Jurnal Rekayasa Elektrika Vol. 19, No. 2, June 2023, Page 71-77
ISSN. 1412-4785; e-ISSN. 2252-620X, Accredited by Kemendikbudristek No. 105/E//KPT/2022
DOI: 10.17529/jre.v19i2.31090

Detection of Intermittent Oscillation in Process Control Loops with Semi-Supervised Learning

Nova Zidane Ibrahim, Awang Noor Indra Wardana*, and Agus Arif
Departement of Nuclear Engineering and Engineering Physics, Faculty of Engineering, Universitas Gadjah Mada
Jl. Grafika 2, Yogyakarta 55281
e-mail: awang.wardana@ugm.ac.id

***Abstract*—Oscillations in the control loops indicate the poor performance of the control loops. The occurrence of oscillations in the process control loop is quite high in the industry, so it needs to be reduced so that the control loop can work properly. The first step for oscillation reduction is oscillation detection. One type of oscillation that is difficult to detect is intermittent oscillation. The smart factory concept encourages the development of the intermittent oscillation detection system using machine learning by being implemented online. Therefore, in this study, an online intermittent oscillation detection program is built using the K-Nearest Neighbor (KNN)-based Semi-Supervised Learning (SSL) method. The SSL method applied is self-training. The training data was obtained by a simulation of the Tennessee Eastman Process (TEP). The data is segmented based on window size and extracted time series features. The extracted data is used to build a model to detect oscillations caused by stiction, tuning errors, and external disturbances in the reactor. The model is implemented online with sliding windows and a standard for internet of things messaging i.e., Message Queuing Telemetry Transport (MQTT). The best accuracy and F1-score of the model obtained are 96.15% and 95.15%. In online detection, the model detects the type of oscillation with an average time of 305 seconds.**

Keywords: ***intermittent oscillation, control loops, semi-supervised learning, k-nearest neighbor***

I. INTRODUCTION

The control loop is an essential part of the process industry. High-performance results must be had by each control loop so that the output quality is well maintained and as required [1]. However, every process industry will surely have a poor performance control loop. Based on a survey [2], 10% of 26,000 PID controllers from different process industries had poor performance results. The poor performance is caused by the appearance of oscillations which could be identified through the data trend of the process variables. The occurrence of an Oscillation in a control loop can reach about 30% up to 41% [3], [4]. The oscillations refer to oscillations that can cause excessive controller movement of the control loops and increase the wear of the instruments in the control loop. Therefore, the occurrence of oscillations in the control loop should be reduced. An initial step is to detect oscillations of a control loop in the process industry.

Oscillation detection is challenging in the process industry, especially in control loops. Oscillations occurring in control loops are usually caused by stiction, dead zone, or hysteresis in control valves, external disturbances received by the process, and poor controller tuning [5], [6]. One type of oscillation that is usually found in industry is intermittent oscillation. An intermittent oscillation is an oscillation that appears intermittently within a period and suddenly [7]. It needs to be detected so that the performance degradation of the control loop can be reduced in the long term and sustainably.

Research on intermittent oscillation detection in control loops has already been published. However, the number of studies is still limited to using ruled-based methods that are implemented offline (manual or without being connected to the network) [8]–[12]. Several offline methods are the Intrinsic Time-scale Decomposition (ITD) method, zero crossing of Auto-Covariance (ACF) method, peaks wavelet method, and Discrete Fourier Transform (DFT) method. These methods have difficulties when applied online (networked) because the determination of performance results and the selection of data window size are not appropriate [10]. In addition, oscillation detection using the analysis method of looking for peaks in the signal power spectrum is not easy. This method only looks for the highest peak amplitude, so it is less suitable for detecting intermittent oscillations in periods that vary each cycle [13]. Along with the development of technology, the use of ruled-based methods and working offline needs to be changed by using simple methods and being able to work online. This is adapted to the smart factory concept.

