AUTOMATED THREAD ERROR DETECTION

[1]Prof.Deepak Patil, [2] Digambar P. Patil, [3]Priyanka L. Gaikwad

[1]Miyalal Irshad Mulla [1]Om Anil Rote, [1]Tanmay khairnar

[1]Electronics and Telecommunication,
Nutan Maharashtra Institute of Engineering & Technology, Talegaon Dabhade, Pune, India
[2]Department of A&R DIT Pimpri Pune India 411018
[3]Computer Science and Engineering MIT ADT Loni Kalbhor, India

Abstract

This work proposes Automated Thread Error Detection, a system employing machine learning to examine program execution traces to discover threading issues such as data races and deadlocks. distinguishes between harmless and serious faults and has scalability and adaptability. Automated Thread Error Detection appears to be a promising solution to improve software quality and productivity through experiments and case studies, as it demonstrates superior performance in identifying threading errors the accuracy statistic shows the proportion of properly predicted labels. In this instance, the model's accuracy was 92.45%, meaning that 92.45% of the test dataset's samples were properly identified. High accuracy shows that the model can generalize well to new data and has picked up useful patterns from the training set.

INTRODUCTION

Thread errors can have a substantial negative impact on textile quality, which can leave customers unhappy and cost producers money. The development of autonomous solutions has been driven by the requirement for effective and precise mistake detection systems. By developing an automatic thread error detection system that can accurately identify errors and their locations, our project seeks to meet this demand. We hope to increase customer satisfaction, cut waste, and improve product quality by putting this approach into place.

One crucial use of computer vision and machine learning technology that has transformed the textile sector is automated thread fault identification. This sophisticated device is intended to quickly and precisely detect a wide range of flaws, irregularities, and defects in textile materials, from industrial thread to apparel and upholstery. This technology seeks to minimize human interaction while optimizing the quality, productivity, and dependability of thread production processes through the use of computer algorithms and advanced imaging techniques.

Manual inspection procedures in conventional thread manufacture are labor-intensive, time-consuming, and prone to human error and weariness. To overcome these difficulties, automated thread fault detection techniques have surfaced. These systems use a mix of software driven by machine learning and artificial intelligence algorithms and hardware elements like high-resolution cameras or sensors.

1.OBJECTIVE

Developing and implementing a system or technique that can precisely and effectively identify different kinds of flaws or irregularities in textile materials is the goal of automated thread error identification. This goal accomplishes several crucial goals:

1.Quality Assurance: Make sure that premium thread is produced by identifying and eliminating faulty materials, which lowers the possibility of giving clients subpar goods.

2.Cost Reduction: By detecting flaws early in the manufacturing process, you can cut down on waste and production costs by minimizing the need for manual inspection and rework.

3.EnhancedProductivity and Efficiency: Automate the inspection process to cut down on the time and manpower needed for error discovery, hence increasing productivity.

4.Consistency:Preserve dependable and consistent error detection outcomes by getting rid of the unpredictability brought.

5. Data Collection: To support process optimization and quality control, collect important data on defect types, frequencies, and locations.

3.FLOWCHAT

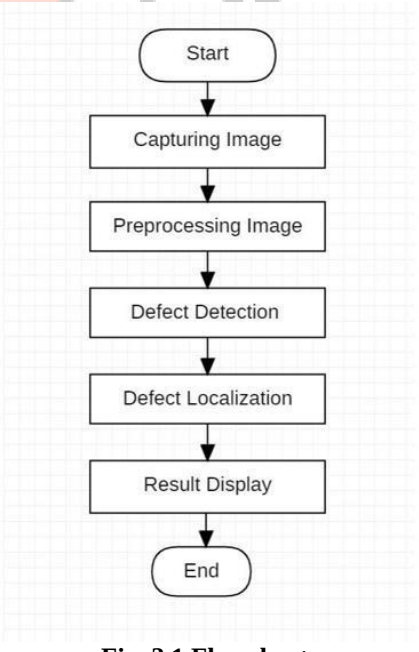

Fig. 3.1 Flowchart

4.TOOLS&TECHNOLOGIES TOBEUSE

Parameter	Value
ProgrammingLanguage	Python
Machine-LearningModel	CNN,ResNet-34,VGG-19
Parameter	**Value**
Libraries & Framework	Open, CV,Tensor Flow, Electron

