Application of machine learning methodology for textile defect detection

DOI: 10.35530/IT.076.03.2024108

KEMAL GOKHAN NALBANT BERKAN BOZKURT

ABSTRACT – REZUMAT

Application of machine learning methodology for textile defect detection

This study investigates the use of artificial intelligence (AI) and machine learning (ML) technologies in the textile industry, particularly emphasising how they improve operational efficiency and enhance product quality. Using a comprehensive dataset obtained from textile manufacturing operations, a specially tailored convolutional neural network (CNN) model and a long-short-term memory (LSTM) model are implemented for the classification of fabric defects. After undergoing intensive training and validation, our model showed significant improvements in performance over a large number of epochs. The CNN model started with 61.15% accuracy initially and reached 92.91% accuracy after training. The validation accuracy increased from 72.44% to 92.05%. On the same dataset, the LSTM model resulted in 86.11% training accuracy and 87.80% validation accuracy. The significant improvements in accuracy highlight the power of AI and ML to not only improve classification accuracy but also boost overall operational performance by continuously learning from fresh data inputs. Moreover, this research highlights the impact of AI and ML breakthroughs on textile production as they optimise procedures, enhance efficiency, and strengthen competitive advantage. The findings demonstrate that these technologies are a substantial advancement for the textile sector, providing powerful tools to reduce faults, streamline production processes, and ultimately provide goods of superior quality. Therefore, the study promotes the wider use of AI and ML technologies in the textile manufacturing industry, emphasising their crucial role in driving future advancements and sustainable growth.

Keywords: *artificial intelligence, convolutional neural networks, long short-term memory, machine learning, textile defect detection, textile industry*

Aplicarea metodologiei de învățare automată pentru detectarea defectelor textile

Acest studiu investighează utilizarea inteligenței artificiale (AI) și a tehnologiilor de învățare automată (ML) în industria textilă, cu un accent deosebit pe modul în care acestea îmbunătățesc eficiența operațională și calitatea produselor. Folosind un set cuprinzător de date obținute din operațiunile de producție textilă, un model de rețea neuronală convoluțională (CNN) special adaptat și un model de memorie pe termen scurt și lung (LSTM) sunt implementate pentru clasificarea defectelor materialelor textile. După ce a fost supus unei instruiri și validări intensive, modelul nostru a prezentat îmbunătățiri semnificative ale performanței pe parcursul unui număr mare de epoci. Modelul CNN a început inițial cu o precizie de 61,15% și a atins o precizie de 92,91% după formare. Precizia validării a crescut de la 72,44% la 92,05%. Pe același set de date, modelul LSTM a avut o acuratețe de 86,11% la formare și 87,80% la validare. Îmbunătățirile semnificative ale preciziei evidențiază puterea AI și ML nu numai de a îmbunătăți precizia clasificării, ci și de a stimula performanța operațională generală prin învățarea continuă de la noi date de intrare. În plus, această cercetare evidențiază impactul descoperirilor AI și ML asupra industriei textile, deoarece acestea optimizează procedurile, sporesc eficiența și consolidează avantajul competitiv. Constatările demonstrează că aceste tehnologii reprezintă un progres substanțial pentru sectorul textil, oferind instrumente puternice pentru reducerea defectelor, eficientizarea proceselor de producție și, în cele din urmă, furnizarea de bunuri de calitate superioară. Prin urmare, studiul promovează utilizarea pe scară mai largă a tehnologiilor AI și ML în industria textilă, subliniind rolul lor crucial în promovarea progreselor viitoare și a creșterii durabile.

Cuvinte-cheie*: inteligență artificială, rețele neuronale convoluționale, memorie pe termen scurt, învățare automată, detectarea defectelor textile, industria textilă*

INTRODUCTION

Artificial intelligence (AI) has revolutionised people's lives by enabling the efficient execution of repetitive jobs with enhanced precision [1]. Globalisation has significantly increased people's awareness and understanding of fashion and high-quality apparel. Under these conditions, textile and garment manufacturing companies face a significant demand that they must fulfil [2]. Various machine learning (ML) methodologies have demonstrated their ability to generalise well, not just by enhancing classification accuracy over time but also by acquiring knowledge from novel instances [3]. The advent of AI has revolutionised the way organisations operate and oversee their operations. AI has demonstrated its efficacy in the textile industry by streamlining processes, enhancing productivity, and bolstering competitiveness [4].

Currently, the textile sector extensively utilises electronic equipment equipped with high acquisition rate sensors to gather real-time and uninterrupted data [5].

Researchers have established a connection between the fundamental structural and chemical properties of textile materials [6]. Various traditional mathematical and statistical models have been employed in several textile research projects to analyse textile data [7]. AI has been responsible for driving significant advancements in developing technologies, leading to improvements in several parts of our everyday lives and industries. AI offers solutions to extract valuable insights from large volumes of data generated by industrial processes and online user activities [8]. The quality of an organisation's human resources is dependent on the collection and proficiency of its employees' competencies [9].

Upon reviewing the existing literature, it was observed that ML studies are scarce, specifically focused on detecting defects in textiles. Yildirim et al. [7] offered a comprehensive explanation of the application of data mining techniques, including classification and clustering, and ML algorithms in the textile industry. They aim to showcase the application of these techniques in addressing various problems that traditional methods cannot effectively solve. Shahrabadi et al. [3] provided a concise overview of defect categories and automated optical inspection (AOI), mostly utilising ML approaches. Arora & Majumdar [10] conducted bibliometric, network, and content analyses of research publications in the field of ML and supply chain applications in the textile and clothing industry. Fang et al. [11] created a personalised mobile application (APP) using an internal algorithm to enable simple exchange of health data and provide data-driven cardiovascular diagnostics with only one click.

Kahraman and Durmusoglu [12] aimed to evaluate the use of deep learning for detecting fabric defects. Therefore, they analysed articles that specifically focus on fabric defect detection using deep learning techniques. They conducted a comparative analysis of these works' methodologies, databases, performance rates, comparisons, and architecture types. Guder et al. [13] investigated to assess the efficacy of various deep learning frameworks in accurately categorising fabric faults that are frequently seen in the textile sector in Turkey. A novel data set is created specifically for this purpose, which includes fabric flaws like lines, wrinkle marks, machine oil leaks, holes, and bleaching. Moreover, the efficacy of the Adam and Ranger optimisation functions in detecting defects has been assessed using different models in conjunction with explainable artificial intelligence. The results suggest that the ResNet18+Adam model, despite its simplicity and shallow architecture, achieved a remarkable level of success with an accuracy of 99.30%. In contrast, the more intricate EfficientNetv2m+Adam model demonstrated a remarkable accuracy of 99.42%.

Dlamini et al. [14] implemented a real-time machine vision system for detecting defects in functional textiles. The model was constructed, trained, and evaluated using functional textiles. Their model was implemented on an industrial computer, which received functional textile fabric data from hardware that they specifically created for defect inspection. Their system has attained a precision rate of 95.3%, along with recall and F1 scores of 93.6% and 94.4%, respectively. Jeyaraj and Samuel Nadar [15] focused on designing and developing a computer-aided system for detecting and classifying fabric defects using advanced machine learning algorithms. A complex convolutional neural network is constructed to acquire knowledge from diverse defect data sets during the training phase. During the testing phase, the authors have employed a learning feature to classify defects. The enhancement in the accuracy of defect categorisation has been accomplished by utilising a deep learning method. The researchers evaluated the precision of fault classification on six distinct fabric materials and achieved an average accuracy of 96.55%. This accuracy was determined with a sensitivity of 96.4% and a success rate of 0.94.

Soma and Pooja [16] introduced a novel methodology for identifying faults and defects in the offered fabric samples. Their proposed methodology achieved superior performance with a 95% accuracy rate utilising a neural network and an 85% accuracy rate using SVM. The neural network (NN) classifier is employed to categorise fabrics as either normal or faulty. Ultimately, fault localisation and fabric identification are utilised to determine whether the fabric sample was defective or normal. Liu et al. [17] presented an enhanced version of the YOLOv4 algorithm that achieves superior accuracy in detecting fabric defects. Their improvement involved the use of a novel SPP structure that utilises SoftPool instead of MaxPool. The enhanced YOLOv4 algorithm, equipped with three SoftPools, efficiently handles the feature map, resulting in a notable reduction in the adverse impacts of the SPP structure and a boost in detection accuracy. Huang et al. [18] proposed a highly effective convolutional neural network to accurately identify and localise defects. Their framework's architecture effectively reduces the expense of manually annotating the dataset. It required just a small number of defect samples, along with standard samples, to learn the possible characteristics of defects and accurately identify their locations. Their network is partitioned into two components: segmentation and decision. Their suggested approach required around 50 defect samples to get reliable segmentation findings and can meet the real-time detection requirement at a speed of 25 frames per second (FPS).