The smart factory concept is a new concept in the industrial revolution 4.0 era that encourages the use of intelligent sensors, devices, and machines so that factories can continuously collect productivity data [14]. The use of Artificial Intelligence (AI) techniques has played a role in building experience-based models from process data so that decisions can be made more quickly and effectively, do not require complex rules, and can

Received 8 April 2023; Revised 4 May 2023; Accepted 5 May 2023

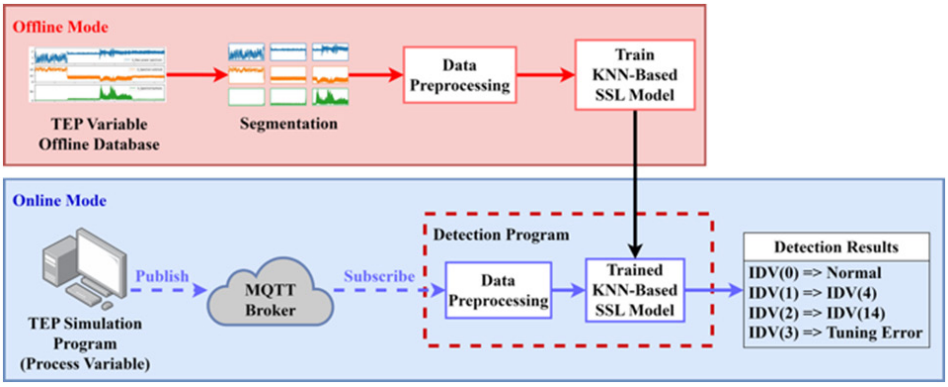


Figure 1. The proposed intermittent oscillation detection model

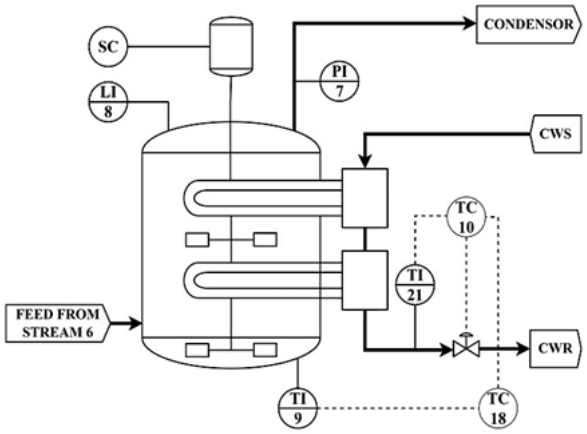


Figure 2. Process and Instrumentation Diagram (P&ID) reactor of TEP where CWS/R is cooling water water steam/return, SC is speed stirred control, TC is temperature control, TI/PI/LI are temperature/pressure/level indicators [25]

adjust according to the given pattern to improve their performance [15], [16]. Many studies have been published about oscillation detection using AI, such as building a detection model using Deep Feedforward Network (DFN) model [16], Deep Convolutional Neural Network (DCNN) [17], Artificial Neural Network (ANN) [18], Deep Belief Network - Adaptive Lifting Wavelet (DBN-ALW) [19], Extreme Gradient Boosting (XGBoost) [20], and Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and recurrent neural network (RNN) [21]. However, most of them still apply offline and deep learning methods so that they cannot detect in real-time when there is a disturbance and the computation time is too long.

Therefore, this study will develop a method to detect intermittent oscillations using the K-Nearest Neighbor (KNN)-based Semi-Supervised Learning (SSL) method that can work online. The KNN-based SSL method is chosen because the model built can combine supervised learning and unsupervised learning. It can handle the amount of labeled data using pseudo-label generation techniques with shorter computation time than deep learning. The SSL method applied is self-training that conducts the learning process by using its prediction results to train itself. The benefits of self-training and the KNN algorithm are chosen as the basis of the model because it has simple and efficient characteristics so that any simple or complex modeling can be incorporated into the self-training framework [22]–[24].

II. METHOD

This study proposes an intermittent oscillation detection model to classify the cause of oscillation in the Tennessee Eastman Process (TEP) shown in Figure 1. The stages in this study are data collection, data preprocessing, design of detection model using SSL KNN-based offline, model implementation online, and model performance evaluation.

A. Data Collection

The data used in this study are time series data of output variables which are the results of the TEP program simulation for 12 hours with a total of 52 variables consisting of 11 manipulated variables and 41 measured variables. The TEP program is an overall chemical plant dynamics first proposed by Down and Vogel [25]. To optimize the results obtained, this study will focus on the processes that occur in the reactor. The reactor used in TEP is an exothermic type. This reactor produces heat, requiring cooling water to maintain the reactor temperature. The control structure of the TEP reactor is shown in Figure 2.

