International Journal of Engineering and Technology Innovation, vol. 13, no. 4, 2023, pp. 284-295

Development of the Abnormal Tension Pattern Recognition Module for Twisted Yarn Based on Deep Learning Edge Computing

Chuan-Pin Lu[1,*],Yan-Long Huang[2], Po-Jen Lai[2]

[1]Department of Information and Communication Engineering, Chaoyang University of Technology, Taichung, Taiwan, ROC

[2]Department of Information Technology, Meiho University, Pingtung, Taiwan, ROC

Received 11 November 2022; received in revised form 13 July 2023; accepted 14 July 2023

DOI: https://doi.org/10.46604/ijeti.2023.11158

Abstract

This study aims to develop an artificial intelligence module for recognizing abnormal tension in textile weaving, The module can be used to address the time-consuming and inaccurate issues associated with traditional manual methods. Long short-term memory (LSTM) recurrent neural networks as the algorithm for identifying different types of abnormal tension are employed in this module. This study focuses on training and validating the model using five common patterns. Additionally, an approach involving the integration of plug-in modules and edge computing in deep learning is employed to achieve the research objectives without altering the original system architecture. Multiple experiments were conducted to search for the optimal model parameters. According to the experimental results, the average recognition rate for abnormal tension is 97.12%, with an average computation time of 46.2 milliseconds per sample. The results indicate that the recognition accuracy and computation time meet the practical performance requirements of the system.

Keywords: abnormal yarn tension, deep learning, edge computing, twisted yarn, long short-term memory

1. Introduction

Nowadays, many automation systems are extensively utilized in various industries to enhance product quality, curtail expenses, and boost production efficiency. The supervisory control and data acquisition system (SCADA) is one of the common automation systems used to automatically collect production data and monitor production status. However, the yarn tension monitoring systems available in today's market are still unable to automatically classify and identify abnormal tension data but can only record abnormal tension data. These records must be manually classified, identified, and marked. This is an affair that requires a significant investment of time and experience. Despite the importance of collecting abnormal yarn tension patterns for improving yarn quality, there remains a scarcity of relevant studies.

In recent years, the rapid development of artificial intelligence and the improvement of chip computing performance have significantly reduced the required computing time for deep learning algorithms. The neural network model of deep learning architecture is composed of several hidden layers [1-2], including recurrent neural networks (RNNs) [3], AlexNet [4], generative adversarial networks [5], and deep reinforcement learning [6]. These classical neural network models have been successfully applied in many fields. For instance, Hadidi et al. [7] utilized seven edge computing modules for deep neural network (DNN) performance, including Raspberry Pi 3B, Jetson TX2, Jetson Nano, Edge TPU, Movidius Neural Compute Stick, PYNQ, and HPC platforms. Chen and Huang [8] use deep learning technology to realize an automatic control program of a robot arm based on hand gesture recognition.

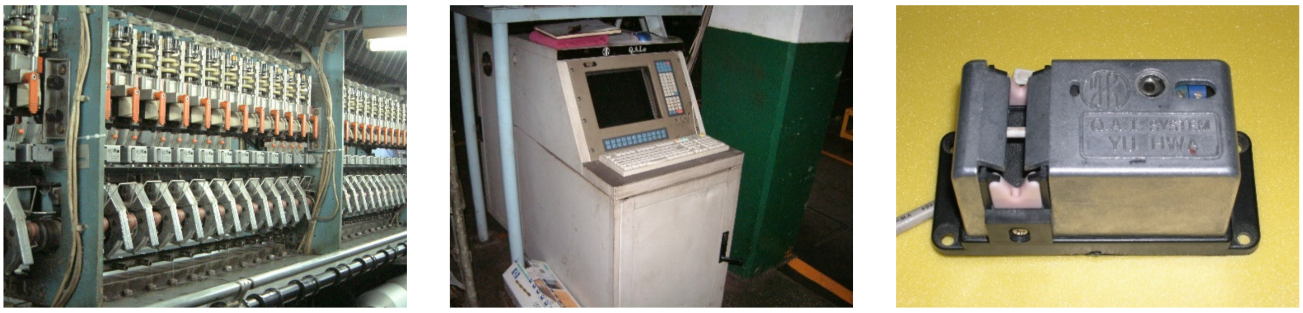
* Corresponding author. E-mail address: cplu@cyut.edu.tw

English language proofreader: Yen-Chun Hsieh

In this study, they propose the gesture recognition function to provide a natural human-machine interaction mode and control the robot arm. Ho et al. [9] also use machine vision technology with deep learning to detect rubber gasket defects, effectively detecting five characteristic defects resulting from the mold-based manufacturing process ranging from 10 to 100 μm. For crop disease detection, Bhagwat and Dandawate [10] utilized convolutional neural networks (CNN) to fuse high-level features with hand-crafted features, achieving a significant improvement compared to state-of-the-art techniques. In addition, the long short-term memory (LSTM) [11] neural network excels in handling evolving sequential data, addressing the issues of vanishing gradients and exploding gradients caused by traditional RNNs in sequence processing [12-14]. Related research on LSTM includes Michielli et al. [15] proposal of an LSTM-based associative RNN network architecture, accurately predicting and analyzing the EEG of human sleep. Jiang et al. [16] compared the prediction accuracy of RNN and LSTM on stock prices and showed that LSTM could better predict stock price fluctuations.

Regarding the automated monitoring of yarn tension, Lu and Liaw [17] proposed a method that utilizes Gaussian density distribution for maximum similarity comparison to identify abnormal tension types. In this study, the abnormal tension data is measured using tension sensors installed on the twister machine with a signal sampling frequency of 0.01 Hz, and real-time data analysis and storage are conducted by the QAI system produced by Yu-Hwa Co. Ltd. The system employs four characteristic values for classifying abnormal tension types. However, the methodology of this study does not apply to abnormal tension data under different production conditions. Cao and Ma [18] achieved high-speed yarn tension sampling by using a 400 KHz signal acquisition card and direct memory access data transmission technology, which replaced the CPU's data processing. In 2022, Khodier et al. [19] employed multispectral imaging and CNN to detect complex pattern jacquard defects. The study employed an EfficientNet CNN with a detection accuracy of up to 99%, demonstrating its potential for use in detecting defects in intricate patterns. The majority of the existing literature focuses on automating yarn detection rather than addressing the challenge of identifying abnormal tension patterns.

