



PMSM fault detection using unsupervised learning methods based on conditional convolution autoencoder

KOZOVSKEÝ, M.; BUCHTA, L.; BLAHA, P.

IECON 2024- 50th Annual Conference of the IEEE Industrial Electronics Society

pages: 1-6

eISBN: 978-1-6654-6454-3

DOI: <https://doi.org/10.1109/IECON55916.2024.10905074>

Accepted manuscript

©2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. KOZOVSKEÝ, M.; BUCHTA, L.; BLAHA, P. „PMSM fault detection using unsupervised learning methods based on conditional convolution autoencoder“, IECON 2024- 50th Annual Conference of the IEEE Industrial Electronics Society. DOI: 10.1109/IECON55916.2024.10905074. Final version is available at <https://ieeexplore.ieee.org/document/10905074>

PMSM fault detection using unsupervised learning methods based on conditional convolution autoencoder

1st Matus Kozovsky 

Central European Institute of Technology
Brno University of Technology
Brno, Czech Republic
matus.kozovsky@ceitec.vutbr.cz

2nd Ludek Buchta 

Central European Institute of Technology
Brno University of Technology
Brno, Czech Republic
ludek.buchta@ceitec.vutbr.cz

3rd Petr Blaha 

Central European Institute of Technology
Brno University of Technology
Brno, Czech Republic
petr.blaha@ceitec.vutbr.cz

Abstract—The challenges of fault detection and condition monitoring in powertrain systems have become increasingly prominent, particularly with the widespread adoption of fail-operational systems. These systems are pivotal in diverse sectors, including the robotics, automotive industry, and various industrial applications. A critical attribute of such systems lies in their capability to identify non-standard behaviour of the system. This study describes a inovative conditional convolutional autoencoder-based fault detection algorithm for the permanent magnet synchronous motor. The study compares a train process of conditional convolutional autoencoder with a classical convolutional autoencoder. The presented autoencoder structure was designed to be implementable into the target microcontroller AURIX TC397 while providing sufficient recognition capabilities of the interturn short-circuit. Autoencoders are trained on data obtained during healthy motor operation and subsequently used to detect interturn short-circuit faults on the experimental dual three-phase permanent magnet synchronous motor with the possibility of emulating an interturn short-circuit fault. The paper provides insights into the achieved autoencoder inference times and the sensitivity in detecting the fault.

Index Terms—autoencoder, conditional convolution, fault diagnostic, permanent magnet synchronous motor (PMSM)

I. INTRODUCTION

Recent advancements in artificial neural networks and the continuous increase in the computing power of microcontrollers have opened up new application fields for deploying machine learning approaches. Machine learning methods such as neural networks have the potential to enhance the fail-operational capabilities of the powertrains significantly, especially from a condition monitoring point of view. Although these methods were traditionally used on powerful computational platforms, they can now be implemented on-the-edge devices or directly integrated into microcontrollers. One of

The work has been performed in the project Archimedes: Trusted lifetime in operation for a circular economy No 101112295/9A23010. The work was co-funded by grants of Ministry of Education, Youth and Sports of the Czech Republic and Chips Joint Undertaking.

The work was supported by the infrastructure of RICAIP that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857306 and from Ministry of Education, Youth and Sports under OP RDE grant agreement No CZ.02.1.01/0.0/0.0/17_043/001/0085.

the biggest challenges highlighted by the extensive study [1] is implementing neural networks directly into microcontrollers without additional hardware modification. The study describes recent trends and the typical approaches for detecting faults in permanent magnet synchronous motors (PMSM). The mechanical, magnetic, and electrical failures in these motors are analysed. The fault detection methods are classified into three categories: signal analysis-based, model-based, and artificial intelligence (AI) based methods.

Electrical faults in motors are often detected from the stator current waveforms, which are often transformed into rotor coordinates. Two main methods are used for signal analysis - frequency domain and time domain. The methods provide excellent results under constant operational conditionals, where data can be measured for a long period. On the other hand, the reaction time is limited by long measurements. The methods also have problems operating under transitional states when the signal processing is significantly more demanding.

Model-based methods generally compare measured data with models. This approach typically includes parameter observers or Kalman filters. The main benefit comes from the possibility of detecting the fault under transient states if the properly designed model is used [2]. Fault detection reactions can be very fast compared to signal analysis methods. Models for fault behaviour simulations are often in stator coordinates, respecting the mechanical construction of the machine. However, designing proper models respecting the stator structure of the machine can be challenging [3].

