DOI: 10.2478/sbeef-2025-0023

DEEP NEURAL NETWORK-BASED FAULT DIAGNOSIS FOR PREDICTIVE MAINTENANCE OF INDUCTION MOTORS

IONUT-CATALIN MUNTEANU, EMIL CAZACU, MARILENA STANCULESCU
National University of Science and Technology POLITEHNICA Bucharest, Romania,
Faculty of Electrical Engineering,
313, Splaiul Independentei, 060042, district 6, Bucharest ROMANIA,
E-mail: ionut.munteanu2409@stud.electro.upb.ro, emil.cazacu@upb.ro, marilena.stanculescu@upb.ro

Abstract. ***Predictive maintenance for electric drives increasingly leverages data-driven diagnostics to detect early-stage faults under varying operating conditions. Building on the analysis and simulation of induction motor behavior under fault conditions in industrial electric drive systems, this paper presents the design, training, and validation of a deep convolutional neural network (CNN) capable of automatically identifying multiple fault types. The CNN operates on a 2D time–feature representation of multivariate signals and is trained using Adam optimization with a 70/30 train–holdout split. Results show 92.31% accuracy on the holdout set, with a confusion matrix indicating minimal inter-class confusion. For deployment, the trained model together with the normalization parameters is integrated into a masked Simulink block to enable automated end-to-end inference on new simulation runs.***

Keywords**:** predictive maintenance; induction motor; fault diagnosis; deep learning; convolutional neural networks; MATLAB/Simulink.

1. INTRODUCTION

Unplanned downtime of industrial electric drives leads to production losses and increased maintenance costs. Predictive maintenance aims to detect early degradation and schedule interventions based on condition indicators rather than fixed intervals [1]–[3]. For induction motors, electrical signature analysis and vibration-based monitoring are among the most used approaches [4]–[6]. However, traditional pipelines often require hand-crafted features and expert tuning, which can limit portability across motor types and operating regimes.

Deep learning models can learn discriminative representations directly from raw or minimally processed signals, reducing the need for manual feature engineering [7]–[9]. Convolutional neural networks (CNNs) can be adapted to multivariate time series by treating synchronized signals as structured arrays and learning local time–feature patterns. This paper presents a simulation-to-implementation workflow in MATLAB/Simulink: a CNN is trained on signals generated by the Simulink induction-motor model and then integrated into Simulink as a masked block for automated fault classification.

Contributions

- A simulation-driven dataset generation method using a MATLAB/Simulink induction-motor model under normal and faulty operating conditions.
- A reproducible pre-processing pipeline (grouping by simulation, temporal resampling, and Z-score normalization) suitable for deep learning input standardization.
- A compact CNN architecture for multi-class fault diagnosis and an experimental evaluation achieving 92.31% validation accuracy.
- A deployment-oriented integration in Simulink via a masked block that performs automated data extraction, normalization, and classification.

2. RELATED WORK

Motor Current Signature Analysis (MCSA) is widely used for condition monitoring due to its non-intrusive nature and compatibility with existing drive instrumentation [4]. Electrical signature analysis has been applied to predictive maintenance with both analytical and data-driven approaches [5], [10]. Complementary signatures such as stray flux measurements can support rotor fault detection and can be fused with current-based methods [11, 12]. Recent open datasets combining vibration, current, and other modalities support benchmarking of learning-based methods [12].

Machine learning for predictive maintenance spans classical classifiers (e.g., tree-based methods) [13] to deep architectures such as CNNs, recurrent neural networks (RNNs), and CNN–LSTM hybrids [9]. While sequential models can capture long-range temporal dependencies, CNNs often provide strong accuracy with lower computational cost and stable training for many diagnostic tasks [9]. Operational variability (load changes, voltage unbalance, harmonics) can significantly affect observed signatures, motivating robustness testing aligned with power-quality and machine standards [14]–[18].

3. DATA GENERATION AND PRE-PROCESSING

Simulation-driven dataset and file structure

A MATLAB/Simulink model of an induction motor drive is used to generate multivariate time-series signals for multiple operating states. For reproducibility, simulation outputs are exported to two tabular files: (i) an input file that stores time-indexed signals together with a simulation_id, and (ii) an output file that stores the operating_state label for each simulation. This structure enables grouping all time samples belonging to the same run and creates a one-label-per-simulation learning problem.