To address the time-consuming and inaccurate nature of traditional manual identification, as well as overcome the limitations of previous research in feature design, this study developed an artificial intelligence abnormal tension recognition module for twisted yarn. The module is built on LSTM and an edge computing embedded system, while LSTM is implemented using the Python programming language and the Keras library. Deep learning is superior to machine learning in identifying signal patterns, particularly when patterns have problems such as shift, scale, threshold, tilt, and feature design. The module utilizes the abnormal yarn tension patterns captured by the QAI system, as illustrated in Fig. 1. In the module, LSTM is implemented with the Python programming language and Keras library. In this study, five abnormal patterns of twisted yarn tension are identified. The practicality of the system was also been verified through experiments.



(a) Twisting machine (b) The workstation of the QAI system (c) Yarn tension sensor

Fig. 1 Equipment of QAI system

2. Methods

In this study, a tension pattern recognition module for twisted yarn was developed using LSTM, incorporating deep learning edge computing techniques. The device uses the Nvidia Jetson Nano as the recognition module (edge computing node) attached to the QAI system. This study identifies five important anomalous yarn tension patterns, and the system architecture is shown in Fig. 2. The QAI system and the abnormal tension pattern recognition module communicate peer-to-peer communication over a local network to store the identification results in the database.

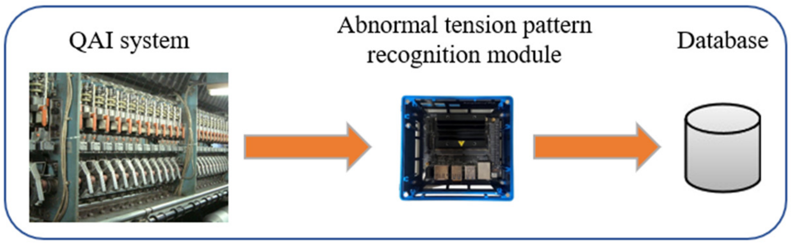


Fig. 2 Abnormal tension pattern recognition system architecture

2.1. *Edge computing by using Nvidia Jetson Nano*

Hadidi et al. [7] used frameworks including TensorFlow, TensorFlow-Lite, Keras, Caffe2, Movidius NCSDK Toolkit, PyTorch, TensorRT, and FPGA, for performing DNN experiments. These experiments aimed to assess the performance of time, power, and thermal output. Based on the experimental results of power consumption and computation time, Jetson Nano has the most stable performance in all edge computing modules with low power consumption and good performance in computation time. Based on the results of this study, the Nvidia Jetson Nano developer kit is selected as the edge operator in this study, as shown in Fig. 3. The CPU is a Quad-core ARM A57 1.43 GHz small embedded system with a 128-core Maxwell core GPU and 4 GB of 64 bits LPDDR4 RAM. Nvidia Jetson Nano's Nvidia ampere architecture further enhances the performance of the TensorRT SDK, enabling the GPU to accelerate model training, and making it a low-cost and low-power platform for developing artificial intelligence systems.

In terms of the system software, JetPack SDK supports all Jetson modules and development kits with JetPack SDK version 4.3. The operating system uses Tegra 32.3.1 Linux, which is a suite of the Linux operating system packages released by Nvidia for the Tegra processor family. Its CUDA version is 10.0.326, cuDNN version is 7.6.3, and it supports C language and Python. Python version 3.6.13 can be used with both TensorFlow and Keras, with TensorFlow version 2.1.0 and Keras version 2.3.1 for deep learning applications such as image classification, object recognition, natural language processing, time series prediction, etc.

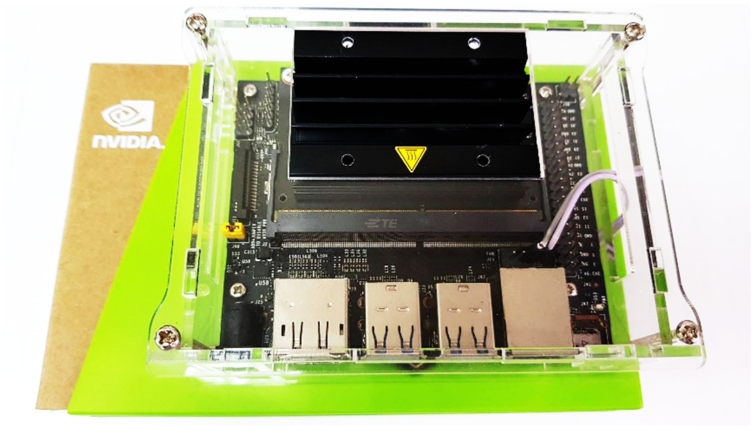


Fig. 3 Nvidia Jetson Nano developer kit

2.2. *Device communication structure*

The QAI system and the twisted yarn anomaly tension recognition module communicate over a wired network using a CAT-5E network cable. The system uses the socket API and Ethernet TCP/IP protocol, which has the advantages of strong security and short transmission time. Each QAI system is equipped with a dedicated identification module to avoid the wrong machine analysis. To maintain the load balance of the identification module and its stability, the abnormal tension data is sent to the module in a queue for processing. The data to be identified is initially stored in the QAI system's temporary storage area. After the previous data has been identified and stored in the database, a return message is sent to the QAI system, and the next data is sent to the identification module by the system. The main reason for this approach is that the module hardware specification is relatively low, while the QAI system is equipped with 64-GB memory, which can set up a larger temporary storage area to store the abnormal tension data to be processed.

2.3. Long short-term memory (*LSTM*)

The LSTM algorithm was proposed by Hochreiter and Schmidhuber [12] in 1997. Different sequence data can be used for four applications and corresponding operations can be performed for each application vector and sequence. The first application, termed "one-to-many," transforms a vector into sequences. The second application, termed "many-to-one," transforms sequences to vector (STV). The third application, termed "many-to-many," transforms sequences into sequences. Lastly, the fourth application, termed "many-to-many," transforms synced sequences to sequences (SSTS). The abnormal tension data is inputted into the neural network and classified as an application of SSTS.