There are three faults in the reactor, consisting of step, random variation, and sticking. However, random variation in the reactor is not applied and is replaced by adding a tuning error failure. The tuning fault type is used in this study by changing the proportional gain constants of TC-10 and TC-18. The details and name labeling of each fault that causes oscillation are modified according to Table 1.

B. Data Preprocessing

The TEP simulated data for 12 hours were segmented according to the window size in Table 2. The window size segmentation reference is based on the research of Henry [26]. Furthermore, the data segments were feature extracted into three domains (time, spectral, and statistical domains) using the Time Series Feature Extraction Library (TSFEL) [27].

Tabel 1. Details of disturbance type applied

Disturbance Name	Process Variable	Oscillation Type	Label
Normal	Normal/no-fault	Normal	IDV(0)
IDV(4)	Reactor cooling water inlet temperature	Step	IDV(1)
IDV(14)	Reactor cooling water valve	Sticking	IDV(2)
Tuning Error	Proportional gain constant	Tuning Error	IDV(3)

Tabel 2. Details of disturbance type applied

Hyperparameter	Default Value	Variation
Window size	100	[100, 150, 200]
k-value	5	[5 – 15]
Distance equation (p)	euclidean	[manhattan, euclidean]

The extracted features were selected using Pearson correlation. This correlation will be compared against the label. The chosen features have a high correlation with the label. The feature selection results are then divided into training and testing data with a composition of 80% and 20%.

C. K-Nearest Neighbor-based Semi-Supervised Learning

The KNN-based SSL model is built using the self-training method first proposed by Scudder [23]. The pseudo-labeling implementation is done by training a KNN model with correctly labeled data and then using the results to label the unlabeled data to get a pseudo-label. After that, the pseudo-labeled data with high prediction probability is combined with the initial true-labeled data to serve as the basis for building a self-training model. Figure 3 shows the flow of the self-training method. The performance results of a proper self-training method depend on the pseudo-labeled data selected in the model training procedure [28].

The evaluation for this offline model uses accuracy and F1-score. If the evaluation result of the initial model is not good, the hyperparameter variation is performed according to Table 2. The best offline evaluation results will be implemented in the online model.

Online implementation is executed by subscribing to the TEP simulation output data from the broker using the Message Queuing Telemetry Transport (MQTT) communication protocol at certain time intervals. Detection will be done when the data length matches the window size of the best model. If it does not fulfill, the detection will wait until the data length fits the window size. The complete data will be extracted with TSFEL, and features will be selected according to the results of the previous feature selection offline. The data is then classified using the KNN-based SSL model. The result of testing the online model is the time it takes for the model to detect and classify the type of oscillation that appears correctly.

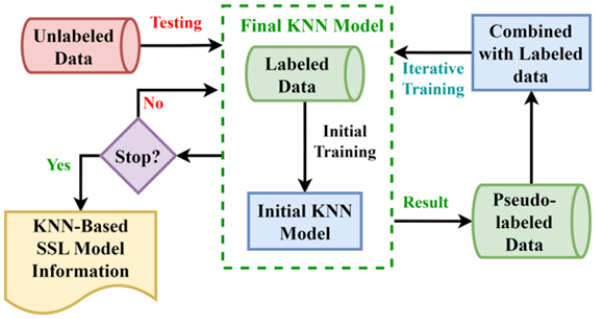


Figure 3. Flowchart of the self-training method [23]

On online detection of oscillations of each disturbance type, the simulation is run for 30 minutes. Each disturbance type was triggered after the first 100 seconds. The correct detection is considered if the model can detect the disturbance type 30 times. Meanwhile, the online intermittent oscillation detection test triggered the disturbance type for 30 minutes and then turned it off for 15 minutes. After that, the other disturbance type was trigged for 30 minutes. The behavior of the model in classifying the intermittent oscillation types were tested. The detection time of each oscillation change was calculated to compare it with the average time of the online detection of each disturbance type. This study uses several Python libraries, such as pandas, numpy, seaborn, matplotlib, scikit-learn, tsfel, and paho-mqtt.

III. Results and Discussion

A. Data Collection and Preprocessing Results

Based on the simulation results for data collection, the number of data types obtained is four types of time series data with a total data length for the three types of oscillations of 129,600 data and the normal type of 43,200 data. Furthermore, the data is segmented based on the window size of 100, 150, and 200. The segmentation results are extracted to obtain characteristics based on the three domains.