Each simulation produces synchronized measurements including rotor currents (*Ir_a, Ir_b, Ir_c*), stator currents (*Is_a, Is_b, Is_c*), input power *P_in*, rotor speed ω, electromagnetic torque *T_e*, and rated torque *T_rated*. Seven classes are considered: normal operation and six fault categories (open-circuit, phase-to-ground, phase-to-phase, under-voltage, over-voltage, and harmonics) (Table I).

Table 1. Signals and Operating Classes Used for CNN Input

Category	Description
Electrical signals	*Ir_a, Ir_b, Ir_c; Is_a, Is_b, Is_c; P_in*
Mechanical signals	Rotor speed ω; electromagnetic torque *T_e*; *T_rated*
Classes (7)	Normal; Open-circuit; Phase-to-ground; Phase-to-phase; Under-voltage; Over-voltage; Harmonics

Temporal grouping and resampling

Simulations may produce sequences with different numbers of time samples; therefore, signals are grouped by *simulation_id* and resampled via linear interpolation to a common length *maxLen*. The resampled data are stored as a 4D tensor compatible with MATLAB.

Feature normalization

To prevent features with large magnitude from dominating gradient updates, Z-score normalization is applied per feature: $x' = (x - \mu)/\sigma$. The mean μ and standard deviation σ are computed over the concatenated training data and saved to disk so that the same normalization is applied during testing and deployment.

Fault injection and operating scenarios

The seven operating classes are implemented as parameterized scenarios in the Simulink model. Normal operation serves as a baseline, while fault cases modify the electrical supply and/or introduce abnormal circuit conditions to emulate common degradation modes observed in industrial drives. Because all scenarios share the same signal extraction interface, new fault types can be added by extending the scenario library and regenerating labeled simulation runs.

Voltage-related cases (under-voltage and over-voltage) are modeled by adjusting supply amplitude, and harmonic distortion is introduced by superimposing non-sinusoidal components on the supply waveform. Open-circuit and short-circuit cases are represented through switch-controlled phase interruptions or phase coupling paths. This digital-twin approach enables systematic exploration of operating regimes and supports the generation of balanced datasets without requiring destructive experiments.

4. CNN ARCHITECTURE AND TRAINING

Network design (MATLAB layer stack)

The CNN treats each resampled simulation as a 2D "signal image" of size L_max×F. Following the implementation described in the MATLAB build script (Table II), the network uses three convolutional blocks; each block includes a 2D convolution layer, batch normalization, a ReLU nonlinearity, and max pooling. Convolutional filters capture local patterns in the time–feature plane associated with distinct fault signatures. A fully connected layer with 64 neurons aggregates extracted features, followed by 50% dropout to reduce overfitting, and a final softmax classifier.

Table 2. CNN Layer Configuration (from the Training Script)

Stage	Layer	Key parameters	Output (conceptual)
Input	imageInputLayer	[L_max × F × 1], no built-in normalization	L_max × F × 1
Block 1	conv + BN + ReLU + pool	Conv: [5×3], 48 filters, stride [2×1], padding same; Pool: [2×1], stride [2×1]	↓ time resolution
Block 2	conv + BN + ReLU + pool	Conv: [7×3], 96 filters, stride [2×1], padding same; Pool: [2×1], stride [2×1]	↓ time resolution
Block 3	conv + BN + ReLU + pool	Conv: [3×3], 128 filters, stride [2×1], padding same; Pool: [2×1], stride [2×1]	↓ time resolution
Classifier	FC + dropout + FC + softmax	FC1: 64; Dropout: 0.5; FC2: 7 classes	7-class probabilities

Training configuration and convergence (Table III)

Table 3. CNN Training Configuration

Parameter	Value
Optimizer	Adam
Epochs	50
Mini-batch size	16
Initial learning rate	1×10^{-4}
Train/test split	70% / 30%
Loss	Cross-entropy (softmax)

The network is trained using Adam optimization with mini-batches of 16 for 50 epochs. In early epochs the model starts near chance-level performance (around 30% accuracy) but rapidly improves after a few epochs as convolutional filters begin capturing discriminative fault patterns (Fig. 1). Training stabilizes toward the end of optimization with a reported validation accuracy of 92.31% (Fig. 2).