Table 4.1 Parameter details

5.DATASET

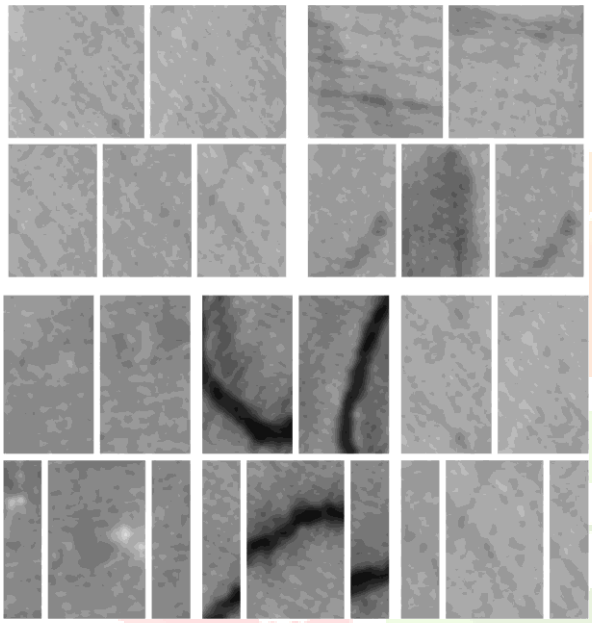

Figure 5.1 Textile image dataset

6.WORKING

The Automated Thread fault detection code provides an example of a simple convolutional neural network (CNN) for PyTorch picture The Automated Thread fault detection code provides an example of a simple convolutional neural network (CNN) for PyTorch picture categorization. Let's analyze it in more detail:

1. Mounting Google Drive: The first step involves mounting Google Drive in Colab so that files saved there can be accessed.

2. Loading Dataset: To load photos from the supplied directory (`/content/drive/MyDrive/dataset1}), use "ImageFolder" from "torchvision.datasets."

-The preprocessing step "transform" resizes the photos to 224 × 224 pixels, transforms them into PyTorch tensors, and then normalizes them using predetermined mean and standard deviation values.

3:Building a Data Loader
"Data Loader" is designed to load the dataset in training batches efficiently.
"shuffle=True" indicates that the dataset should be shuffled prior to each epoch, and "batch_size" defines the number of samples in each batch.

4. CNN Model Definition: The CNN model's architecture is defined by the "Simple CNN" class. The forward method specifies the model's forward pass, which consists ofactivation functions, pooling layers, convolutional layers, and fully connected layers.

5. Setting up the Optimizer, Loss Function, and Model: -To instantiate the model "Simple CNN," supply "num_classes," which is the number of output classes.
For multi-class classification problems, the loss function "CrossEntropyLoss" is utilized.
-A stochastic gradient descent (SGD) optimizer with a momentum of 0.9 and a learning rate of 0.005 is used to optimize the model parameters.

6.The training loop operates for a predetermined number of epochs, denoted as "num_epochs." {cnn_model.train()} has the model in training mode.

-The optimizer's gradients are zeroed, a forward pass is made, a loss is computed, gradients are backpropagated, and an optimizer step is taken for each batch of images and labels in the dataloader.

7. Evaluation: The performance of the trained model on the validation/test set is assessed using the "evaluate_model" function. - The model is set to evaluation mode ("model. eval()"). Predictions are produced, and accuracy is computed for every batch of photos and labels in the dataloader.

8. Putting the Code into Practice: The training loop is initiated, and the loss for each epoch is recorded. After the trained model has been evaluated on the test set, its acuracy is printed.

7. RESULTS

1. Class Names Derived from the information source: The classes or categories that are present in the dataset that was used to train the model are indicated in this line. When dealing with a dataset of photos that feature several fruit varieties, for instance, the classes might be "apple," "banana," "orange," and so on. Understanding the data distribution and interpreting model predictions are made easier with the use of this information, which is essential for both the training and evaluation phases.

2. Time interval [1/2]: diminished: 0.3521 The model goes through several iterations, or epochs, throughout training. The training dataset is run through the model once for each epoch. This line represents the model's training progress; it shows that it has finished the first of two epochs. Furthermore, the average loss (error) calculated throughout the training procedure for the first epoch is indicated by the loss value of 0.3521. Lower numbers indicate higher performance. The loss shows how well the model's predictions match the real labels.