In this study, a classification process was performed on the textile defect detection dataset using both convolutional neural networks (CNN) and long short-term memory (LSTM) models. In the Materials and Methods section, datasets and pre-processing steps are detailed. In the classification section, the architecture of the CNN model was created, the training process was analysed, and model performance was evaluated. Similar steps were followed for the LSTM model for comparison. After the completion of the training processes, the performances of both models were compared. In the Discussion section, the results of the models used were compared with four different studies in the literature. Finally, in the Conclusions section, a general evaluation of the study was made, and the obtained findings were discussed comprehensively.

MATERIALS AND METHODS

In this study, the dataset underwent numerous pre-processing stages and model training. The model's performance was assessed using the test dataset, and the results were thoroughly studied. Thus, a highly efficient technique was devised for identifying uncommon irregularities in textile materials. This approach may be regarded as a crucial measure to enhance quality control in textile manufacturing procedures and streamline the identification of faulty items.

Dataset

MVTec provided the dataset used in this study, which was designed to detect unusual abnormalities in textiles. Rare anomalies that occur during textile production can have a significant impact on fabric quality, and detecting them is critical to quality control. This dataset was created for educational purposes and may be used to conduct different anomaly detection research. MVTec is a dataset used for industrial image processing and is widely preferred, especially for anomaly detection. This dataset consists of high-quality images and is suitable for the detection of various industrial defects. The dataset used in our study is optimised for industrial defect detection, similar to MVTec.

The collection comprises photos with pixel sizes of 32x32 and 64x64, as well as several anomaly classifications. These classes include "good", "colour", "cut", "hole", "thread", and "metal contamination". Each picture includes eight possible rotation angles: 0, 20, 40, 60, 80, 100, 120, and 140 degrees. The training and test datasets comprise randomly produced patches, whereas the patches acquired from the source photos do not overlap. This dataset facilitates the conduct of diverse activities across a range of research and application domains. These include class type classification, angle classification, and texture representation learning (also known as self-supervised learning). Class-type categorisation seeks to differentiate several sorts of abnormalities. Angle classification may be used to categorise angles using just "good" photographs while testing images from other classes. Texture representation learning, also known as self-supervised learning, seeks to develop a robust representation of texture rather than typical image processing characteristics. The dataset, in .h5 format, comprises 64×64 pixel patch files classified by error type and randomly sampled. The patches are extracted at various angles, allowing the model to detect abnormalities from several perspectives. This dataset is based on MVTec's publicly accessible dataset. The publication "MVTec AD: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection" by Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger describes the dataset in full. This study was presented at the 2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Table 1 provides details regarding our data collection, whereas figure 1 displays a representative picture [19, 20].

Table 1

DATASET INFORMATION

Feature	Description
Dataset provider	MVTec company
Purpose of the dataset	Detection of rare anomalies in textile fabrics
Image sizes	32x32 pixels, 64x64 pixels
Classes	good, colour, cut, hole, thread, metal_contamination
Rotations	0, 20, 40, 60, 80, 100, 120, 140
Dataset structure	Training and test datasets contain randomly generated patches. Patches from source images are non-overlapping
File format	.h5 format

Fig. 1. Example image from the dataset

Pre-processing

Data pre-processing is a critical step in improving the accuracy and dependability of machine learning models. This study employs a textile defect dataset made up of 64×64 pixel grayscale pictures with six distinct categories (['good', 'colour', 'cut', 'hole', 'thread', 'metal_contamination']) and various rotation angles. Initially, training and testing datasets were seeded with random patches selected from non-overlapping source pictures. The initial part entailed preparing the dataset for the binary classification problem. The original dataset had six distinct classifications, but these were reduced to two main categories: "damaged" and "good". This update makes the dataset easier to manipulate and analyse. TensorFlow completed the class transformation algorithm during this stage. The model's performance may suffer because of the dataset's class imbalance. Because the "good" class includes fewer examples, data augmentation techniques were applied to it. Three extra copies were created for each training example, yielding four additional records with "good" category data augmentation. Techniques employed to enhance the data included rotation, cropping, and scaling. These tactics allowed the model to train on a wider range of data, which improved its generalisation skills. The H5ToStorage object was used to process and save the data. This technique reduced the dataset's memory usage and improved data loading times. The training and test datasets were kept separate and processed by the TensorFlow dataset API. These strategies ensured that the dataset was properly used and sped up the model training process. Several graphs were generated to highlight the effectiveness of the data preparation methods. Figure 2 depicts the distribution of raw data into distinct classifications. This graph demonstrates the dataset's class imbalances. Initially, the "damaged" class takes up a large portion of the dataset, and this imbalance has the potential to undermine the model's performance.

Figure 3 depicts the class distribution of a balanced dataset using data augmentation approaches. This graph demonstrates that the "good" and "damaged" classes in the dataset have a more balanced distribution. Data augmentation enabled the model to learn both classes equally well.

Figure 4 depicts samples of training data, demonstrating the diversity and structure of pictures from the "damaged" and "good" classes. These photos show that the dataset is appropriately tagged.

As a result, the data pre-treatment methods in this study include those necessary to eliminate class imbalances and make the dataset TensorFlow compliant. These pre-processing techniques guarantee that the model is trained more evenly and efficiently. Data augmentation and data loading techniques help the model train faster and perform better overall. This improves the ML model's accuracy and dependability dramatically.

CLASSIFICATION

Convolutional neural networks (CNNs) are models that have been extremely successful in the field of deep learning, particularly in image processing applications. CNNs employ convolutional techniques to extract features from pictures. The initial layers extract simpler and low-level characteristics, whereas the subsequent layers extract more complicated and high-level information. In this approach, the model learns to spot essential patterns in photos. CNN's fundamental structure includes convolution layers, pooling layers, nonlinear activation functions, and fully connected layers. CNN design starts with convolution layers. These layers use filters to extract various characteristics from the incoming data. Each filter focuses on a certain feature, such as edge detection. For example, in the model employed in this work, the first convolution layer recovers low-level features using three 1×1 filters, whereas the succeeding layers extract more complicated features with bigger filters. In this model, using an L2 regularizer prevents overfitting. After the convolution layers, the pooling layers appear. The pooling technique is utilised to minimise the size of the convolution-derived feature maps as well as the computational burden. For example, Max Pooling determines the maximum value for each depth by traversing with a specific filter size and step interval. This procedure strengthens the model's translation invariance.