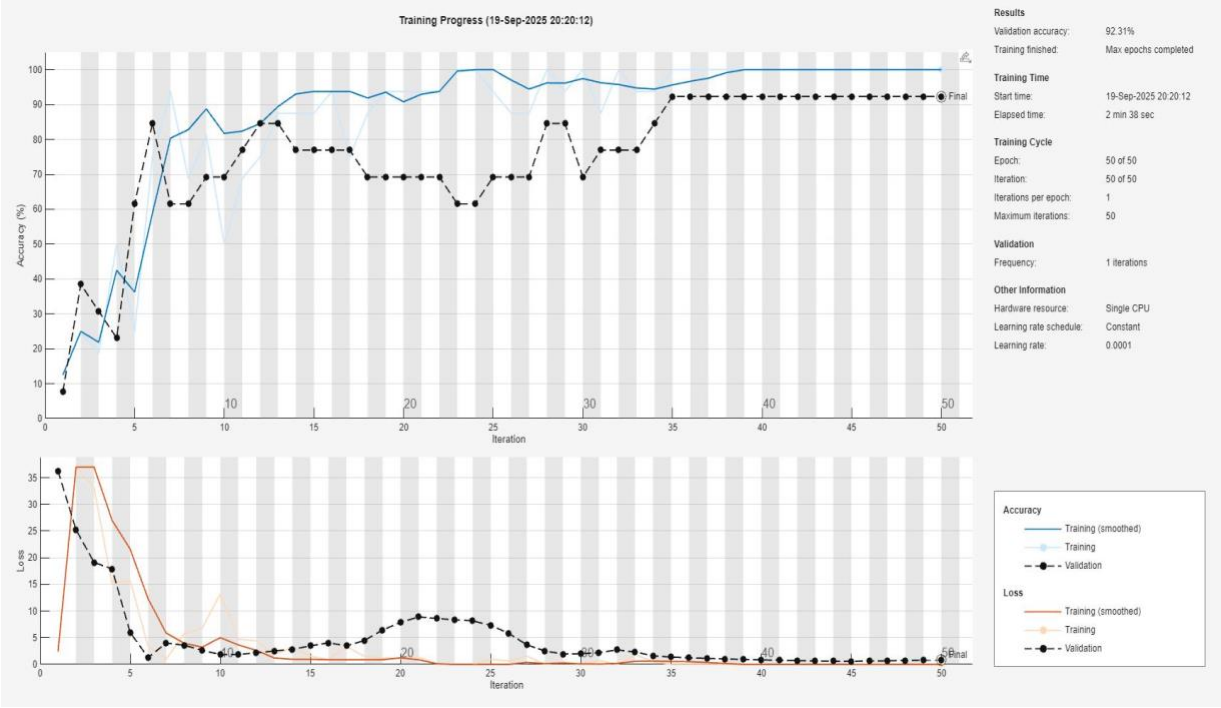

Fig. 1. CNN training progress (accuracy and loss)

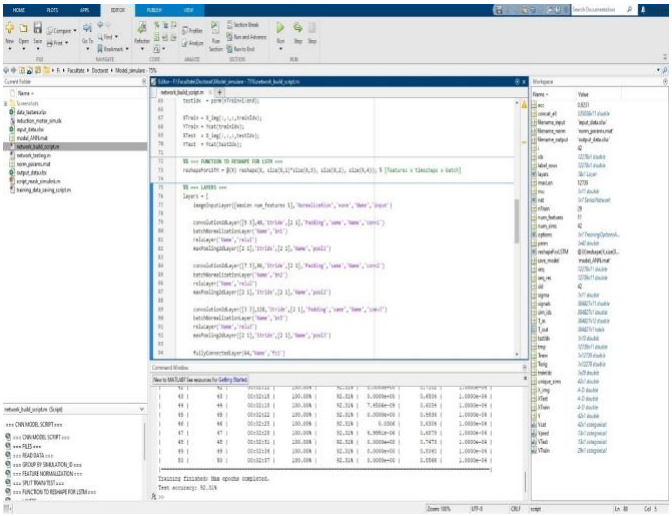

Fig. 2. End of training showing stabilized validation accuracy.