LSTM is a variation of RNN, based on the RNN network structure. It introduces additional components, such as the input gate, forget gate, output gate, and memory cell. Compared to the traditional RNN, it has one more output state. Each gate filters data through the sigmoid activation function (AF) and element-wise product. Three gates control the memory cell. The input gate controls whether to input, the forget gate controls whether to clear the memory value in the memory cell, and the output gate controls whether to output the current computed value. By this method, the problems of vanishing gradient and exploding gradient caused by traditional RNN are improved. The overall model is shown in Fig. 4, where t denotes the node; x_t is the current node input; σ denotes the sigmoid AF; the h_{t-1} is the previous node output; the C_{t-1} is the value remembered in the previous node memory cell; h_t is the current node output; and C_t is the value remembered in the current memory cell.

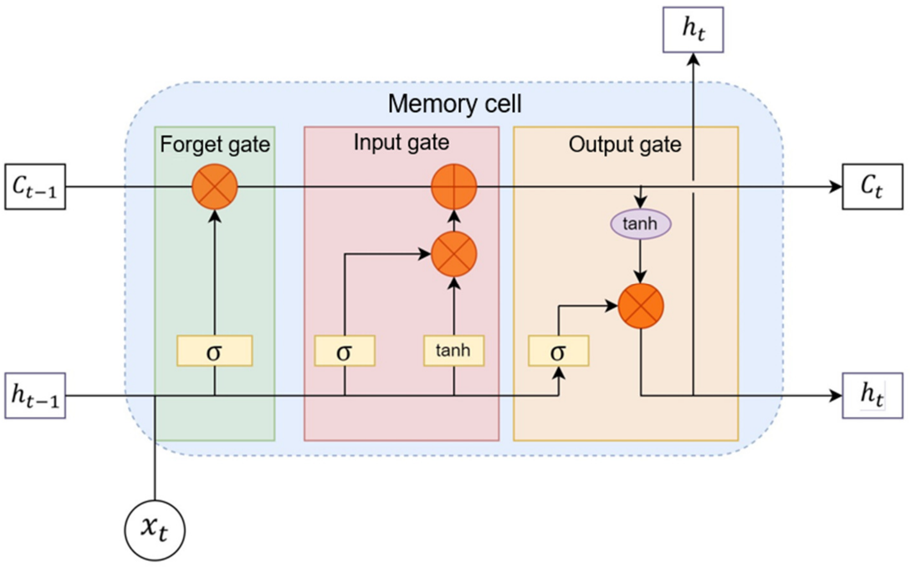


Fig. 4 LSTM network model [12]

3. Experiments

According to the instructions provided in Nvidia's official forum [20], the Jenson Nano environment should be configured with Python 3.6, Jetpack 4.3, TensorFlow 2.1.0, and Keras 2.3.1. The LSTM model architecture, as shown in Fig. 5, was constructed. It consists of an input layer, LSTM layer, flatten layer, and dense layer. The LSTM layer employs the LSTM class from the Keras API [21] to implement the LSTM model. Each signal sample contains 513 data points. The objective of the experiment is to identify samples of five abnormal tension patterns. In the model, the AF used is the hyperbolic tangent (Tanh), while the recurrent step employs the sigmoid AF. The softmax function is utilized as the AF in the dense layer of the neural network models, which predicts the type of abnormal yarn tension pattern.

The input of the LSTM layer must have three dimensions, such as "(None, 513, 1)". The first dimension represents the batch size, indicating the number of samples in a batch. The value "None" indicates that the parameter is set during sample training. The second dimension is the time step, referring to the number of time units in a sample. The third dimension is the input dimension, which represents the vector dimension per unit of time. The output shape is determined by the batch size and output dimension, such as "(None, 5)".

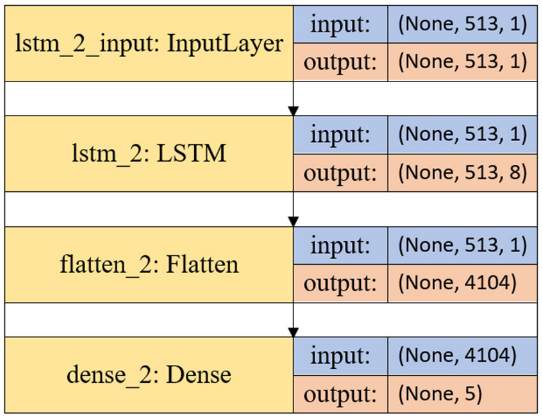


Fig. 5 LSTM model architecture

The optimal model weights for identifying abnormal yarn tension patterns were obtained through ablation studies of five cases. Four model evaluation metrics, including accuracy, precision, recall, and F1-score, were calculated based on confusion matrices. They consist of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) to assess the effectiveness of LSTM in recognizing abnormal yarn tension patterns.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

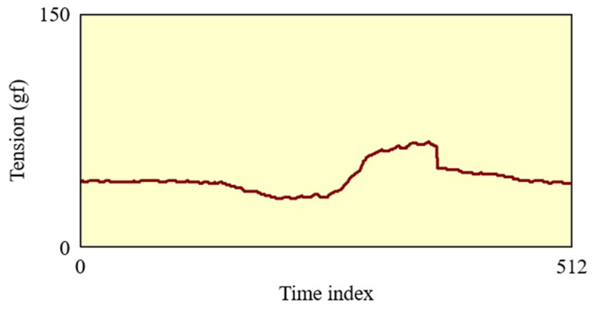
$$\text{Precision} = \frac{TP + TN}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

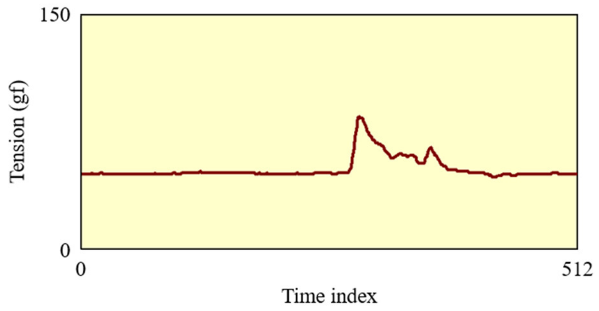
$$\text{F1-score} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{4}$$

Accuracy, as defined in Eq. (1), evaluates the overall accuracy of the model. Precision, as defined in Eq. (2), assesses the proportion of correct predictions for a single category among all predicted positive samples. Recall, as defined in Eq. (3), also evaluates a single category and represents the proportion of correct predictions for that category among all positive samples. F1-score, calculated according to Eq. (4) using precision and recall for the category, provides an approximate evaluation of the model's performance.