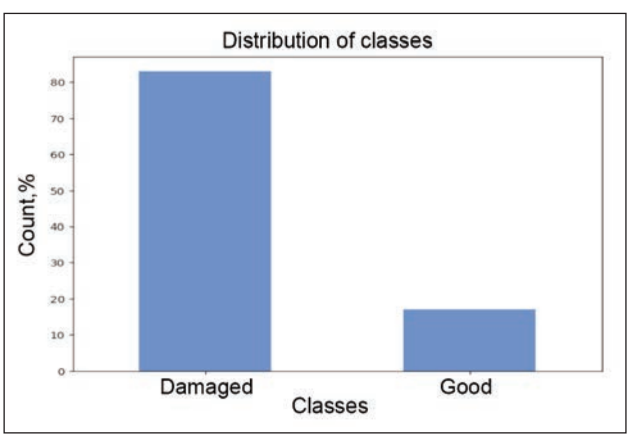

Fig. 2. Initial class distribution

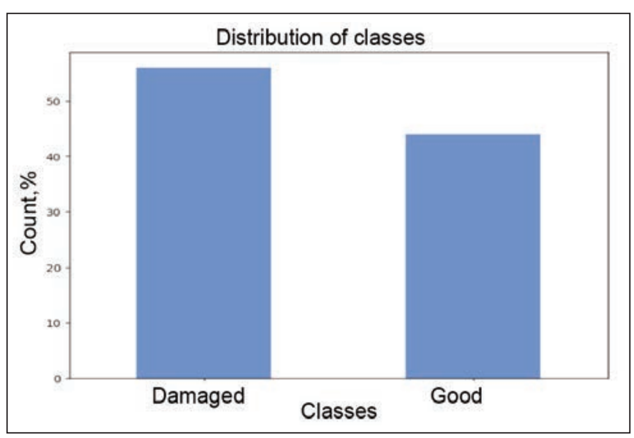

Fig. 3. Balanced class distribution after data augmentation

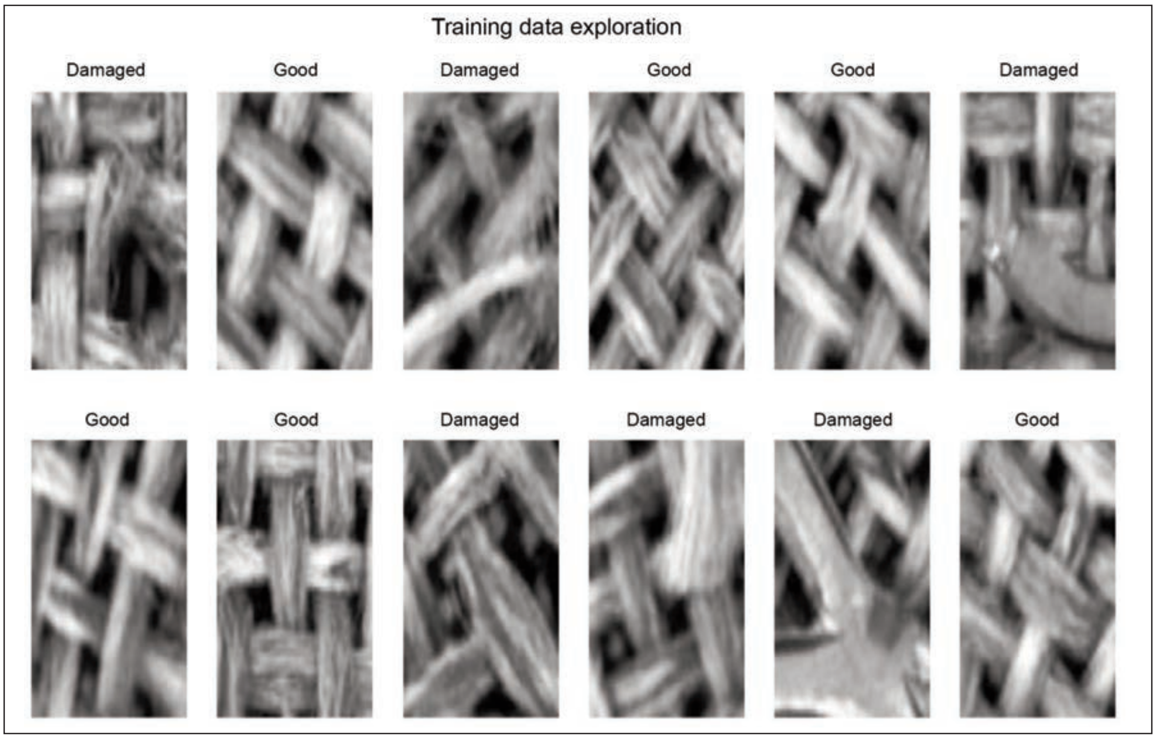

Fig. 4. Training data exploration

Activation functions are critical components that improve the performance of CNNs. This model employs the ReLU (Rectified Linear Unit) activation function. ReLU reduces negative pixel values to zero while keeping positive ones constant. These speed up the network's learning process and increase its effectiveness. The flattening procedure is then followed. The flattening process transforms the final matrix into a vector structure, which artificial neural network models can use as input. This change makes the characteristics extracted from the photos compatible with more typical artificial neural network models. Finally, fully linked layers appear. These layers incorporate the characteristics from the last level of the network and carry out the classification procedure. Fully linked layers enable the model to learn complicated relationships. However, true communication between these layers necessitates the tuning of a huge number of parameters. As a result, strategies like dropout may be employed to minimise the network's overall complexity while also speeding up training. The basic CNN model employed in this work has convolution layers with sizes of 1×1, 2×2, 3×3, 4×4, and 4×4, respectively. The first layer has three filters, while the subsequent levels have 16, 32, 64, 128, and 512 filters, respectively. The model is composed of two pooling layers, both of which use Max Pooling to reduce the size of the feature maps. A single neuron with a sigmoid activation function performs binary classification after the Global Average Pooling Layer reduces the feature maps to a single vector. The purpose of this structure is to efficiently execute tasks such as feature extraction and classification. The model's overall structure and function explain why CNNs perform so well in image processing tasks.

In this study, the CNN model is used as the basis to perform the image classification task. CNNs are considered to be a suitable model for this task, especially because they successfully capture spatial relationships and features in image data. However, to provide a broader perspective by comparing the performance of the CNN model with another model, the LSTM model is also evaluated. Although LSTM is designed to process sequential data and time series data, it is tried as a different approach and applied to the same dataset as the CNN model.

LSTM (long short-term memory) networks are a type of recurrent neural network developed specifically to capture long-term dependencies in sequential data. The main advantage of LSTM is that it can retain previous information in memory for a longer period, thanks to the cell state. The forgetting gates in this structure determine when previous information is preserved and when it is forgotten, and this mechanism strengthens learning by providing access to past information at each step in the sequence. This feature makes LSTM successful in sequential data such as natural language processing and time series analysis. So, LSTM was applied to the image classification task and compared with the CNN model.

The CNN and LSTM evaluation criteria used in this study are dimensionless, meaning that these metrics do not include any physical dimensions or units when measuring model performance. Thus, the results of both models can be compared universally. The inclusion of the LSTM model was performed to provide an alternative perspective and to be able to perform a comparative analysis with CNN. Considering the success of LSTM on sequential data, it was wondered how it would perform on the image classification task. The performance of both models on the dataset was compared, thus providing an additional analysis to establish model preferences on a more solid basis. As a result, the LSTM model was used for comparison and was subjected to performance analysis on the same task as the CNN model.

Applying CNN to the dataset

This study used the MVTec dataset and a CNN to detect unusual abnormalities in textiles. This dataset permits a study into the detection of infrequent abnormalities that play an important role in quality control procedures.

Creating the model architecture

Table 2 details the construction of a TensorFlow sequential model named "baseline_model". The CNN architecture, a deep learning approach, serves as the foundation for this model. The model has several distinct layers and a complex structure, each serving a specific function. First, the model's input layer receives single-channel (grayscale) 64×64 pixel pictures. The convolutional layers come next, and they provide the model's feature extraction capabilities. These layers use convolution to learn various visual properties. Each convolution layer generates a particular number of filters and a certain output size. For example, the first convolution layer applies three 1×1 filters to the 64×64 input pictures to generate a 64×64×3 output. The pooling layers follow the convolution layers. These layers are used to minimise the size of feature maps while also representing the scale and placement of learned features over a larger region. For example, the first pooling layer uses a pooling technique to reduce the 64×64 feature maps to 2×2. The last layer of the model is known as the Fully Connected (Dense) Layer. This layer is used to merge the characteristics from the preceding layers and complete the classification process. In this model, the sigmoid activation function generates the output from a single neuron in the last layer. The total number of parameters indicates the weights and biases that the model may learn during training. There are 1,201,591 parameters in this model. The model can modify all these trainable parameters as it undergoes training. There are no untrainable parameters, which means that the model's parameters all have fixed structures. Table 2 is a significant tool for assessing the model's complexity and learning potential.

Compiling and training the CNN Model

During the training process, the model was trained on a dataset consisting of 108,000 images. This dataset consists of images belonging to two classes, 'damaged' and 'good'. After the model training was completed, a test dataset consisting of 72,000 images was used to evaluate the performance of the model. This test dataset also consists of images belonging to two classes.