Artificial intelligence-based techniques have the potential to fuse the advantages of both methods mentioned earlier [4], [5]. The study mentioned above also presents the main shortcomings of using neural networks for PMSM fault diagnostics. Most summarised methods are unsuitable for microcontroller implementation due to high computational demands. Analysed AI methods mainly use multi-layer perceptron networks (MLPN), convolutional neural networks (CNN), or long-short-term memory (LSTM) networks. All these approaches can be classified as state-of-the-art. The majority of the analysed neural networks are based on supervised learning. In this case,

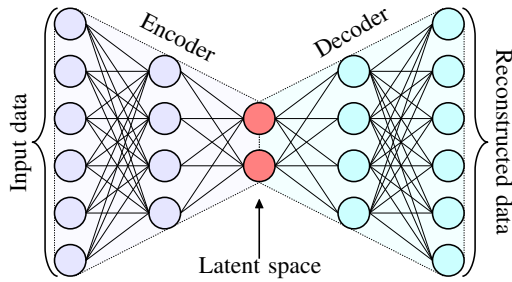


Fig. 1: The structure of a basic autoencoder based on a multi-layer perceptron network

behaviour under the fault condition must be measured [6] or simulated [5]. The simulated behaviour of the fault can be inaccurate, which can negatively affect the classification capabilities of artificial neural networks ANNs. On the other hand, some types of faults cannot be easily emulated on real devices. From this point of view, using healthy behaviour training data and identifying anomalies have many benefits. Anomaly detection methods utilise unsupervised learning methods. In this case, Self-organizing Kohonen maps [7] or autoencoders are typically used [8], [9].

This paper aims to demonstrate the detection capabilities of the modern convolutional autoencoders used to detect anomalies in the behaviour of the PMS motors. The presented autoencoder fall into the category of unsupervised learning methods. They are trained on data sets from healthy motor operations, which include steady and transient states, and tested on measured data with emulated fault.

II. ANOMALY DETECTION USING NEURAL NETWORKS

Autoencoders find applications in various fields, including data compression, feature learning, and anomaly detection. The primary goal of an autoencoder is to learn a compressed representation of input data. The architecture consists of an encoder, which maps the input data to a lower-dimensional representation, and a decoder, which reconstructs the input from this representation. During training, the autoencoder learns to minimise the difference between the input and the reconstructed output, forcing it to capture meaningful features and patterns in the data.

Considering healthy training data measured on the motor, the autoencoder extracts the main features essential for encoding the signal into latent space. Latent space is significantly smaller compared to input data size. The reconstruction process performed by the decoder is also based on features extracted from training data. During the normal operation, the mean squared error (MSE) of the difference between the input and reconstructed signal is low because the autoencoder was trained on the healthy data. On the other hand, the fault condition signal contains additional components caused by an active fault in the motor unknown to the trained autoencoder. For this reason, the performance of the autoencoders is decreased, and MSE is significantly higher.

Various layers can be used to create the autoencoder. One of the most accessible options is to use fully connected layers

and reduce the number of neurons until the desired dimension of the latent space is reached. The reconstruction part can be a mirror version of the encoded part. In this case, we speak about autoencoders based on MLPN. Figure 1 shows the autoencoder structure.

Autoencoder can also be implemented using convolutional layers. This version of autoencoders is suitable for one-dimensional input vectors or two-dimensional images. In our case, we use an autoencoder to detect anomalous motor behaviour. For this reason, suitable inputs are vectors of measured currents or vectors of control voltages. The properties of signals entering the autoencoder are strongly dependent on the motor speed. The impact of the speed is present on signals in stator coordinates as well as in rotor coordinates. In this case, a simple convolutional autoencoder requires a larger number of filters (kernels) to process the signal at different operating points. The more significant number of filters causes longer inference times. For instance, this problem can be solved by using a set of simpler autoencoders prepared for the various speeds.

Another approach for anomaly detection using neural networks involves training a model to match the motor behaviour. In this case, MLPN or LSTM can be utilised. The neural network takes the input voltages sent to the motor and estimates the measured motor currents in the next step. The neural network inference must occur every sampling period to achieve reliable performance while using this method.