5. EXPERIMENTAL RESULTS

Class balance

Fig. 3 illustrates the distribution of classes in the training and test subsets. Maintaining comparable class frequencies reduces bias toward dominant categories and supports fair evaluation.

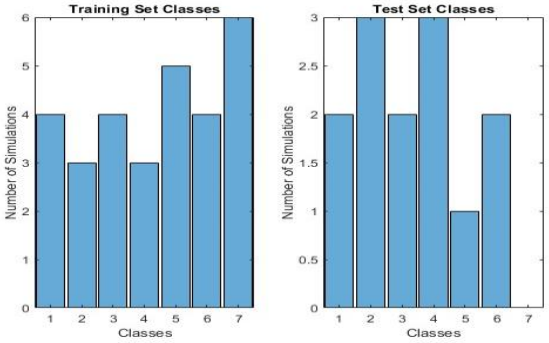

Fig. 3. Class distribution in training and test subsets.

Confusion matrix and error patterns

The confusion matrix in Fig. 4 shows that most predictions lie on the diagonal, indicating correct classification for most samples. Misclassifications appear mainly between conceptually close fault types (e.g., voltage-related categories), suggesting that additional operating scenarios or constraints could further improve separability.

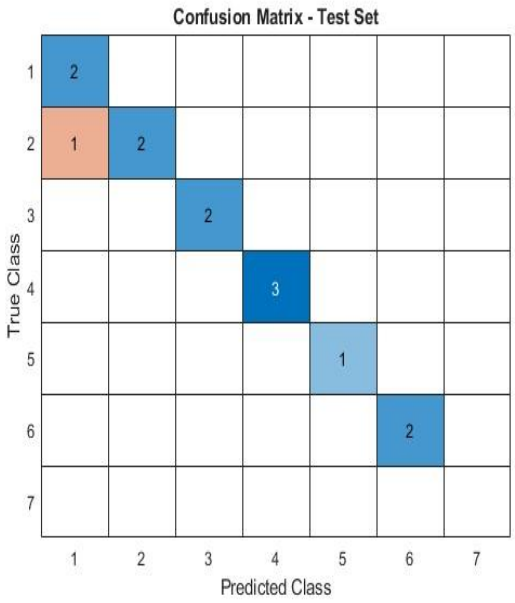

Fig. 4. Confusion matrix on the test set.

Predictions and qualitative validation

Beyond the held-out test subset, the trained model is exercised on new simulation runs using an automated testing script. Fig. 5 and Fig. 6 show examples of predicted labels obtained through the scripted inference pipeline and through the diagnostic output display, confirming end-to-end operability.

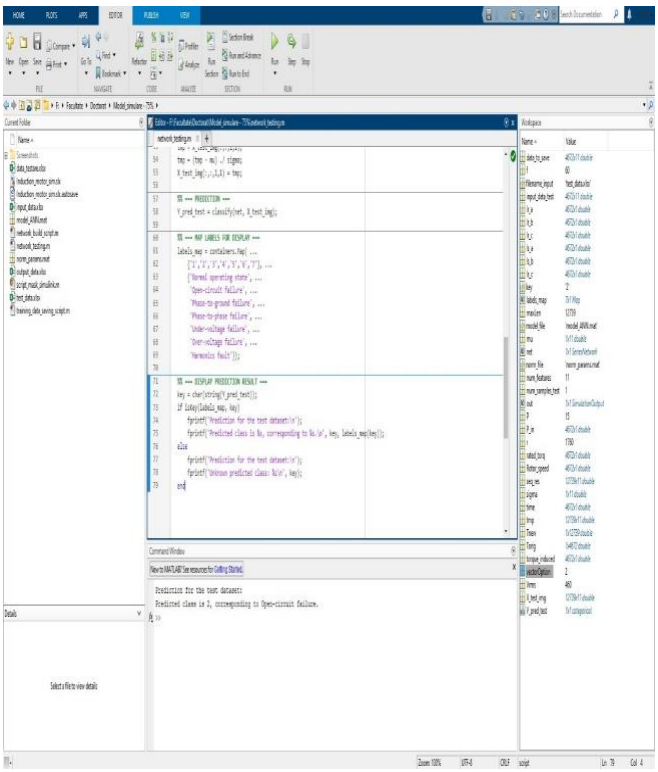

Fig. 6. Diagnostic label produced when evaluating the model on test data.