The extraction results obtained are 9620 features. These features are extracted from the factors i.e., Fast Fourier Transform (FFT) mean coefficient, mean absolute deviation, Empirical Cumulative Distribution Function (ECDF) percentile, spectral distance, variance, spectral kurtosis, peak-to-peak distance, human range energy, and turning point. Each factor can produce more than one feature. These features must be selected so that the model built has high accuracy. Feature selection uses Pearson correlation coefficient. The selection results obtained were 142 features out of 9620, or about 1.5%, used to create the model. The feature selection results were split into test and training data based on Table 3. The training data is again divided into labelled data to train the initial KNN model and unlabelled data to be used as pseudo-labelled data. The division for labelled data is 40% and 60% for unlabelled data.

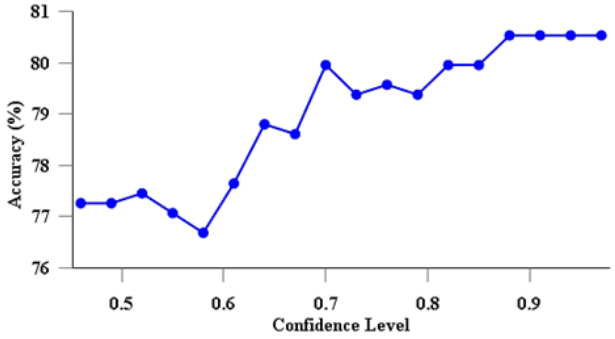


Figure 4. Effect of confidence level on accuracy for the KNN-based SSL model

Tabel 3. Details of the data used

Window Size	Labeled Data	Unlabeled Data	Test Data	Total
100	483	726	519	1728
150	322	484	346	1152
200	241	363	260	864

B. Self-Training Results

The KNN-based SSL model will use the KNN algorithm to create pseudo-labels for unlabeled training data and make predictions for new data. This is due to the use of the self-training method so that the learning process of the ML model can use its predictions to teach itself. The initial KNN model and the initial KNN-based SSL were built using default parameters according to Table 2, namely a k-value of 5, the use of Euclidean equations, and a window size of 100. The test results obtained for the initial KNN model for accuracy are 80.92%, and the F1-score value is 80.96%. These performance results will affect the results of the pseudo-label prediction, so it is necessary to observe the training data and the probability of confidence level so that the pseudo-label prediction results can be even better.

The results of the effect of different limit values for the probability of confidence level with the accuracy results of the KNN-based SSL model are shown in Figure 4. The higher the probability of confidence level, the higher the results' accuracy. The greatest increase in accuracy occurs when the probability value is more than 0.88, which is 80.92%, and the rest, there is no increase and tends to be stable until the probability is 1. This is because the observation is very similar to the label.

The results of the initial KNN model are compared with the KNN-based SSL model to see the effect of self-training. Based on Figure 5, the results of the initial KNN-based SSL model have no difference from the initial KNN model for pseudo-labeling because the KNN-based SSL model created only adds new training data, namely pseudo-label training data, without eliminating the old, labeled training data. Therefore, the results will be the same as the testing and validation results of the KNN model for pseudo-labeling. In this model, self-training does not affect the applied performance. However, the application for this research still uses the KNN-based SSL model.

