




Multiclass Classification of Denim Fabric Defects Using Convolutional Neural Networks

Thamiris Gire Zine Neves^{*1}, Roberto Gutierrez Beraldo^{*1}, and Cinara Guellner Ghedini[†]

^{*}National Service for Industrial Training (SENAI-SP) - São Bernardo do Campo, Brazil.

[†]Institute of Computing - Unicamp - Campinas, Brazil

¹Corresponding Author: thamiris.zine@sp.senai.br

Abstract—In denim fabric manufacturing, quality assessment traditionally relies on visual inspection by weavers, as guided by ABNT NBR 13484. This manual, post-production process can delay defect detection, allowing issues to persist and recur, which disrupts the production chain. To address this, we developed a deep neural network for multiclass classification of common fabric defects, including double yarns, coarse ends, harness misdraws, slack ends, broken wefts, and soft wefts. Our model integrates convolutional layers for feature extraction and linear layers for classification. It was trained on 2,985 images captured using a low-cost camera mounted above a textile manufacturing loom in a denim manufacturing laboratory, replicating real-world inspection conditions. Our model achieved a classification accuracy of 89%, outperforming other state-of-the-art machine learning algorithms. These results demonstrate the potential for real-time defect detection during production, reducing material waste and improving overall fabric quality.

I. INTRODUCTION

The textile industry significantly impacts the economies of many countries worldwide. According to the Brazilian Textile and Apparel Industry Association (Abit), the textile and clothing sector in Brazil generated revenues of R\$ 203.9 billion in 2023 [1]. Despite advancements in technologies and modern textile production methods, challenges related to fabric quality persist, including lack of uniformity, stains, and other defects that reduce market value [2]. In Brazil, the Associação Brasileira de Normas Técnicas (ABNT) established the standard NBR 13484:2025, that outlines the procedures for defect scoring inspection in fabrics [3].

These defects often result from poor-quality yarns, inadequate maintenance of automated looms, or operational errors, such as improper handling of equipment and processes by human operators. The negative impact of fabric defects in the textile industry is significant, resulting in increased waste and decreased profitability. Therefore, monitoring each production stage and implementing an effective action plan is essential to effectively minimize these defects.

A specific case is denim manufacturing, which produces the raw material used in jeans. Denim is produced by interlacing horizontal threads, called warps, and vertical threads, called wefts. During production, different types of defects may alter the fabric's texture and pattern, reducing the overall quality of the final product. Illustrative cases for jeans defects are provided in [4].

Rather than relying on human inspection, fault defection can be automated through the implementation of a real-time system that integrates sensors, digital signal processing, computer vision, and machine learning (ML) or deep learning (DL) algorithms. A fabric inspection system can be on-loom if it's mounted onto the weaving machine, or off-loom, after the manufacturing and using dedicated equipment [5]. On-loom real-time systems can analyze data during the manufacturing process, identifying fabric defects, and providing alerts, enabling weavers to promptly and accurately address their causes. This approach aims to minimize the number of defective pieces, reduce waste, and optimize manufacturing processes, thereby improving overall efficiency. However, fabric defect classification remain inherently challenging due to the wide variety of fabric defect types and the limited availability of labeled datasets for training ML and DL models.

To address these challenges, this work proposes a novel approach tailored for real-time fabric defect detection in denim production. The main contributions of this work are:

- A custom dataset was acquired *in loco*, capturing images from the weaver's point of view (on-loom) during fabric production. This perspective enables earlier defect detection compared to the traditional post-manufacturing inspection.
- A convolutional neural network (CNN) designed specifically for the multiclass classification of six types of denim fabric defects, as well as defect-free fabric. This model was proposed and evaluated as a key contribution of this work, and its performance was benchmarked against state-of-the-art machine learning algorithms.

It is important to note that both training and inference were performed offline, apart from the data acquisition. The focus of this work is not the deployment of a real-time monitoring system, but rather an investigation into whether data collected under this specific setup, combined with the proposed model, is sufficient for accurate multiclass defect classification.

The remainder of this paper is organized as follows: Section II reviews relevant literature and situates this study within it. Section III details the methodological approach. Section IV presents the results and Section V discusses its implications and limitations. Finally, Section VI concludes the paper and indicates possible future work.

II. RELATED WORKS

Several approaches have been proposed for fabric inspection across different contexts, many of which do not rely on deep learning-based models [7]–[10]. However, in recent years, significant advances in computer vision have been driven by deep learning techniques, particularly the emergence of CNNs [6]. CNNs leverage convolutional layers—specialized filters that process input images to capture spatial hierarchies of visual features—leading to substantial improvements in the performance of vision-based inspection systems.

A key property of the convolution operation is translation equivariance: when a visual feature (such as an edge, texture, or shape) is translated within the input image, the output of the convolution reflects the same translation. This allows CNNs to recognize features regardless of their position in the image.