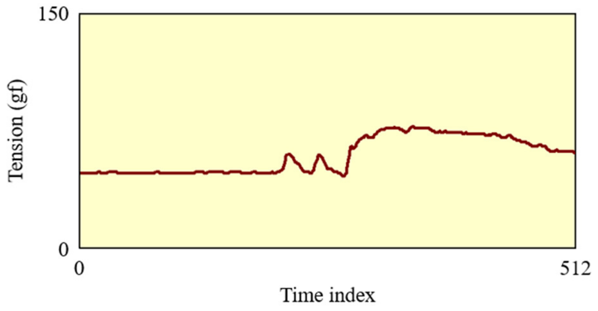
Fig. 6 illustrates the experimental anomalous yarn tension signal samples retrieved using the QAI system. These patterns were collected from twisting machines in a factory. The abnormal yarn tension patterns shown in Fig. 6 are (a) knot transfer, (b) foreign matter, (c) splice transfer, (d) tail transfer, and (e) broken yarn. Each pattern consists of 513 signals, measured in gram-force (gf), which form a series of data with fixed time and length. A total of 250 signals were used for training the model, with an 80% (200 signal patterns) training set, a 10% (25 signal patterns) validation set, and a 10% (25 signal patterns) test set.



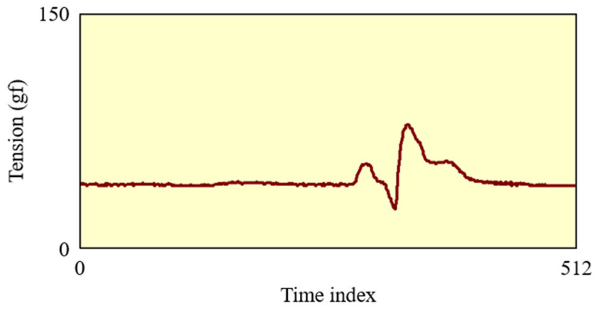
(a) Knot transfer



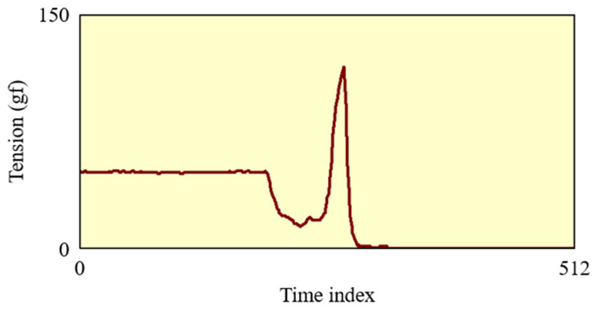
(b) Foreign matter



(c) Splice transfer



(d) Tail transfer



(e) Broken yarn

Fig. 6 Five abnormal tension patterns

3.1. Long short-term memory training experiments

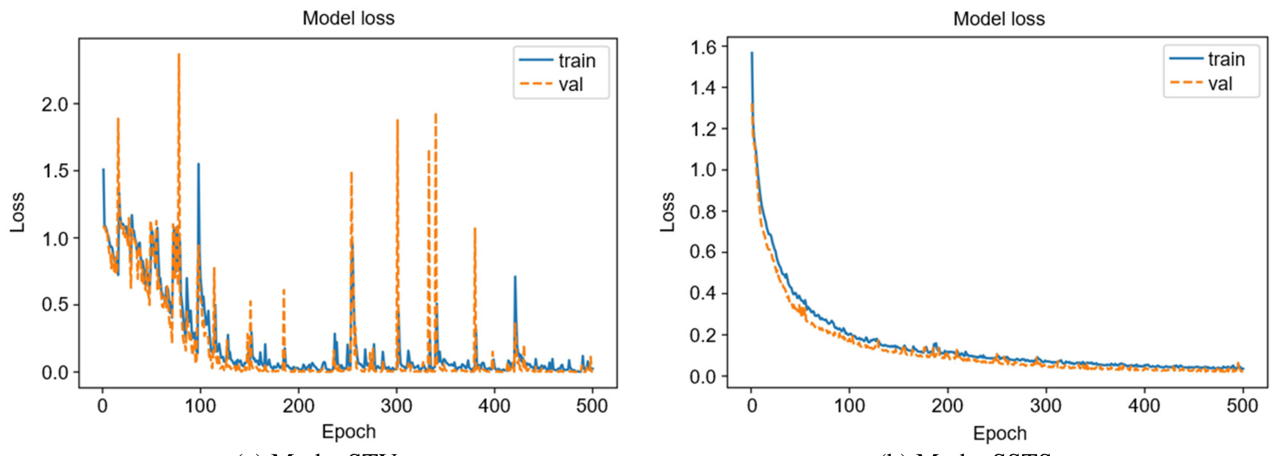
To classify the five anomalous yarn tension data patterns, the LSTM model framework was utilized to train the neural network weights using a batch size of 50, categorical cross entropy (CCE) as the loss function, and Adam [22] as the optimizer. The results of the training and validation sets for the ablation study are shown in Table 1, which summarizes all the results of tuning the input and output modes of the LSTM and the model hyperparameters epoch, units, learning rate (LR), and AF.

