Time Series Anomaly Detection using Convolutional Neural Networks in the Manufacturing Process of RAN

Cristina Landin*, Jie Liu†‡, Katerina Katsarou‡, Sahar Tahvili†§
*School of Science and Technology, Örebro University, Örebro, Sweden
cristina.landin@oru.se
†Product Development Unit, Cloud RAN Development Support, Ericsson AB, Stockholm, Sweden
{anna.a.liu, sahar.tahvili}@ericsson.com
§ Innovation and Product Realisation, Mälardalens University, Eskilstuna, Sweden
‡Technical University of Berlin, Germany
a.katsarou@tu-berlin.de

***Abstract*—The traditional approach of categorizing test results as "Pass" or "Fail" based on fixed thresholds can be labor-intensive and lead to dropping test data. This paper presents a framework to enhance the semi-automated software testing process by detecting deviations in executed data and alerting when anomalous inputs fall outside data-driven thresholds. In detail, the proposed solution utilizes classification with convolutional neural networks and prediction modeling using linear regression, Ridge regression, Lasso regression, and XGBoost. The study also explores transfer learning in a highly correlated use case. Empirical evaluation at a leading Telecom company validates the effectiveness of the approach, showcasing its potential to improve testing efficiency and accuracy. Despite its significance, limitations include the need for further research in different domains and industries to generalize the findings, as well as the potential biases introduced by the selected machine learning models. Overall, this study contributes to the field of semi-automated software testing and highlights the benefits of leveraging data-driven thresholds and machine learning techniques for enhanced software quality assurance processes.**

***Keywords*–Software Testing, Test Optimization, Machine Learning, Imbalanced Learning, Moving Block Bootstrap**

I. Introduction

The concept behind anomaly detection in software testing is examined, classified, and improved based on the belief that software testing can be automated to a greater extent [1]. Real-world applications and testing of these applications have been emphasized throughout this study. In fact, wireless device companies are facing fierce competition in the market, where cost efficiency and innovative solutions play a critical role in their success. To remain competitive, these companies need to continually design and manufacture products that meet the ever-increasing demand for high-quality wireless connectivity. Additionally, telecom operators globally are striving to optimize their network performance to enhance testing efficiency and capacity. Despite significant research efforts and industry advancements in recent decades, achieving this common goal has proven challenging [2], [3]. In addition, the radio access network (RAN) testing, data collection, and debugging analysis, demand engineers to incorporate automatic management algorithms to relieve data and help the Mobile Networks operate. Typically, a software engineer (SE) will go through several steps to ensure that the software complies with specifications before releasing it to the clients [2]. However, these processes demand engineers to incorporate automatic management algorithms to relieve data and reduce manual work [4]. Unfortunately, the interpretation of requirement specifications by SEs can be either too strict or too open, leading to inefficient testing sequences and wasting important resources at the production stage. As software complexity increases, software testing becomes more challenging for the SEs. Semi-automatic and automatic software testing is becoming increasingly necessary to address this issue. Although some semi-automated tools have improved software testing methodologies, many software tests are still executed manually or use rule-based methods, which are ineffective, expensive, and error-prone [5]. Besides, several issues remain unanswered, including when to stop testing software is economically feasible. Therefore, there is a pressing need to develop more efficient and effective software testing solutions.

To overcome these challenges, companies need to optimize their test processes and find possible anomalies early in the development cycle. This can help to reduce costs, improve the efficiency of the testing process, and ensure that the product meets the quality standards before release. Moreover, with the advent of new technologies such as machine learning (ML) and deep learning (DL), it is now possible to automate some parts of the testing process and improve the effectiveness of the testing. By incorporating these new techniques into the testing process companies can stay ahead of their competitors and deliver high-quality products to their customers.

This study aims to improve the product test cycle by optimizing the software test execution and finding possible anomalies ahead of time. It introduces a new framework to detect the divergent points which have potential risks of failing on the final test results based on data-driven thresholds. Our

2835-3560/23/$31.00 ©2023 IEEE
DOI 10.1109/AITest58265.2023.00023

model predicts future outcomes using different AI techniques and labels the inputs, where the test process raises the alarm if a new point crosses these thresholds. Combining and enhancing testing techniques using AI can improve effectiveness while investigating how software testing results can be automated increases efficiency and optimize production resources.

The manufacturing testing process of the latest products often involves collecting data that is time-series in nature, making it challenging to analyze and identify potential anomalies. This paper proposes a solution that models the time-series data using DL techniques and applies data-driven thresholds to predict future outcomes and identify potential risks ahead of time. Furthermore, as it is well known, in industrial applications, data sets are often imbalanced due to the nature of the manufacturing process, which leads to biased predictions and inaccurate results. To address this matter, the paper discusses the use of data augmentation techniques to balance the data and improve the accuracy of the model. By incorporating DL solutions, data augmentation techniques, and dynamic thresholds, the proposed framework aims to improve the efficiency and effectiveness of the manufacturing testing process and reduce the risk of product failures.