CNNs are typically trained end-to-end to automatically extract features from structured data, eliminating the need for manual feature engineering. Compared to fully connected multi-layer perceptrons (MLPs), CNNs require fewer parameters due to weight sharing, where the same filter (or kernel) is applied across the input image. Additionally, pooling layers

downsample the feature maps, further reducing computational complexity. Given these characteristics, CNNs are particularly well-suited for image classification tasks.

Table I summarizes and compares recent approaches to fabric defect detection based on CNNs, except for [21], which classifies denim defects in a similar setting using a fully connected neural network (FCNN). It reports dataset availability (public or private), fabric types, defect categories, and camera positioning relative to the fabric (on-loom or off-loom), as well as whether close-up imaging was used. For the learning setup, it specifies the deep learning task—binary classification (BC), multiclass classification (MCC), object detection (OD), or segmentation (Seg.)—together with the model architecture, learning paradigm (supervised or unsupervised), use of transfer learning (TL), and test accuracy (ACC). Accuracy values are rounded to the nearest whole number for classification tasks. In cases involving non-classification or multiple tasks, the ACC field is marked as N/A (Not Applicable).

Although our overall accuracy is lower than that reported in related works, this result reflects specific challenges inherent to the specific data acquisition process, which will be detailed in the following sections. Importantly, our study tackles a distinct

TABLE I: Comparison between fabric defect detection approaches

Ref. (year)	Dataset	Fabric (defect) classes	Task	Architecture	TL	Learning	ACC
[11] (2021)	Closed	Rotary screen-printed fabric.(2 - Defective colour spots and misprints)	MCC	Custom CNN	No	Supervised	61%
[12] (2021)	TILDA ^b	Different types (5 - holes, oil stains, yarn defects, flying objects and wrinkles)	MCC	DenseNet, InceptionV3, and Xception (Ensemble)	Yes	Supervised	98%
[13] (2021)	Tianchi ^a + custom	Different types (3, weft direction, warp direction, and no direction.)	BC, MCC	Deep CNN, LeNet-5	No	Supervised	93%
[14] (2021)	Tianchi ^a	Different types (Specific defects were not highlighted).	BC+OD	SE-YOLOv5	Yes	Supervised	96%
[15] (2022)	AITEX ^c	20 types of fabrics (12 types of defects)	OD	YOLOv5	Yes	Supervised	N/A
[16] (2022)	Open ^d	Denim. (2 - warp and horizontal deformation)	BC	Custom	No	Supervised	97%
[17] (2023)	Multiple	Different types (4 - slub, hole, stain, and knot).	BC, MCC, OD	VGG16, VGG19, ResNet101, DarkNet53, YOLOv3	Yes	Supervised	N/A
[18] (2024)	MVTEC AD ^e	Grid, carpet, texture 1 and texture 2	Seg.	Convolutional autoencoder	No	Unsupervised	N/A
[19] (2024)	Roboflow + DAGM 2007 ^f	Different types (4 + 10 defect types)	MCC	ResNet-50	Yes	Supervised	95%–100%
[20] (2024)	Multiple + live test session	Different types (14 types of defects).	OD	Faster R-CNN (backbone: Resnet50)	No	Supervised	N/A
[21] (2024)	Closed	Denim (3 - cut weft, soft weft, and double thread)	BC	FCNN	No	Supervised	96%
Ours	Closed	Denim (6 - Double yarns, coarse ends, harness misdraws, slack ends, broken wefts, and soft wefts)	MCC	Custom CNN	No	Supervised	89%

^a<https://tianchi.aliyun.com/competition/entrance/231666/information>

^b<https://lmb.informatik.uni-freiburg.de/resources/datasets/tilda.en.html>

^c <https://www.aitex.es/afid/>

^d<https://github.com/MahdiHatami/denim-fabric-dataset>

^e<https://www.mvtec.com/company/research/datasets/mvtec-ad>

^f<https://www.kaggle.com/datasets/mhskjelvareid/dagm-2007-competition-dataset-optical-inspection>

gap: prior work on CNNs and datasets has not explored multiclass classification of denim defects using images captured directly over the loom during manufacturing.

III. MATERIALS AND METHODS

This section outlines our deep learning pipeline, including dataset acquisition and data preprocessing, model architecture and experimental setup, and the evaluation metrics and baseline models.

A. Dataset Acquisition

The dataset consists of 2D images capturing the denim fabric manufacturing process from raw cotton. These images were captured in a controlled training room environment, which minimizes common issues such as poor lighting and dust. Additionally, this controlled setting allows for the deliberate generation of fabric defects on demand.

Fig. 1 presents the experimental setup, with the camera positioned above the loom. Further details about the camera and the settings are provided in Appendix A. The images were intended to simulate the weaver’s perspective during the manufacturing process. For example, the camera was positioned as close as possible to the denim output in the loom. However, placing the camera too close to the fabric was not feasible, as it could interfere with maintenance operation on the loom. As a result, the level of detail in the captured defects – measured in pixels per centimeter – was somewhat reduced.



Fig. 1: Position of the camera over the loom.