Figure 5 details the training and performance of a deep learning model. This data depicts the model's progress during each epoch, as well as the many metrics used to evaluate its success. Large datasets gathered over numerous epochs are routinely used to train deep learning models. Each epoch is a dataset that the model processes only once. During each epoch, the model refines its random parameters at a set learning rate. This method assesses metrics like as model accuracy and loss across both the training and validation datasets. At the beginning of the training phase, the model had low accuracy and a significant loss. This shows that the model hasn't fully mastered the dataset's patterns. In the first epoch, the training accuracy was 61.15%, while the training loss was 0.9853. Validation accuracy is 72.44%, with a loss of 0.4640. In succeeding epochs, the model begins to learn patterns from the training dataset, and training accuracy quickly improves as loss reduces. For example, by the third epoch, training accuracy had increased to 88.51% and validation accuracy to 89.89%. Similarly, training loss fell to 0.3207, while validation loss dropped to 0.2657.

Table 2

DETAILS OF THE MODEL ARCHITECTURE

Layer (type)	Output shape	Parameter
conv2d (Conv2D)	(None, 64, 64, 3)	6
conv2d_1 (Conv2D)	(None, 63, 63, 16)	208
conv2d_2 (Conv2D)	(None, 62, 62, 32)	2,080
conv2d_3 (Conv2D)	(None, 60, 60, 64)	18,496
max_pooling2d (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_4 (Conv2D)	(None, 27, 27, 128)	131,200
max_pooling2d_1 (MaxPooling2D)	(None, 13, 13, 128)	0
conv2d_5 (Conv2D)	(None, 10, 10, 512)	1,049,088
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
output_layer (Dense)	(None, 1)	513

```
Epoch 1/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1253s 741ms/step - accuracy: 0.6115 - loss: 0.9853 - val_accuracy: 0.7244 - val_loss: 0.4640 - learning_rate: 0.0010
Epoch 2/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1355s 803ms/step - accuracy: 0.7920 - loss: 0.4670 - val_accuracy: 0.8704 - val_loss: 0.3225 - learning_rate: 0.0010
Epoch 3/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1318s 781ms/step - accuracy: 0.8851 - loss: 0.3207 - val_accuracy: 0.8989 - val_loss: 0.2657 - learning_rate: 0.0010
Epoch 4/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1331s 788ms/step - accuracy: 0.8942 - loss: 0.2963 - val_accuracy: 0.9040 - val_loss: 0.2509 - learning_rate: 0.0010
Epoch 5/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1317s 780ms/step - accuracy: 0.8997 - loss: 0.2825 - val_accuracy: 0.9056 - val_loss: 0.2506 - learning_rate: 0.0010
Epoch 6/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1333s 790ms/step - accuracy: 0.9029 - loss: 0.2723 - val_accuracy: 0.9169 - val_loss: 0.2270 - learning_rate: 0.0010
Epoch 7/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1329s 787ms/step - accuracy: 0.9063 - loss: 0.2650 - val_accuracy: 0.9181 - val_loss: 0.2239 - learning_rate: 0.0010
Epoch 8/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 0s 681ms/step - accuracy: 0.9090 - loss: 0.2579
Epoch 8: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1254s 743ms/step - accuracy: 0.9090 - loss: 0.2579 - val_accuracy: 0.9089 - val_loss: 0.2440 - learning_rate: 0.0010
Epoch 9/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1249s 740ms/step - accuracy: 0.9192 - loss: 0.2362 - val_accuracy: 0.9225 - val_loss: 0.2117 - learning_rate: 2.0000e-04
Epoch 10/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1238s 733ms/step - accuracy: 0.9206 - loss: 0.2311 - val_accuracy: 0.9240 - val_loss: 0.2082 - learning_rate: 2.0000e-04
Epoch 11/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1243s 736ms/step - accuracy: 0.9218 - loss: 0.2279 - val_accuracy: 0.9269 - val_loss: 0.2041 - learning_rate: 2.0000e-04
Epoch 12/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1253s 743ms/step - accuracy: 0.9226 - loss: 0.2254 - val_accuracy: 0.9278 - val_loss: 0.1998 - learning_rate: 2.0000e-04
Epoch 13/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1090s 645ms/step - accuracy: 0.9235 - loss: 0.2233 - val_accuracy: 0.9275 - val_loss: 0.1974 - learning_rate: 2.0000e-04
Epoch 14/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.9244 - loss: 0.2215
Epoch 14: ReduceLROnPlateau reducing learning rate to 4.0000001899898055e-05.
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 2058s 1s/step - accuracy: 0.9244 - loss: 0.2215 - val_accuracy: 0.9291 - val_loss: 0.1980 - learning_rate: 2.0000e-04
Epoch 15/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 0s 667ms/step - accuracy: 0.9282 - loss: 0.2124
Epoch 15: ReduceLROnPlateau reducing learning rate to 8.000000525498762e-06.
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1234s 731ms/step - accuracy: 0.9282 - loss: 0.2124 - val_accuracy: 0.9084 - val_loss: 0.2384 - learning_rate: 4.0000e-05
Epoch 16/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 0s 688ms/step - accuracy: 0.9291 - loss: 0.2099
Epoch 16: ReduceLROnPlateau reducing learning rate to 1.6000001778593287e-06.
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 1258s 746ms/step - accuracy: 0.9291 - loss: 0.2099 - val_accuracy: 0.9205 - val_loss: 0.2154 - learning_rate: 8.0000e-06
```

Fig. 5. CNN model training process

As the model's performance improves, validation accuracy and loss stabilise or barely change. During this phase, the model frequently lowers its learning rate to allow for more precise adjustments. For example, in the ninth epoch, the learning rate decreased from 0.001 to 0.0002. This update increased validation accuracy and loss. Training accuracy was 90.90%, while validation accuracy was 90.89%. As the training procedure advances, the model's accuracy and loss become more consistent. For example, in the thirteenth epoch, training accuracy grew to 92.35%, validation accuracy increased to 92.75%, and training loss fell to 0.2233 and 0.1974, respectively. The learning rate is gradually reduced to improve the model's performance. In the sixteenth epoch, the learning rate was lowered to 8.0000e-06, which resulted in 92.91% training and 92.05% validation accuracy. As a result, the model was changed to maintain a specific level of accuracy and loss during the training stage. Reducing the learning rate improved the model's performance on the validation dataset without causing overlearning. This approach shows how to train and upgrade deep learning models. These training metrics can be used to evaluate the model's performance and plan future enhancements.

CNN model evaluation

During model training, accuracy and loss values on the training and validation datasets at the end of each epoch must be monitored. Visualising how these variables change over time helps us understand the model's learning and performance. This representation allows us to see how well the model matches training data and performs on new data.

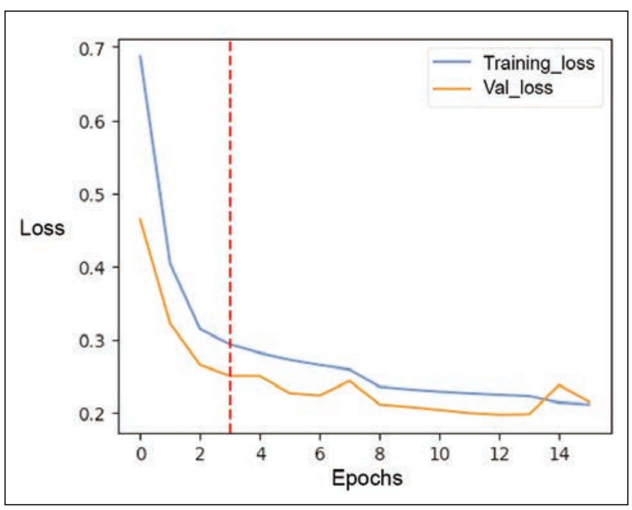

Fig. 6. Training and validation loss of the CNN model

The loss graph in figure 6 depicts the model's error rates in the training and validation datasets on an epoch basis, allowing us to assess the model's learning effectiveness and generalizability. This graph contains two key components: training loss and validation loss. At the start of the graph, the training loss is relatively significant, indicating that the model is having trouble recognising patterns in the dataset. In the first epoch, the training loss is 0.9853. However, as the epochs advance, the model begins to understand patterns in the dataset, and the training loss drops dramatically. This loss decreases dramatically in the second epoch to 0.4670. This demonstrates that the model optimises immediately at the start of the training session and learns patterns rapidly. Validation loss exhibits a similar declining pattern. The validation loss is significant in the first epoch but gradually declines and reaches 0.2657 in the third epoch. This demonstrates that the model not only learns the patterns in the training dataset but also accurately generalises to the patterns in the validation dataset. The red dashed line in the graph represents the point at which the learning rate initially decreased. From this point on, both the training and validation losses become more consistent. Reducing the learning rate allows the model to be tuned in smaller stages, allowing for more precise tweaking. During this phase, the training loss decreases gradually but consistently. In the sixteenth epoch, the training loss fell to 0.2099, while the validation loss fell to 0.2154. Overall, the loss graph indicates that the model finishes the learning process swiftly and effectively. The simultaneous drop in training and validation losses suggests that the model works well on both datasets while avoiding overfitting. This demonstrates that the model improves in a balanced manner on both the training and validation datasets, indicating a successful training procedure.

