UDC 004.93 **doi: 10.26906/SUNZ.2025.1.77-82**

Vladyslav Huts, Oleksii Gorokhovatskyi

Simon Kuznets Kharkiv National University of Economics, Kharkiv, Ukraine

EFFICIENT FAULT DETECTION IN INDUSTRIAL EQUIPMENT USING PCA AND SMOTE ENHANCED NEURAL NETWORKS

Abstract. This research addresses the challenge of fault detection in industrial equipment using high-dimensional vibration data with limited labeled examples. The goal was to develop a neural network model capable of accurately classifying measurement vectors into normal and faulty categories. The dataset consisted of 1158 samples, each with 93,752 numerical features, representing two classes: 865 normal and 293 faulty instances. A comprehensive preprocessing pipeline was employed, including standardization, dimensionality reduction using Principal Component Analysis (PCA), and Synthetic Minority Over-sampling Technique (SMOTE) for class balancing. The developed neural network achieved a baseline accuracy of 94.40% with 100 PCA components. Further experiments demonstrated that reducing the architecture and using only 50 PCA components improved accuracy to 98.81%, highlighting the effectiveness of the proposed approach. These findings emphasize the utility of combining PCA, SMOTE, and neural networks for fault detection in industrial equipment in high-dimensional, imbalanced datasets. Future research directions include exploring advanced neural network architectures, investigating the impact of PCA component count on model performance, and studying the feasibility of training effective models on synthetic data.

Keywords: fault detection; neural networks; PCA; SMOTE; dimensionality reduction.

Introduction

Ensuring the reliability of mechanical systems is a critical task in modern industrial environments. Equipment failures can cause severe disruptions, leading to costly repairs, operational delays, and, in some cases, serious safety risks. As a result, early detection of faults becomes an essential component of preventive maintenance strategies. Machine learning (ML) became a key tool in automating the fault detection process, enabling fast and accurate identification of potential issues. However, many traditional ML models require large labeled datasets, which are often difficult to obtain in real-world scenarios.

The object of study in this paper is the process of building the neural network (NN) model to classify the probable faults of equipment through the analysis of vibration data. Building and training of the model requires time and computational resources as well as the construction of the effective architecture of the model that could solve the problem. Additionally, this process is data-dependent and should utilize methods to address the unequal distribution of classes and data scarcity.

The goal of the work is to develop a robust fault detection model that can accurately identify both nominal and faulty instances in the equipment's operational data.

Related work

Fault detection in complex systems is a critical area of research, particularly in industries where early identification of equipment malfunction is essential for preventing costly breakdowns and ensuring operational safety. Traditional machine learning techniques are commonly applied in this field, yet they often face significant challenges, such as handling high-dimensional data and imbalanced class distributions. The dataset used in this study contains 1158 files, each containing over 93,000 features—exemplifies a such scenario in industrial monitoring, where large amounts of sensor data are collected for fault detection.

High-dimensional data, such as the dataset representatives in this research, present several computational challenges, including overfitting and increased computational costs. Moreover, when working with neural networks, the risk of the “curse of dimensionality” arises, which can affect model performance by diluting the significance of individual features. To address this, principal component analysis (PCA) is commonly used as a dimensionality reduction technique. PCA transforms the high-dimensional data into a lower-dimensional space by extracting the most important features, or principal components, that explain the largest amount of variance in the dataset. PCA has been widely validated in the literature as an effective way to reduce computational complexity while retaining the most relevant information, enabling more efficient training of machine learning models, including neural networks [1 – 3].

The unequal class distribution is a common issue in fault detection problems, as there are significantly fewer instances of faulty data (class 1) compared to normal operation data (class 0). This distribution can skew model predictions, making it more likely to favor the majority class (class 0). The synthetic minority over-sampling technique (SMOTE) [2] was employed to address this. SMOTE generates synthetic samples from the minority class by interpolating between existing examples, effectively balancing the dataset. By providing a balanced dataset, SMOTE helps ensure that the model is able to detect faults as effectively as it recognizes normal operating conditions. Previous studies have demonstrated the usefulness of SMOTE in addressing class imbalance, especially in industrial fault detection scenarios where faulty conditions are rare [4, 5].