The autoencoders presented in this publication utilise a combination of both methods. The input vector is formed by voltages, while the reconstructed output vectors represents motor currents. Real measured currents contain ripples given by nonlinearities in the motor and power inverter. The nonlinearities depend on the polarities of the currents and also on the actual rotor position. For this reason, additional input vectors with Sine and Cosine electrical position signals are used. This signal also indirectly contains information about the motor speed.

$$\mathbf{V}_{\text{In}} = [\mathbf{u}_d, \mathbf{u}_q, \sin(\theta_e), \cos(\theta_e)]^T \quad (1)$$

Input matrix for autoencoder is constructed according to (1) where \mathbf{u}_d denotes d-axis voltage vector and \mathbf{u}_q stands for q-axis voltage vector. θ_e represents the vector of the electrical angle of the motor. All vectors have a length of 125 elements and contain the latest data.

$$\mathbf{u}_d = [u_{d(k-124)}, u_{d(k-123)}, \dots, u_{d(k)}] \quad (2)$$

(2) shows construction of \mathbf{u}_q vector where k denotes the current step of the control algorithm. The autoencoder is trained to reconstruct motor currents in dq coordinates corresponding to the input voltages. In this case, the autoencoder needs to decrease the dimension of the input voltage vector and reconstruct output currents given by the motor behaviour. We use a traditional convolutional autoencoder with three convolutional layers in the encoder and three transposed convolution layers in the decoder. The overall 330 trainable parameters are in the

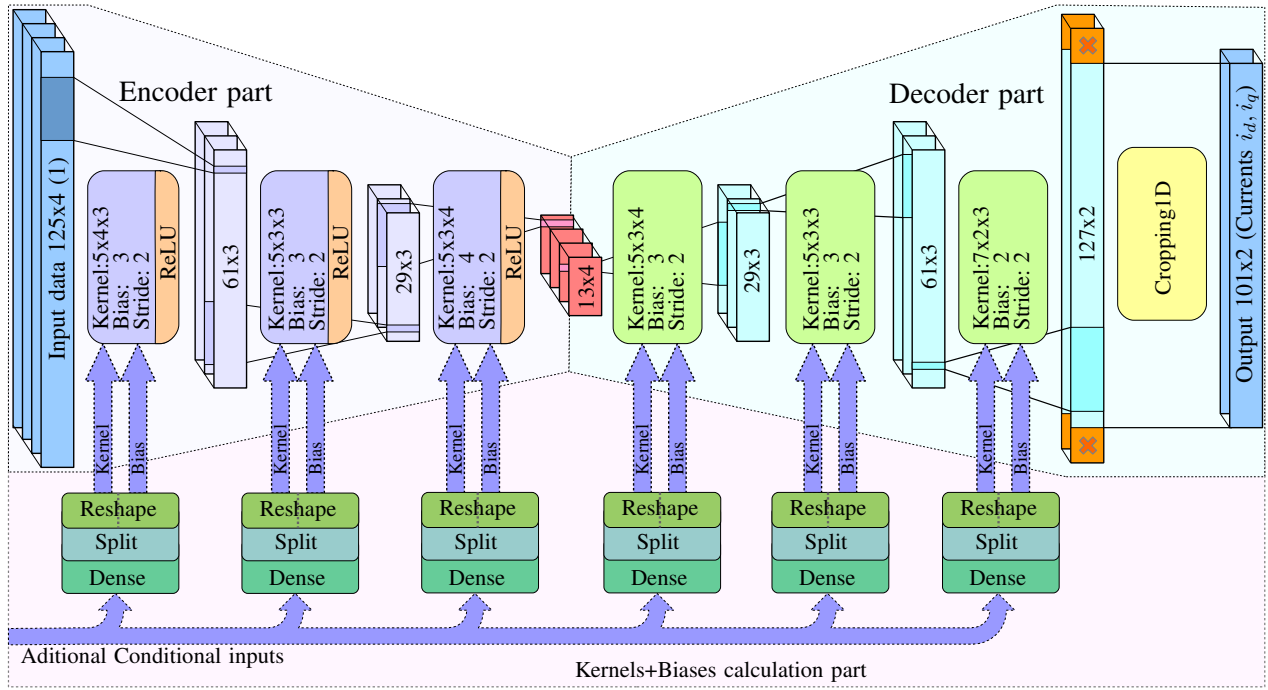


Fig. 2: The structure of a basic autoencoder based on a multi-layer perceptron network

autoencoder. Convolutional layers in the encoder part of the autoencoder calculate outputs Out_C using (3).