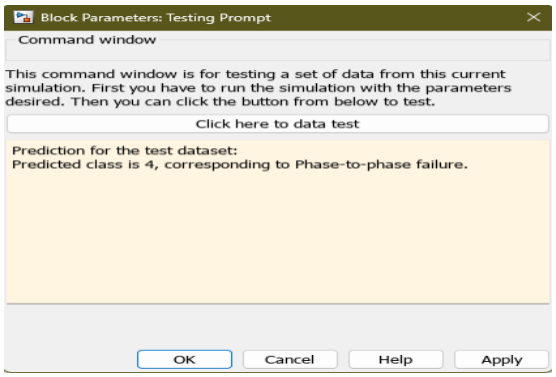

Fig. 5. Example prediction output on a new simulated dataset.

6. TESTING AND DEPLOYMENT WORKFLOW

Automated testing script for new simulations

A dedicated MATLAB script (*network_testing.m*) automates the entire inference flow for new Simulink runs. First, it checks the presence of the simulation output structure (out) in the MATLAB workspace and extracts the required signals (currents, power, speed, torque, and rated torque). Signals are exported to an Excel file (test_data.xlsx) that follows the same ordering and format used during training, ensuring full compatibility with the CNN input pipeline. Next, the script loads the trained network (model_ANN.mat) together with the stored normalization parameters (norm_params.mat). Because the temporal length of test signals may differ from training sequences, linear interpolation resampling is performed to match L_max defined by the trained network. Finally, the normalized tensor is passed to classify, which returns the predicted label; a mapping from numeric codes to descriptive class names is used to display a user-readable diagnosis in the MATLAB console.

Algorithmic summary

1) Extract signals from Simulink output structure and export to tabular format.
2) Group by simulation_id and resample each feature sequence to length L_max using linear interpolation.
3) Apply Z-score normalization using stored (μ, σ) from training.
4) Run CNN inference (softmax classifier) to obtain class probabilities and predicted label.
5) Display and log the diagnosis; optionally trigger downstream maintenance actions.

Simulink integration

For workflow integration, the testing procedure is encapsulated in a Simulink masked block that calls the automated testing routine and returns the predicted class (Fig. 7). This approach enables automated evaluation across multiple simulated scenarios in MATLAB/Simulink, and the trained CNN is integrated in Simulink (together with

normalization parameters) to perform end-to-end inference on new simulation outputs.