The implementation of neural network model used in this study was built using the Keras framework [6, 7]. The model architecture consists of three fully connected layers, each followed by Leaky ReLU activation functions. Leaky ReLU was chosen over the standard ReLU to prevent the issue of "dying neurons," where neurons stop updating during training. Dropout layers were applied after each dense layer to prevent overfitting by randomly deactivating a portion of neurons during training. Dropout is a well-known regularization technique used to improve model generalization by preventing overfitting to the training data [8, 9].

The final layer of the network uses a sigmoid activation function, which is suitable for binary classification problems, distinguishing between nominal (non-faulty) and faulty data. Adam optimizer [10] was used which in an adaptive learning rate optimization algorithm that has been shown to achieve better performance on deep learning tasks due to its ability to adjust learning rates for each parameter dynamically. Binary cross-entropy was used as the loss function, which is standard for binary classification tasks. Additionally, early stopping was employed during training to prevent overfitting.

While other techniques such as prototype selection and construction methods have been applied to reduce dataset size and improve model performance, they are often computationally expensive and risk discarding important information. Prototype selection methods focus on identifying the most relevant instances from the dataset, whereas prototype construction generates new instances to represent the data more effectively. However, these approaches often face limitations in terms of processing time and accuracy, especially in high-dimensional datasets like the one used in this research. Instead, the combination of PCA and SMOTE provides a more efficient solution for managing the large dataset while retaining critical information necessary for fault detection [11, 12].

Prototype selection methods focus on identifying a representative subset of data points from the entire dataset. These methods aim to choose the most relevant instances avoiding redundancy and noise. However, in fault detection tasks, where there is a significant difference in the number of samples between classes (with fewer faulty samples compared to normal ones), traditional prototype selection techniques often face challenges. While these methods can help reduce dimensionality, they might not effectively address the issue of unequal class distribution. In this paper, the SMOTE was applied to mitigate this by generating synthetic samples, ensuring that the fault detection neural network model receives a balanced set of examples for training.

Conversely, prototype construction methods [12] focus on generating new samples that represent the underlying structure of the data. These methods employ clustering techniques or neural networks to create synthetic prototypes. While prototype construction can address some issues with the unequal distribution of classes, it may also introduce complexity and noise. PCA was utilized in this paper to perform dimensionality reduction, effectively capturing the main variance of the dataset while eliminating redundancy. This preprocessing step enhanced memory usage (and, probably, model efficiency) by reducing the dimensionality of the vector from over 93,000 dimensions to 100.

Both prototype selection and construction methods come with trade-offs, especially in high-dimensional datasets like this. A significant drawback of these methods is the high computational cost involved in processing involved in large datasets processing the potential introduction of noise through synthetic sample generation. For our case, the leveraging PCA helped alleviate these challenges by preserving the dataset's core features, while SMOTE effectively handled class imbalance without the need for complex prototype construction methods.

The model building pipeline

In this paper, the traditional prototype selection and construction techniques were not directly implemented, as SMOTE and PCA provided a more efficient solution for high-dimensional fault detection datasets with unequal class distribution. This combined approach significantly improved the model's ability to generalize, effectively addressing both dimensionality and the challenge of having fewer faulty data samples (class 1) compared to normal data (class 0).

Data Preprocessing. To prepare the data for analysis, the first step involved the standardizing dataset. This was accomplished using the StandardScaler from the scikit-learn library, which transforms the data so that each feature has a mean of zero and a standard deviation of one. The standardization formula can be expressed as (1):

$$x' = (x - \varphi)/\sigma, \quad (1)$$

where x' is the standardized value of feature x after normalization, μ – mean of the feature values, σ – standard deviation of the feature values.

Dimensionality Reduction. Given the high dimensionality of the vibration data, PCA was applied to reduce the number of features. It helps to retain the most significant variance in the data while minimizing the number of dimensions. The transformation can be described by the following equation (2):

$$Z = XW, \quad (2)$$

where Z is matrix representation of data after dimensionality reduction, X is the original feature matrix, W – weight matrix for the transformation in dimensionality reduction.