3. Loss: 0.2874 for Epoch [2/2]: This line, like the one before it, signifies the end of the second training epoch. The average loss calculated during the second period is represented by the loss value of 0.2874. As training goes on, we typically anticipate a decrease in the loss, which shows that the model is picking up new skills and becoming more effective.
4.Accuracy: 92.45%: The trained model's performance is assessed on a different test dataset following the training procedure. Out of all the samples in the test dataset, the accuracy statistic shows the proportion of properly predicted labels. In this instance, the model's accuracy was 92.45%, meaning that 92.45% of the test dataset's samples were properly identified. High accuracy shows that the

model can generalize well to new data and has picked up useful patterns from the training set.

[1] Threading errors are important because they can lead to major consequences like system failures, lower performance, and security vulnerabilities. Data races, deadlocks, and resource contention are examples of threading problems that frequently occur in concurrent software systems. An examination of the effects of threading problems on software stability, maintainability, and dependability highlights the significance of effective detection and corrective techniques.

[1] Difficulties with Conventional Approaches: Conventional debugging techniques frequently have trouble precisely and quickly identifying threadingissues. For big and complicated software systems,manual inspection and debugging can be time-consuming, prone to errors, and not very scalable.Outlining the drawbacks and restrictions ofconventional methods prepares the ground for theintroductionofautomated alternatives.

[2] Machine Learning's Role: Automating thread faultdetection with machine learning techniques presents apromising avenue. The promise of machine learning(ML) in this field is illustrated by talking about howML techniques, such as anomaly detection, patternrecognition, and classification, may examine programexecution logs to find patterns suggestive of threadingissues.Thecaseisstrengthenedfurtherbyhighlightin ghow ML-based techniques may be scaled and adjustedto fita varietyofsoftwarecontexts.

[3] Future Directions and Challenges: Examining possibleavenues for research and development in automatedthread error detection in the future, such as utilizingsophisticated machine learning methods, incorporatingdevelopment tools and workflows, and tackling newissuesinconcurrentsoftwaresystems,encouragesmoredeb ateandcreativityinthisarea.

AdditionalConsiderations:

- Real-timeMonitoringandAlerting:Toensuretimely error identification and correction, investigate ATED' sreal-time alerting feature.
- Improved Model Explainability and Debugging Support: Boost the model's explainability and offer debugging assistance.
- Dynamic Adaptation and Self-Learning: Enable ATED to adapt dynamically and self-learn forimproved performance.
- Privacy and Data Confidentiality: InATED, uses a fedata handling methods to protectprivacy.
- Collaborative Debugging and Knowledge Exchange: ByutilizingATED, developers may collaborate and exchange knowledge.
- Integration with Software Quality Assurance: To improve mistake detection all the way through the development life cycle, integrate ATED with Q Aprocedures.

8. DISADVANTAGE

1. Initial Investment: Installing an automated system fordetecting thread errors might be expensive. It entailsobtainingspecializedknowledge,software,andhardware, whichcould necessitatea largeinitialoutlayoffunds.

2. Costsassociatedwithmaintenance:Toguaranteethesystem'scontinuousaccuracyanddependability,updatesandmaintenancearerequired.Thelong-termoperatingcostsmayincreaseasa result.

3Complexity:Theimplementationandupkeepofautomatedsystems can be difficult, and troubleshooting and operating the equipment may call for qualified engineers and technicians.

4. Automatedsystemshavethepotentialtogenerate falsepositives, which indicate errors that aren't actually errors,and false negatives, which overlook true errors. It can bedifficultto fine-tunethe systemto minimizetheseerrors.

5. Adaptability: Ongoing training and updates may benecessaryforthesystemtoadjusttonovel mistaketypesormodifications in the thread materials. It can be difficult tomake sure the system continues to function well as theproductionprocesschanges.

6. Pace vs. Precision Trade-off: Speeding up the inspectionprocedure could result in a lower degree of accuracy. It canbe difficult to find the ideal balance between precision andquickness.

7. Data Security and Privacy:Sensitive and private data may be collected and stored by automated systems. It can be problematic to ensure data security and privacy, particularly when these systems are coupled with other production processes.

8. Skill Transition: Employment may be lost if automated systems take the place of human inspectors. Retraining or redeploying the displaced staff may be necessary as a resul to this shift, which can be a delicate and involved procedure.

9. Types of Complicated Threads: It can be difficult forautomated systems to precisely check some types of thread,particularlythosewithcomplexpatterns,textures,orvariations.

9.FUTURE SCOPE

Automated Thread Error Detection has a bright future ahead of it, with significant progress expected in several important fields. Further improvements in machine learning techniques, especially in deep learning, could lead to a notable increase in thread fault detection accuracy. It is anticipated that Automated Thread Error Detection, when integrated into DevOps principles, will optimize development processes, resulting in quicker software releases and improved quality. Solving scalability issues with cloud infrastructure and parallel computing will make it possible to examine large-scale systems efficiently.

