution include:

1. **Automated and Hierarchical Feature Extraction:** The CNN architecture extracts hierarchical features from fabric images, starting from simple edges and textures to complex patterns, without requiring manual feature engineering. This enables the model to learn the distinguishing characteristics of each defect type.
2. **Data Augmentation for Better Generalization:** To enhance model generalization and simulate variations in real-world fabric conditions, the training dataset undergoes augmentation. Techniques applied include the following:
    - Rotation, width/height shifting
    - Shear transformation and zoom
    - Horizontal flipping
    - Brightness adjustments
    - Channel shifting to simulate lighting and color variations

These augmentations reduce overfitting and strengthen the model's ability to handle diverse fabric images in different conditions unseen.

3. **Class Imbalance Handling:** Fabric datasets often have uneven distributions of defect types. To address this, **class weights** are computed using scikit-learn's `compute_class_weight` method. This ensures that underrepresented classes contribute proportionally during training, improving overall classification performance.
4. **Regularization to Prevent Overfitting:** Given the limited dataset size (which only ranged from 350-390), the model integrates multiple regularization methods to improve generalization:
   - **L2 Regularization** is applied to deeper convolutional and dense layers to penalize overly complex weights
   - **Dropout layers**, particularly after feature pooling, to reduce co-adaptation of neurons
   - **Batch Normalization** to stabilize and accelerate training

   Together, these strategies enable the model to learn robust patterns while minimizing the risk of memorizing the training data.

5. **Training Workflow and Multi-Optimizer Evaluation:** The model is trained using multiple optimizers (Adam, SGD with Momentum, and RMSProp) with early

stopping to prevent unnecessary training once performance plateaus. Evaluation metrics include **test accuracy**, **confusion matrices**, and **classification reports** to quantify precision, recall, and F1-score for each defect category.

6. **Real-Time Prediction Capability:** The system allows uploading of new fabric images for prediction. Preprocessing ensures images are resized and normalized consistently, and the trained CNN outputs predicted labels along with confidence scores.

III. Neural Network Structure

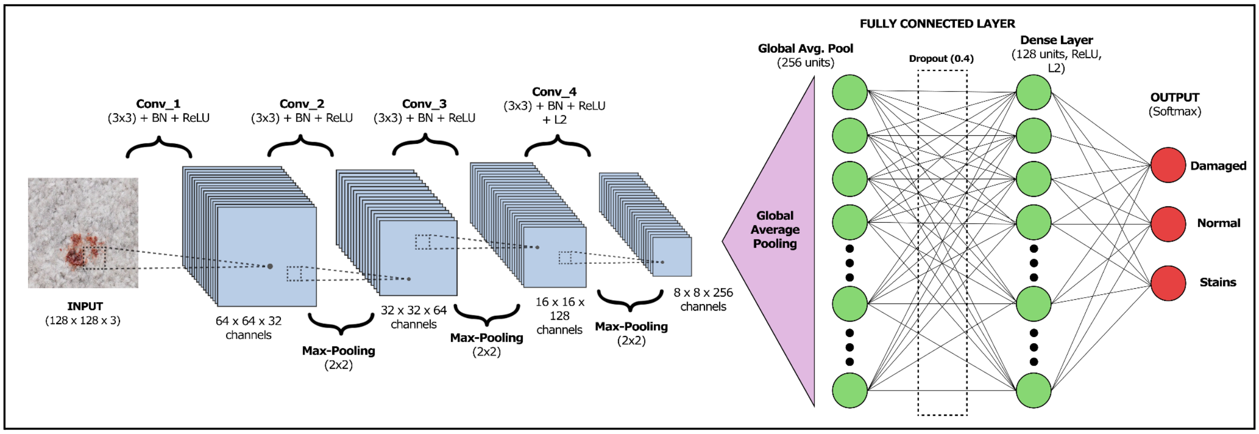


Fig. 1: Model Diagram of the Neural Network (CNN) Architecture

The proposed fabric defect detection model is built using a Convolutional Neural Network (CNN) architecture specifically designed for multiclass classification of fabric images. The architecture emphasizes both feature extraction and regularization to ensure robust performance despite dataset limitations. The overall structure is composed of four convolutional blocks followed by a global pooling layer and fully connected layers for final classification.

The following subsections describe each component of the model in detail:

1. Input Layer

The model accepts RGB fabric images resized to 128 × 128 × 3. This standardized input shape ensures compatibility with the data generators and maintains computational efficiency while preserving essential texture details in fabric samples.