Data Balancing. Data Balancing. SMOTE generates synthetic examples for the minority class to create a more balanced dataset. The new synthetic instances are created by interpolating between existing instances of the minority class (3):

$$x_{new} = x_i + \lambda(x_j - x_i), \quad (3)$$

where x_{new} is new synthetic instance generated by SMOTE, x_i is the existing instance from the minority class, x_j is the other instance from the minority class for interpolation, and λ is the interpolation factor for generating synthetic instances.

Neural Network Architecture. Three hidden layers utilize the LeakyReLU activation function, which allows small gradients for inactive neurons. The LeakyReLU activation can be expressed as (4):

$$LeakyReLU(x) = \begin{cases} x \; if \; x > 0; \\ ax \; if \; x \leq 0, \end{cases} \quad (4)$$

where *a* is the small constant slope, typically set to 0.1.

Dropout layers are applied after each hidden layer to mitigate overfitting by randomly setting 25% of the neurons to zero during training. The output layer consists of a single neuron with a sigmoid activation function for binary classification (5):

$$y = \frac{1}{1+e^{-z}}, \quad (5)$$

where *z* is the input to the sigmoid function, and y is the predicted output of the model after applying the sigmoid function.

The model was trained using the Adam optimizer and binary cross-entropy as the loss function (6):

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\begin{pmatrix} y_i log(\hat{y}_i) + \\ +(1+y_i) log(1-\hat{y}_1) \end{pmatrix}, \quad (6)$$

where L is the loss function, calculated using binary cross-entropy, N is the total number of instances in the dataset, y_i is the actual class label for the i-th instance, and $\hat{y}_i$ is the predicted probability for the i-th instance, representing the model's confidence that the instance belongs to the positive class.

Early stopping was implemented to monitor the validation loss and prevent overfitting.

The performance of the model was rigorously evaluated on the test dataset, focusing on metrics such as accuracy and loss. The training process was monitored across a specified number of epochs with a batch size of 32, ensuring efficient use of computational resources while tracking the total training time.

Data preprocessing

In this paper, various methods were employed to develop an effective fault detection model. The primaкy focus was on the use of vibration data collected from mechanical systems.

To ensure uniformity across the dataset, the data was standardized using the StandardScaler from the sklearn.preprocessing module. This normalization is crucial for optimizing the performance of the machine learning model. Given the large number of features, PCA was applied to reduce the dimensionality of the dataset. The number of principal components was set to 100, significantly reducing computational complexity while retaining essential variance in the data.

The dataset was then split into training and testing sets using train_test_split from sklearn.model_selection, with 80% allocated for training and 20% for testing. To address class imbalance—where faulty samples were significantly fewer than nominal ones—the SMOTE was utilized to generate synthetic samples for the minority class. This step was essential to ensure the model trained effectively on a balanced dataset.

The architecture of the neural network includes three layers. The first one contained 512 units, the second layer had 256 units, and the third layer included 128 units. Each layer employed the LeakyReLU activation function to prevent inactive neurons, with a dropout rate of 0.25 applied to each dense layer to mitigate overfitting. The final layer utilized a sigmoid activation function for binary classification, distinguishing between nominal and faulty data.

The model was compiled using the Adam optimizer with a learning rate of 0.0003, and binary cross-entropy was used as the loss function. Early stopping procedure was implemented to prevent overfitting, halting training if the validation loss did not improve after two consecutive epochs. The model's performance was evaluated on the test set, yielding accuracy and loss metrics to assess its effectiveness in fault detection.

Artificial neural network modeling

Experiments were conducted on a computer with an Intel Core i5-10500H processor and 16 GB of RAM. The Python 3.8 programming environment in Jupyter was utilized, with Keras 2.4 and scikit-learn 0.24 versions, providing the necessary performance for deep learning tasks.