$$Out_C = g(\mathbf{K} *_2 \mathbf{In}_C + \mathbf{B}) \quad (3)$$

$$g(x) = \max(0, x) = (x + |x|)/2 \quad (4)$$

The variable K denotes the Convolutional kernel, and B stands for the biases. Convolutional operation with stride two is represented by $*_2$ operand. The nonlinear function g , which represents the rectified linear unit ReLU, is also used in the encoders (4). The Decoder part of the autoencoder uses Transposed convolutional (5).

$$Out_{C^T} = \mathbf{K}^T *_2 \mathbf{In}_{C^T} + \mathbf{B} \quad (5)$$

In the decoder part, nonlinear functions are not present. The last layer trims the edges of the reconstructed signal to prevent inaccurately reconstructed border data from compromising the precision of the estimation during training as well as in the anomaly detection process. This layer removes 26 elements, i.e. 13 elements from each side. For this reason, the output dimension from the autoencoder differs from the input dimensions, and the output vector represents the current in control steps according to (6).

$$\mathbf{I}_{Out} = [\mathbf{i}_d, \mathbf{i}_q]^T = \begin{bmatrix} I_{d(k-112)} & I_{d(k-111)} & \cdots & I_{d(k-12)} \\ I_{q(k-111)} & I_{q(k-111)} & \cdots & I_{q(k-12)} \end{bmatrix}^T \quad (6)$$

Coder presented by the paper uses condition-dependent kernels and biases. Figure 2 shows the overall structure of the autoencoder. The upper part of the figure illustrates the structure of the simple autoencoder we mentioned earlier. Figure also indicates the data dimensions in all parts of the autoencoder. The lower part of the figure explains the kernels

and biases calculation from the additional feature input used by the conditional convolutional autoencoder. Kernels and biases are used as inputs for the convolutional layer. And the output is calculated using (7) and (8).

$$Out_{CC} = g(\mathbf{K}_{(\omega_e)} *_2 \mathbf{In}_{CC} + \mathbf{B}_{(\omega_e)}) \quad (7)$$

$$Out_{CC^T} = \mathbf{K}_{(\omega_e)}^T *_2 \mathbf{In}_{CC^T} + \mathbf{B}_{(\omega_e)} \quad (8)$$

Generally, kernels and biases can depend on multiple features and include additional nonlinearity, making kernel computation computationally expensive. However, we only implemented a simple motor speed dependency to highlight the substantial contribution of using conditional convolutional layers in autoencoders. Electrical motor speed is represented by ω_e . Each Kernel and Bias variable is calculated using (9) and (10).

$$K_{n(w,ic,oc)} = wk_{n(w,ic,oc)}(\omega_e) + wkb_{n(w,ic,oc)} \quad (9)$$

$$B_{n(oc)} = wb_{n(oc)}(\omega_e) + wbb_{n(oc)} \quad (10)$$

$wk_{n(l,j,k)}$ and $wkb_{n(l,j,k)}$ denotes weights in dense layer for specific convolutional layer n related to the layers Kernel while $wb_{n(k)}$ and $wbb_{n(k)}$ stands for convolutional layer bias weights. w, ic, oc represent the width, input channel, and output channel indexes.

The number of trainable parameters is increased to 660. However, the computation time of the autoencoder will be only slightly higher because the main computational difficulty lies in the repeated use of the kernel. Encoder and decoder part computing require over 22,000 numerical operations, mainly multiplications and addition. In contrast, the construction of the kernels and biases requires only 330 multiplications and 330 additions.