The accuracy graph in figure 7 displays the model's accuracy rates on the training and validation datasets on an epoch basis, allowing us to assess how effectively the model classifies. The graph has two key components: training accuracy and validation accuracy. The training accuracy was quite poor at the start of the procedure. In the first epoch, the training accuracy was 61.15%. This demonstrates that the model initially struggles to recognise patterns in the dataset, resulting in several false classifications. However, as the epochs advance, the model begins to recognise patterns in the dataset, and training accuracy rapidly

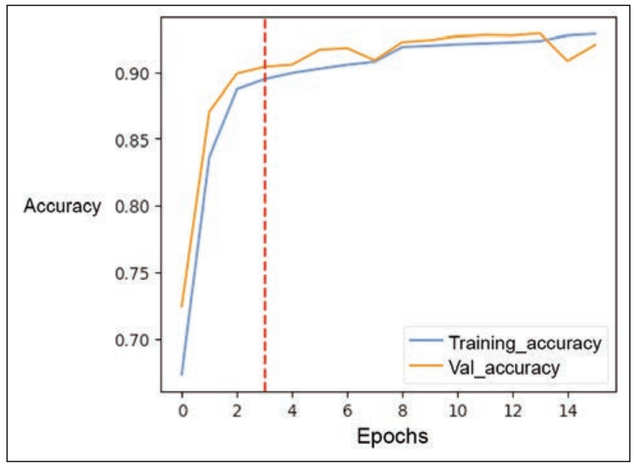

Fig. 7. Training and validation accuracy of the CNN model

improves. A significant improvement in training accuracy is noticed, particularly during the first few epochs. In the third period, training accuracy reaches 88.51%. This demonstrates that the model optimises effectively at the start of the training session and learns patterns rapidly. Validation accuracy examines the model's capacity to generalise. Validation accuracy is likewise poor at the start of the training process, but it improves quickly in tandem with training accuracy. During the first few epochs, validation accuracy improves noticeably. In the third epoch, the validation accuracy was roughly 89.89%. This demonstrates that the model generalises effectively to both the patterns in the validation and training datasets. The graph's red dashed line indicates the initial decrease in the learning rate. From this point on, both the training and validation accuracy are more stable. Reducing the learning rate enables the model to be optimised in smaller stages, allowing for more precise tweaking. Throughout this procedure, training accuracy improves gradually but steadily. In the sixteenth epoch, the training accuracy was 92.91%, and the validation accuracy was 92.05%. Overall, the accuracy graph indicates that the model finished the learning process quickly and effectively. The simultaneous improvement in training and validation accuracy implies that the model performed well on both datasets while avoiding overtraining. This demonstrates that the model improved both accuracy and loss measures in a balanced manner and that the training procedure was effective. These graphs highlight the model's performance and optimisation

Table 3

CLASSIFICATION REPORT				
Parameter	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.98	0.94	0.96	60000
1	0.74	0.88	0.80	12000
accuracy			0.93	72000
macro avg	0.86	0.91	0.88	72000
weighted avg	0.94	0.93	0.93	72000

during the training phase. The parallel trend in training and validation losses and accuracies demonstrates that the model has a high generalisation capacity and minimises overfitting concerns. The model learned the patterns in the dataset rapidly and performed well on both the training and validation datasets. This is a positive measure of the model's overall success and efficiency. Table 3 displays the model's accuracy, precision, recall, and F1 score.

Compiling and training the LSTM model

Figure 8 presents statistical data obtained during the training process of an LSTM model. The training process was monitored for 20 epochs, and the performance of the model on the training and validation sets was evaluated at the end of each epoch. In the first epoch, the training accuracy of the model was 54.51%, while the training loss was recorded as 0.6894. At this stage, the model had not yet learned the patterns in the dataset well enough. However, as the training process progressed, the accuracy of the model increased continuously and reached 86.11% in the 20th epoch. The training loss decreased to 0.3175 in the 20th epoch as the errors of the model gradually decreased. This shows that the optimisation process of the model was successful.

The model's performance on the validation set has also improved similarly. The validation accuracy, which was 54.20% in the first epoch, increased to 87.80% in the 20th epoch. This increase shows that the model can generalise well not only on the training data but also on the validation data. Similarly, the validation loss decreased from 0.6735 in the 1st epoch to 0.2783 in the 20th epoch. This decrease in the validation loss reveals that the model's errors on the validation sets are reduced, and overfitting is avoided. The learning rate dynamics also played an important role in the performance of the model.

The learning rate, which was initially 0.0010, was gradually reduced by the ReduceLROnPlateau algorithm when the validation loss stopped improving. For example, when no significant improvement in validation loss was observed at epoch 9, the learning rate was reduced to 0.0002. This process continued until

```
Epoch 1/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 190s 110ms/step - accuracy: 0.5451 - loss: 0.6894 - val_accuracy: 0.5420 - val_loss: 0.6735 - learning_rate: 0.0010
Epoch 2/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 159s 94ms/step - accuracy: 0.5882 - loss: 0.6641 - val_accuracy: 0.5977 - val_loss: 0.6097 - learning_rate: 0.0010
Epoch 3/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 159s 94ms/step - accuracy: 0.6979 - loss: 0.5669 - val_accuracy: 0.7350 - val_loss: 0.4972 - learning_rate: 0.0010
Epoch 4/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 158s 94ms/step - accuracy: 0.7409 - loss: 0.5130 - val_accuracy: 0.7493 - val_loss: 0.4787 - learning_rate: 0.0010
Epoch 5/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 162s 96ms/step - accuracy: 0.7625 - loss: 0.4835 - val_accuracy: 0.8035 - val_loss: 0.4070 - learning_rate: 0.0010
Epoch 6/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 161s 95ms/step - accuracy: 0.7851 - loss: 0.4534 - val_accuracy: 0.8062 - val_loss: 0.3987 - learning_rate: 0.0010
Epoch 7/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 165s 98ms/step - accuracy: 0.7974 - loss: 0.4318 - val_accuracy: 0.8161 - val_loss: 0.3871 - learning_rate: 0.0010
Epoch 8/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 166s 98ms/step - accuracy: 0.8067 - loss: 0.4149 - val_accuracy: 0.8389 - val_loss: 0.3417 - learning_rate: 0.0010
Epoch 9/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - accuracy: 0.8210 - loss: 0.3921
Epoch 9: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 170s 101ms/step - accuracy: 0.8210 - loss: 0.3921 - val_accuracy: 0.8175 - val_loss: 0.3864 - learning_rate: 0.0010
Epoch 10/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 166s 99ms/step - accuracy: 0.8371 - loss: 0.3656 - val_accuracy: 0.8621 - val_loss: 0.3097 - learning_rate: 2.0000e-04
Epoch 11/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 162s 96ms/step - accuracy: 0.8465 - loss: 0.3477 - val_accuracy: 0.8637 - val_loss: 0.3070 - learning_rate: 2.0000e-04
Epoch 12/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 165s 98ms/step - accuracy: 0.8490 - loss: 0.3421 - val_accuracy: 0.8680 - val_loss: 0.2879 - learning_rate: 2.0000e-04
Epoch 13/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 170s 101ms/step - accuracy: 0.8502 - loss: 0.3384 - val_accuracy: 0.8729 - val_loss: 0.2862 - learning_rate: 2.0000e-04
Epoch 14/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - accuracy: 0.8568 - loss: 0.3328
Epoch 14: ReduceLROnPlateau reducing learning rate to 4.0000001899898055e-05.
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 170s 100ms/step - accuracy: 0.8568 - loss: 0.3328 - val_accuracy: 0.8725 - val_loss: 0.2902 - learning_rate: 2.0000e-04
Epoch 15/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 175s 104ms/step - accuracy: 0.8581 - loss: 0.3250 - val_accuracy: 0.8777 - val_loss: 0.2790 - learning_rate: 4.0000e-05
Epoch 16/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step - accuracy: 0.8610 - loss: 0.3220
Epoch 16: ReduceLROnPlateau reducing learning rate to 8.000000525498762e-06.
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 175s 104ms/step - accuracy: 0.8610 - loss: 0.3220 - val_accuracy: 0.8656 - val_loss: 0.2982 - learning_rate: 4.0000e-05
Epoch 17/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 177s 105ms/step - accuracy: 0.8618 - loss: 0.3209 - val_accuracy: 0.8792 - val_loss: 0.2757 - learning_rate: 8.0000e-06
Epoch 18/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.8634 - loss: 0.3177
Epoch 18: ReduceLROnPlateau reducing learning rate to 1.6000001778593287e-06.
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 178s 105ms/step - accuracy: 0.8634 - loss: 0.3177 - val_accuracy: 0.8698 - val_loss: 0.2927 - learning_rate: 8.0000e-06
Epoch 19/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step - accuracy: 0.8608 - loss: 0.3198
Epoch 19: ReduceLROnPlateau reducing learning rate to 3.200000264769187e-07.
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 178s 106ms/step - accuracy: 0.8608 - loss: 0.3198 - val_accuracy: 0.8712 - val_loss: 0.2902 - learning_rate: 1.6000e-06
Epoch 20/20
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step - accuracy: 0.8611 - loss: 0.3175
Epoch 20: ReduceLROnPlateau reducing learning rate to 1e-07.
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 180s 107ms/step - accuracy: 0.8611 - loss: 0.3175 - val_accuracy: 0.8780 - val_loss: 0.2783 - learning_rate: 3.2000e-07
```

