Feature Extraction Using Convolutional Neural Networks for Anomaly Detection

Rodrigo de P. Monteiro
Electrical Engineering Postgraduate Program
Federal University of Pernambuco
Recife, Brazil
rodrigo.paula@ufpe.br

Carmelo J. A. Bastos-Filho
Computer Engineering Postgraduate Program
University of Pernambuco
Recife, Brazil
carmelofilho@ieee.org

***Abstract*—Anomaly detection is an import field of study, which has many applications, *e.g.*, fraud and disease detection. It consists of identifying non-conforming patterns regarding an expected behavior. Despite the improvements provided by deep learning techniques in several areas, their use for anomaly detection is not widespread. The main reason is the difficulty to learn discriminative models when all the information available regards one class, or the classes are highly unbalanced. We propose a new deep learning-based solution for the anomaly detection problem. It consists of a hybrid system, composed of a feature extractor and a one-class classifier. The feature extractor is a convolutional neural network, trained as a regressor to learn a predefined distribution. The classifier is the one-class support vector machine, which performs the anomaly detection by using the outputs provided by the feature extractor. We used a gearbox failure diagnosis data set to assess the performance of our proposal. We also compared our anomaly detection system with other deep learning-based techniques commonly found in the literature. Our proposal presented an average accuracy close to 0.95, outperforming techniques based on the reconstruction error and hybrid models.**

Keywords— Anomaly Detection; Deep Learning; One-Class Support Vector Machine; Convolutional Neural Network

I. INTRODUCTION

Anomaly detection has gained importance in many applications, and it can be applied to multiple domains, *e.g.*, fraud detection [1], network security [2], among others. It consists of identifying non-conforming patterns (anomalies) regarding an expected behavior [3]. Anomalous behaviors are commonly caused by errors, defects, unseen patterns, presence of contaminants, among other reasons [4]. The detection of anomalies is an essential step in decision-making processes [3, 4], *e.g.*, to plan the maintenance of machines in a manufacturing line.

The last decade witnessed the expansion of deep leaning to several application domains, *e.g.*, time series prediction, object detection, and classification [4]. Deep learning is a subfield of machine learning characterized by processing the information through multiple layers. The advantage of these models is the capability to represent implicitly features regarding the inputs along with their hidden structure, which generates different abstraction levels of a problem without human interference [5]. The performance of deep learning-based techniques proved to be superior to traditional machine learning approaches, *e.g.*, support vector machines, especially regarding problems with images and a large amount of data [4, 5].

A standard solution that employs deep learning for anomaly detection is the analysis of the reconstruction error [1, 6, 7]. In this case, an algorithm learns to reconstruct a given input pattern. Then, if we present an input with a different pattern to the algorithm, the reconstruction error tends to be higher than expected. The autoencoder is an example of an algorithm used for this purpose. Another common anomaly detection approach is to train the reconstruction algorithm, *e.g.*, an autoencoder, and use the outputs of intermediate layers as input features for shallow anomaly detection algorithms [8, 9], *e.g.*, one-class support vector machine.

The anomaly detection also can be faced as a binary or multi-class classification problem. However, it is necessary to have prior knowledge concerning the anomalous classes to train the model. One or more classes are defined as normal, while the remaining ones are defined as anomalies [10, 11]. Generative adversarial networks have been employed in anomaly detection, as well [12]. They are often used to increase the number of samples to improve the learning process of the anomaly detector.

Despite the improvements provided by employing deep learning techniques in many areas, their use for anomaly detection is not widespread, especially compared to other applications, *e.g.*, multi-class classification and time series prediction. The main reason is the difficulty to learn discriminative models when the only information available belongs to one class, or the classes are highly unbalanced [4].

Aware of this scenario, we propose a new deep learning-based anomaly detection approach. It consists of a hybrid system that combines implicit feature extraction by a deep architecture with a one-class classifier, as already seen in the literature [8, 9]. Our contribution focuses on the feature extractor, which is a convolution neural network trained as a regressor to learn a predefined distribution, randomly chosen. The network training used the target distribution and data belonging to the normal class. The structure of the CNN consists of alternating convolutional and max-pooling layers,

followed by a fully connected output layer. By doing this, we expect the inputs to belong to the normal class result in patterns like the one used to train the network. On the other hand, anomalous inputs result in outputs that differ from the expected one. The anomaly detection is performed by the one-class support vector machine, which is a shallow machine learning architecture. The inputs are the features extracted by the CNN model. We validate the proposed solution with a gearbox diagnosis data set, already used in works like [13, 14]. We compare our results with the ones obtained by deep learning-based techniques commonly found in the literature.

The remainder of the paper is organized as follows: Section II presents the theoretical background, which contains some information about the algorithms used in this work. Section III introduces the proposed model. Section IV describes the methodology, in which we explain the experiments. Section V shows the results and discussions, Section VI presents the conclusion, and in Section VII we acknowledge for the supports.