The dataset consists of 1158 files with vibration measurements, each containing high-dimensional information with 93752 features that needed to be processed and analyzed. There are 865 vectors representing class 0 and 293 for class 1. Each file was loaded into Python using the numpy and glob libraries, resulting in a two-dimensional array where each row represented a sample and each column corresponded to a feature. This problem was introduced by Eric Bechhoefer in the scope of Ukrainian hackathon of scientific works of young scientists in the field of intellectual information technologies [13]. The initial dataset is available in [14].

The neural network was trained over 50 epochs, with early stopping implemented to prevent overfitting. The training process recorded a significant increase in accuracy, starting from 73.48% and reaching a maximum accuracy of 100% by the 11th epoch. The loss during training decreased rapidly, beginning at 7.51 and converging close to zero. The model's performance was evaluated on the test dataset, achieving an accuracy of 94.40% and a loss of 0.2953.

The loss and accuracy curves indicate that the training and validation losses steadily decreased, while the training accuracy significantly increased, approaching 1.0. To assess the impact of the number of PCA components on model performance, tests were conducted with different values of n_components (200, 300, 400). The results showed that as the number of components increased, the model's accuracy on the test data decreased. Specifically, with n_components = 200, the model achieved an accuracy of 90.95%, while for n_components = 300, accuracy dropped to 89.66%. Further increasing the number of components to 400 resulted in a test accuracy of 84.91%. At the same time, the training accuracy remained high across all variations. Based on these results, it was decided to retain 100 components as the optimal number, as it provided the best balance between test accuracy (94.40%) and computational complexity.

Results

The conducted experiments explored the impact of different neural network configurations on fault detection accuracy. The primary goal was to determine how changes in architecture, activation functions, and the number of PCA components affect the model's accuracy and loss values. The results of experiments are shown in Table 1.

The baseline model consisted of three layers three layers with 512, 256, and 128 neurons, used the LeakyReLU activation function and 100 PCA components. It achieved a test accuracy of 94.40% with a loss value of 0.3125. This configuration served as a reference point for comparing the results of subsequent experiments. Replacing the LeakyReLU activation with regular ReLU neurons keeping the other parameters unchanged resulted in a slight improvement in test accuracy, reaching 94.83%, although the loss value increased to 0.3662. This suggests that while ReLU can improve performance, it may also introduce greater fluctuations during training.

A configuration with fewer neurons (256, 128, and 64 neurons across three layers) led to the highest test accuracy of 95.69%. The loss value for this experiment was 0.3201, suggesting that although the model was effective, it might have been overly specialized to the training data.

Increasing the number of neurons and adding a fourth layer (1024, 512, 256, 128 neurons) resulted in a slight improvement in test accuracy to 94.83%, with a lower loss value of 0.2952. This configuration helped the model generalize slightly better, as indicated by the reduced loss, though the overall accuracy did not significantly increase. Training accuracy remained at 98.63%, matching the baseline. Reducing the network to two layers (512, 256 neurons) allowed us to reach 98.81% test accuracy with a loss value of 0.3392. This shows that a simple architecture can still perform well, potentially benefiting from the reduced complexity.

Reducing the number of PCA components to 50 showed significant improvement. This model achieved a test accuracy of 98.71% with a very low loss of 0.1225, which suggests that reducing the dimensionality helped to filter out noise and preserve critical patterns.

On the other hand, increasing the number of PCA components to 150 led to a drop in test accuracy to 92.24%, with the loss value rising to 0.5240. This suggests that adding more components introduced unnecessary complexity, making it harder for the model to learn effectively.

Finally, increasing the PCA components further to 200 reduced test accuracy to 91.81% with a loss of 0.5021. The training accuracy was 98.63%, consistent with the baseline. However, the increase in complexity from the additional components did not improve performance and, in fact, resulted in poorer generalization. Probably, it is required to add more neurons/layers in the neural network to achieve better results for 150 or 200 PCA components.

Overall, these experiments show that finding the optimal configuration of neural network architecture and PCA components is critical for achieving high performance. In particular, the reduction to 50 PCA components proved to be highly effective, significantly improving the model's ability to generalize, as evidenced by the low test loss and high accuracy. Meanwhile, the simpler architecture with two layers also demonstrated that reduced complexity can lead to better performance, highlighting that a more complex model is not always the better option. The baseline architecture of the neural network used in this study is shown in Fig. 1.