2. Convolutional Feature Extraction Blocks

The CNN is composed of four convolutional blocks, each responsible for progressively learning more complex and abstract visual features:

- **Block 1:** This block focuses on capturing simple patterns such as edges and color gradients.
  - Conv2D with 32 filters, 3×3 kernel, ReLU activation
  - Batch Normalization
  - MaxPooling2D (2×2)
- **Block 2:** In this stage, it learns intermediate features such as small stains and minor distortions.
  - Conv2D with 64 filters, 3×3 kernel, ReLU activation
  - Batch Normalization
  - MaxPooling2D (2×2)
- **Block 3:** This deeper block detects more complex textures, shapes, and structural irregularities in fabric.
  - Conv2D with 128 filters, 3×3 kernel, ReLU activation
  - Batch Normalization
  - MaxPooling2D (2×2)
- **Block 4 (*with L2 Regularization*):** Lastly, this block learns high-level defect characteristics while preventing overfitting due to the limited dataset.
  - Conv2D with 256 filters, 3×3 kernel, ReLU activation

- Batch Normalization
- MaxPooling2D (2×2)
- L2 regularization (0.001) to limit weight magnitude

3. Global Average Pooling Layer

Instead of flattening the feature maps, the model uses GlobalAveragePooling2D, which reduces each feature map to a single value by averaging. This technique:

- Makes the model more location-invariant (helpful since defects may appear anywhere in the fabric)
- Reduces total parameters
- Improves generalization

4. Fully Connected Layers

- Dropout Layer: with a Dropout rate of 0.4

  This is used to prevent co-adaptation of neurons and control overfitting.
- Dense Layer
  - 128 units
  - ReLU activation
  - L2 Regularization (0.001)

  This layer integrates extracted features for class separation.

5. Output Layer

- Dense layer with 3 units (one for each class: Damaged, Normal, Stains)
- Softmax activation for multiclass probability output

The softmax function outputs a normalized probability distribution, allowing the system to identify the most likely class along with confidence scores.

6. Architectural Rationale

This architecture was chosen due to the following considerations:

- Multi-level feature extraction suits the complex nature of fabric textures.
- Batch normalization stabilizes learning and allows faster convergence.
- L2 regularization and dropout mitigate overfitting given the limited dataset.
- Global average pooling improves robustness to defect location variability.

IV. Experimental Setup

The study evaluates the performance of a **fabric defect detection** using a Convolutional Neural Network (CNN) trained under different hyperparameter configurations. The experimental setup is designed to ensure reproducibility and fair comparison across multiple training strategies.

1. Dataset

- **Fabric images** were collected from Pathirana, P. (n.d.). *Fabric Stain Dataset* [Data set], Kaggle, and Ramesh, S. (n.d.). *Fabric Defect Dataset* [Data set], Kaggle.
- Images that were preprocessed or unsuitable were removed, and additional stock images from online sources were added.
- Then images are then stored in **Google Drive**, organized into **training** and **test** directories.
- Categories: **Damaged, Normal, Stains** (3 classes).
- Images resized to **128 × 128 × 3**.
- **Training-validation split:** 80%-20%.

2. Data Preprocessing

- **Training:** Images rescaled and augmented to increase diversity and reduce overfitting: rotation, shift, shear, zoom, horizontal flip, brightness adjustment, and channel shift.
- **Validation and Test:** Only rescaled.

3. Class Imbalance Handling

- Class weights computed from the training dataset to balance learning.
- Class weights applied during model training to avoid bias toward majority classes.

4. Training Parameters

- **Batch size:** 8
- **Loss function:** Sparse categorical crossentropy
- **Early stopping:** Monitors validation accuracy, patience of 5 epochs, restores best weights

5. Hyperparameter Experiments

Three different optimizer configurations were tested to evaluate model performance:

Experiment	Optimizer	Learning Rate	Momentum	Epochs
1	Adam	0.0001	–	20
2	SGD + Momentum	0.001	0.9	20
3	RMSProp	0.001	–	25

- Each experiment used the **same training and validation datasets** for a fair comparison.
- Class weights and early stopping were applied in all cases.

6. Evaluation Metrics

- **Quantitative:** Test accuracy, precision, recall, F1-score.
- **Visual:** Confusion matrices and a bar chart comparing test accuracies across optimizers.
- **Qualitative:** Real-world testing on newly uploaded fabric images to evaluate practical performance, including predicted labels and confidence scores.