Nevertheless, our proposed framework is not only applicable to our company but can also be widely employed for similar cases of anomaly detection for one-dimensional data with self-correlations in time-series data, and for detecting potential risks at an early stage. This study also aims to contribute to the broader industry knowledge of DL and transfer learning and to encourage the adoption of ML methods to address real production problems in the telecommunication domain.

II. Background

A. Test Optimization on the production process

Base stations (BSs) make possible mobile communication. Since a BS radiates and receives electromagnetic waves, it needs to comply with safety limits. Therefore, Telecom companies adhere to national and international regulators which influence directly the definition of the product's test requirements and dictate what is expected from the product. The test requirements contain a set of test cases, which in turn define what and how to test and the expected results out of fixed thresholds. A test suite is a collection of test cases and describes how these are being executed. The test suite can be executed sequentially or in parallel, based on several conditions such as test case dependencies or test infrastructure limitations. If a test suite is executed sequentially, the total test time is assumed to be the sum of all test cases. Furthermore, if a failure occurs, (i.e., results fall beyond the fixed thresholds), the test case should be either re-tested or the total test sequence needs to be stopped for this unit and continues for another unit. However, test outcomes based on fixed thresholds do not usually show complete information on how the test results are distributed within the fixed limits or how they are evolving through time as long they fall within their specifications. Test optimization on the test suite execution is desired due to the cost of production steps. Therefore, early prediction of manufacturing outcomes can help the testing team to avoid the waste of unnecessary resources while still keeping the quality of the product. The above-mentioned issue becomes a challenge to real applications in the industry due to the multidimensional input data and the different parameters involved in the production process of different domains as described in the following subsections.

B. Time-series modeling

Time-series data is a sequence of observations of the same source that are measured consecutively over time and can be linear or nonlinear. In either case, historical data can be modeled using regression models [6]. In the context of testing, the execution of a unit in a software system is typically performed as a sequence of test cases, and the output of each test case is recorded over time. In our testing process, for each test case, there are three pieces of data: *the test case number*, *the test unit*, and *the time stamp*. When a new unit is introduced to the testing infrastructure, it undergoes a series of test cases, and its final outcome, i.e., the sum of the test case outputs, is saved for future analysis. To simplify the time series modeling, we can consider a single test case executed on different units over time within the same infrastructure, in order to avoid external influences on the data record [7].

C. Anomaly detection in time-series

Anomaly detection in time-series data has been extensively studied using traditional statistical methods such as autoregressive integrated moving averages (ARIMA), exponentially weighted moving averages (EWMA), and cumulative sum (CUSUM). However, with recent advances in DL, there has been growing interest in using deep learning-based methods for anomaly detection in time-series data, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs). CNNs are particularly well-suited for processing time series data with local patterns, as they can learn local features through convolutional layers and capture long-term dependencies through pooling layers. CNN-based anomaly detection methods have been applied to various types of time-series data, including sensor data, financial data, and medical data. In the context of production testing, CNN-based anomaly detection methods have been proposed to detect anomalies in production test data, which might help improve product quality and reduce production costs [8].

D. Time series data augmentation

It has been shown that learning systems exhibit lower classification performance on imbalanced datasets due to the limited amount of data in the minority class [9]. Experiments demonstrate that over-sampling methods produce a better classification performance than under-sampling, and for datasets with a larger amount of positive samples, random over-sampling can produce significant results [2], [10].

In this paper, each test case is considered as time-series data and is classified using regression models with different alarm levels depending on dynamic thresholds. Due to the limited number of negative samples (which negatively affects

classification performance in most real applications), data augmentation (DA) techniques are necessary. In this paper, we employ the Moving Block Bootstrap (MBB) as a DA technique suitable for dependent observations such as time-series data. MBB creates new synthetic data by resampling and replacing the original time-series data using defined blocks [11]. For instance, for a data set of *n* samples, the MBB split the data into $N = n - l + 1$ overlapping blocks of length *l*. The optimal block length can be found using Lahiri's method at [12] chapter 7. MBB allows the blocks to overlap, which is useful for small samples. The interested reader can find more about block bootstrap methods for time series in [13].

E. Classification using deep learning

Classification is the task of assigning input data to one of several predefined categories or classes. Recently, deep learning approaches have revolutionized the field of classification by achieving state-of-the-art performance on various benchmark datasets [14]. Deep learning models for classification typically consist of multiple layers of interconnected artificial neurons, which learn hierarchical representations of the input data. These models are trained using large amounts of labeled data and an optimization algorithm, such as stochastic gradient descent, to minimize the difference between the predicted outputs and the ground truth labels.