Table 1 Ablation study regarding LSTM

Case Study 1: Change the input and output mode of the LSTM model						
Model	Mode	Params	Training CCE	Training accuracy (%)	Validation CCE	Validation accuracy (%)
LSTM	STV	16M	0.1928	95.55	0.1854	96.00
	SSTS	20,845	0.1802	96.42	0.1456	97.26
Case Study 2: Change the number of Epoch						
Model	Epoch	Params	Training CCE	Training accuracy (%)	Validation CCE	Validation accuracy (%)
LSTM	200	20,845	0.2488	93.53	0.2068	94.84
	500	20,845	0.1802	96.42	0.1456	97.26
	800	20,845	0.1725	96.73	0.1456	97.26
Case Study 3: Change the number of Units						
Model	Units	Params	Training CCE	Training accuracy (%)	Validation CCE	Validation accuracy (%)
LSTM	1	2,582	0.7552	73.56	0.5036	77.34
	8	20,845	0.1802	96.42	0.1456	97.26
	16	42,197	0.1524	96.97	0.1455	97.28

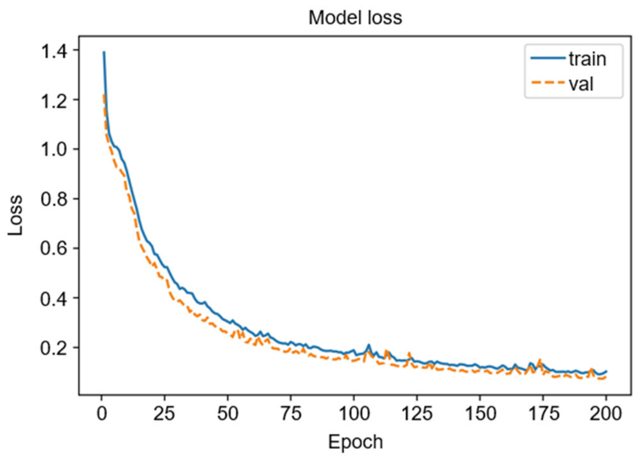
Table 1 Ablation study regarding LSTM (continued)

Case Study 4: Change the learning rate value						
Model	LR	Params	Training CCE	Training accuracy (%)	Validation CCE	Validation accuracy (%)
LSTM	0.01	20,845	0.1802	96.42	0.1456	97.26
	0.001	20,845	0.2668	92.40	0.2231	94.01
Case Study 5: Change activation function						
Model	AF	Params	Training CCE	Training accuracy (%)	Validation CCE	Validation accuracy (%)
LSTM	Tanh	20,845	0.1802	96.42	0.1456	97.26
	ReLu	20,845	0.2130	95.68	0.2113	95.23