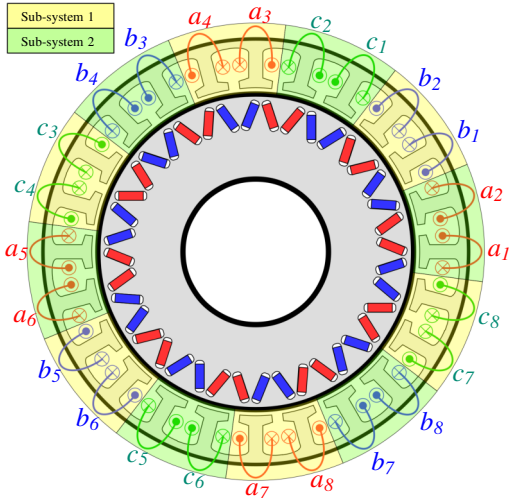


Fig. 3: The experimental motor structure and assignment of individual coils into sub-systems

Simple convolutional autoencoder as well as conditional convolutional autoencoder were trained in TensorFlow. The first autoencoder uses only the standard layers supported by the framework. On the other hand, conditional convolution is not natively supported by TensorFlow. Custom layers providing conditional convolution were designed and implemented for this reason. Custom conditional convolution layers use multiple inputs, one for the input data, the second for the convolution kernel and the last for the biases, in contrast to the natively supported layers, which have all weights as internal class variables.

Both autoencoders were also implemented into AURIX TC397 microcontroller. Performance tests measuring computational complexity revealed that the convolutional autoencoder with constant kernels requires 274 μ s per inference run, whereas the presented conditional version incurs only an additional 8 μ s.

III. TEST PLATFORM

All tests use the experimental motor to get closer to real applications. The motor is designed to ensure fail-operational functionality and is connected as a dual three-phase machine with interlaced sub-systems. Sub-systems have electrically isolated middle points, though the whole motor is connected to

TABLE I: Experimental motor parameters

Name	Symbol	Value	Unit
Nominal DC-link voltage	U_{DC}	400	V
Used DC-link voltage	U_{DC}	200	V
Maximum continues motor current	I_s	107	A
Nominal motor power	I_s	30.16	kW
Winding resistance	R_s	6.5	m Ω
Back-EMF constant	ψ_m	0.0117	Vs/rad
Winding inductance	L_{u-v}	180	μ H
Nominal speed	ω_n	8000	rpm
Maximum speed	ω_M	10500	rpm
Nominal power	P_c	30160	W
Number of pole pairs	P_p	10	-
Characteristic current	I_{ch}	81	A

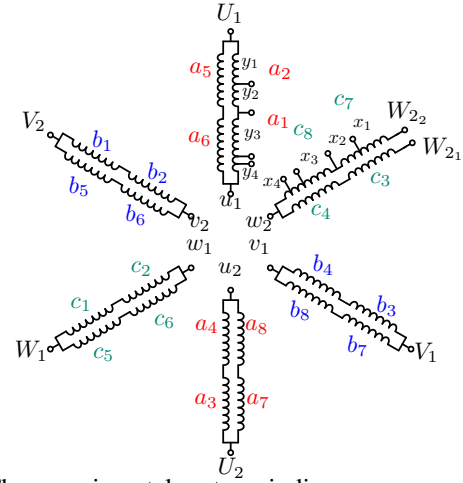


Fig. 4: The experimental motor winding arrangement, winding taps y_{1-4} and x_{1-4} can be used to emulate interturn short-circuit fault

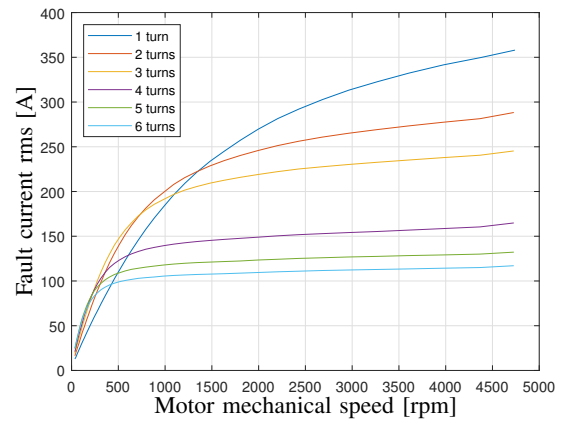


Fig. 5: Motor fault current depending actual motor speed for various inter-turn short-circuit depths including resistance and inductance of additional wires and FIU (0.6 Ω 0.25 μ H)

one six-phase power inverter. The power inverter is controlled by the AURIX TC397 microcontroller, which is equipped with our control software. The individual motor coils are assigned to the sub-systems as shown in figure 3. The electrical connection of the individual coils into motor windings is presented in Figure 4. Each tooth coil has seven turns, while four tooth-coils form each phase winding. Some coils have prepared winding taps y_{1-4} and x_{1-4} used for interturn short-circuit emulation. During the experiment, a short-circuit can be created permanently or switched by a fault insertion unit (FIU). The FIU uses a 500A LEV200A4NAF relay parallel to the power thyristor module STT800N16P55XPSA1.