V. Tools and Environment Used

This section outlines the **platforms, libraries, and utilities** utilized to develop, train, test, and evaluate the CNN model for fabric defect detection.

1. Development Platform

- **Google Colab** – Used as the primary environment for coding, training, testing, and evaluating the neural network models.
- **Python** – Main programming language for implementing machine learning and deep learning workflows.

2. Core Machine Learning & Neural Network Libraries

- **TensorFlow / Keras**
  - Building ANN/CNN architectures
  - Model training, tuning, and evaluation
  - Layers, optimizers, and activation functions
  - EarlyStopping callback for overfitting prevention

3. Data Preprocessing & Image Handling

- **ImageDataGenerator** – For data augmentation and image preprocessing.
- **Keras image module** – Loading and preprocessing images for prediction.

- **OS module** – File and directory management for dataset handling and model storage.

4. Data Manipulation and Numerical Computation

- **NumPy** – Performing numerical operations on arrays.
- **Pandas** – Handling tabular data and preprocessing structured datasets.

5. Visualization Tools

- **Matplotlib** – Plotting training/validation accuracy and loss curves.
- **Seaborn** – Creating heatmaps, such as confusion matrices, and advanced visualizations.

6. Evaluation and Validation Tools

- **Scikit-learn (sklearn)**
  - Classification report for precision, recall, F1-score
  - Confusion matrix for performance evaluation
  - Class weight computation to handle imbalanced datasets

VI. Summary and Results

This section presents the **model's performance outcomes** based on quantitative evaluation metrics, comparative accuracy across different optimization methods, and final testing on real-world fabric images.

1. Quantitative Outcome

The quantitative results include both the overall accuracies and the class-wise performance of the models.

Test Accuracies:

Optimizer	Test Accuracy	Notes
Adam	64.22%	Balanced performance across classes; moderate precision and recall for all defect types.
SGD + Momentum	56.86%	High precision for Damaged fabrics but very low recall; Normal fabrics show high recall but low precision; overall uneven class predictions.

RMSProp	69.61%	Strong recall for Damaged and Stains fabrics; Normal fabrics recall is low, leading to some misclassification; overall best balanced performance.

Class-wise Performance:

- **Adam Optimizer**

Class	Precision	Recall	F1-score	Support	Interpretation
Damaged	0.71	0.62	0.66	65	Most predicted Damaged fabrics were correct; some misclassified as Normal or Stains.
Normal	0.60	0.47	0.53	77	Better at detecting actual Normal fabrics (high recall), some misclassification lowered precision.

Stains	0.62	0.89	0.73	62	Moderate performance; some Stains misclassified as Normal or Damaged.
Macro Avg	0.65	0.66	0.64	204	Average performance across all classes.
Weighted Avg	0.64	0.64	0.63	204	Class-size-weighted average, reflects overall model performance.

- **SGD + Momentum**

Class	Precision	Recall	F1-score	Support	Interpretation
Damaged	0.72	0.52	0.61	65	High precision but very low recall; many actual

					Damaged fabrics misclassified.
Normal	0.55	0.31	0.40	77	High recall indicates most Normal fabrics correctly detected, but low precision shows false positives.
Stains	0.51	0.94	0.66	62	Moderate performance; some misclassification as Normal or Damaged.
Macro Avg	0.59	0.59	0.56	204	Indicates uneven class performance.
Weighted Avg	0.59	0.57	0.54	204	Reflects lower overall accuracy due to class imbalance in predictions.

- **RMSProp**

Class	Precision	Recall	F1-score	Support	Interpretation
Damaged	0.76	0.88	0.81	65	Strong recall; most Damaged fabrics correctly detected, moderate precision.
Normal	0.90	0.36	0.52	77	High precision but low recall; many Normal fabrics misclassified as other classes.
Stains	0.58	0.92	0.71	62	Excellent recall; most Stains correctly detected, moderate precision.

Macro Avg	0.75	0.72	0.68	204	Balanced across classes.
Weighted Avg	0.76	0.70	0.67	204	Reflects slightly higher overall performance than Adam.

Observations Across Optimizers:

Damaged Class

- **RMSProp:** Best recall (0.88), meaning most Damaged fabrics were correctly detected; moderate precision (0.76).
- **SGD + Momentum:** High precision (0.72) but lower recall (0.52), indicating some Damaged fabrics were misclassified.
- **Adam:** Balanced performance with precision 0.71 and recall 0.62.