Fig. 8. LSTM model training process

the learning rate dropped to a low level of 3.2000e-07 across epochs. This reduction in the learning rate allowed the model to reach a better minimum with smaller steps and improved the optimisation process. As a result, the LSTM model successfully learned the patterns in the dataset with the statistical performance it showed throughout the training process and was able to generalise this information to the validation set. The high validation accuracy and low validation loss obtained show that the model works effectively and has a good generalisation ability.

LSTM model evaluation

This graph shows the loss values recorded on both the training (training_loss) and validation (val_loss) sets during the training process of the LSTM model. The X-axis represents the epochs, and the Y-axis represents the loss values. The graph shows in detail how the model performs over time and how it progresses in the learning process.

At the beginning of the graph, both the training loss (blue line) and the validation loss (orange line) are at a high level. This shows that the model has not yet fully learned the patterns in the data in the early stages of training. However, after a few epochs, both loss values start to decrease rapidly. This decrease shows that the model starts to perform better on the dataset and that the training process is effective.

The validation loss initially follows a course close to the training loss, and both tend to decrease during this process. This decrease in the validation loss shows that the model's ability to learn not only from the training data but also from the validation data increases. As the epochs progress, both loss values become stable. This indicates that the model has now largely learned the patterns in the dataset. There is limited room for further improvement.

Another striking point in the graph is that the validation loss stabilises at a level very close to the training loss after a certain epoch. This indicates that the generalisation capacity of the model is high. That is, its ability to adapt to new data is strong. The fact that the training loss is slightly lower than the validation loss indicates that the model is slightly more adapted to the training set (fitting), but since this difference is quite small, it cannot be considered as a sign of overfitting.

In general, figure 9 shows graph shows that the model successfully reduces its losses throughout the training process, performs well on the validation set, and eventually both the training and validation losses stabilise at low levels. These results reveal that the model's learning process is effectively managed, and the result obtained is satisfactory in terms of generalisation.

The graph obtained in figure 10 shows that the accuracy and validation accuracy metrics have successfully increased during the training process of the model. When the graph is examined, it is observed that both the training accuracy and validation accuracy have increased rapidly. This increase reveals that

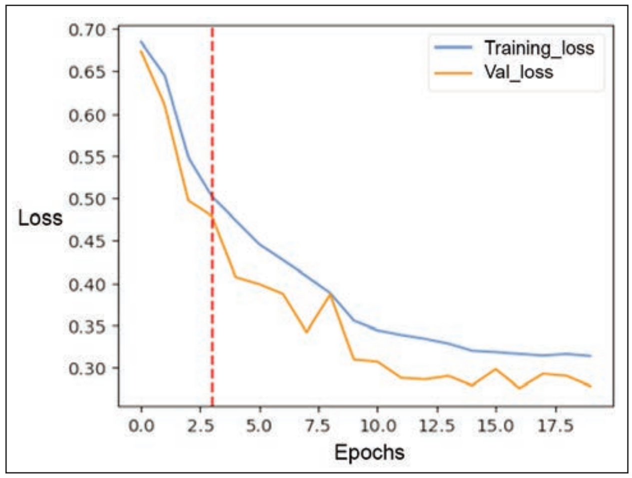

Fig. 9. Training and validation loss of the LSTM model

the model has the capacity to effectively learn the patterns and relationships in the dataset.

Throughout the training process, the steady increase in training accuracy indicates that the model is increasingly grasping the information in the dataset and is continuously improving its performance. Similarly, the increase in validation accuracy parallel to the training accuracy indicates that the model exhibits a successful generalisation ability on new and unseen data. This indicates that the model performs well not only on the training data but also on the validation data.

The accuracy curves observed in the graph reveal that the model has a high learning capacity and maintains its overall performance consistently by maintaining its validation accuracy throughout the training period. These results show that the model performs successfully on both the training and validation sets and has a strong generalisation ability. It can be concluded that the model effectively learns from the dataset by increasing its accuracy over the specified number of epochs and can exhibit robust performance against new data.

As a result, the high accuracy values obtained throughout the training process of the model indicate

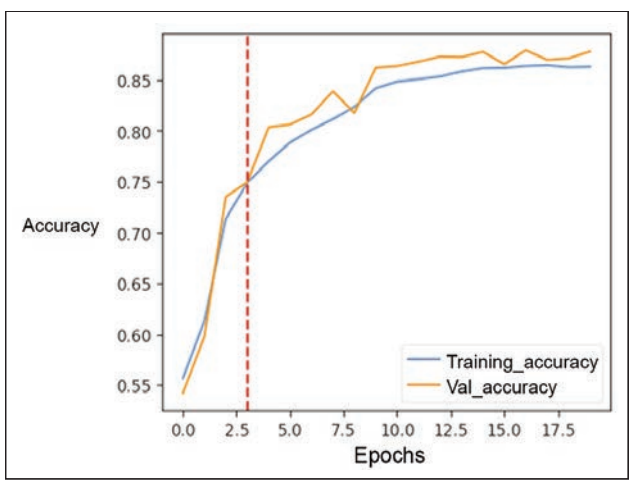

Fig. 10. Training and validation accuracy of the LSTM model

that the model successfully improves its learning ability on the training data and its generalisation capacity on the validation data. These findings demonstrate that the overall performance of the model improves continuously throughout the training period and exhibits strong generalisation ability on the validation data.

Table 4 summarises the classification performance of the LSTM model. The model achieved 87% overall accuracy, exhibiting high accuracy (99% precision, 92% F1-score) in class 0 and strong recall (95%) in class 1. Weighted average metrics reveal that the model demonstrates a successful generalisation ability on the dataset.

Table 5 presents the performance evaluations of the Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) models applied to the dataset. Performance evaluations were made according to the training accuracy, training loss, validation accuracy, and validation loss criteria.

The accuracy rate of the CNN model in the training phase was determined as 92.91%, and the training loss remained at a low level of 0.2099. These results show that the CNN model processed the training data with a high accuracy rate and that the model had a low error margin during the learning process. In the validation phase, CNN also performed quite successfully with an accuracy rate of 92.05% and a loss value of 0.2154. These findings reveal that CNN provided superior overall performance in both the training and validation processes and that the model effectively learned the general features of the data.