F. System fault tolerance in machine learning systems

Over the years, fault tolerance has been studied as part of the system's dependability and the capacity to recover from errors [15]. As machine learning systems are non-deterministic statistical approaches, fault prevention (model accuracy) is not enough to avoid failures in the systems, also fault tolerance is needed. In this context, fault tolerance occurs when the errors are detected and the resulting errors will not propagate into the systems [16]. Myllyaho et. al [17] study how fault tolerance is used in practice regarding the dependability of machine learning systems and they conclude that the term is still young in the software engineering field. Using mean absolute error (MAE) as a measure of fault tolerance implies assessing the system's resilience to errors or deviations in predictions or estimations. By measuring the average magnitude of errors between predicted and actual values, MAE can provide insights into the system's ability to maintain accuracy and robustness even in the presence of uncertainties or faults. This perspective aligns with the broader concept of fault tolerance, which aims to ensure that a system continues to function effectively despite the occurrence of faults or failures. By evaluating the accuracy of predictions or estimations through MAE, one can indirectly assess the system's capability to tolerate and mitigate potential faults, making it a relevant metric for evaluating fault tolerance.

III. Related work

In the past years, there has been increasing interest in the application of machine learning algorithms in the industry, particularly deep learning techniques, to the detection of anomalies especially in time series data. In manufacturing, anomaly detection can be used to identify defects or deviations from expected production processes, which can lead to improved quality control and reduced costs, including testing and optimization of complex systems. Among other solutions, anomaly detection can be performed by regression [18]. This is accomplished by first predicting the relevant variable using regression, and then comparing the predicted value to the actual value to determine if it is anomalous or not. For instance, Weiss et al. in [19] and [20] explained the advantages of early predictions of final outcomes for the manufacture of microprocessors. This will allow to apply correction actions early in the production stage with the aim of a specific task, e.g., microprocessor speed. The authors use regression to model a stationary population of chips. However, the process is not stationary which brings forward the challenges this approach has and the need for specialized algorithms such as deep learning. In a previous work [21], the authors developed a prototype monitoring system to predict yield drop based on the early prediction of manufacturing outcomes. They use support vector machines (SVM) to classify and label the new inputs, thus, if the new input will most probably produce a yield drop, then actions must be taken. This approach makes the user part of the decision-making process. Nevertheless, they do not study the feature input dependencies as such, because they use SMOTE for DA purposes. SMOTE is not applicable to all data types, such as time-series data.

For test optimization purposes, the use of machine learning and deep learning has also been investigated for time series anomaly detection in various applications. To solve the data sparsity issue proper of industrial applications, some studies have explored the use of pre-trained networks and transfer learning for the classification of anomalies. Kashiparekh et al. [22] applied transfer learning to a one-dimensional CNN-based approach for detecting anomalies in univariate time series classification. They utilized pre-trained deep networks to handle multiscale learning for varying time series, improving the classification performance across domains. Likewise, Wen et al. use CNN for time-series anomaly detection in three types of anomalies: additive outliers, anomalous temporary changes, and violations of cyclic patterns [23]. Additive outliers are one of the most common anomalies found in univariate time series and represent anomalies present in a very short time span, e.g., spike. The authors discuss that there are challenges in the learning process, such as the absence of anomalies ground truth and the difficulties CNNs have to learn the anomaly features since they are not as easily recognizable as image features. Moreover, in anomaly detection, the learned features may overlap, which makes it difficult to measure the goodness of the detection. This paper grounded the principle of our study since we aim to detect additive outliers from time-series data using CNNs and use transfer learning to predict anomalies in unseen data. However, our proposal uses dynamic thresholds to define the class of anomaly importance during the training. As much as we are aware, this is the closest research done regarding our work. Moreover, we will explore the use of sliding windows to identify patterns in the data, something the mentioned authors

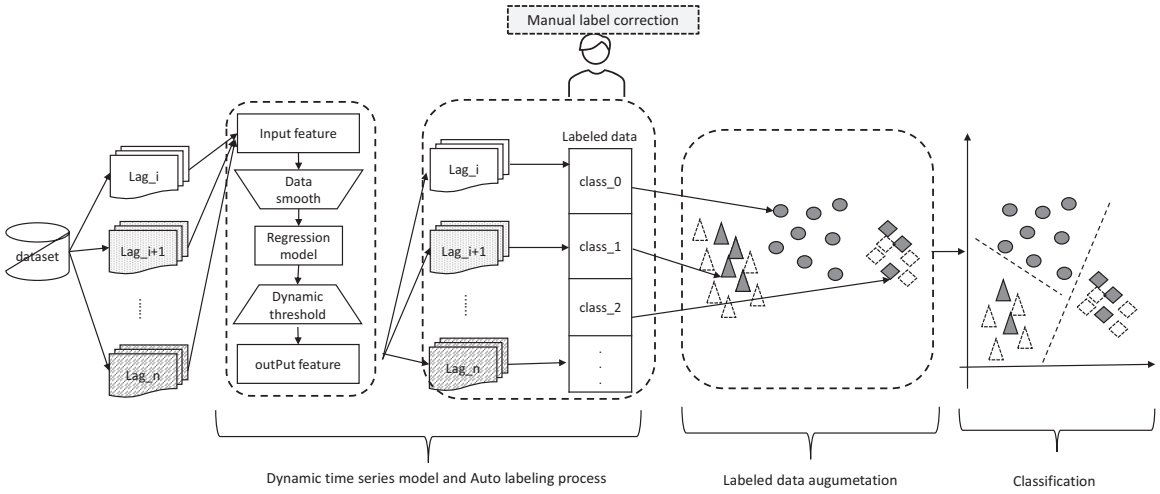

Figure 1. An overview of the proposed solution in this study.

left as future work. While these approaches show promise in the detection of anomalies, there are still limitations that need to be addressed, such as handling imbalanced datasets and real-time processing requirements. Thus, this study aims to propose a new framework that addresses these limitations and improves the effectiveness and efficiency of the testing process.