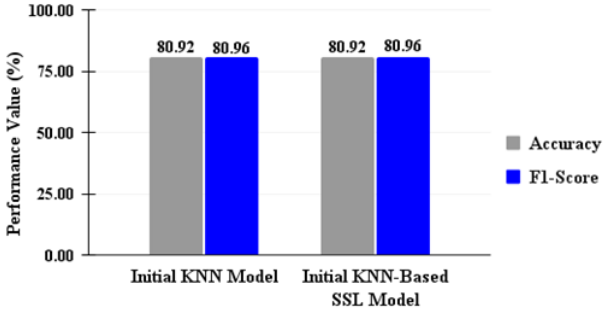


Figure 5. Effect of the self-training model

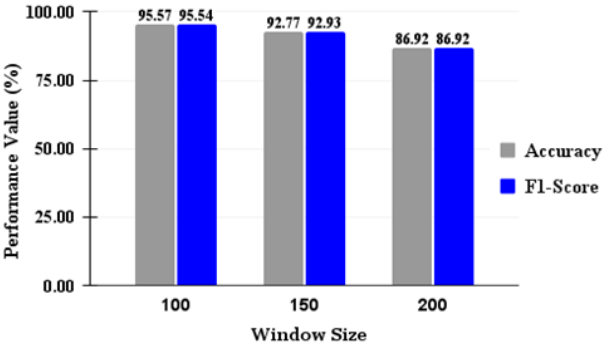


Figure 6. Window size variation validation results

C. Hyperparameters Tuning Results

Hyperparameter tuning is carried out here to optimize the performance results of the KNN-based SSL model. The hyperparameter variations performed for the KNN-based SSL model are the k value and the distance equation, as shown in Table 2. For the confidence level, the threshold value will be fixed, which is 0.88. In addition, adding window size parameters also determines the results obtained. The hyperparameter tuning results for the model built with the best accuracy value and F1-score value will be selected as the final model for oscillation detection model building. The tuning validation is validated by cross-validation to obtain optimal results.

The best model from the hyperparameter tuning results of the KNN-based SSL model has a k value of 12, using the Manhattan equation, and a window size of 100. The window size variation validation is shown in Figure 6. The results of this tuning are appropriate because it is mainly related to the use of the Manhattan distance, which is suitable for high dimensionality in the data used. Suppose the test results of this final model are compared with the test results of the initial model, as shown in Figure 7. In that case, the initial model has an accuracy and F1-score of 80.92% and 80.92%, while the final model has an accuracy and F1-score of 96.15% and 96.15%. These results have significantly improved compared to the initial KNN-based model, preventing the model from being overfitted. However, the accuracy and F1-score values did not reach perfection because the most misdetections were IDV(1),