The parameters for the PMS motor that was used are outlined in Table I. The field weakening index of the machine is lower than one, and the sub-system affected by the fault can be switched into active short circuit (ASC) mode to suppress the amplitude of the fault current. Otherwise, the short-circuit will spread to other coil turns due to local overheating. The amplitude of the fault current depends primarily on the actual

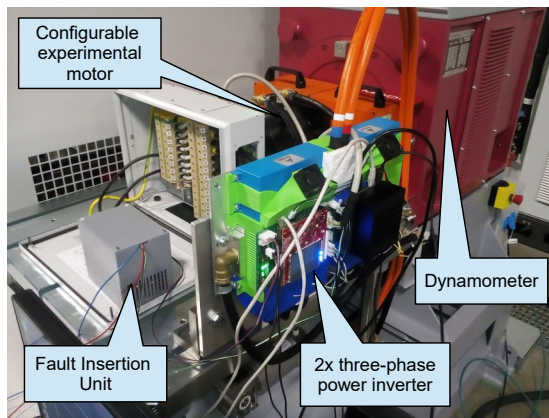


Fig. 6: Dynamometer and experimental PMSM under test

speed of the motor and the number of short-circuited turns. Figure 5 shows the measured short-circuit current values for the motor without load. It follows from the figure that for a given motor, it is necessary to detect the fault for speeds above 500 rpm reliably. Figure 6 shows the used experimental testbench.

IV. ACHIEVED RESULTS

Both autoencoders were trained using the same data. Training data set was collected during 15 minutes of motor running with a sampling frequency of 10kHz. The motor changed the working speed (0-5000 rpm), and the load torque on the dynamometer was gradually adjusted to measure the entire working area (0-35 Nm) during the data measurement. The entire operating area is covered evenly. Validation data were obtained from a separate measurement. This dataset includes densely measured low-speed operation points and less densely measured high-speed operation points. Thanks to this approach, it is possible to draw a detailed performance map up to 1200 rpm shown in Figure 10. On the other hand validation loss is higher than training loss because of majority of low-speed data in validation dataset.

The data generator for the training process was created in Python. The generator randomly selects sections of 125 elements from the measured data and compiles the input and training output vectors according to (1) and (6). During each training epoch, 50,000 randomly selected data segments were used in the autoencoder. Validation uses 10,000 randomly selected data segments from validation dataset. A comparison of learning processes between a convolutional autoencoder and a conditional convolutional autoencoder can be seen in Figure 7. The figure clearly demonstrates a significant difference between both structures. Convolutional autoencoder achieved a loss function value of 16.5. On the other hand, the conditional convolutional layer achieved a loss function value of 3.4. These results show a significant improvement at the cost of a small increase in computational complexity.

The designed autoencoder was tested on the dual three-phase machine. However, the evaluation of each sub-system was performed individually for each sub-system. Fault detec-

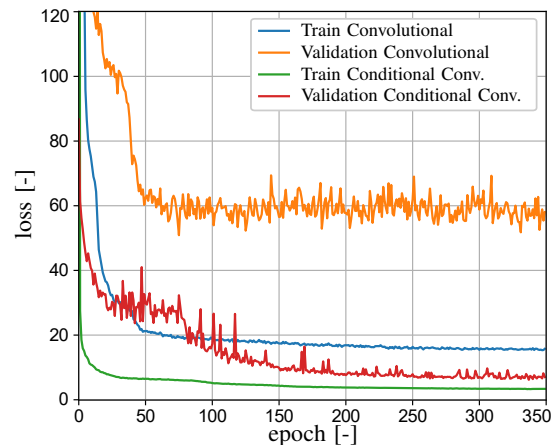


Fig. 7: Comparison of the convolutional autoencoder and conditional convolutional autoencoder during training process