For a wider view of the latest deep learning techniques to find anomalies in time-series data applied in real use cases, the authors suggest the readers review the work done by Choi in [24].

IV. Proposed Solution

Fixed thresholds have several drawbacks, making it challenging to detect potential system failures and they are insensitive to changes in time-series data. To address these limitations, the proposed solution involves identifying dynamic thresholds that can recognize data patterns, capture golden samples from historical data, and detect new anomalies. Figure 1 provides an overview of the proposed solution presented in this paper, which consists of three primary components: a dynamic time series model and auto-labeling process, data labeling augmentation, and classification.

A. Dynamic time series model and auto labeling process

The input data is one-dimensional data recording the product function test in values and in time series. A time lag is a relatively long period of time between one event and the subsequent occurrence of another related event. The original data set is split into high-dimensional data with the same time lags input to the system. The data is smoothed to remove unwanted noise and to identify the data trend even better than before. Then, the regression model is implemented to make a prediction of the current value at time t. The dynamic threshold can be considered as the error tolerance, which is based on the different times of the mean absolute error in the models. In this case, the actual measured value is compared with different dynamic threshold values at the current time t to make a labeling decision. For instance, if the value is within the most inner dynamic threshold, a safe limit, it is labeled as $'class - 0'$, and so on, as shown in the first part of Figure 1. There are four labels in this study, class 0 representing no anomaly present, class 1 warning, class 2 worse, and class 3 stop. All of them represent actions to take. However, manual correction of the labeling is required at the beginning of the whole system to obtain golden labeling for the later semi-supervised models to obtain a more accurate result. Thus, the process of manual labeling is necessary. The output of the first part of the system will be a labeled data set representing high-dimensional data based on time intervals with appropriate labels after the dynamic thresholds.

B. Labeled data augmentation

The outliers or anomalies are minorities in most anomaly detection tasks. Therefore, the labeled data set would be very unbalanced after the labeling process. Transition data expansion, such as SMOTE, duplicates the smaller labels to balance the data set. However, this method does not work for time-series data because the temporal dependencies in the data set have to be preserved. Each dimension of the data is not independent, but a data series with temporal relationships. This is where the bootstrap data extension was introduced, using the residuals between the different lags of the data. The original

data set to which these residuals are added establishes the same dependencies of the data set as the original data set. To resolve the labeling of the expanded data sets, a semi-supervised model can be used to label these synthetic data. After the process of expanding the labeled data set, the output is a balanced data set ready for the classification task. For comparison, the data set was divided into a training set and a test set, with the training set containing 80% of the unique data points and the test set containing 20% of the unique data points. This was done because the performance of these models is motivated by unseen data. In short, the classification is based on the balanced data set from the second part of the system to produce the output.

C. CNN for classification

Each trained model in the proposed system of work was evaluated for its ability to detect anomalies at different dynamic thresholds, and the regression models in the first part were evaluated for their ability to estimate model accuracy. The difference between the predicted values and the target values was used to predict which data points were anomalous using regression models. Various dynamic thresholds for anomalies, e.g., class 0 to class 3, were used to evaluate anomaly detection. The classification task was evaluated by accuracy, confusion matrix, and the ROC curve which is also the final evaluation of the whole system.

V. Implementation

The proposed system framework aims to monitor the performance of test cases in the target system using a labeled data set. Specifically, given a test case T and a time-series data set D that records the system performance over time, the framework starts by replacing the original timestamps in D with standardized time intervals of 30 seconds, which is the average execution time for the test cases. Before analyzing D, the proposed method first checks if it conforms to the characteristics of time-series data, using its p-value, and if it exhibits white noise, using a combination of p-value and rolling average value. If D is verified to be a non-white noise time-series data, time-series models can be applied. However, to reveal the underlying data pattern from the raw data set, data smoothing is necessary. This can be achieved by rolling a window over the data set, where the size of the window affects the degree of smoothing. For example, wider windows produce smoother trends, while narrower windows are more sensitive to noisy data points.