II. Theoretical Background

A. Convolutional Neural Networks

Convolutional neural networks (CNN) are models inspired by biological processes. They are composed of small processing units, called neurons. The connections among neurons present patterns like the ones observed in the animal visual cortex [15]. They proved to be a useful tool regarding applications such as object detection [16], fault diagnosis [13], among others. The basic CNN is composed of an input layer, alternating convolutional and pooling layers, fully connected layers and an output layer [15]. Such configuration may be modified according to the application. We explain the role of each layer in the following:

1) Input layer: It receives the raw input data, and also defines the width, height and number of channels of the input [17].

2) Convolutional layer: It learns features, *i.e.* new representations, from a set of inputs and generate feature maps. The maps are created by convolving their inputs with a set of learned weights [17]. Activation functions are used to bound the outputs of this layer and to provide a given behavior, *e.g.*, non-linear [15]. Equation (1) shows the general formulation of this kind of layer:

$$x_j^l = f\left(\sum_{i \in M_j} x_i^{l-1} * k_{ji}^l + b_j^l\right) \quad (1)$$

in which l refers to the current layer, i and j are the indexes of the elements of the previous and current layers, respectively, M_j is a set of input maps, k is the weight matrix of the i-th convolutional kernel of the l -th layer applied to the j-th input feature map. b is the bias.

3) Pooling layer: This layer reduces the spatial resolution of feature maps. It also improves the spatial invariance to input distortions and translations [17]. Most of the recent works employ a variation of this layer called max-pooling [15]. It propagates to the next layer the maximum value of a neighborhood of elements. Equation (2) defines this operation.

$$y_{jrs} = max_{(p,q) \in R_{rs}} x_{kpq} \quad (2)$$

in which y_{jrs} is the output of the pooling process regarding the j-th feature map and x_{kpq} is the element at location (p,q) contained by the pooling region R_{rs} . The pooling process is also called subsampling [17].

4) Fully-connected and output layers: These layers interpret the features provided by the previous layers and perform high-level reasoning [15]. The output layer provides the output of the model, which can be a class score, a prediction, among others [17].

B. Autoencoders

Autoencoders (AE) are models which aim to reproduce their inputs on their outputs, with the least possible amount of distortion. They play a crucial role in unsupervised and semi-supervised learning, as well as in applications such as data compression, noise removal and anomaly detection [18].

Autoencoders can be modeled by using different techniques, *e.g.* neural networks and restricted Boltzmann machines. Also, their structure can present different features, *e.g.* shallow or deep architectures, convolutional or densely connected layers, among others [18]. Fig. 1 depicts an example of an autoencoder that presents a simple structure.

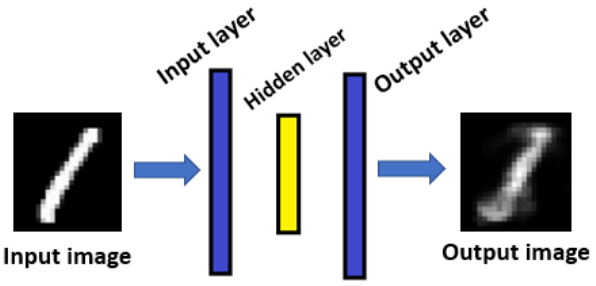

Fig. 1. Scheme of an autoencoder with one hidden layer.

C. One-class support vector machines

The one-class support vector machine (OCSVM) is a semi-supervised variant of the well-known support vector machine (SVM). It is commonly employed on classification tasks in which only data belonging to one class are available, *e.g.* anomaly, outlier and novelty detection problems [3]. The OCSVM proposed by Schölkopf *et al.* uses a hyperplane to separate the normal data from the origin in the feature space. At the same time, it maximizes the distance from this hyperplane to the origin [19]. The quadratic error minimization process works according to Equations (3), (4) and (5):

$$min_{\omega,\,\varepsilon_i\,,\rho}\,\frac{1}{2}\|\omega\|^2 + \frac{1}{\vartheta n}\sum_{i=1}^{n}\varepsilon_i - \rho \tag{3}$$

subject to:

$$\left(\omega \cdot \varphi(x_i)\right) \geq \rho - \varepsilon_i \tag{4}$$

$$\varepsilon_i \geq 0 \tag{5}$$

in which ω is a weight vector, ρ is an offset that parameterizes a hyperplane in the feature space associated with a kernel, x_i is the *i*-th sample from a data set with *n* samples, ϑ is a parameter that characterizes the solution by setting an upper bound on the fraction of outliers and, ε_i is a variable that allows some data to lie within the margin.

III. Proposed Model: Feature Extraction Based on Convolutional Neural Networks + OCSVM

The proposed feature extraction system is modeled as a convolutional neural network. It consists of three pairs of alternating convolutional and max-pooling two-dimensional layers, followed by a densely connected one-dimensional layer, which provides the system output. We adopted a two-dimensional configuration for the convolutional and max-pooling layers due to the kind of input data, *i.e.* two-dimensional 128x128 images. The strides of those layers were 1 and 2, respectively. The model architecture was inspired by [13, 14], and it is illustrated in Fig. 2.