Normal Class

- **RMSProp:** Best precision (0.90) but low recall (0.36), showing few false positives but many Normal fabrics misclassified.
- **SGD + Momentum:** Best recall (0.31), but low precision (0.55).

- **Adam:** Moderate performance (precision 0.60, recall 0.47).

Stains Class

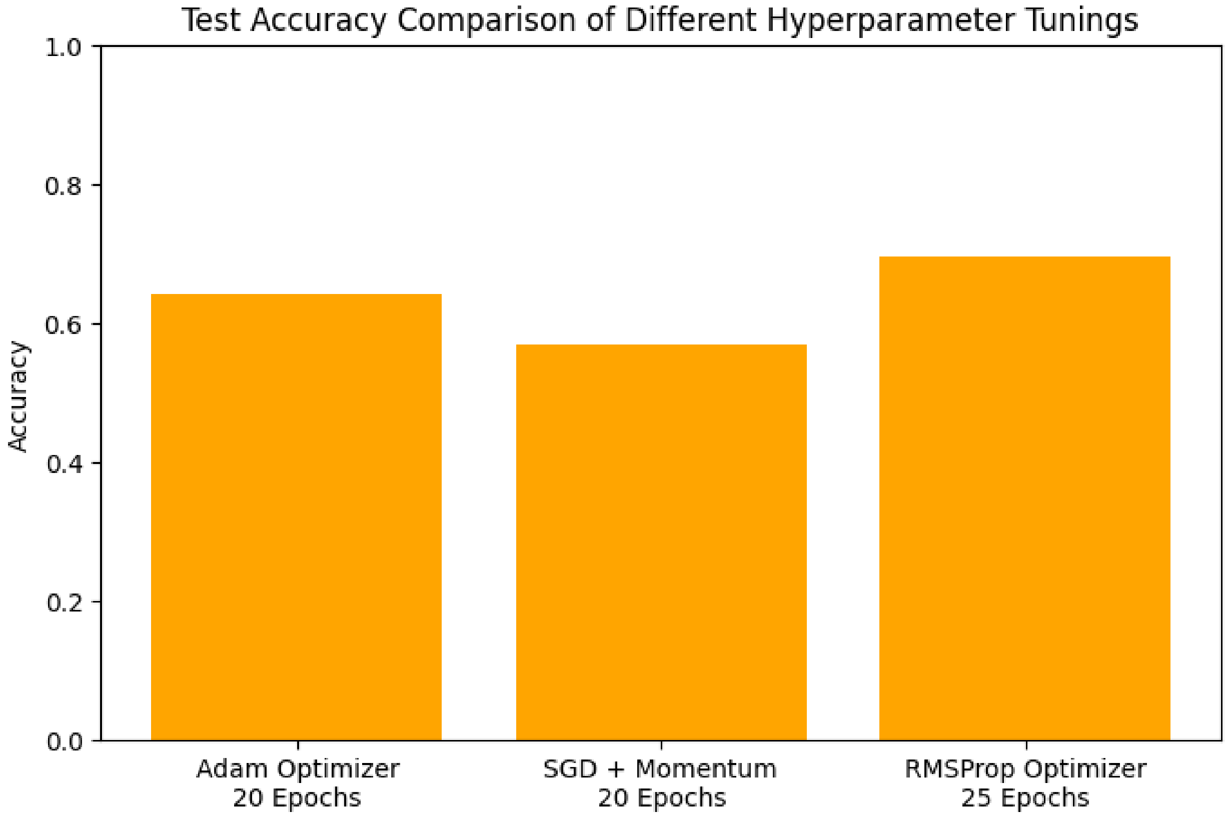
- **RMSProp:** Best recall (0.92) and moderate precision (0.58), meaning most Stains were correctly identified.
- **SGD + Momentum:** Moderate recall (0.94) but low precision (0.51).
- **Adam:** Balanced performance (precision 0.62, recall 0.89).

Overall Performance

- **Macro Average:** RMSProp (precision 0.75, recall 0.72, F1-score 0.68) slightly better balanced across classes than Adam (0.65 / 0.66 / 0.64) and SGD + Momentum (0.59 / 0.59 / 0.56).
- **Weighted Average:** RMSProp (0.76 / 0.70 / 0.67) slightly outperforms Adam (0.64 / 0.64 / 0.63), while SGD + Momentum lags behind (0.59 / 0.57 / 0.54).
- **Overall Accuracy:** RMSProp (65.69%) slightly outperforms Adam (64.71%), and SGD + Momentum (56.37%) shows the lowest performance due to uneven class predictions..