The fabric used consists of blue warp yarns and three different materials of white-filling yarns (wefts): 100% cotton, cotton-elastane, and polyester-elastane. Denim images were collected both without any defects and with the six most common types of defects. The defects were intentionally introduced during production by the weaver, and the corresponding ground truth labels were annotated accordingly. Fig. 2 illustrates the types of defects present in the dataset.

The number of images for each class, already divided into training and test sets, is presented in Table II. Tests were conducted with data augmentation techniques, but no performance improvement was observed. Consequently, only the acquired data were used for training.

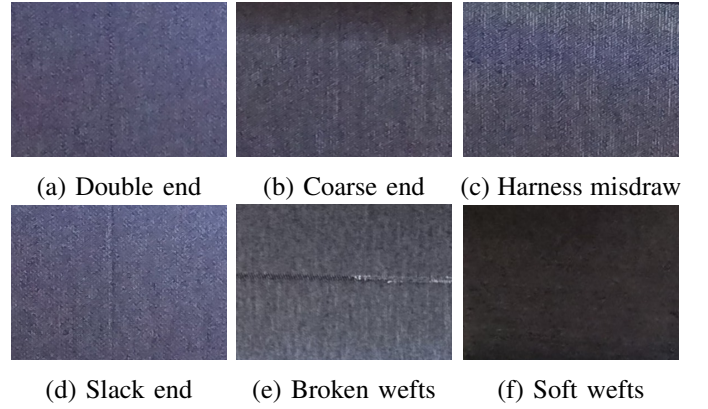


Fig. 2: Examples of each type of defect.

TABLE II: Number of images for each class

	Train	Test
Double end	184	47
Coarse end	338	85
Harness Misdraw	302	76
Slack end	492	123
No defects	340	85
Broken wefts	496	125
Soft wefts	233	59
Total	2385 ($\approx 80\%$)	600 ($\approx 20\%$)

B. Data preprocessing

The region of interest (ROI) in the image is the rectangular region of the fabric. Fig. 3 shows examples of the resulting images with different lens modes, after cropping the ROI.



Fig. 3: Cropped image samples from the dataset. The two above are from the class “broken wefts” and the two below are “No defects”.

The images were acquired on different days, with defects introduced by different weavers. Additional variability was introduced from changes in room illumination, slight camera rotations, and minor translations—none of which were corrected during preprocessing. Furthermore, depending on the camera mode and mounting height, both the cropping region and crop window size varied across captures.

In addition, preprocessing was necessary to prepare the images for input into the proposed and baseline models (See Subsection III-E). Images were resized to 256×512 pixels with PyTorch’s anti-aliasing Gaussian filter for the proposed CNN and ML models, except for ResNet50, which required

256×256 pixels. Subsequently, the images were converted to tensors with pixel values normalized to the $[0,1]$ range. The mean (μ) and standard deviation (σ) of pixel values were computed across the training set for each color channel. These statistics were then used to standardize all image pixels (x_i) from both the training and test sets, according to Eq. 1.

$$x_i = \frac{x_i - \mu}{\sigma}. \quad (1)$$

C. Model Architecture

The task was formulated as a multiclass classification problem with seven 2D-image classes, where the “without defects” class was treated equally to the six defect classes. We evaluated CNN architectures with varying numbers of convolutional layers, but increasing depth did not yield performance improvements. Therefore, we report the best-performing architecture with respect to layer depth.

Fig. 4 shows the proposed model, a CNN with four convolutional layers (kernel size of 3×3 and padding = 1) for feature extraction, progressively increasing the number of filters in each layer (32, 64, 128, and 256, respectively). Each convolutional layer is followed by batch normalization, a Rectified Linear Unit (ReLU) activation function, and 2D Max Pooling (kernel size of 2×2 and stride = 2 for dimensionality reduction).

Image features were extracted automatically by the CNN, in contrast to our previous study that relied on manual feature engineering followed by a multilayer perceptron for binary classification [21]. The extracted features were flattened into a 1D tensor to support the classification task performed by a linear layer with 256 units, succeeded by another linear layer whose size corresponds to the number of classes (seven). A Dropout of 40% is applied between the linear layers to prevent overfitting.

D. Hyperparameter and parameter optimization

For hyperparameter tuning, we performed a grid search over batch size, learning rate, dropout, scheduler, step size, γ , and weight decay. In total, 62 hyperparameter combinations were evaluated, and the best-performing configuration was selected. The resulting network comprises 33,945,863 trainable parameters, with weights initialized from a uniform distribution.

Training was performed using the cross-entropy loss function with the Adam optimizer. Due to the limited dataset size, no separate validation set was created; instead, 5-Fold Cross-Validation with 200 epochs per fold was employed. Model evaluation was conducted on the corresponding test subset every 20 epochs, and within each fold the best-performing model was retained for final evaluation. Hardware and software configurations are provided in Appendix B.

After hyperparameter tuning via grid search, the Adam optimizer was configured with a learning rate of 0.0001, weight decay of 1×10^{-4} , and a batch size of 16. A StepLR scheduler was applied with a decay factor of $\gamma = 0.3$ every 50 steps.