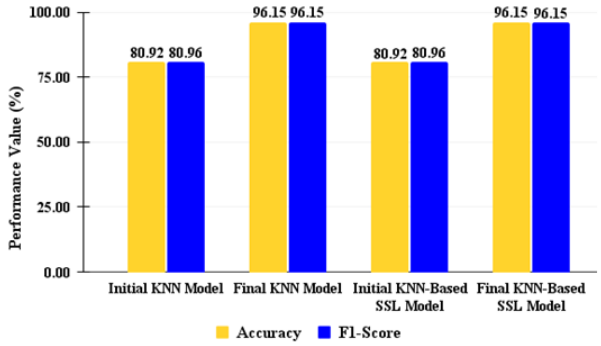


Figure 7. Comparison of the initial model with the final model

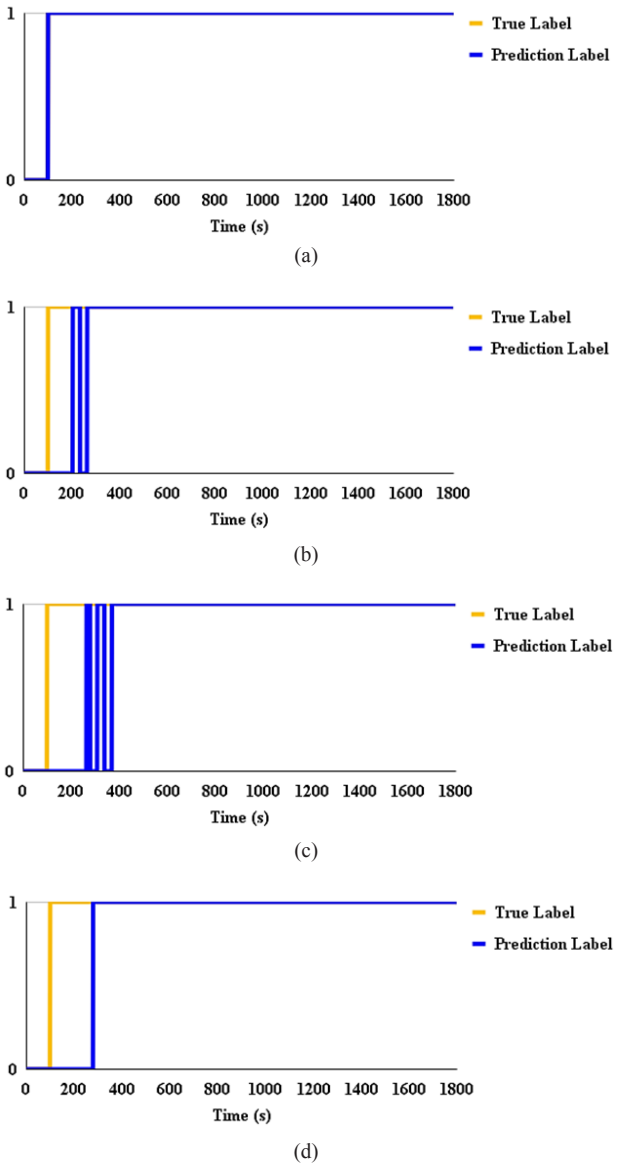


Figure 8. The result of prediction on (a) IDV(0); (b) IDV(1); (c) IDV(2); and (d) IDV(3)

as the true label was predicted to be IDV(0). This is due to the damping process of the tuning controller in the TC so that its oscillation signal merges into the natural signal of the system. However, the results of this best model will be applied online.

Tabel 4. Details of the detection time

Oscillation Type	Detection Time (s)
IDV(1)	265
IDV(2)	370
IDV(3)	280
Average	305

D. Prediction Results

The prediction results obtained by the program when performing online predictions for each type of fault are shown in Figure 8. In the graph, condition 1 indicates that the detection result is correct. In contrast, condition 0 after the 100th second suggests that the detection result is wrong, and condition 0 before the 100th second indicates that no detection has been made.

The details of the detection time required for the program to detect the stability of each disturbance are shown in Table 4. The average detection time obtained is 305 seconds or 5 minutes and 5 seconds. The performance result for the accuracy of the online program brought is 92.11%.

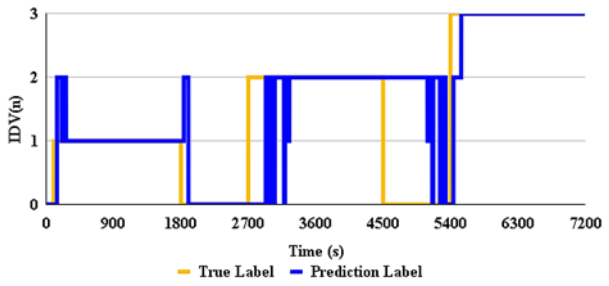
The program can detect very well without errors during normal or undisturbed conditions. This excellent detection needs to be the foundation because the normal operation is the default value that should not have any errors in the detection. Meanwhile, the condition when the disturbance is given is not as it should be. At the beginning of the IDV(1) or IDV(2) disturbance, the program does not immediately detect IDV(1) or IDV(2) stably, but a detection error occurs. This is due to the influence of the ripple-type disturbance that affects the initial moment, so the detection looks similar to other disturbances. After passing the 265th second, the oscillation detection starts to stabilize to predict IDV(1), while IDV(2) only stabilizes after passing the 370th second. For IDV(3), the detection experiences errors from the start of the disturbance until the 280th second. This is because when this disturbance is applied, there will be a very high increase in value compared to the other two disturbances so that for the first 280 seconds, the system predicts it more like IDV(1) and IDV(2).

We compared the proposed system's detection time with the DCNN and XGBoost methods. Different IDV names are applied from each model, so the IDV naming is adapted to this study. In addition, the test conditions are also different. Wu et al. used sampling lengths of 20 and 10 with a three-minute interval [17], while this study and Rabba et al. used a data length of 100 with a one-second interval [20]. However, the sampling period is the same, which is 15 seconds. Two disturbances were selected for the DCNN model, namely IDV(1) and IDV(2), while the XGBoost model selected three disturbances, namely IDV(1), IDV(2), and IDV(3). These choices are compared and shown in Table 5. The detection time calculation is based on the time it takes for the model to detect the

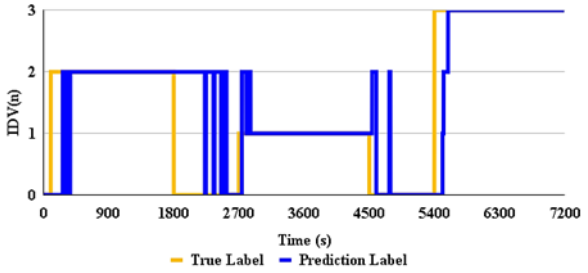
Tabel 5. Details of the detection time

Oscillation Type	DCNN (C*=20) [17]	DCNN (C*=10) [17]	XGBoost [20]	This Study
IDV(1)	480	480	870	265
IDV(2)	720	720	330	370
IDV(3)	-	-	350	280
Average	600	600	517	305

C*: sample length time



(a)



(b)

Figure 9. The result for online detection of intermittent oscillations for (a) test 1 and (b) test 2

disturbances 30 sequential times correctly. Overall, the average detection time required for each fault type since initiation was 305 seconds. The model detected the cause of the oscillation faster than the DCNN model, with sample length times of 10 and 20, by 96.72%, and the XGBoost model by 69.51%.