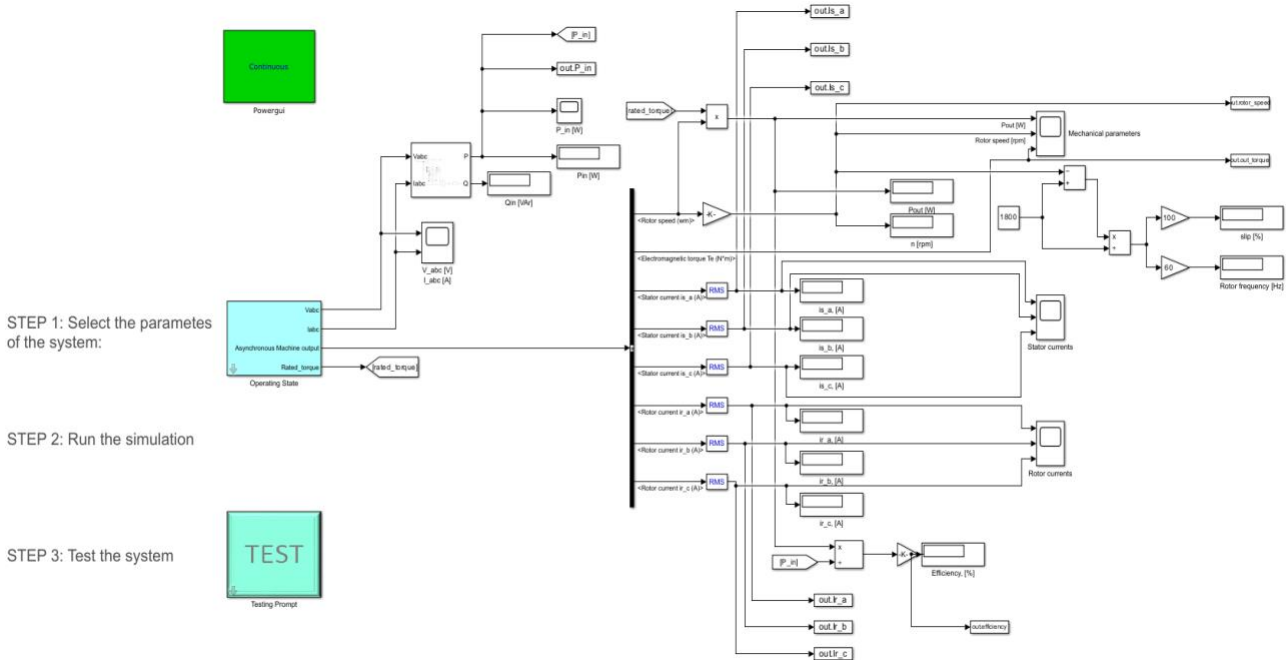

Fig. 7. Modeling of the Operating Modes of an Induction Motor in MATLAB/Simulink.

7. IMPLEMENTATION DETAILS

To facilitate replication and integration in engineering workflows, the implementation follows a file-based interface between Simulink simulations and MATLAB training/testing scripts. During training, the build script exports a unified dataset, computes normalization statistics, trains the CNN, and saves all parameters required for later inference (Table IV).

During deployment/testing, the inference script loads the saved model and normalization parameters, applies the same resampling and Z-score normalization, and returns a human-readable diagnostic label. This separation between data export, model training, and inference supports versioning and traceability of datasets and models.

Table 4. Key Files and Deliverables Used in Training and Deployment

Artifact	Role in the workflow
input_data.xlsx / simulation export	Time-indexed multivariate signals with simulation_id.
output_data.xlsx / labels	Operating_state labels for each simulation run.
network_build_script.m	Dataset grouping, resampling, normalization, training, and model saving.
norm_params.mat (μ, σ)	Stored normalization parameters reused in inference.
model_ANN.mat (trained net)	Saved CNN used for classify during deployment.
network_testing.m	Automated inference on new Simulink runs and label mapping.
Simulink masked block	Calls the testing routine and injects diagnosis into the model workflow.

Script excerpt (pre-processing core)

```
maxLen = max(histc(sim_ids, unique_sims));
Tnew = linspace(1,size(seq,1),maxLen);
seq_res = interp1(Torig, seq, Tnew);
mu = mean(concat_all,1);  sigma =
std(concat_all,0,1);
tmp = (tmp - mu) ./ sigma;  % Z-score
normalization
Ypred = classify(net, XTest);
```

8. DISCUSSION AND FUTURE WORK

The reported results indicate that a compact CNN can accurately classify induction motor fault modes from multivariate operational signatures. In additional experiments described in the source material, alternative architectures such as simple RNNs and CNN–LSTM hybrids were explored; these models showed less stable training and lower accuracy than the pure CNN, while incurring higher computational cost for inference.