A. Regression model

The regression model is based on the library Scikit-learn version $0.24.2$ (see: [25]). To get the best performance of different regression models based on the test, linear regression, LassoCV, RidgeCV, and XGBoost were implemented. D is in data frame format with time series. To perform the split between training and testing in terms of time-series structure, a time lag is used to define a fixed period of time for the time unit. That is, the xth lag represents the amount of time that occurs from "k" time points before time x. For example, $Lag1(Y2) = Y1$ and $Lag4(Y9) = Y5$. The lags are defined by the data format integer. The lag start is the first step back in time to intersect the target variable. In this work, the lag start of 6 means that the model uses the values of the last 6 minutes to predict the next minute. The lag *end* means the last step back in time to intersect the target variable. The Lag *end* equals 25 means that the model will see up to 25 minutes in the time series to predict the outcome of the current test unit. By using the range function to separate the input data set D, D is split into multiple "batches". The input features, x, are vectors taken from a list of historical values with predefined time delays, while the labels, y, are based on the actual measured value at the endpoint of the delay. The model prediction is compared to the ground truth to calculate the score.

Creating lagged features by the time period defined from Lag_{Start} to Lag_{End}:

$$Lag_{range} = range(Lag_{Start}, Lag_{End+1}) \tag{1}$$

Then the lag_{range} formed a new data set, which will be put into the regression models to be evaluated, e.g., linear, Lasso, Ridge, and XGBoost regression models.

B. Dynamic threshold

Dynamic thresholds are inspired on the principle of fault tolerance. This fault tolerance comes from the mean absolute error (MAE) definition, i.e., it measures the absolute average distance between the actual values and the predictions.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{D}|y_i - \hat{y_i}| \tag{2}$$

The dynamic thresholds are calculated by Equation 3,

$$DT = \text{ Estimated value } \pm \text{ margin of tolerance} \tag{3}$$

where *DT* is the dynamic threshold and the *margin of tolerance* can be described as:

$$\text{margin of tolerance } = \text{MAE} + \text{scale} * \sigma_{MAE} \tag{4}$$

σ_{MAE} refers to the standard deviation of MAE, it measures the amount of variation in the MAE across different samples and *scale* is a constant that is used to adjust the margin of error based on the sample statistical confidence level, i.e., 95% or 2 sigma, 99.7% or 3 sigma, and monitoring to define the dynamic test limits for the classes 0 to 3 mentioned in Section IV.

C. Time-series data augmentation and label balancing

The goal of this process is to augment a data set using the MBB technique, which preserves the time dependencies of the original data set and then assigns the correct labels to this augmented data. The augmented data combined with the original labeled data set, creates a balanced data set for the subsequent classification task. Time-series DA is illustrated in Figure 3, where the block size and step size are set to

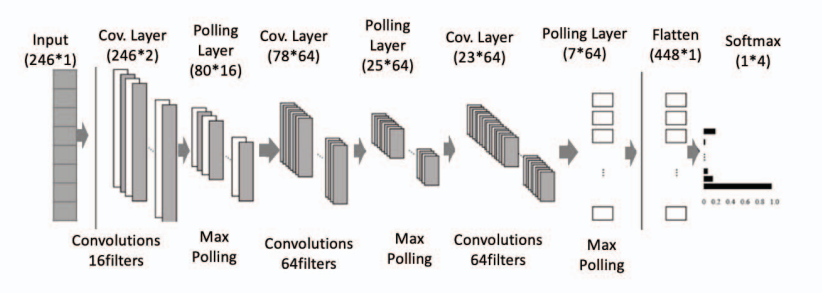

Figure 2. The CNN network visualization.

1. The original data set is split into Y_n blocks of equal length, each containing a fixed number of time lags. The MBB method is then used to generate residuals by performing successive subtractions between adjacent blocks, resulting in $n-1$ residuals. The augmented data set is generated by adding these residuals to the original data set.

D. Labeled data set re-sampling

The augmented data acquires the labels from the label propagation. The test experts correct those labels for the minority classes, i.e., 'Warning(1)', 'Worse(2)', and 'Stop(3)'. To get a balanced data set for the classification task, the implementation combines two random sampling techniques and obtains better overall performance results compared to the original data. The idea behind re-sampling in this section is to use over-sampling of the minority class, and simultaneously apply under-sampling to the majority class from the original labeled data set, which reduces the bias toward the majority class examples. The output results after re-sampling are shown in Table I.

Label	Quantity	Sampling method	Output
Normal	1953	Under-sampling	233
Warning	16	Over-sampling	177
Worse	41	Over-sampling	193
Stop	64	Over-sampling	195

TABLE I. A SUMMARY OF THE ORIGINAL AND AUGMENTED DATA RESAMPLING.

E. CNN on classification

The CNN network diagram is depicted in Figure 2. The input data is a one-dimensional time series with a shape of $246*1$. The data is first processed by two convolutional layers, each consisting of 16 filters. Subsequently, a max pooling layer is applied, resulting in a shape of $80*16$. Afterward, the data passed through two additional convolutional layers, each with 64 filters, followed by another max pooling layer. The resulting data shape is $25*64$. This is then followed by two more convolutional layers, each with 64 filters, and a final max pooling layer, resulting in a data shape of $7*64$. The flattened data is then converted into a one-dimensional format, resulting in a data shape of $448*1$. Finally, the data passed through a dense layer with softmax activation, which produces the classification output, representing one of four possible classes.