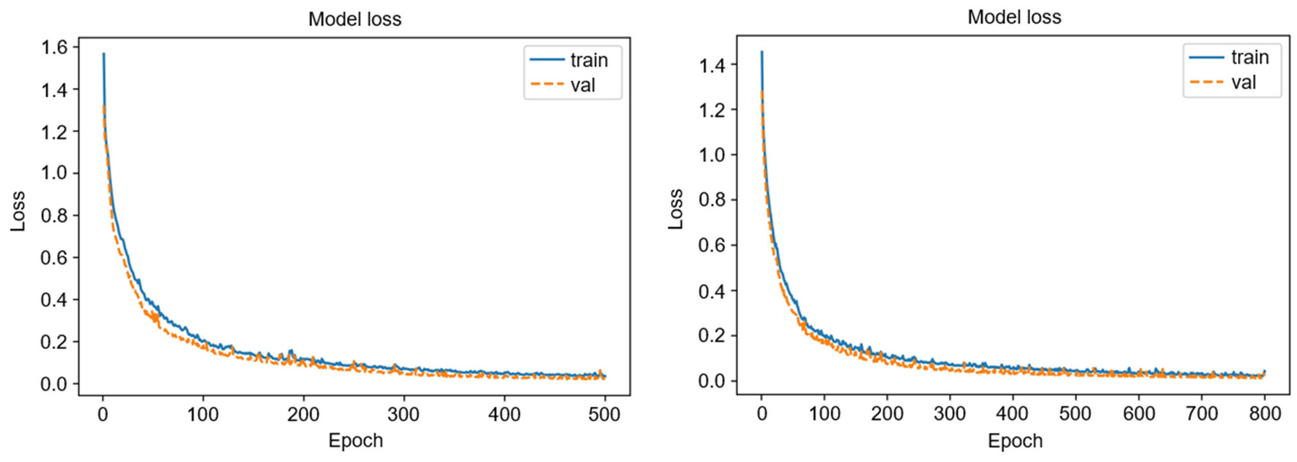


(a) Mode: STV (b) Mode: SSTS

Fig. 7 Model input and output analysis



(a) Epoch: 200

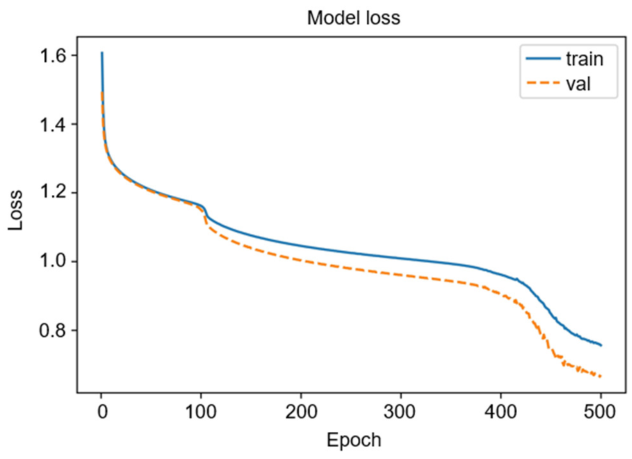


(b) Epoch: 500 (c) Epoch: 800

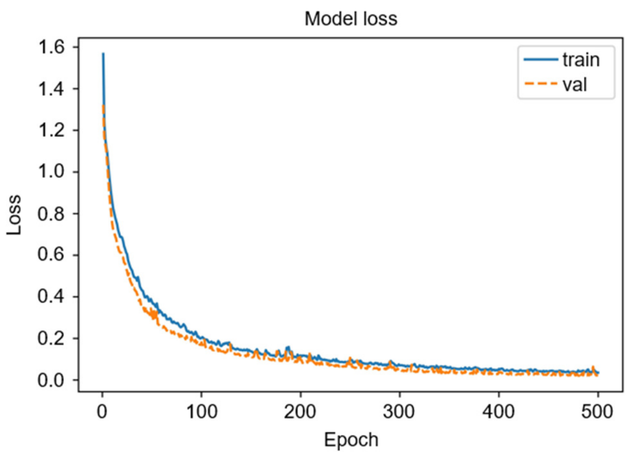
Fig. 8 Epoch performance analysis

In Table 1, the loss value of the training set is represented as training CCE, while the loss value of the validation set is denoted as validation CCE. As the tension data pattern exhibits time-dependent characteristics, experimentation was conducted to explore different input and output modes of the LSTM model. It was observed that the single signal with time series (SSTS) approach outperformed the single signal with time vector (STV) approach. The SSTS approach achieved a training accuracy of 96.42%, a training loss of 0.1802 (training CCE), a validation accuracy of 97.26%, and a validation loss of 0.1456 (validation CCE), as illustrated in Fig. 7 and presented as Case Study 1 in Table 1. Furthermore, the number of epochs was set to 200, 500, and 800 respectively to reduce the computational workload. The results show that when the epoch is set to 500 times, the Loss function shows the phenomenon of convergence stability. The validation accuracy was 97.26% for both 500 and 800 epochs, so the follow-up study was conducted with 500 epochs, as shown in Fig. 8 and in Case Study 2 in Table 1.

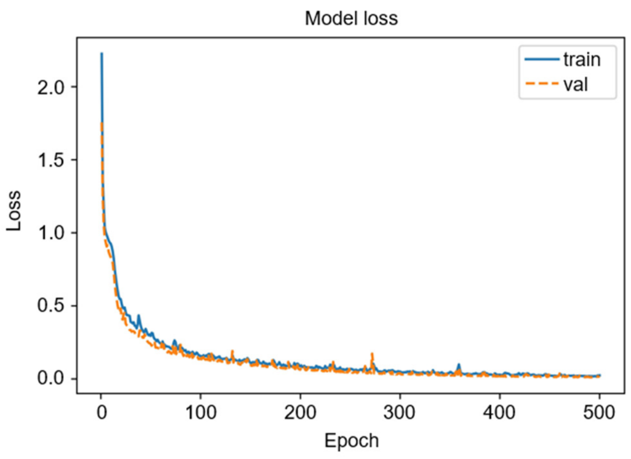
The number of units in the LSTM cell affects the feature vector extracted by the model, which represents the number of components in the system and significantly influences the total number of network parameters involved (abbreviated as params). However, if the parameters of the model increase, the relative hardware performance requirements will also increase, which may lead to overfitting. One unit, which represents the minimum number of LSTM cells in the network, is presented here to demonstrate the increase in accuracy that can be achieved by increasing the number of cells from this lowest level. Setting the unit to 16 results in the highest accuracy. However, it also leads to a significant increase in hardware load as the number of parameters rises from 20,845 to 42,197. The difference in verification accuracy between eight and sixteen units is only 0.02%, but the latter has more than twice as many parameters as the former. To alleviate the hardware load, "8 units" have been selected for subsequent ablation experiments, as illustrated in Case Study 3 in Fig. 9 and Table 1.