E. Evaluation metrics

The performance of the multiclass classification models was assessed using four standard evaluation metrics [22]: overall accuracy, precision, recall, and F1-score. These metrics were derived from the confusion matrices corresponding to each class. Given the seven-class classification problem, a one-vs-all strategy was used to compute class-specific metrics.

For each class k , the confusion matrix is treated as a binary classification matrix, where class K is considered the positive class and all others are considered negative. Based on this, true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) were defined accordingly.

The overall accuracy is computed as:

$$\text{Accuracy} = \frac{TP}{TP + TN + FP + FN}. \quad (2)$$

Precision and recall per class are defined, respectively, as:

$$\text{Recall}_k = \frac{TP_k}{TP_k + FN_k}, \quad (3)$$

$$\text{Precision}_k = \frac{TP_k}{TP_k + FP_k}. \quad (4)$$

To account for the balance between precision and recall, we also computed the F1-score for each class:

$$\text{F1-score}_k = \frac{2 \cdot \text{Precision}_k \cdot \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k}. \quad (5)$$

Each metric ranges from 0 (lowest) to 1 (highest) performance, providing complementary information not captured by overall accuracy alone.

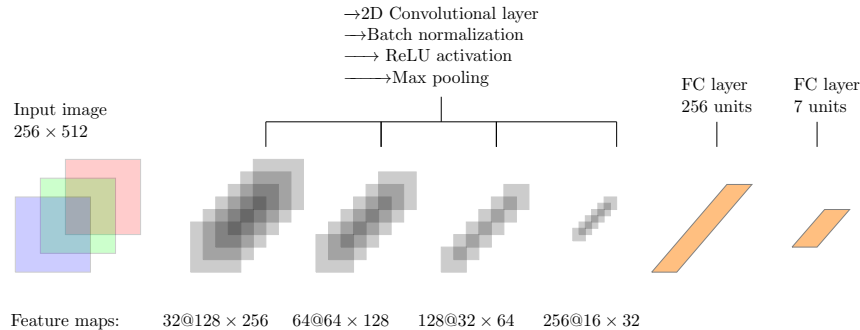


Fig. 4: Convolutional Neural Network Architecture.

F. Baseline models

For comparison with the proposed model, the multiclass classification task was also performed using conventional machine learning algorithms and another deep learning model, all summarized in Table III. The same preprocessing pipeline was applied across all models, including pixel value standardization using Equation (1). The Adam optimizer’s learning rate, batch size, and other training hyperparameters were kept identical to those used in the proposed model.

TABLE III: ML and DL algorithms used for the benchmark.

Algorithm	Abbrev.	Properties
Decision Tree [23]	DT	Random state = 0 (Deterministic)
Random Forest [24]	RF	100 estimators, Gini impurity criteria, maximum depth = 5
K-Nearest Neighbors [25]	KNN	Number of neighbors = 5
Support Vector Machine [26]	SVM	Linear kernel, C = 1.0
50-Layer Residual Network [27]	ResNet50	256 × 256 Input size. Transfer learning using Imagenet1K [28]. Fine-tuning using our custom dataset.

IV. RESULTS

Fig. 5 presents the loss function and the accuracy during the model training. The training curves indicate rapid convergence, with the training loss stabilizing before 100 epochs and the training accuracy approaching 100%. Although a gap of approximately 10% is observed between training and validation accuracy during the 5-fold cross-validation, the model demonstrates consistent generalization across folds. It is important to clarify that the “Validation Loss” refers to the performance on the test subsets of the cross-validation procedure, not on an independent validation set.

To complement the cross-validation analysis, Table IV and Fig. 6 report the performance metrics per class and the corresponding confusion matrix. The results demonstrate that the proposed model achieved high precision, recall, and F1-score across most classes. In particular, the *Coarse end*, *Slack end*, and *Harness misdraw* classes achieved F1-scores of 0.94, 0.96, and 0.91, respectively, indicating the model’s capacity to distinguish well-defined defect patterns with high reliability.

TABLE IV: Proposed model: Results per class.

Class	Precision	Recall	F1-score
Double end	0.97	0.79	0.87
Coarse end	0.90	0.98	0.94
Harness misdraw	0.92	0.91	0.91
Slack End	0.96	0.97	0.96
No defects	0.85	0.74	0.79
Broken wefts	0.82	0.90	0.85
Soft wefts	0.88	0.90	0.89

Some variability in performance is observed in classes such as *Double end* and *No defects*, with recall values of 0.79 and 0.74, respectively. This behavior can be attributed to several factors. First, certain defect types exhibit high visual similarity, as suggested by the confusion between *Double end* and *Coarse*

end. Second, some images may contain more than one defect, despite being annotated with a single label, which introduces unavoidable ambiguity into the learning process. Finally, the classification of *No defects* is particularly challenging in practical scenarios, as it involves distinguishing subtle or borderline cases under realistic manufacturing conditions.