In comparison, the training accuracy of the LSTM model was recorded as 86.11%, and the training loss reached a higher value of 0.3175. This result shows that LSTM offers lower learning accuracy and a higher error rate on the training data. In the validation phase, LSTM achieved an 87.80% accuracy rate and 0.2783 loss value. Although LSTM provided slightly higher accuracy in the validation phase, it generally fell behind the performance provided by CNN, considering the limitations in training performance.

When evaluated in general, it reveals that the CNN model provides a significant superiority over LSTM on the dataset. CNN's high training accuracy and low training loss show that the model grasps the characteristics of the data better and is more effective in the learning process. The high accuracy rate in the validation phase also emphasises that the CNN model's generalisation ability is stronger than LSTM. As a result, it has been scientifically clearly demonstrated that CNN provides better results and, therefore, is considered a more effective model in solving the textile defect detection problem.

ALGORITHM TIME PERFORMANCE AND ANALYSIS

The results obtained for the CNN (convolutional neural network) model show that the data was processed quickly and effectively during the training of the model. 72,000 data samples were processed in approximately 14 seconds, and each step of the model was completed in an average of 163 milliseconds. This situation reveals that the CNN model has strong parallel processing ability and can process complex data effectively. This model provides a great advantage, especially when working with high-dimensional and large data sets, such as the analysis of visual data. CNNs are superior in capturing spatial features in the data and are extremely competent in recognising local contexts and patterns.

The results obtained for the LSTM (Long Short-Term Memory) model reveal the model's capacity to process temporal data. The same data set was processed by the LSTM in 13 seconds, and each step was completed in an average of 37 milliseconds. LSTMs are known to exhibit strong performance, especially on time series and sequential data. However, although each step is longer, the data processing speed of the LSTM model is generally higher than the CNN. This is because LSTM learns

Table 4

CLASSIFICATION REPORT				
Parameter	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.99	0.86	0.92	60000
1	0.57	0.95	0.71	12000
accuracy			0.87	72000
macro avg	0.78	0.90	0.82	72000
weighted avg	0.92	0.87	0.89	72000

Table 5

PERFORMANCE COMPARISON OF BOTH MODELS				
Model	**Training accuracy (%)**	**Training loss**	**Validation accuracy (%)**	**Validation loss**
CNN	92.91	0.2099	92.05	0.2154
LSTM	86.11	0.3175	87.80	0.2783

temporal dependencies faster and is better at modelling these dependencies.

When comparing CNN with LSTM models, it becomes clear that while LSTM is best suited for sequential and temporal data sets, CNN has a significant edge over larger and more complicated structures. In particular, CNN models outperform LSTM in fields like computer vision and visual data analysis because of their capability for parallel processing and ability to grasp spatial correlations of data. LSTM, on the other hand, stands out as a more effective model in cases where time series data and sequential dependencies are important. Therefore, when choosing a model, it should be taken into account that the CNN model may be more suitable than the LSTM model, considering the visual structure of the dataset and the characteristics of the problem to be solved. CNN models are generally a preferred approach in our dataset, as they show high performance, especially in visual datasets. Table 6 presents the performance comparison of the Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) models applied to the dataset.

Table 6

PERFORMANCE COMPARISON OF BOTH MODELS

Model	Solution time for the performances (%)	Each step
CNN	14 seconds	163 milliseconds
LSTM	13 seconds	37 milliseconds

DISCUSSION

In this section, a comparative analysis of four related studies is presented. Table 7 summarises the key comparisons between these studies to provide a clearer understanding of their findings and methodologies. Guder et al. [13] focused on the detection of frequently encountered defects in the textile sector in Turkey and used a new dataset specially created for this purpose. In their study, the performance of models such as ResNet18+Adam and EfficientNetv2m+Adam was compared. While the ResNet18 model achieved a high accuracy rate of 99.30% despite its simple and superficial structure, the EfficientNetv2m model, which has a more complex structure, provided a slight superiority with an accuracy rate of 99.42%. Their study is remarkable in terms of showing how effective optimisation functions and deep learning models are in detecting textile defects.

Dlamini et al. [14] developed a real-time machine vision system to detect functional textile defects. This system is integrated on an industrial computer and collects and processes functional textile data with specially designed hardware. The results show that the system is quite successful in industrial applications with a 95.3% accuracy rate, 93.6% recall, and 94.4% F1 score. Their study provides an example that highlights the performance and practical use of deep learning systems in real-time applications.

In the study conducted by Jeyaraj and Samuel Nadar [15], a complex CNN model based on deep learning was used to detect and classify various fabric defects. The study achieved an average accuracy of 96.55% and a sensitivity of 96.4% on 6 different fabric types. The highlight of this study is that deep learning methods can successfully categorise defects using learning features in the defect classification process. These results show that advanced artificial intelligence models such as CNN have great potential for the textile industry. Soma & Pooja [16] presented a new method to detect defects in textile samples using artificial neural networks (NN) and support vector machines (SVM). While the NN model showed higher performance with a 95% accuracy rate, the SVM model reached an 85% accuracy rate. Their study developed an effective method to determine whether textile samples are defective or not, especially by providing localisation and classification of defects.

As a result, these studies show that deep learning and machine learning algorithms provide quite successful results in the detection and classification of textile defects. While Guder et al. [13] and Jeyaraj and Samuel Nadar [15] stand out with their high accuracy rates, Dlamini et al. [14] make a significant contribution in terms of usability in real-time applications.

Table 7

COMPARATIVE ANALYSIS

Authors	Dataset	Methods used	Success rate	References
Guder et al.	Fabric defects (lines, wrinkles, oil leaks, holes)	ResNet18+Adam, EfficientNetv2m+Adam	ResNet18: 99.30%, EfficientNetv2 m: 99.42%	[13]
Dlamini et al.	Real-time data from functional textiles	Real-time machine vision system	95.3% (Precision), 93.6% (Recall), 94.4% (F1)	[14]
Jeyaraj and Samuel Nadar	6 different fabrics materials	Deep learning based CNN	96.55%	[15]
Soma and Pooja	Textile samples	Neural network (NN) ve SVM	NN: 95% SVM: 85%	[16]

Soma and Pooja [16] offer a more general perspective by comparing the performance of different models. These studies reveal that deep learning is an effective solution in quality control processes in the textile industry. Similarly, in this study conducted on the textile defect detection dataset, the performance of CNN and LSTM models also revealed significant findings. The CNN model exhibited high performance, with a 92.91% accuracy rate in the training phase and a 92.05% accuracy rate in the validation phase. This result is similar to the high accuracy rates in the studies of Guder et al. [13] and Jeyaraj and Samuel Nadar [15]. The LSTM model achieved an 86.11% accuracy rate in the training phase and an 87.80% accuracy rate in the validation phase. These results show that although LSTM provides high accuracy in some cases, the CNN model generally performs better.

Comparing the studies in the literature with our results, it is seen that the CNN model is generally more effective in detecting textile defects. This supports the wide application potential of CNN in the field of deep learning, while the LSTM model can also be useful in certain scenarios. These findings emphasise that deep learning methods offer powerful and effective solutions in quality control processes in the textile industry.

CONCLUSIONS

This study investigated the application of artificial intelligence (AI) and machine learning (ML) models – specifically, convolutional neural networks (CNN) and long short-term memory (LSTM) networks – in the detection of textile defects. The primary objective was to evaluate the performance of these models in identifying anomalies in textile images, thereby contributing to enhanced quality control processes in the textile manufacturing industry. The results demonstrated that these advanced technologies hold significant potential in automating defect detection, optimising production processes, and improving the overall efficiency and accuracy of textile manufacturing.

From a general perspective, AI and ML technologies have increasingly become integral tools in the textile sector, where precision and operational efficiency are critical to maintaining high-quality standards. By leveraging these technologies, manufacturers can streamline processes, reduce human error, and increase productivity while maintaining consistent product quality. The automation of defect detection through machine learning models also offers the potential to reduce labour costs and mitigate the risks associated with manual inspection, which can be prone to oversight and fatigue.