E. Online Intermittent Oscillation Detection Test

The online intermittent oscillation test was conducted twice, as shown in Figure 9. The program was able to correctly detect the type of disturbance, although the model was still unable to predict directly when the type of disturbance was given. This is normal because oscillations will not be formed immediately when a new disturbance is given. It takes some time for the oscillation to be visible so that it can be detected. In addition, when the disturbance is removed, oscillations still occur, and detection is still not predicted to be normal immediately because it still takes time for the oscillations to return to normal and not immediately.

IDV(1) has an initial error when a disturbance is applied and a final error when the disturbance is stopped. That error is a detection error into IDV(2). This is because when the disturbance is applied, there is a spike in oscillation similar to IDV(2), but it stabilizes afterward. Even so, the program created is appropriate despite a slight misdetection when the disturbance is given.

IV. Conclusion

The research that has been conducted has the main objective of building a KNN-based SSL model in an intermittent oscillation detection program that can work online and in real time. The online periodic oscillation detection program that has been created is influenced by several factors, namely the window size selection and hyperparameter values. Based on offline testing and validation results, the KNN-based SSL model has the best performance results with a data window size of 100, the number of nearest neighbours or k value of 5 and using the Manhattan distance equation as it is suitable for high dimensionality in the data used. The model performs well in classifying the oscillation type from various causes with an accuracy value and F1-score of 96.15% and 96.15%. The result of testing the online detection program that has been carried out can detect oscillations with an average time of 305 seconds for each class. It was faster than the detection time using the DCNN model, which needs 600 seconds, and the detection time XGBoost, which requires 517 seconds.

References

[1] D. Zheng, X. Sun, S. K. Damarla, A. Shah, J. Amalraj, and B. Huang, "Valve stiction detection and quantification using a k-means clustering based moving window approach," *Industrial & Engineering Chemistry Research*, vol. 60, no. 6, pp. 2563–2577, 2021.

[2] R. Srinivasan and R. Rengaswamy, "Control loop performance assessment. 1. A qualitative approach for stiction diagnosis," *Industrial Engineering Chemistry Research*, vol. 44, no. 17, pp. 6708–6718, 2005.

[3] W. L. Bialkowski, "Dreams vs. Reality: A view from both sides of the Gap," *JDream III for advanced search and analysis*, vol. 94, pp. 19–27, 1994.

[4] B. S. Torres, F. B. Carvalho, M. O. Fonseca, and C. S. Filho, "Performance assessment of control loops-case studies," *in Proc. International Symposium on Advanced Control of Chemical Processes*, 2006, pp. 1–6.

[5] V. Akavalappil and T. K. Radhakrishnan, "Comparison of current state of control valve stiction detection and quantification techniques," *Transactions of the Institute of Measurement and Control*, vol. 44, no. 3, pp. 562–579, 2022.

[6] Y. A. Yazdi, H. T. Shandiz, and H. G. Narm, "Automatic oscillations detection and classification of control loop using generalized machine learning algorithms," *Transactions of the Institute of Measurement and Control*, vol. 45, no. 3, pp. 476–491, 2023.

[7] M. Jelali and B. Huang, *Detection and Diagnosis of Stiction in Control Loops*. London, UK: Springer London, 2010.

[8] Z. Guo, L. Xie, T. Ye, and A. Horch, "Online detection of time-variant oscillations based on improved ITD," *Control Engineering Practice*, vol. 32, pp. 64–72, 2014.

[9] Z. Guo, J. Shen, L. Xie, X. Chen, and H. Su, "Automatic detection of multiple oscillations by wavelet analysis," *Computers & Electrical Engineering,* vol. 40, no. 7, pp. 2167–2177, 2014.