Table 1 – **Model Performance Across Different Configurations**

Exp.	Parameters (Number of Neurons, Layers)	Activation function	Number of PCA Components	Test Accuracy (%)
1	512, 256, 128 (3 layers)	LeakyReLU	100	94.40
2	512, 256, 128 (3 layers)	ReLU	100	94.83
3	256, 128, 64 (3 layers)	LeakyReLU	100	95.69
4	1024, 512, 256, 128 (4 layers)	LeakyReLU	100	94.83
5	512, 256 (2 layers)	LeakyReLU	100	98.81
6	512, 256, 128 (3 layers)	LeakyReLU	50	98.71
7	512, 256, 128 (3 layers)	LeakyReLU	150	92.24
8	512, 256, 128 (3 layers)	LeakyReLU	200	91.81

Fig. 1. Architecture of the neural network used for fault detection

It consists of an input layer that processes 100 principal components, followed by three dense layers with 512, 256, and 128 neurons respectively, each using LeakyReLU activation and a dropout rate of 0.25. The

final output layer utilizes a sigmoid activation for binary classification.

The training process of this network, as shown in Fig. 2 while Fig. 3, demonstrates a smooth convergence of both training and validation loss, indicating effective learning without overfitting. Additionally, the accuracy plot reflects that the model achieves high accuracy on both training and validation data, confirming the robustness of the neural network.

Discussion

The results from the experiments highlight several interesting insights into optimizing neural network models for fault detection, particularly when dealing with high-dimensional data and imbalanced datasets. By systematically varying the network architecture, activation functions, and the number of PCA components, we were able to identify configurations that improved the model's performance, as well as those that led to less effective results. Below, we discuss the implications of these findings.

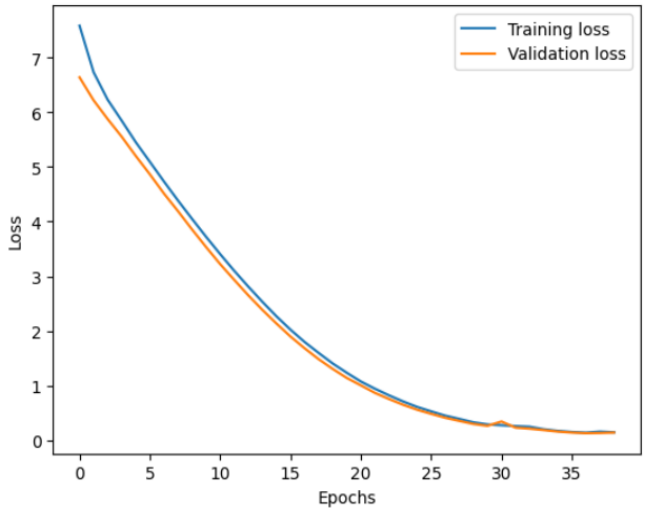

Fig. 2. Training and validation loss over epochs

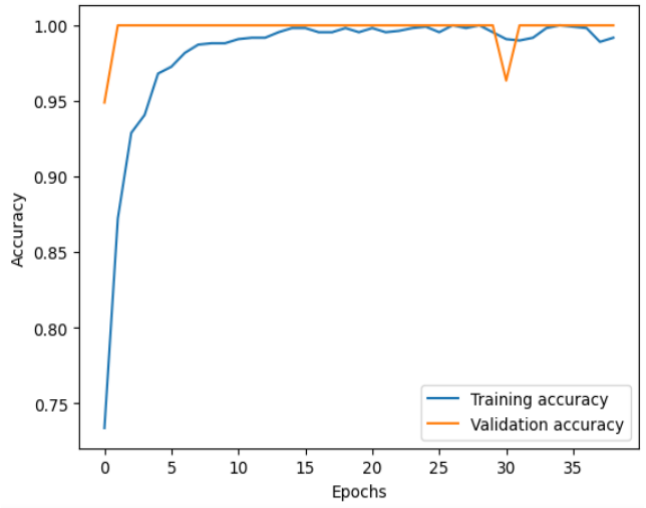

Fig. 3. Training and validation accuracy over epochs

One of the most significant observations was the effect of dimensionality reduction through PCA. Reducing the number of components to 50 significantly improved the model's performance, leading to a test accuracy of 98.71%. This suggests that removing excess dimensions helped the model focus on the most critical features, effectively filtering out noise and reducing overfitting.