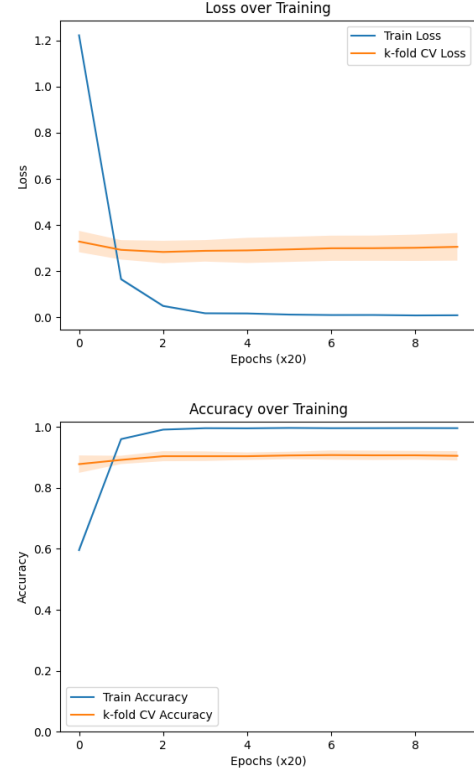


Fig. 5: Loss function and accuracy during the training of the proposed model.

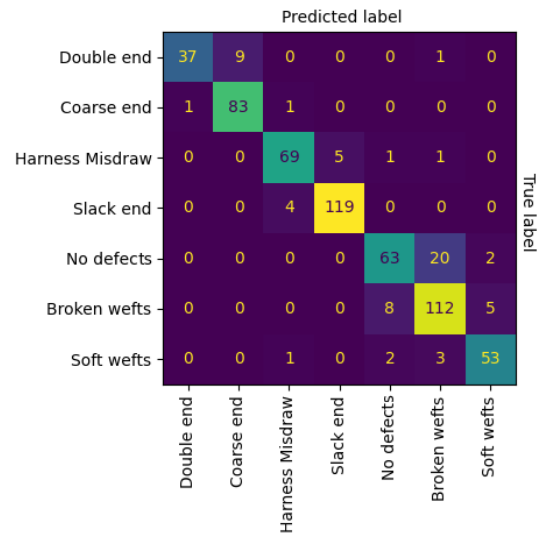


Fig. 6: Confusion matrix of the proposed model in the test set.

Despite these challenges, the model maintains strong performance across all classes, with all F1-scores exceeding 0.79. These findings confirm the robustness of the proposal and its suitability for real-world deployment in inspection tasks, even under conditions of label noise and class similarity.

Table V presents the quantitative results of the proposed and baseline models, rounded to two decimal places. The best results are highlighted in bold.

TABLE V: Benchmark with ML and DL algorithms. All values presented are macro averages.

Classifier	Accuracy	Precision	Recall	F1-score
DT	0.84	0.83	0.84	0.83
RF	0.78	0.81	0.75	0.77
KNN	0.80	0.81	0.78	0.79
SVM (Linear)	0.87	0.88	0.87	0.87
Resnet50	0.84	0.84	0.85	0.84
Our model	0.89	0.90	0.88	0.89

V. DISCUSSION

This section addresses three main topics: (i) factors influencing the performance comparison between CNN and SVM, where the former achieved the best results; (ii) the use of dimensionality reduction techniques for 2D visualization of CNN-extracted features prior to classification; and (iii) practical limitations and decisions made during data acquisition that affect the results and suggest directions for future study.

A. Performance using different models

This study addresses a scenario where deep learning models do not always have a clear advantage over traditional machine learning approaches. As shown in Table V, classical algorithms such as linear SVM achieved competitive performance on the dataset, even surpassing a pretrained ResNet-50. This finding highlights the importance of carefully evaluating model choice in contexts with limited data.

To mitigate the challenges of a small dataset, several strategies were adopted to enhance generalization. Model complexity was controlled by limiting network depth, as deeper architectures did not improve performance. In addition, K-Fold Cross-Validation, data augmentation, and regularization techniques (weight decay, batch normalization, and dropout) were applied. With these measures, the CNN achieved robust results, and importantly, outperformed all competing models across every evaluation metrics. This demonstrates that, when carefully designed and regularized, deep learning can still deliver superior performance even under data-constrained conditions.

Beyond accuracy, additional factors influence model selection. Training time, for example, varied across models (CNN: ≈ 10 hours, SVM: ≈ 6 hours), though such differences depend on hardware and implementation efficiency rather than the learning paradigm itself. Hardware requirements also differ: CNNs demand more GPU memory, while SVMs rely more heavily on RAM. Thus, in real-world scenarios such as factory-floor deployment, the choice of model must balance predictive performance with infrastructure constraints,

latency requirements, and whether the solution is intended for embedded edge computing.

B. Visualizing model embeddings

After training the proposed CNN, low-dimensional (2D) embeddings were generated to visualize the learned feature representations. For each image in the test set, features were extracted from the convolutional layers and reduced using the T-distributed Stochastic Neighbor Embedding (t-SNE) technique [29]. Ideally, well-separated clusters for each class indicate that the model effectively distinguishes among classes. The visualization results are shown in Fig. 7.