VI. RESULTS

The project effort resulted in an implementation proposal including a number of measures that could improve test efficiency. Primarily, the actions include data-driven thresholds, which detect the failures based on the data behavior and trend from historical time-series data. The implementation plan also provided a new solution for the data set with similar features and properties.

A. Regression evaluation performance

All the regression models studied in this work show better prediction results on the data set after smoothing. Table II is an overall view of the performance of the regression models implemented in this study. From the table, XG-Boost has the largest MAE value, likely caused by the hyper-parameters tuning did not reach the best scope of data in the built-in cross-validation process. Furthermore, linear, Lasso, and Ridge regression models have lower computational costs than XGBoost given the accuracy results. However, linear regression does not have a regularization item to improve the speed performance as the Lasso regression nor help to avoid over-fitting such as Ridge regression. Therefore, linear prediction performance shows slightly worse results than Ridge regression for this data set.

Corresponding to the model score, the Lasso and Ridge have low MAE. However, the Ridge regression model has the best RMSE which is the dynamic threshold impacted. Furthermore, the Ridge regression model is very helpful when there are dependencies between variables. The regularization in the Ridge regression model helps reduce the dimensionality of the training set to prevent the over-fitting issue. As the time-series data studied here includes many lags of data as input features, whereas, the Ridge regression simplifies the features by regularization as well to lower the computational cost. According to the result analysis on model scores and model error rates, the Ridge regression is the best-fit model chosen in this part of the work to fulfill the labeling on the data set.

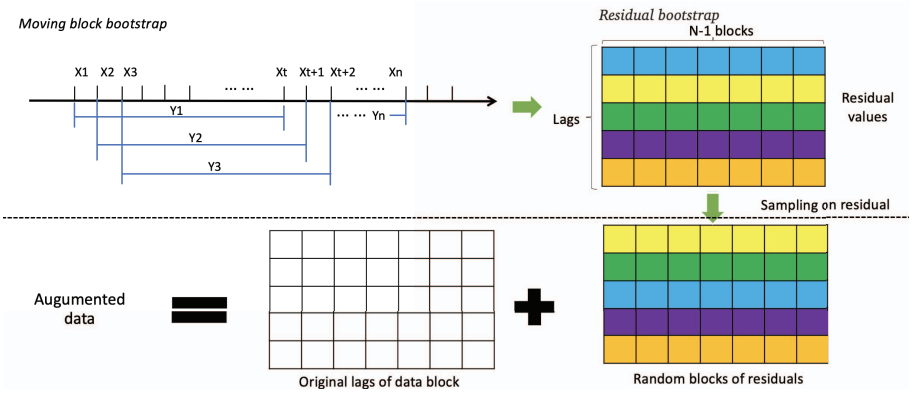

Figure 3. Data augmentation process.

Model Name	RMSE	MAE	Accuracy	
			Training	Testing
Linear Regression	0.032	0.0021	0.96	0.95
Ridge regression	0.032	0.0020	0.96	0.95
Lasso Regression	0.039	0.022	0.96	0.95
XGBoost	0.031	0.036	0.93	0.91

TABLE II. THE PERFORMANCE SUMMARY OF THE DIFFERENT REGRESSION MODELS.

B. Classification performance

The CNN classifier was trained to recognize and classify different test anomalies in real-time. The models performance was evaluated on a dedicated test set consisting of a representative sample of test anomalies. The test set included a range of test anomalies, including ones with varying degrees of failures.

The evaluation of the model's performance was based on its ability to accurately classify anomalies into one of four categories: *Normal* (class label 0), *Warning* (class label 1), *Worse* (class label 2), and *Stop* (class label 3). The classification results were then presented as a confusion matrix, which shows the number of correct and incorrect classifications for each category as shown in Figures 4. The confusion matrix revealed the dynamic thresholds used in the CNN classifier influenced the classification of the "Worse" category. The close proximity of the dynamic thresholds for class "Worse" to those of classes "Warning" and "Stop" created challenges for the model in accurately distinguishing between them. As a result, there were more misclassifications and confusion between the "Worse" category and the "Warning" and "Stop" categories. On the other hand, the confusion matrix also showed that the model performed well in classifying the "Normal," "Warning," and "Stop" categories, suggesting that the dynamic thresholds for these classes effectively differentiated them from one another.

To further evaluate the performance of the model, a receiver operating characteristic (ROC) curve was also plotted in Figure 5, which displays the model's true positive rate (sensitivity) against its false positive rate (1-specificity) for different classification thresholds. The ROC curve demonstrated that all categories were well-classified, with a highest area under the curve (AUC) of 0.94. This indicates that the model had a high overall accuracy in classifying test anomalies into the correct category.

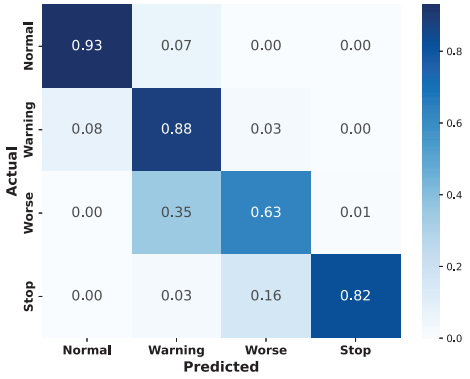

Figure 4. CNN classification test set confusion matrix.