This configuration not only the accuracy but also streamlined the learning process, allowing the model to generalize better. Conversely, increasing the number of PCA components to 150 and 200 degraded the performance, with test accuracies dropping to 92.24% and 91.81%, respectively.

These results underscore the importance of finding the right balance in dimensionality; too many components can lead to unnecessary complexity and noise, hindering the model's ability to generalize. This also shows that the high-dimensional vector (with more than 93 000 dimensions) could be effectively reduced for this problem.

The experiments with different activation functions showed that replacing LeakyReLU with ReLU led to a slight increase in test accuracy to 94.83% but also increased the loss to 0.3662, suggesting a potential trade-off between stability and performance. While ReLU may boost performance under certain conditions, it can introduce more fluctuations during training, which may lead to less consistent learning outcomes. On the other hand, LeakyReLU proved to be more stable across various configurations, as reflected in the baseline model's performance.

The analysis of neural network architecture also provided valuable insights. Reducing the model to two layers with 512 and 256 neurons produced a high test accuracy of 98.81% with a loss of 0.3392. This finding suggests that simpler architectures can still achieve strong performance, particularly when the data are well-preprocessed.

Simpler models often have fewer parameters, which can help prevent overfitting and make them more efficient.

In contrast, adding complexity by increasing the network to four layers with 1024, 512, 256, and 128 neurons did not yield a significant improvement, maintaining a test accuracy of 94.83% but slightly reducing the loss to 0.2952.

This indicates that additional layers and neurons do not always lead to better generalization, and may instead complicate the learning process without adding real value.

Furthermore, the experiment that reduced the number of neurons in each layer to 256, 128, and 64 achieved high test accuracy of 95.69%.

Overall, the experiments demonstrate that simplicity and balance are the keys to optimizing model performance. Effective dimensionality reduction, as seen with 50 PCA components, can streamline the learning process and reduce the risk of overfitting by focusing the model on the most relevant features. Similarly, the success of the two-layer architecture suggests that complex, deep networks are not always necessary for high performance, especially when the data are processed effectively.

These findings highlight the importance of data preprocessing and careful architecture selection in building models that are not only accurate but also efficient and robust.

Conclusions

The paper addresses the problem of detecting equipment faults through approach that uses both machine learning techniques and advanced data processing strategies.

The contribution of this research includes the entire machine learning pipeline of using data preprocessing methods like PCA, SMOTE to address the classification problem presented with imbalanced dataset.

The practical significance of this research provides the machine learning model effective automatic tool for predictive maintenance of the equipment.

The prospects for further research may include but are not limited with the search for more effective architecture of the neural network, deeper research of the dependency of accuracy on the quantity of elements in PCA analysis, finally, it would be interesting to investigate the possibility to train the effective model only using synthetic data.