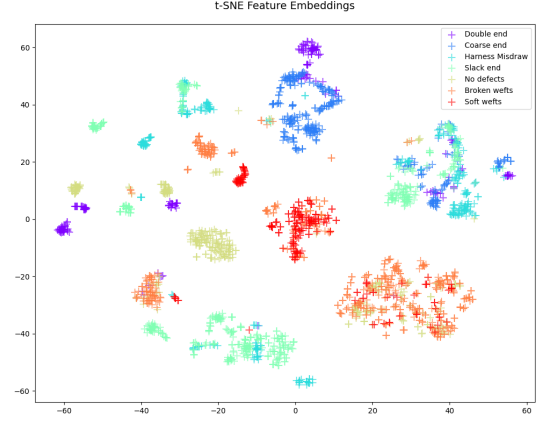


Fig. 7: CNN embeddings: The t-SNE dimensionality reduction algorithm represents in the 2D-plane how the CNN is classifying the test set samples.

We observe distributed and partially separated class clusters, although some regions show overlap. This suggests that the CNN's feature extraction may not perfectly distinguish all seven classes. However, the reduction to two dimensions is highly restrictive and may not fully reflect the model's ability to separate classes in higher-dimensional spaces.

It is important to note that the defects are often subtle relative to the fabric, typically involving small areas, widths, or lengths. Furthermore, certain defects are visually similar, particularly when oriented in the same direction, such as broken and soft wefts. These characteristics, combined with factors such as the camera-to-fabric distance and the need to resize images, may affect model performance.

In real-world deployment, it would be essential to periodically acquire new images and update the CNN-based model. With continual updates, it is expected that the feature extraction process to better distinguish classes, reducing overlap in the 2D t-SNE visualization.

C. Considerations and Challenges

Several challenges were encountered during image acquisition, particularly related to the camera and its support structure, and computational resources for image processing and model development were also limited. From these constraints, several important consequences emerged, discussed as follows.

Image acquisition was not fully standardized, resulting in a heterogeneous dataset of images and defects. Parameters such as camera mode and height were varied, and the training laboratory conditions differ from those in the factory. To address this, it will be necessary to collect a new dataset in the factory environment, ideally following a standardized acquisition protocol. For instance, a fixed camera support would prevent translation and rotation, preserving the relative position to the loom and avoiding manual adjustments. This would allow an assessment of whether standardization can lead to higher accuracies.

A multiclass classification approach was adopted to address the complexity of denim fabric defects. However, the presence of additional defect types and the possibility of multiple defects occurring simultaneously indicate that a multi-label classification approach could be explored in future work.

Certain weave types, such as black weave, were not included in the dataset. As a result, performance in these cases may differ from the reported results. Although classification was not explicitly based on weft types, it remains important to incorporate all weave variations in future datasets to enable the development of a more generalizable model.

Image resizing was necessary given the computational resources available (See Appendix B). As shown in Fig. 3, the original images exhibit a relatively small height-to-width ratio, which was altered during the resizing process, resulting in information loss, particularly along the horizontal axis. Consequently, model performance may be lower than what could be achieved with the original, uncropped images.

During training, CNN filters are automatically learned to extract meaningful features for the task. Nevertheless, the image preprocessing techniques—such as artifact removal and defect enhancement—can further support classification, as demonstrated in [21], [30]–[32]. For example, moiré patterns present in some images could potentially be mitigated through appropriate preprocessing methods.

VI. CONCLUSION AND FUTURE WORKS

We proposed a CNN-based model to classify denim fabric images into seven distinct classes: one defect-free and six corresponding to common fabric defects. The main contribution of this study lies in evaluating a deep learning performance on real-world data acquired *in loco*, with images captured directly above an automated loom during active denim production—a realistic, industrially relevant on-loom setting. This setup simulates the perspective of a human weaver and introduces the natural variability found in operational environments.

Despite the challenges posed by a relatively small and heterogeneous dataset, the proposed model achieved an accuracy of 89%, outperforming conventional machine learning baselines and demonstrating strong generalization across classes. These results demonstrate robustness and potential for practical deployment in textile manufacturing environments. Given the acquisition conditions and processing pipeline, the proposed system already approximates real-time inspection.

We therefore anticipate similar performance in online deployment, paving the way for intelligent, automated defect detection systems that reduce waste, improve quality control, and support the digital transformation of the fabric industry.

Future work will focus on addressing the challenges outlined in Subsection V-C to enable the development of a fully real-time fabric inspection system. Another promising direction involves extending the deep learning task beyond classification to incorporate spatial information about defect locations. This could be achieved by dividing fabric images into smaller patches and applying multiclass classification at the patch level, by reformulating the problem as object detection to localize defects with bounding boxes, or by exploring unsupervised anomaly detection methods capable of segmenting defective regions from the defect-free background without requiring extensive labeled data. These directions aim to enhance the practical applicability of the system in industrial settings by improving both interpretability and localization accuracy.

APPENDIX

A. Camera and Capture Setup details

We used a GoPro Hero 9 camera in the time-lapse mode and the advanced Protune settings shown in Table VI.