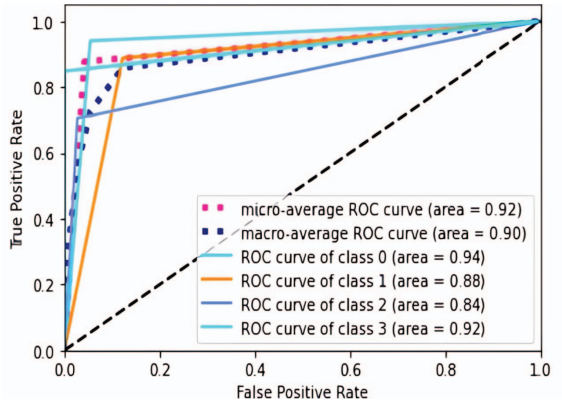

Figure 5. CNN classification test set evaluation visualization.

C. Transfer learning on a correlated data set

In the case of object recognition using a CNN, transfer learning involves taking a pre-trained CNN model and adapting

it to recognize objects in a new data set. Typically, this is achieved by freezing the early convolutional layers of the network, which are responsible for identifying low-level features in the image, and only training the final dense layer, which is responsible for making predictions based on those features. This approach can be effective because the low-level features identified by the early layers are often useful for recognizing objects in different data sets, while the final layer can be fine-tuned to produce accurate predictions on the new data set. In the context of evaluating transfer learning on another data set, the ROC curve was used to assess the performance of the adapted model as shown in Figure 6. The evaluation of transfer learning on an industry data set revealed that the adapted CNN model performed reasonably well, but not as well as the pre-trained model on the original data set. The low Pearson's correlation coefficient between the two data sets, i.e., 0.63, indicated that they were not highly correlated, which likely contributed to the lower performance of the adapted model. The class imbalance in the 'Stop' class also presented a challenge for the adapted model, as there were very few labeled examples of this class compared to the other classes. This made it difficult for the model to learn to recognize this class effectively, resulting in lower classification accuracy for this class. Despite these challenges, transfer learning still showed promise for adapting pre-trained models to new data sets. However, careful consideration of the similarities and differences between the data sets is crucial to ensure effective knowledge transfer and avoid negative transfer [26]. This includes taking into account important factors such as class imbalance, differences in the characteristics of the data sets, and the similarity (correlation) between source task and target task [27]. In general, while transfer learning can be a useful technique for adapting pre-trained models to new data sets, it is important to approach it with caution and to carefully evaluate the performance of the adapted model to ensure that it meets the desired accuracy and performance standards.

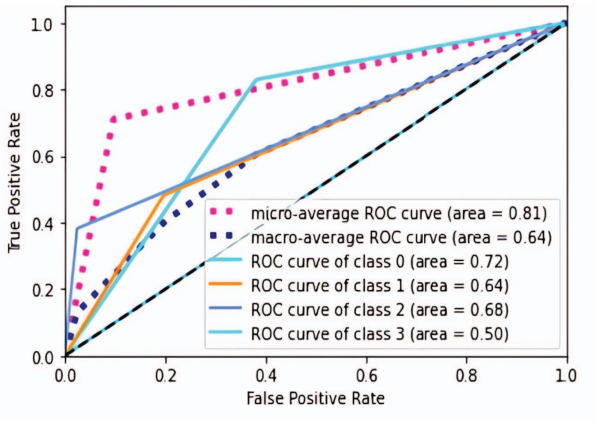

Figure 6. Transfer learning classification ROC curve on a correlated data set.

VII. Discussion and Conclusions

Our study focuses on improving the test case efficiency and effectiveness for radio access network testing by exploring the potential of data-driven thresholds, anomaly detection, and performance prediction. The proposed CNN-based anomaly detection framework automatically labels the test inputs according to the level of an anomaly on large amounts of data from numerous sources and domains. Furthermore, the framework has several potential applications, including real-time anomaly detection, feature extraction, network diagnosis, and network optimization/simulations. The study confirms that the proposed framework represents a significant advancement in the field of radio access network test optimization as it allows for quick identification of potential issues beforehand and thus, lead to faster, more efficient, and accurate test. However, the study also revealed some challenges in transfer learning, such as class imbalance and anomalies feature overlapping that can affect the classification performance of the framework when applied in cases of data scarcity. The final test results achieved 86% classification accuracy, and the transfer learning results on correlated time-series data also achieved promising results on input data with enough samples, i.e., up to 72%. Nevertheless, further research is necessary to explore the full potential of the proposed framework for test optimization purposes. The use of machine learning and deep learning techniques can help identify trends and patterns that may not be immediately apparent to human testers, leading to more effective test execution than using fixed thresholds without monitoring the changes till the failure has occurred.