[10] L. Xie, X. Lang, A. Horch, and Y. Yang, "Online oscillation detection in the presence of signal intermittency," *Control Engineering Practice*, vol. 55, pp. 91–100, 2016.

[11] M. F. Ullah, L. Das, S. Parmar, R. Rengaswamy, and B. Srinivasan, "On developing a framework for detection of oscillations in data," ISA Transaction, vol. 89, pp. 96–112, 2019.

[12] J. W. V. Dambros, J. O. Trierweiler, and M. Farenzena, "Oscillation detection in process industries-Part I: Review of the detection methods," *Journal of* Process Control, vol. 78, pp. 108–123, 2019.

[13] M. Jelali, Control Performance Management in Industrial Automation. London, UK: Springer London, 2013.

[14] R. Rai, M. K. Tiwari, D. Ivanov, and A. Dolgui, "Machine learning in manufacturing and industry 4.0 applications," *International Journal of Production Research,* vol. 59, no. 16, pp. 4773–4778, 2021.

[15] Z. Ge, Z. Song, S. X. Ding, and B. Huang, "Data mining and analytics in the process industry: the role of machine learning," *IEEE Access*, vol. 5, pp. 20590–20616, 2017.

[16] J. W. V. Dambros, J. O. Trierweiler, M. Farenzena, and M. Kloft, "Oscillation detection in process industries by a machine learning-based approach," *Industrial & Engineering Chemistry Research*, vol. 58, no. 31, pp. 14180–14192, 2019.

[17] H. Wu and J. Zhao, "Deep convolutional neural network model based chemical process fault diagnosis," *Computers & Chemical Engineering*, vol. 115, pp. 185–197, 2018.

[18] S. Sharma, V. Kumar, and K. P. S. Rana, "Machine Learning Application for Oscillation Detection in Control Loops," in *Soft Computing: Theories and Applications*, M. Pant, T. K. Sharma, O. P. Verma, R. Singla, and A. Sikander, Eds. Singapore: Springer Singapore, 2020, pp. 1067–1075.

[19] Y. Yao, J. Zhang, W. Luo, and Y. Dai, "A Hybrid intelligent fault diagnosis strategy for chemical processes based on penalty iterative optimization," *Processes*, vol. 9, no. 8, pp. 1266, 2021.

[20] D. F. Rabba, A. N. I. Wardana, and N. Effendy, "Intermittent oscillation diagnosis in a control loop using extreme gradient boosting," *Jurnal Nasional Teknik Elektro*, vol. 11, no. 3, 2022.

[21] N. T. Tsanya, A. N. I. Wardana, and N. Effendy, "Deteksi osilasi kontrol pada proses industri dengan menggunakan jaringan saraf tiruan - review," *Journal Electrics, Electronics, Communications, Controls, Informatics, Systems* vol. 16, no. 3, pp. 71–78, 2022.

[22] W. Xueli, J. Zhiyong, and Y. Dahai, "An Improved knn algorithm based on kernel methods and attribute reduction," in *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control*, 2015, pp. 567–570.

[23] Z. Ge, "Semi-Supervised Data Modeling and Analytics in The Process Industry: Current Research Status and Challenges," *IFAC Journal of Systems & Control*, vol. 16, pp. 100150, 2021.

[24] N. Z. Ibrahim, "Detection of Intermittent Oscillation in Tennessee Eastman Process Control Loop using KNN-Based Semi-Supervised Learning Method," Undergraduate Final Project, Universitas Gadjah Mada, Yogyakarta, 2022.

[25] J. J. Downs and E. F. Vogel, "A Plant-wide industrial process control problem," *Computers & Chemical Engineering*, vol. 17, no. 3, pp. 245–255, 1993.

[26] Y. Y. S. Henry, C. Aldrich, and H. Zabiri, "Detection and severity identification of control valve stiction in industrial loops using integrated partially retrained CNN-PCA frameworks," *Chemometrics and Intelligent Laboratory Systems*, vol. 206, pp. 104143, 2020.

[27] M. Barandas et al., "TSFEL: time series feature extraction library," *SoftwareX*, vol. 11, pp. 100456, 2020.

[28] J. Tanha, M. van Someren, and H. Afsarmanesh, "Semi-supervised self-training for decision tree classifiers," *International Journal of Machine Learning and Cybernetic*, vol. 8, no. 1, pp. 355–370, 2017.