TABLE VI: GoPro Hero 9 acquisition settings

Parameter	Value	Parameter	Value
Camera Height ^a	72 to 80 cm	(Time) Interval	5 seconds
Sharpness	High	CompEX	0
Color	GoPro	White Balance	Automatic
ISO min	100	ISO max	3200
Lens mode	Linear or wide	Resolution	5184 × 3888
Format	JPG		

^aDistance from the camera lens to the denim fabric.

B. Hardware and software details

The deep learning pipeline was conducted using the setup described in Table VII. We used Notebook 1 for Dataset preparation, image processing, CNN training, and inference; Notebook 2 for baseline models training; and a workstation for hyperparameter tuning (grid search).

TABLE VII: Equipments used for the experiments

	Notebook 1	Notebook 2	Workstation
CPU	Intel i7-9750H	Intel i7-9750H	Two Intel Xeon Silver
GPU	RTX2060	GTX1660Ti	Tesla T4
RAM	16.0 Gb	24.0 Gb	128.0 Gb

We preprocessed the images using OpenCV [33], implemented and trained the CNN using PyTorch [34] and the baseline models using Scikit-Learn [35]. Table VIII describes the software used in this work.

TABLE VIII: Software used for the experiments

Software	Version	Software	Version
Operating System	Windows 11/WSL2	Python	3.12
Cuda	12.4	Numpy	1.26
OpenCV	4.10	Pytorch	2.5.1
Scikit-image	0.24	Scikit-learn	1.6
Scipy	1.14		

ACKNOWLEDGMENTS

The authors would like to thank the National Service for Industrial Training (SENAI-SP) for supporting this research through the Center for Skills Development in Artificial Intelligence (CDC-IA) program.