Overall, the proposed CNN-based anomaly detection framework shows great potential to contribute to the test optimization for radio access networks, but further research is necessary to address the challenges and explore its full potential like the training in larger data sets where different anomaly features can be learned. Likewise, transfer learning is desirable in data sets where there are not enough failures. Though, the unseen data need to share the same data characteristics as the one used for the training to avoid the negative transfer. Additionally, more experimentation on different time lags of data series is necessary to account for changes in time-series data, which are expected to grow and shrink with the span between the dynamic thresholds calculated from the regression model.

Acknowledgments

This work has been supported by the Industrial Graduate School Collaborative AI & Robotics funded by the Swedish Knowledge Foundation (KKS) Dnr:20190128 and Swedish Governmental Agency for Innovation Systems (VINNOVA) D-RODS (2023-00244).

References

[1] B. Lane, M. Poole, M. Camp, and J. Murray-Krezan, "Using machine learning for advanced anomaly detection and classification," in *Advanced Maui Optical and Space Surveillance Tech. Conf.(AMOS)*, 2016.

[2] S. Tahvili and L. Hatvani, *Artificial Intelligence Methods for Optimization of the Software Testing Process With Practical Examples and Exercises*. Elsevier, July 2022.

[3] S. Tahvili, *Multi-Criteria Optimization of System Integration Testing*. PhD thesis, M
"alardalen University, December 2018.

[4] C. Landin, X. Zhao, M. Längkvist, and A. Loutfi, "An intelligent monitoring algorithm to detect dependencies between test cases in the manual integration process," in *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 353–360, 2023.

[5] C. Kaner, J. Falk, and H. Q. Nguyen, "Automated software testing," *Software testing and analysis: process, principles, and techniques*, pp. 173–206, 2002.

[6] D. Montgomery, E. Peck, and G. Vining, *Introduction to Linear Regression Analysis*. John Wiley & Sons, 2015.

[7] I. Bessa, M. Valente, and F. Dantas, "Time series analysis in software engineering: An empirical study on test suites," *Empirical Software Engineering*, vol. 22, no. 4, pp. 1884–1924, 2017.

[8] K. Yang, X. Yu, J. Wu, Q. Zhang, and X. Lu, "Deep convolutional neural networks for industrial anomaly detection in the production of printed circuit board assemblies," in *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 1830–1835, IEEE, 2019.

[9] H. He and E. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[10] N. Chawla, K. Bowyer, L. Hall, and P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[11] H. R. Kunsch, "The Jackknife and the Bootstrap for General Stationary Observations," *The Annals of Statistics*, vol. 17, no. 3, pp. 1217 – 1241, 1989.

[12] S. Lahiri, *Resampling Methods for Dependent Data*. Springer Science & Business Media, 2013.

[13] J.-P. Kreiss and S. N. Lahiri, "1 - bootstrap methods for time series," in *Time Series Analysis: Methods and Applications* (T. Subba Rao, S. Subba Rao, and C. Rao, eds.), vol. 30 of *Handbook of Statistics*, pp. 3–26, Elsevier, 2012.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[15] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[16] J. Knight, *Fundamentals of dependable computing for software engineers*. CRC Press, 2012.

[17] L. Myllyaho, M. Raatikainen, T. Männistö, J. K. Nurminen, and T. Mikkonen, "On misbehaviour and fault tolerance in machine learning systems," *Journal of Systems and Software*, vol. 183, p. 111096, 2022.

[18] S. Brahma, R. Kavasseri, H. Cao, N. Chaudhuri, T. Alexopoulos, and Y. Cui, "Real-time identification of dynamic events in power systems using pmu data, and potential applications-models, promises, and challenges," *IEEE transactions on Power Delivery*, pp. 294–301, 2016.

[19] S. Weiss, A. Dhurandhar, and R. Baseman, "Improving quality control by early prediction of manufacturing outcomes," KDD '13, p. 1258–1266, 2013.

[20] S. Weiss, A. Dhurandhar, R. Baseman, B. White, R. Logan, J. Winslow, and D. Poindexter, "Continuous prediction of manufacturing performance throughout the production lifecycle," *J. Intell. Manuf.*, vol. 27, p. 751–763, aug 2016.

[21] C. Landin, J. Liu, and S. Tahvili, "A dynamic threshold based approach for detecting the test limits," in *The Sixteenth International Conference on Software Engineering Advances*, October 2021.

[22] K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, and G. Shroff, "Convtimenet: A pre-trained deep convolutional neural network for time series classification," in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2019.

[23] T. Wen and R. Keyes, "Time series anomaly detection using convolutional neural networks and transfer learning," *ArXiv*, vol. abs/1905.13628, 2019.

[24] K. Choi, J. Yi, C. Park, and S. Yoon, "Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines," *IEEE Access*, vol. 9, pp. 120043–120065, 2021.

[25] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[26] J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, no. 10, pp. 1345–1359, 2010.

[27] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. Van Der Maaten, "Exploring the limits of weakly supervised pretraining," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 181–196, 2018.