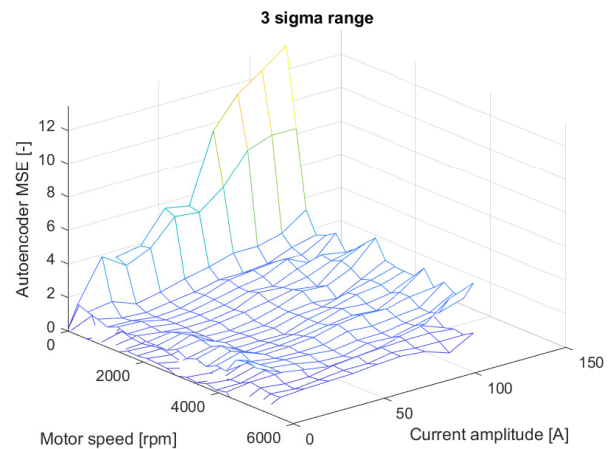


Fig. 8: Evaluation algorithm map showing the $\pm 3\sigma$ range

tion in the PMSM is based on statistical approaches. The operation area is divided into a mesh of points, and autoencoder performance is analysed using the training dataset for each mesh point. The mean value and variance of the MSE are calculated and stored for each point. In a real deployment of the system, the first step involves the inference of the autoencoder, followed by the second step, which evaluates the probability of achieving the result within permissible deviations. Acceptable sigma range which directly influence to sensitivity can be easily configured. Figure 8 shows the evaluation map with the configured range of three sigma. The system returns the probability of the reconstructed state aligning with the training conditions. Figure 9 shows the performance of the conditional convolutional autoencoder with a described evaluation under healthy conditions and during one shorted turn in the evaluated sub-system. A probability of 0.5 was chosen as a threshold for classifying healthy/faulty classes. High MSE value in low-speed areas is given by the low voltages and low noise-to-signal ratio.

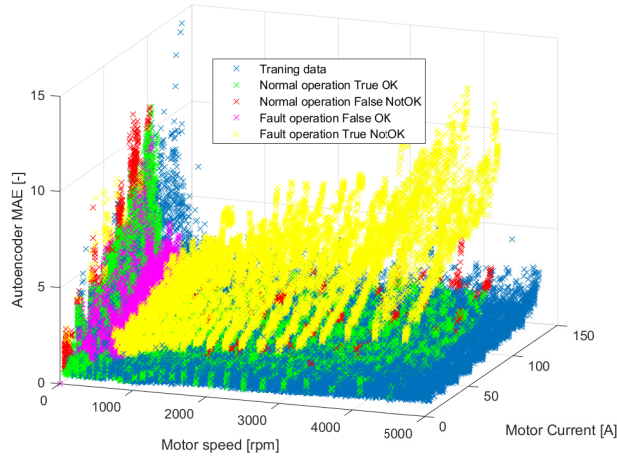


Fig. 9: Difference between normal behaviour and one shorted turn in autoencoder performance

Last Figure 10 shows the performance of the system under different operation conditions. Left column represent results achieved on sub-system 1 and right column of results belongs to sub-system 2. Faults are emulated only in the first sub-system. However decrease of autoencoder performance is visible also in the second sub-system. This behaviour is given by mutual inductance coupling between the sub-systems nevertheless faulty part can be easily determined. Achieved results are comparable to methods based on supervised learning [10]; however, autoencoders use only easily obtainable standard operation data.

V. CONCLUSION

The paper discusses the use of a conditional convolutional autoencoder for detecting deviations from standard behaviour in PMSM powertrains. The study shows an improvement in the training process compared to the classical convolutional autoencoder that has the same number and dimensions of filters. The presented conditional convolutional autoencoder was designed with the implementability into a microcontroller in mind. The results achieved in this study demonstrate the benefits of using innovative conditional convolutional layers in autoencoders, which provide significantly higher performance than classical convolutional layers. The computational complexity of the conditional convolutional layers depends on the number of related features. The paper proposes the use of a simple dependence on speed, which brings a significant improvement in performance with only a slight increase in computation time. Computation time increases from 274 μ s for classical convolutional autoencoder to 282 μ s for conditional version. The fault detection algorithm based on the conditional convolutional autoencoder achieved comparable results to supervised learning methods.