(a) Units: 1



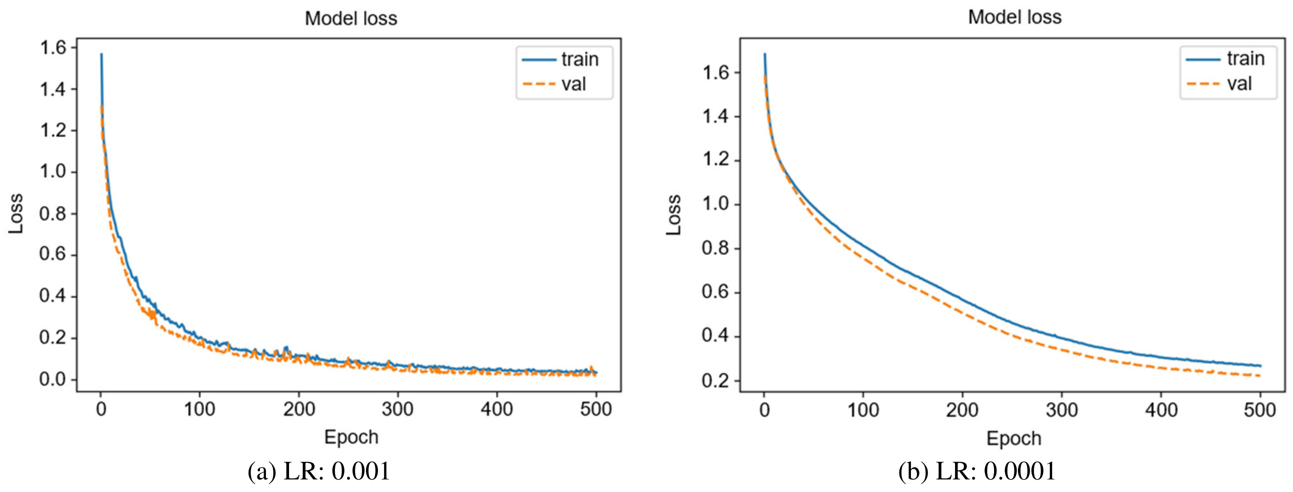
(b) Units: 8



(c) Units: 16

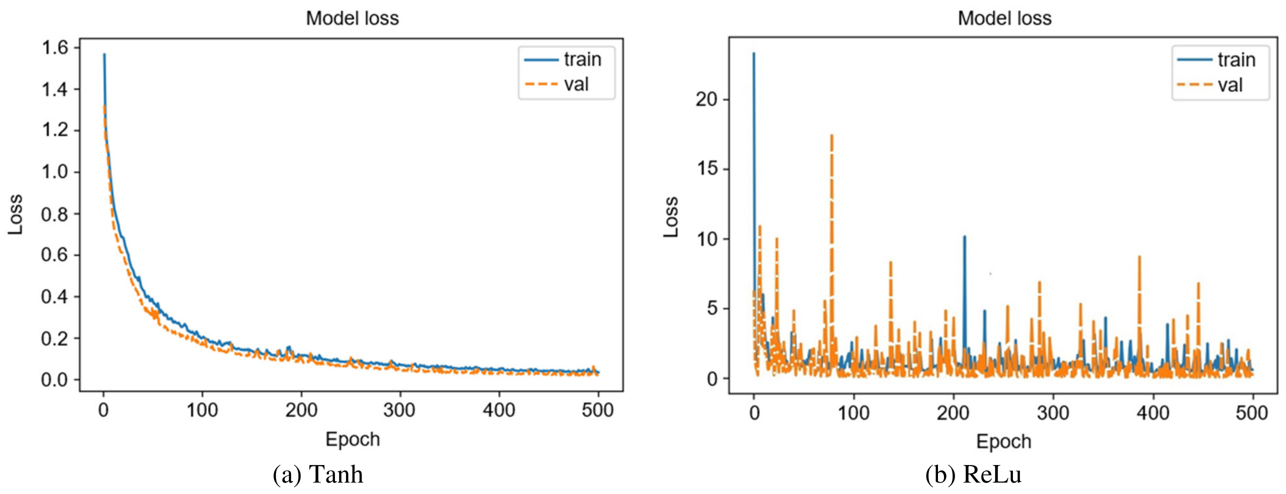
Fig. 9 Units performance analysis

Additionally, in the pursuit of finding an optimal LR, a comparison was made between LR values of 0.01 and 0.001. It was observed that when the LR was set to 0.001, the training accuracy reached 96.42%, with a training loss of 0.1802, a validation accuracy of 97.26%, and a validation loss of 0.1456. The experimental results are depicted in Fig. 10 and presented as Case Study 4 in Table 1. Since the output values of different AF regularizations affect the model performance, it is important to choose an optimal AF value. Consequently, experiments were conducted using two activation functions, namely Tanh and ReLu. The experimental results revealed that Tanh exhibited higher Training Accuracy and Validation Accuracy by 0.74% and 2.03%, respectively, compared to ReLu. These findings are depicted in Fig. 11 and presented as Case Study 5 in Table 1. The best hyperparameters of the model obtained through the above experiments are shown in Table 2.



(a) LR: 0.001 (b) LR: 0.0001

Fig. 10 Learning rate performance analysis



(a) Tanh (b) ReLu

Fig. 11 Activation function performance analysis

Table 2 Optimal hyperparameters of LSTM for abnormal tension pattern recognition

Model	LSTM
Hidden layers	1 LSTM layer
Optimizer	Adam
Activation function	Tanh
Recurrent activation function	Sigmoid
Loss function	Categorical cross-entropy
Batch size	50
Epoch	500
Units	8
Learning rate	0.001

3.2. Abnormal tension pattern recognition experiments

In this study, Nvidia Jetson Nano is used as an edge computing device to perform an anomalous yarn tension pattern recognition algorithm. The algorithm was developed and executed using Python 3.6, TensorFlow 2.1.0, and Keras 2.3.1. The abnormal yarn tension pattern recognition algorithm uses hyperparameters in Table 2 to recognize the test set. The average loss was 0.1472 and the average recognition time per signal was 43.2 ms, as shown in Fig. 12. The figure is a normalized confusion matrix. The diagonal elements of the matrix plot indicate the proportion of predicted labels that match the true labels, while the off-diagonal elements represent the proportion of misclassified samples. A greater number of correct predictions is indicated by higher values on the diagonal of the confusion matrix. By analyzing the confusion matrix, the accuracy of LSTM in identifying the 5 types of tension patterns in the experiments can be observed. However, due to the similarities in appearance between tail transfer (TT) and foreign matter (FM), LSTM has misidentified a few samples. Although it is not 100% accurate, the overall identification accuracy is sufficient to meet the system’s requirements in this application.

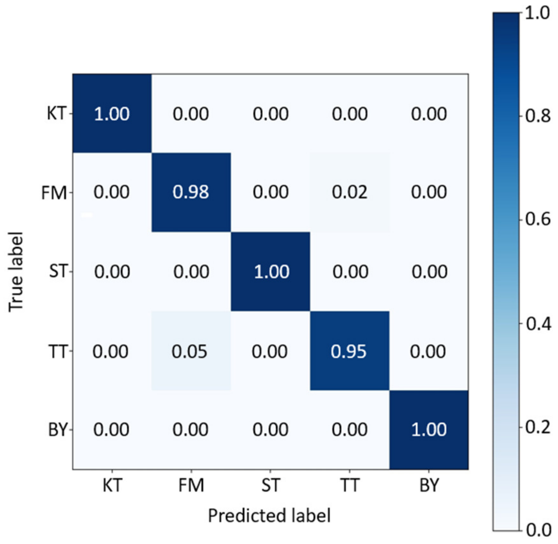


Fig. 12 Normalized confusion matrix

As to the confusion matrix, the accuracy is 97.12%, precision is 97.29%, recall is 97.12% and F1-score is 97.16%, as shown in Table 3. For further confirmation of the stability of the LSTM model, five sets of new samples were randomly sampled from the test set to test the model, and the identified results were plotted as box plots, as shown in Fig. 13. Box plot is a statistical graph used to display the dispersion of a set of accuracy. In Fig. 13, the individual accuracies of 1, 8, and 16 units are displayed. It is apparent from the figure that the accuracy of 8 units and 16 units is close, and both are significantly superior to the accuracy obtained using 1 unit. Considering that the number of parameters in the 16 units configuration is twice that of the 8 units configuration, and taking into account the necessity of implementing LSTM in an embedded system with limited memory, the decision has been made to utilize 8 units of memory cells as the core of the LSTM.

Table 3 LSTM model performance evaluation indicators

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
LSTM	97.12	97.29	97.12	97.16

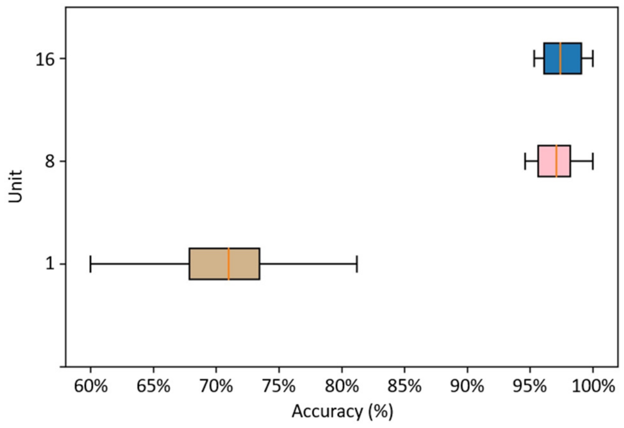


Fig. 13 Performance of LSTM models in accuracy

4. Conclusions

This study developed an artificial intelligence recognition module for identifying abnormal tension patterns in twisted yarn. The module enhances the functionality of the original yarn monitoring system in analyzing abnormal tension data. It also has the advantage of not requiring changes to the hardware architecture of the original system. Only the software of the QAI system needs to be updated to achieve a system-wide upgrade.