2. Test Accuracy Comparison

The overall test accuracy comparison of the CNN model with different hyperparameter tuning experiments are presented in figure below:



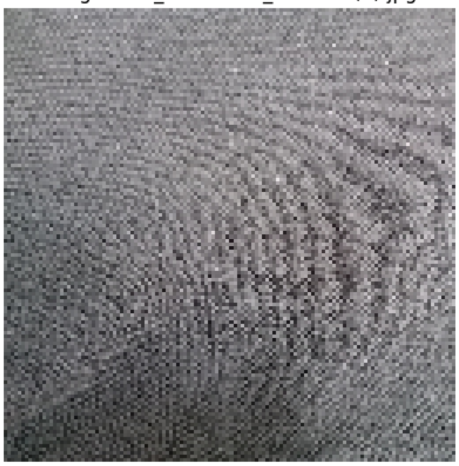
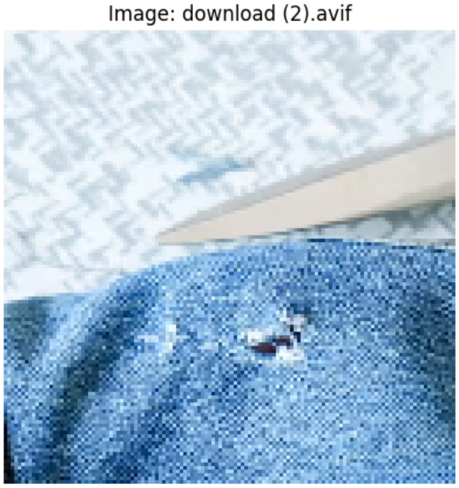
3. Real-World Image Testing

To evaluate practical performance, a set of new fabric images were uploaded and processed through all three trained models (Adam, SGD + Momentum, RMSProp). For each image, the predicted label and confidence score were recorded:

Images	Adam Prediction (Confidence)	SGD + Momentum Prediction (Confidence)	RMSProp Prediction (Confidence)
Image: IMG_20251210_140405 (3).jpg 	Normal (0.49)	Normal (0.55)	Normal (0.74)

Image: IMG_20251210_135913 (3).jpg 	Damaged (0.84)	Damaged (0.99)	Damaged (0.85)
Image: 288635af6a62b2c9754478b663abf6e4 (3).jpg 	Stains (0.50)	Normal (0.56)	Damaged (0.51)

Image: 787b9c140fb0e53f81d73aad9de0006a (4).jpg 	Damaged (0.41)	Stains (0.47)	Damaged (0.56)
Image: IMG_20251209_125123 (1).jpg 	Damaged (0.52)	Damaged (0.42)	Damaged (0.81)

Image: IMG_20251209_125149 (3).jpg 	Normal (0.61)	Damaged (0.37)	Damaged (0.53)
Image: download (2).avif 	Damaged (0.83)	Damaged (0.76)	Damaged (0.70)

Observations:

- **Adam:** Produced generally balanced predictions across classes with moderate confidence. Some Stains were misclassified as Normal, but overall performance was consistent.
- **SGD + Momentum:** Predictions often biased toward Normal or Damaged classes, reflecting patterns seen in test set confusion. Confidence scores were high but occasionally misleading due to class bias.
- **RMSProp:** Excelled at identifying high-recall classes (Damaged and Stains) but sometimes misclassified Normal fabrics. Confidence scores indicated moderate certainty and stability in predictions.

VII. Conclusion

The system demonstrates practical applicability by accurately predicting defect classes for unseen fabric images. Among the optimizers, **RMSProp** achieves the highest overall accuracy and balanced class-wise performance, while **Adam** also delivers reliable results. **SGD + Momentum**, however, exhibits noticeable inconsistencies and may require further tuning or a more balanced dataset to improve performance in real-world scenarios. These findings are consistent with the quantitative results and confirm that the model generalizes beyond controlled test conditions. Overall reliability can still be enhanced with a larger, more diverse dataset and additional optimization strategies.

VIII. References

Pathirana, P. (n.d.). *Fabric Stain Dataset* [Data set]. Kaggle. https://www.kaggle.com/datasets/priemshpathirana/fabric-stain-dataset?resource=download

Ramesh, S. (n.d.). *Fabric Defect Dataset* [Data set]. Kaggle. https://www.kaggle.com/datasets/rmshashi/fabric-defect-dataset/data