The CNN model used in this study exhibited exceptional performance, achieving a training accuracy of 92.91% and a validation accuracy of 92.05%. This model's architecture allowed it to effectively capture spatial relationships and intricate features within the textile images, making it highly suitable for the defect detection task. In comparison, the LSTM model, which is traditionally designed for sequential data and time series tasks, was also applied to the same textile defect detection problem. While LSTM networks are typically known for their capacity to retain information across long sequences, they did not perform as well as the CNN model on this particular image-based dataset. The LSTM model achieved a training accuracy of 86.11% and a validation accuracy of 87.80%, which, while respectable, was still inferior to the CNN model's performance. This outcome highlights the inherent limitations of LSTM when applied to spatial data like images, where CNNs excel due to their convolutional layers' ability to detect and learn spatial hierarchies.

The comparison of the two models provides significant insights into their applicability in different contexts within the textile industry. CNN models, with their strong performance on image-based tasks, are better suited for tasks such as visual quality control, where detecting specific patterns, textures, or irregularities is paramount. On the other hand, LSTM models may still hold value in scenarios where sequential data is involved, such as monitoring time-based production data or tracking real-time changes in textile characteristics.

Moving from the general conclusions to the specific implications of this research, it becomes evident that the integration of AI and ML in textile manufacturing is not merely a theoretical advancement but a practical solution with tangible benefits. The CNN model's ability to detect defects with high accuracy indicates that machine learning can be directly applied to industrial production lines, leading to real-time defect detection and correction. This can significantly enhance quality control processes by identifying defects at early stages of production, reducing waste, and ensuring that only high-quality products reach the market. Furthermore, the implementation of AI-driven systems in the textile industry can support predictive maintenance, allowing manufacturers to anticipate and address equipment failures before they cause defects. This proactive approach to maintenance can further optimise production efficiency and reduce downtime, contributing to cost savings and increased profitability. In addition to quality control, the use of AI and ML in the textile sector can enable more flexible and responsive manufacturing processes. With the increasing demand for customisation and fast fashion, AI models can be used to adapt production lines to meet specific customer requirements more quickly and accurately. This capacity to deliver personalised products while maintaining high standards of quality gives manufacturers a competitive edge in the global market.

The limitations of this study should also be acknowledged. While the CNN model performed admirably in detecting textile defects, the dataset used in this research may not fully represent the diverse range of defects encountered in real-world manufacturing settings. Future work could involve training and testing the model on larger and more varied datasets to enhance its robustness and applicability across different textile types and production environments.

The integration of AI and ML technologies into the textile industry in the future will not only enhance current processes but also represent a transformative transition toward more intelligent, adaptive, and efficient manufacturing systems. Defect identification, predictive analytics, and manufacturing optimisation should all show considerably more progress as artificial intelligence develops. The use of AI in green manufacturing practices, reducing waste and improving resource efficiency, is another area ripe for exploration.

In conclusion, this study demonstrates the effectiveness of AI and ML models, particularly CNNs, in enhancing the quality control processes in textile manufacturing. By automating defect detection and improving operational efficiency, these technologies provide manufacturers with powerful tools to meet the increasing demands for high-quality, customised textile products. The findings of this research underscore the importance of continued investment in AI and ML technologies to drive innovation and maintain competitiveness in the textile industry. Future research should focus on expanding the scope of AI applications in textiles, exploring hybrid models, and addressing sustainability challenges through AI-driven solutions.

REFERENCES

[1] Halepoto, H., Gong, T., Noor, S., Memon, H., *Bibliometric analysis of artificial intelligence in textiles,* In: Materials, 2022, 15, 8, 2910

[2] Khan, H., *The application of artificial intelligence in the supply chains of textile industries,* In: Journal for Business Education and Management, 2021, 1, 2, 21–34

[3] Shahrabadi, S., Castilla, Y., Guevara, M., Magalhães, L.G., Gonzalez, D., Adão, T., *Defect detection in the textile industry using image-based machine learning methods: a brief review*, In Journal of Physics: Conference Series, IOP Publishing, 2022, 2224, 1, 012010

[4] Bravo, M.V.C., Iturralde, W.M.P., *Artificial intelligence in the management of textile companies: a contextual analysis*, In: Open Journal of Business and Management, 2024, 12, 2, 945–960.

[5] Lee, C.Y., Lin, J.Y., Chang, R.I., *Improve quality and efficiency of textile process using data-driven machine learning in industry 4.0*, In: International Journal of Technology and Engineering Studies, 2018, 4, 2, 4–10004

[6] Sikka, M.P., Sarkar, A., Garg, S., *Artificial intelligence (AI) in textile industry operational modernisation.* In: Research Journal of Textile and Apparel, 2024, 28, 1, 67–83

[7] Yildirim, P., Birant, D., Alpyildiz, T., *Data mining and machine learning in textile industry*, In: Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2018, 8, 1, e1228

[8] Chatbri, H., Jemmali, S., Hannachi, S., *Empowering the Tunisian textile industry with artificial intelligence*, Preprint, 2019

[9] Kalinouskaya, I.N., *Selection of training programs for textile industry personnel using artificial intelligence*, In: AIP Conference Proceedings, AIP Publishing, 2022, 2430, 1

[10] Arora, S., Majumdar, A., *Machine learning and soft computing applications in textile and clothing supply chain: Bibliometric and network analyses to delineate future research agenda*, In: Expert Systems with Applications, 2022, 200, 117000

[11] Fang, Y., Zou, Y., Xu, J., Chen, G., Zhou, Y., Deng, W., Chen, J., *Ambulatory cardiovascular monitoring via a machine-learning-assisted textile triboelectric sensor*, In: Advanced Materials, 2021, 33, 41, 2104178

[12] Kahraman, Y., Durmusoglu, A., *Deep learning-based fabric defect detection: A review,* In: Textile Research Journal, 2023, 93, 5–6, 1485–1503

[13] Guder, O., Isik, S., Anagun, Y., *Fabric defects identification for textile industry with a deep learning approach,* In: The Journal of the Textile Institute, 2024, 1–10

[14] Dlamini, S., Kao, C.Y., Su, S.L., Jeffrey Kuo, C.F., *Development of a real-time machine vision system for functional textile fabric defect detection using a deep YOLOv4 model*, In: Textile Research Journal, 2022, 92, 5–6, 675–690

[15] Jeyaraj, P.R., Samuel Nadar, E.R., *Computer vision for automatic detection and classification of fabric defect employing deep learning algorithm*, In: International Journal of Clothing Science and Technology, 2019, 31, 4, 510–521

[16] Soma, S., Pooja, H., *Machine learning system for textile fabric defect detection using GLCM technique*, In: Proceedings of Second International Conference on Advances in Computer Engineering and Communication Systems: ICACECS 2021, Singapore: Springer Nature Singapore, 2022, 171–182

[17] Liu, Q., Wang, C., Li, Y., Gao, M., Li, J., *A fabric defect detection method based on deep learning*, In: IEEE access, 2022, 10, 4284–4296

[18] Huang, Y., Jing, J., Wang, Z., *Fabric defect segmentation method based on deep learning*, In: IEEE Transactions on Instrumentation and Measurement, 2021, 70, 1–15

[19] Bergmann, P., Fauser, M., Sattlegger, D., Steger, C., *MVTec AD—A comprehensive real-world dataset for unsupervised anomaly detection*, In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, 9592–9600

[20] Textile Defect Detection, *Dataset*, Available at: https://www.kaggle.com/datasets/belkhirnacim/textiledefectdetection [Accessed on May 22, 2024]

Authors:

KEMAL GOKHAN NALBANT[1], BERKAN BOZKURT[2]

[1]Istanbul Beykent University, Faculty of Engineering, Architecture, Software Engineering Department,
Hadim Koruyolu Street, 34396, Sariyer, Istanbul, Türkiye

[2]Iskenderun Technical University, Postgraduate Education Institute, Computer Engineering, İskenderun Technical University (İSTE) Rectorate Central Campus, 2nd Floor, 31200, İskenderun, Hatay, Türkiye
e-mail: berkanbozkurt.lee23@iste.edu.tr

Corresponding author:

KEMAL GOKHAN NALBANT
e-mail: kemalnalbant@beykent.edu.tr