REFERENCES

- [1] Associação Brasileira da Indústria Têxtil e de Confecção (ABIT), “Indústria têxtil e de confecção aumentou a produção e gerou 30 mil empregos em 2024,” 2025. Available at <https://www.abit.org.br/noticias/industria-textil-e-de-confeccao-aumentou-a-producao-e-gerou-30-mil-empregos-em-2024>.
- [2] M. Hossain, M. Shahid, M. Limon, I. Hossain, and N. Mahmud, “Techniques, applications, and challenges in textiles for a sustainable future,” *Journal Of Open Innovation: Technology, Market, And Complexity*, 10, 100230, 2024. DOI:10.1016/j.joitmc.2024.100230
- [3] Associação Brasileira de Normas Técnicas, “NBR 13484:2025: Tecidos planos - Método de classificação baseado em inspeção por pontuação de defeitos,” 3^a ed. 2025. Available at <https://www.abntcatalogo.com.br/>.
- [4] ABNT and SEBRAE, “Guia de implementação: Normas para confecção de jeans,” Convênio ABNT/SEBRAE, Rio de Janeiro, 2012. ISBN: 978-85-07-03612-8. Available at <https://bis.sebrae.com.br/bis/>.
- [5] X. J. Zhou and J. Wang, “A real-time computer vision-based platform for fabric inspection part 2: Platform design and real-time implementation,” *The Journal Of The Textile Institute*, 107:264–272, 2016. DOI: 10.1080/00405000.2015.1025559
- [6] J. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and Milan Parmar, “A review of convolutional neural networks in computer vision,” *Artif Intell Rev*, 99, 2024. DOI: 10.1007/s10462-024-10721-6
- [7] M. Eldessouki, “Computer vision and its application in detecting fabric defects,” In W. K. Wong, editor, *Applications of Computer Vision in Fashion and Textiles*, The Textile Institute Book Series, chapter 4, pages 61–101. Woodhead Publishing, 2018. DOI: 10.1016/B978-0-08-101217-8.00004-X
- [8] C. Li, J. Li, Y. Li, L. He, X. Fu, and J. Chen, “Fabric defect detection in textile manufacturing: A survey of the state of the art,” *Security and Communication Networks*, 2021(1):9948808, 2021. DOI: 10.1155/2021/9948808
- [9] Y. Kahraman and A. Durmuşoğlu, “Deep learning-based fabric defect detection: A review,” *Textile Research Journal*, 93(5-6):1485–503, 2023. DOI: 10.1177/00405175221130773
- [10] P. Guo, Y. Liu, Y. Wu, R. H. Gong and Y. Li, “Intelligent quality control of surface defects in fabrics: A comprehensive research progress,” *IEEE Access*, 12:63777–808, 2024. DOI: 10.1109/ACCESS.2024.3396053
- [11] S. Chakraborty, M. Moore, and L. Parrillo-Chapman, “Automatic defect detection for fabric printing using a deep convolutional neural network,” *International Journal of Fashion Design, Technology and Education*, 15(2):142–57, 2022. DOI: 10.1080/17543266.2021.1925355
- [12] X. Zhao, M. Zhang, and J. Zhang, “Ensemble learning-based CNN for textile fabric defects classification,” *International Journal of Clothing Science and Technology*, 33(4):664–78, 2021. DOI: 10.1108/ijcst-12-2019-0188
- [13] X. Jun, J. Wang, J. Zhou, S. Meng, R. Pan, and W. Gao, “Fabric defect detection based on a deep convolutional neural network using a two-stage strategy,” *Textile Research Journal*, 91(1-2):130–42, 2021. DOI: 10.1177/0040517520935984
- [14] L. Zheng, X. Wang, Q. Wang, S. Wang, and X. Liu, “A fabric defect detection method based on improved YOLOv5,” In *7th International Conference on Computer and Communications (ICCC)*, IEEE, 620–4, 2021. DOI:10.1109/ICCC54389.2021.9674548
- [15] R. Seidel, H. Seibel Júnior, and K. Komati, “Textile defect detection using YOLOv5 on AITEX dataset,” in *Anais do XIX Encontro Nacional de Inteligência Artificial e Computacional*, Campinas/SP, 2022, pp. 763–74. DOI: 10.5753/eniac.2022.227396
- [16] M. Talu, K. Hanbay, and M. H. Varjovi, “CNN-Based Fabric Defect Detection System on Loom Fabric Inspection,” *Tekstil Ve Konfeksiyon*, 32:208–19, 2022. DOI: 10.32710/tekstilvekonfeksiyon.1032529
- [17] R. Thakur, D. Panghal, P. Jana, Rajan, and A. Prasad, “Automated fabric inspection through convolutional neural network: An approach,” *Neural Computing and Applications*, 35(5):3805–23, 2023. DOI: 10.1007/s00521-022-07891-1
- [18] H. M. Ferreira, D. R. Carneiro, M. Â. Guimarães, and F. V. Oliveira, “Supervised and unsupervised techniques in textile quality inspections,” *Procedia Computer Science*, vol. 232, pp. 426–435, 2024, 5th International Conference on Industry 4.0 and Smart Manufacturing (ISM 2023). DOI: 10.1016/j.procs.2024.01.042
- [19] H. Mewada, I. M. Pires, P. Engineer, and A. V. Patel, “Fabric surface defect classification and systematic analysis using a cuckoo search optimized deep residual network,” *Engineering Science and Technology, an International Journal*, 53:101681, 2024. DOI: 10.1016/j.jestech.2024.101681
- [20] H. Lin, D. Cai, Z. Xu, J. Wu, L. Sun, and H. Jia, “Fabric4show: Real-time vision system for fabric defect detection and post-processing,” *Visual Intelligence*, 2(1), 2024. DOI: 10.1007/s44267-024-00047-w
- [21] T. Neves, D. Minatel, A. Saito, and M. Baffa, “Two-stage preprocessing approach for background normalization and defect segmentation on denim fabric image analysis,” in *Anais Estendidos da XXXVII Conferência on Graphics, Patterns and Images*, Manaus/AM, 2024, pp. 179–185. DOI: 10.5753/sibgrapi.est.2024.31669
- [22] M. Grandini, E. Bagli, and G. Visani, “Metrics for multi-class classification: an overview,” *arXiv preprint arXiv:2008.05756*, 2020. DOI: 10.48550/arXiv.2008.05756
- [23] L. Breiman and J. H. Friedman and R. A. Olshen and C. J. Stone, “*Classification And Regression Trees*,” Routledge, 2017. DOI: 10.1201/9781315139470
- [24] L. Breiman, “Random Forests,” *Machine Learning*, 45(1):5–32, 2001. DOI: 10.1023/a:1010933404324
- [25] T. M. Cover and P. E. Hart, “Nearest neighbor pattern classification,” *IEEE Transactions on Information Theory*, 13(1):21–7, 1967. DOI: 10.1109/TIT.1967.1053964
- [26] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, 20(3):273–97, 1995. DOI: 10.1007/BF00994018
- [27] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, 770–8, 2016.
- [28] J. Deng, W. Dong, R. Socher, L. -J. Li, K. Li and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, 2009. DOI: 10.1109/CVPR.2009.5206848
- [29] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of Machine Learning Research*, 9:2579–605, 2008.
- [30] B. S. Anami and M. C. Elemmi, “Comparative analysis of SVM and ANN classifiers for defective and non-defective fabric images classification,” *The Journal of The Textile Institute*, 113(6):1072–82, 2022. DOI: 10.1080/00405000.2021.1915559
- [31] C. Chaudhari, R. K. Gupta, and S. Fegade, “A hybrid method of textile defect detection using GLCM, LBP, SVD and Wavelet Transform,” *Internat. J. Recent Technol. and Engrg*, 8(6):5356–60, 2020. DOI: 10.35940/ijrte.F9569.038620
- [32] M. Boluki and F. Mohanna, “Inspection of textile fabrics based on the optimal Gabor filter,” *Signal, Image and Video Processing*, 15(7):1617–25, 2021. DOI: 10.1007/s11760-021-01897-3
- [33] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal Of Software Tools*, 2000. URL: <https://github.com/opencv/opencv/>
- [34] A. Paszke et al, “PyTorch: An imperative style, high-performance deep learning library,” 2019. DOI: 10.48550/arXiv.1912.01703
- [35] F. Pedregosa et al, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, 12:2825–30, 2011.