REFERENCES

1. B. Krawczyk, "Learning from imbalanced data: Open challenges and future directions," Progress in Artificial Intelligence, vol. 5, no. 4, Apr. 2016. doi:10.1007/s13748-016-0094-0.
2. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, Jun. 2002. doi: 10.1613/jair.953.
3. Y. Zhang, and S. Wang, "Deep learning for time series forecasting: A survey," Big Data, vol. 9, no. 1, Dec. 2020. doi: 10.1089/big.2020.0159.
4. X. Chen, D. Liu, and H. Zha, "Predictive data mining techniques for fault diagnosis of electric equipment: A review," Applied Sciences, vol. 10, no. 3, p. 950, Feb. 2020. doi: 10.3390/app10030950.
5. B. S. Panigrahi, T. T, M. Tamilselvi, S. B. G. Tilak Babu, P. G and B. Shaik, "Deep Learning Techniques for Fault Detection in Industrial Machinery," 2024 5th International Conference on Recent Trends in Computer Science and Technology (ICRTCST), Jamshedpur, India, 2024, pp. 221-226, doi: 10.1109/ICRTCST61793.2024.10578499.
6. Y. Zhang, et al., "Deep learning-based intelligent fault diagnosis methods toward rotating machinery," IEEE Access, vol. 8, pp. 9335–9346, 2019. doi: 10.1109/ACCESS.2019.2963092.
7. O. Matania, E. Bechhoefer, and J. Bortman, "Digital Twin of a Gear Root Crack Prognosis," Sensors, vol. 23, no. 24, p. 9883, Dec. 2023. doi: 10.3390/s23249883.
8. O. Matania, R. Cohen, E. Bechhoefer, and J. Bortman, "Anomaly Detection and Remaining Useful Life Estimation for the Health and Usage Monitoring Systems 2023 Data Challenge," PHM Society Proceedings, vol. 8, no. 1, Jun. 2024. doi: 10.36001/phme.2024.v8i1.4125.
9. O. Matania, E. Bechhoefer, D. Blunt, W. Wang, and J. Bortman, "Anomaly Detection and Remaining Useful Life Estimation for the Health and Usage Monitoring Systems 2023 Data Challenge," Sensors, vol. 24, no. 13, p. 4258, Jun. 2024. doi: 10.3390/s24134258.
10. O. Matania, L. Bachar, E. Bechhoefer, and J. Bortman, "Signal Processing for the Condition-Based Maintenance of Rotating Machines via Vibration Analysis: A Tutorial," Sensors, vol. 24, no. 2, p. 454, Jan. 2024. doi: 10.3390/s24020454.
11. T. Chistiakova, J. Zambrano, and B. Carlsson, “Application of machine learning methods for fault detection in wastewater treatment plants,”, in Reglermöte, 2014. doi: 10.13140/2.1.3733.4403.
12. S. Windmann, "Data-Driven Fault Detection in Industrial Batch Processes Based on a Stochastic Hybrid Process Model," in IEEE Transactions on Automation Science and Engineering, vol. 19, no. 4, pp. 3888-3902, Oct. 2022, doi: 10.1109/TASE.2021.3138925.
13. Ukrainian hackathon of scientific works of young scientists in the field of intellectual information technologies URL: https://zp.edu.ua/vkiit
14. The dataset. URL: https://pz.zp.ua/files/vkiit/zeroShot_vkiit.rar

Received (Надійшла) 30.11.2024
Accepted for publication (Прийнята до друку) 05.02.2025

Ефективне виявлення відмов у промисловому обладнанні з використанням PCA та покращених нейронних мереж на основі SMOTE

В. В. Гуць, О. В. Гороховатський

Анотація. У цьому дослідженні розглядається проблема виявлення відмов у промисловому обладнанні за допомогою високовимірних даних вібрації з обмеженою кількістю мічених прикладів. Метою було розробити модель нейронної мережі, здатну точно класифікувати вектори вимірювань на нормальні та несправні категорії. Набір даних складався з 1158 зразків, кожен із яких містив 93,752 числові ознаки, що представляли два класи: 865 нормальних та 293 несправних випадки. Було використано комплексний конвеєр попередньої обробки, який включав стандартизацію, зменшення розмірності за допомогою методу головних компонент (PCA) і техніку синтетичного збільшення меншості (SMOTE) для балансування класів. Розроблена нейронна мережа досягла базової точності 94,40% із 100 компонентами PCA. Подальші експерименти показали, що зменшення розмірності архітектури та використання лише 50 компонентів PCA підвищило точність до 98,81%, підкреслюючи ефективність запропонованого підходу. Ці результати акцентують увагу на корисності комбінування PCA, SMOTE та нейронних мереж для виявлення відмов у високовимірних незбалансованих наборах даних. Майбутні напрями досліджень включають вивчення передових архітектур нейронних мереж, аналіз впливу кількості компонентів PCA на продуктивність моделі та дослідження можливості навчання ефективних моделей на синтетичних даних.

Ключові слова: виявлення відмов; нейронні мережі; PCA; SMOTE; зменшення розмірності.