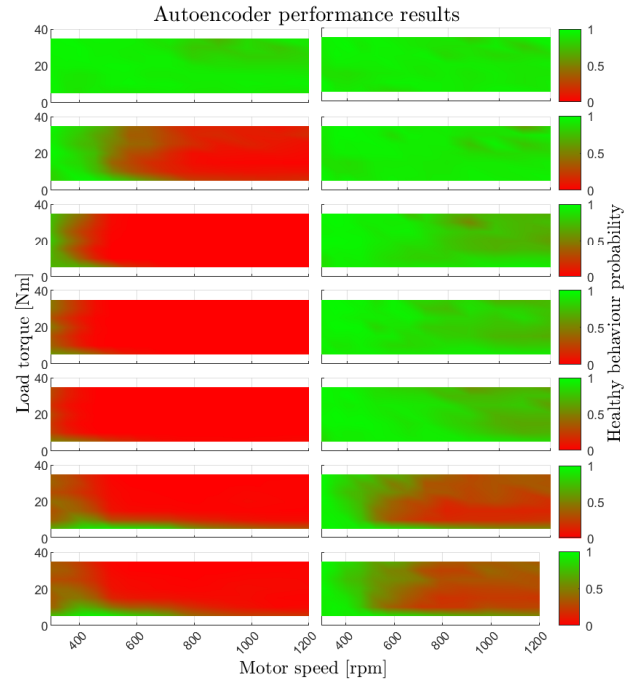


Fig. 10: Deployed autoencoder performance for various operation points. Left column sub-system 1, Right Column sub-system 2, from up to down number of shorted turns in sub-system 1 0-6.

REFERENCES

- [1] T. Orłowska-Kowalska, M. Wolkiewicz, P. Pietrzak, M. Skowron, P. Ewert, G. Tarchala, M. Krzysztofak, and C. T. Kowalski, "Fault Diagnosis and Fault-Tolerant Control of PMSM Drives—State of the Art and Future Challenges," *IEEE Access*, vol. 10, pp. 59 979–60 024, 2022.
- [2] L. Zezula, M. Kozovsky, and P. Blaha, "Diagnostics of Interturn Short Circuits in PMSMs With Online Fault Indicators Estimation," *IEEE Transactions on Industrial Electronics*, pp. 1–11, 2024.
- [3] M. Kozovsky, L. Buchta, and P. Blaha, "Interturn short circuit modelling in dual three-phase PMSM," in *IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, oct 2022, pp. 1–6.
- [4] S. Moosavi, A. Djerdjir, Y. Ait-Amirat, and D. Khaburi, "ANN based fault diagnosis of permanent magnet synchronous motor under stator winding shorted turn," *Electric Power Systems Research*, vol. 125, pp. 67–82, aug 2015.
- [5] L. S. Maraaba, Z. M. Al-Hamouz, A. S. Milhem, and M. A. Abido, "Neural Network-Based Diagnostic Tool for Detecting Stator Inter-Turn Faults in Line Start Permanent Magnet Synchronous Motors," *IEEE Access*, vol. 7, pp. 89 014–89 025, 2019.
- [6] M. Skowron, T. Orłowska-Kowalska, and C. T. Kowalski, "Application of simplified convolutional neural networks for initial stator winding fault detection of the PMSM drive using different raw signal data," *IET Electric Power Applications*, vol. 15, no. 7, pp. 932–946, jul 2021.
- [7] K. Jankowska and P. Ewert, "Effectiveness Analysis of Rolling Bearing Fault Detectors Based On Self-Organising Kohonen Neural Network – A Case Study of PMSM Drive," *Power Electronics and Drives*, vol. 6, no. 1, pp. 100–112, jan 2021.
- [8] Y. Huang, C.-H. Chen, and C.-J. Huang, "Motor Fault Detection and Feature Extraction Using RNN-Based Variational Autoencoder," *IEEE Access*, vol. 7, pp. 139 086–139 096, 2019.
- [9] Z. Chen, R. Guo, Z. Lin, T. Peng, and X. Peng, "A Data-Driven Health Monitoring Method Using Multiobjective Optimization and Stacked Autoencoder Based Health Indicator," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 6379–6389, sep 2021.
- [10] M. Kozovsky, L. Buchta, and P. Blaha, "Implementation of ANN for PMSM Interturn Short-Circuit Detection in the Embedded System," in *IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, oct 2023, pp. 1–6.