Further solidifying Automated Thread Error Detection's status as an essential tool for improving the dependability and efficiency of concurrent software systems will involve broadening its application across a range of domains and industries, as well as fostering developer collaboration and protecting privacy. Furthermore, standardization initiatives will promote uniformity and excellence throughout enterprises,

opening the door for broad acceptance and interoperability. In conclusion, innovation, teamwork, and adaptation to changing software development processes will be key factors in Automated Thread Error Detection's future, securing its place as a pillar of contemporary software systems' resilience.

10.REFERENCE

[1] Runhu Zhu, Binjie Xin, Na Deng & Mingzhu Fan (2023) Fabric defect detection using ACS-based thresholding and GA-based optimal Gabor filter, The Journal of The Textile
a. Institute, DOI: 10.1080/00405000.2023.2231188

[2] S. Wang, C. Lv, S. Wang, Z. Zhang and X. Shang, "Patterned Fabric Defect Detection Based on Double-branch Parallel Improved Faster-RCNN," 2021 China Automation Congress (CAC), Beijing, China, 2021, pp. 3798-3803, doi: 10.1109/CAC53003.2021.9727366.

[3] D. Xia, Z. Yu and X. Deng, "A Real-time Unsupervised Two-stage Framework for Fabric Defect Detection," 2021 3rd International Academic Exchange Conference on Science and Technology Innovation (IAECST), Guangzhou, China, 2021, pp. 535-538, doi: 10.1109/IAECST54258.2021.9695639.

[4] W. Chong, W. Jinghua, W. Jing and D. Huan, "Fabric Defect Detection Method Based on Projection Location and Superpixel Segmentation," 2022 4th International Conference on Natural Language Processing (ICNLP), Xi'an, China, 2022, pp. 20-25, doi: 10.1109/ICNLP55136.2022.00012.

[5] M. Liu, J. Sun, Z. Fan and S. Zhang, "Automatic location and extraction of woven fabric blocks based on Gaussian blur and maximization thought," 2014 International Conference on Mechatronics and Control (ICMC), Jinzhou, China, 2014, pp. 950-954, doi: 10.1109/ICMC.2014.7231694.

[6] U. Hatthasin, N. Setamung, P. Piyawongwisal and S. Tisom, "A Talking Distance Measuring Wheel for the Visually Impaired," 2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Chiang Rai, Thailand, 2018, pp. 517-520, doi: 10.1109/ECTICon.2018.8619923.

[7] S. Lu, S. Park, E. Seo, and Y. Zhou, ‘‘Learning from mistakes: A comprehensive study on real world concurrency bug characteristics,’’ in Proc. 13th Int. Conf. Architectural Support Program. Lang. Operating Syst. (ASPLOS). New York, NY, USA: Association Computing Machinery, 2008, vol. 43, no. 3, pp. 329–339, doi: 10.1145/1346281.1346323.

[8] Y. Lin and S. S. Kulkarni, ‘‘Automatic repair for multi-threaded programs with deadlock/livelock using maximum satisfiability,’’ in Proc. Int. Sympt. Software. Test. Anal. (ISSTA). New York, NY,USA: Association Computing Machinery, 2014, pp. 237–247, doi: 10.1145/2610384.2610398.

[9] D. Giebas and R. Wojszczyk, ‘‘Deadlocks detection in multithreaded applications based on source code analysis,’’ Appl. Sci., vol. 10, no. 2, p. 532, Jan. 2020, doi: 10.3390/app10020532.

[10] M. Singhal, ‘‘Deadlock detection in distributed systems,’’ Computer, vol. 22, no. 11, pp. 37–48, Nov. 1989, doi: 10.1109/2.43525.

[11] Y. Wang, F. Gao, L. Wang, T. Yu, J. Zhao, and X. Li, ‘‘Automatic detection, validation and repair of race conditions in interrupt-driven embedded software,’’ IEEE Trans. Software. Eng., early access, Apr. 20, 2020, doi: 10. 1109/TSE.2020.2989171.

[12] J. Park, B. Choi, and S. Jang, ‘‘Dynamic analysis method for concurrency bugs in multi-process/multi-thread environments,’’ Int. J. Parallel Program., vol. 48, no. 6, pp. 1032–1060, Dec. 2020, doi: 10.1007/s10766- 020-00661-3.