The module employs an LSTM neural network as the core of the pattern recognition algorithm and is implemented on an embedded system hardware architecture using Nvidia Jetson Nano for edge computing. The experimental process focuses on five common types of abnormal tension patterns. The experimental process includes searching for the optimal parameters of LSTM, training, and validation of the data. The LSTM parameters include SSTS, 8 units of memory cells, and the Tanh activation function. The LSTM achieved a sample recognition rate of 97.12% based on the results of the confusion matrix. The average computation time required for single data recognition is 43.2 milliseconds, which is faster and more accurate than manual identification methods. However, considering the limitations of the dataset and the need to enhance the robustness of the model, generative adversarial networks will be employed to generate an effective dataset.

Additionally, the potential of alternative algorithms, such as CNN, will be explored for the identification of abnormal twisted yarn tension patterns. The performance of these algorithms will be compared with LSTM to determine the optimal method for identifying abnormal yarn tension patterns. In future research, emphasis will be placed on the following areas for improvement:

(1) Broadening the scope of abnormal tension pattern types.

(2) Consistently collecting abnormal tension signals from diverse machines.

(3) Investigating and comparing the identification effectiveness of different deep learning models on abnormal tension samples.

(4) Endeavoring to evaluate the performance of deep learning models implemented on edge computing devices.

Acknowledgment

This study was supported by the National Science and Technology Council, Taiwan, ROC (grant number MOST 111-2221-E-324-022).

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep Learning”, Nature, vol. 521, no.7553, pp. 436-444, May 2015.

[2] A. Canziani, A. Paszke, and E. Culurciello, “An Analysis of Deep Neural Network Models for Practical Applications,” https://arxiv.org/abs/1605.07678, April 14, 2017.

[3] R. Zhang, F. Meng, Y. Zhou, and B. Liu, “Relation Classification via Recurrent Neural Network with Attention and Tensor Layers,” Big Data Mining and Analytics, vol. 1, no. 3, pp. 234-244, September 2018.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” Communications of the ACM, vol. 60, no. 6, pp. 84-90, June 2017.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, et al., “Generative Adversarial Networks,” Communications of the ACM, vol. 63, no. 11, pp.139-144, November 2020.

[6] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, “Deep Reinforcement Learning That Matters,” Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1, pp. 3207-3214, 2018.

[7] R. Hadidi, J. Cao, Y. Xie, B. Asgari, T. Krishna, and H. Kim, “Characterizing the Deployment of Deep Neural Networks on Commercial Edge Devices,” 2019 IEEE International Symposium on Workload Characterization, pp. 35-48, November 2019.

[8] S. L. Chen and L. W. Huang, “Using Deep Learning Technology to Realize the Automatic Control Program of Robot Arm Based on Hand Gesture Recognition,” International Journal of Engineering and Technology Innovation, vol. 11, no. 4, pp. 241-250, September 2021.

[9] C. C. Ho, E. Su, P. C. Li, M. J. Bolger, and H. N. Pan, “Machine Vision and Deep Learning Based Rubber Gasket Defect Detection,” Advances in Technology Innovation, vol. 5, no. 2, pp. 76-83, April 2020.

[10] R. Bhagwat and Y. Dandawate, “A Framework for Crop Disease Detection Using Feature Fusion Method,” International Journal of Engineering and Technology Innovation, vol. 11, no. 3, pp. 216-228, June 2021.

[11] A. Sherstinsky, “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network,” Physica D: Nonlinear Phenomena, vol. 404, article no.132306, March 2020.

[12] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735-1780, November 1997.

[13] S. Hochreiter and J. Schmidhuber, “LSTM Can Solve Hard Long Time Lag Problems,” Proceedings of the 9th International Conference on Neural Information Processing Systems, pp. 473-479, December 1996.

[14] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, “Learning Precise Timing with LSTM Recurrent Networks,” Journal of Machine Learning Research, vol. 3, pp. 115-143, January 2003.

[15] N. Michielli, U. R. Acharya, and F. Molinari, “Cascaded LSTM Recurrent Neural Network for Automated Sleep Stage Classification Using Single-Channel EEG Signals,” Computers in Biology and Medicine, vol. 106, pp. 71-81, March 2019.

[16] Q. Jiang, C. Tang, C. Chen, X. Wang, and Q. Huang, “Stock Price Forecast Based on LSTM Neural Network,” Proceedings of the Twelfth International Conference on Management Science and Engineering Management, pp. 393-408, 2019.

[17] C. P. Lu and J. J. Liaw, “Yarn Tension Pattern Retrieval System Based on Gaussian Maximum Likelihood,” International Journal of Innovative Computing, Information and Control, vol. 7, no. 11, pp. 6261-6272, November 2011.

[18] G. Cao and C. Q. Ma, “A High Speed Measuring System of Yarn Tension Based on Direct Memory Access,” 2011 Third International Conference on Measuring Technology and Mechatronics Automation, vol. 1, pp. 66-69, January 2011.

[19] M. M. Khodier, S. M. Ahmed, and M. S. Sayed, “Complex Pattern Jacquard Fabrics Defect Detection Using Convolutional Neural Networks and Multispectral Imaging,” IEEE Access, vol. 10, pp. 10653-10660, 2022.

[20] NVIDIA, “Official TensorFlow for Jetson Nano,” https://forums.developer.nvidia.com/t/official-tensorflow-for-jetson-nano/71770, March 16, 2022.

[21] Keras, “LSTM Layer,” https://keras.io/api/layers/recurrent_layers/lstm, March 20, 2022.

[22] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Proceedings of the 3rd International Conference on Learning Representations, May 2015.