Several extensions are important for industrial-grade predictive–adaptive maintenance:

a) addressing simulation-to-reality domain shift via calibration, noise injection, and transfer learning
b) expanding operating scenarios consistent with power-quality and machine standards [14], [15];
c) adding explainability (saliency maps, relevance propagation) to support maintenance decisions; and
d) extending from diagnosis to prognostics such as remaining useful life estimation [18]

9. CONCLUSION

This paper presented a CNN-based approach for automated fault diagnosis of industrial induction motors within a predictive–adaptive maintenance framework. Using simulated operational signatures generated in MATLAB/Simulink, the model achieved 92.31% validation accuracy on seven operating classes. A standardized pipeline for resampling and normalization ensured consistent training and inference, while integration via a Simulink masked block enabled automated end-to-end fault classification on new simulation outputs

10. REFERENCES

[1] N. Hivarekar et al., "Preventive and predictive maintenance modeling," in Proc. RAMS, 2020, pp. 1–6, doi: 10.1109/RAMS48030.2020.9153636.

[2] H. Meriem, H. Nora, and O. Samir, "Predictive maintenance for smart industrial systems: a roadmap," Procedia Computer Science, vol. 220, pp. 645–650, 2023, doi: 10.1016/j.procs.2023.03.082.

[3] P. Mallioris, E. Aivazidou, and D. Bechtsis, "Predictive maintenance in industry 4.0: A systematic multi-sector mapping," CIRP Journal of Manufacturing Science and Technology, vol. 50, pp. 80–103, 2024, doi: 10.1016/j.cirpj.2024.02.003.

[4] S. Nikolic and C. Rados, "Motor current signature analysis in predictive maintenance," Journal of Energy – Energija, vol. 67, no. 4, pp. 3–6, 2018, doi: 10.37798/201867462.

[5] E. L. Bonaldi, L. L. Oliveira, J. G. da Silva, and G. Lambert-Torres, "Predictive maintenance by electrical signature analysis to induction motors," 2012, doi: 10.5772/48045.

[6] D. Miljković, "Brief review of motor current signature analysis," CrSNDT Journal, vol. 5, 2016.

[7] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol. 313, pp. 504–507, 2006, doi: 10.1126/science.1127647.

[8] Y. Bengio, "Learning deep architectures for AI," Foundations and Trends in Machine Learning, 2009, doi: 10.1561/2200000006.

[9] Y. Yang, M. M. M. Haque, D. Bai, and W. Tang, "Fault diagnosis of electric motors using deep learning algorithms and its application: A review," Energies, vol. 14, 2021, doi: 10.3390/en14217017.

[10] O. V. Thorsen and M. Dalva, "Failure identification and analysis for high voltage induction motors in petrochemical industry," in Proc. IEEE IAS Annual Meeting, 1998, pp. 291–298, doi: 10.1109/28.777188.

[11] W. Jung et al., "Vibration and current dataset of three-phase permanent magnet synchronous motors with stator faults," Data in Brief, vol. 47, 2023, doi: 10.1016/j.dib.2023.108952.

[12] W. Jung et al., "Vibration, acoustic, temperature, and motor current dataset of rotating machine under varying operating conditions for fault diagnosis," Data in Brief, vol. 48, 2023, doi: 10.1016/j.dib.2023.109049.

[13] Z. A. Bukhsh, A. Saeed, I. Stipanovic, and A. G. Doree, "Predictive maintenance using tree-based classification techniques: a case of railway switches," Transportation Research Part C, vol. 101, pp. 35–54, 2019, doi: 10.1016/j.trc.2019.02.001.

[14] IEC 61000-2-2:2002, Compatibility levels for low-frequency conducted disturbances and signaling in public low-voltage power supply systems, 2002.

[15] IEC 60034-25:2014, Rotating electrical machines – Part 25: Effects of unbalanced voltages on the performance of three-phase cage induction motors, 2014.

[16] J. Renfigo, H. Salazar, A. Bueno, and J. M. Aller, "Experimental evaluation of the voltage unbalance in the efficiency of induction motors," in Proc. PEPQA, 2017, pp. 1–6.

[17] S. H. Kia, H. Henao, G.-A. Capolino, and C. Martis, "Induction machine broken bars fault detection using stray flux after supply disconnection," in Proc. IECON, 2006, pp. 1498–1503, doi: 10.1109/IECON.2006.347595.

[18] M. Binder, V. Mezhuyev, and M. Tschandl, "Predictive maintenance for railway domain: A systematic literature review," IEEE Engineering Management Review, vol. 51, no. 2, pp. 120–140, 2023, doi: 10.1109/EMR.2023.3262282.