The model is expected to work as a regressor, *i.e.* we trained it to fit a set of target distributions, and not to classes. Those distributions were randomly defined because we do not intend to find an optimal set of values for this problem, but to show that training neural networks, *i.e.* the feature extractors, to learn a predefined pattern can improve the anomaly detection process. The target values are features that are used by the OCSVM to perform the anomaly detection. We used the backpropagation algorithm to train the model in a supervised way. The model was trained for 50 epochs, and the batch size was 144.

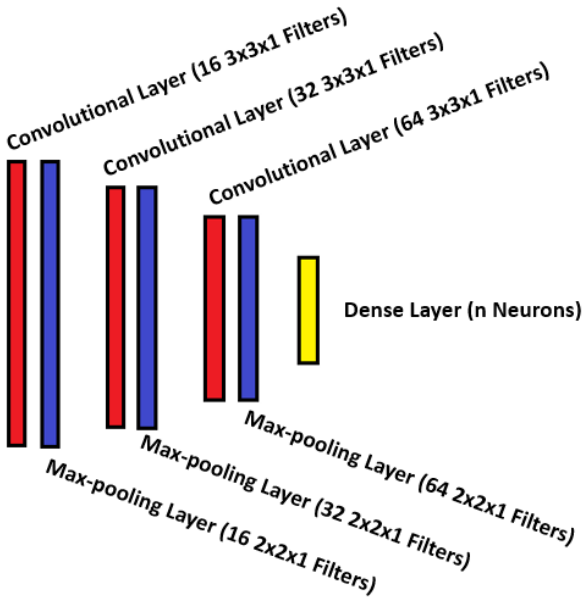

Fig. 2. The configuration of the proposed feature extraction model.

The activation functions employed on the convolutional layers are rectified linear units (ReLU) since they have provided good results in several applications when convolutional neural networks are used [15]. We also used the ReLU as the activation function of the output layer, since the proposed solution does not require the model output to have an upper boundary. The loss function adopted is the mean squared error, since this is a regression problem.

IV. Methodology

A. Data set

We employed a gearbox fault diagnosis data set to assess the performance of the proposed solution. This data set contains 18,000 spectrograms of vibration signals collected from a gearbox operating according to multiple scenarios, *e.g.* different rotation frequencies and load conditions. All spectrograms are 128 x 128 grayscale images. One example is depicted in Fig. 3.

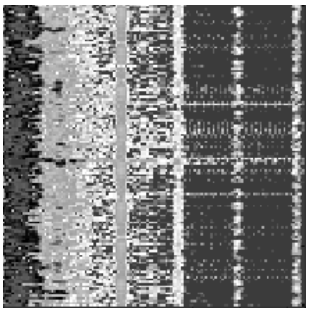

Fig. 3. Spectrogram that belongs to the gearbox fault diagnosis data set.

Those spectrograms are divided into 10 classes, in which the first one regards the normal scenario, and the other 9 classes refer to 9 fault severity levels. Each level regards a tooth breakage percentage of one gear in the gearbox. Those levels are exposed in Fig. 4 and Table I. The normal scenario does not present tooth breakage.

Fig. 4. All ten classes of gear tooth breakage [14].

Table I – Damage severity levels of the gear tooth breakage

Code	Damage (mm)	Tooth remaining percentage (%)
P1	0.00	100.00
P2	2.37	88.42
P3	4.00	80.42
P4	5.73	71.94
P5	7.60	62.81
P6	10.57	48.29
P7	12.37	39.48
P8	14.33	29.85
P9	17.15	14.36
P10	20.43	0.00

B. *Experimental setup*

We compared the proposed solution, which was described in Section III, with two anomaly detection techniques commonly found in the literature. The first technique regarded using autoencoders to reconstruct the input data, so we could define a threshold based on the reconstruction error and use it to detect the anomalies. The architecture of those autoencoders is presented in Fig. 5. The encoder presents the same structure informed in Fig. 2, while the decoder is symmetrical to the encoder relative to the densely connected layer.

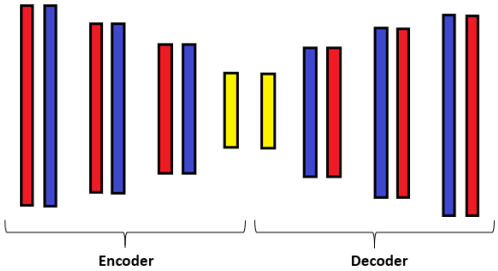

Fig. 5. The configuration of the autoencoder.

The second technique employed the encoders of those autoencoders as feature extractors. They worked together with a shallow machine learning technique, the OCSVM, which performed the anomaly detection by using the features extracted.

We did not regard in this work the comparison with a classifier-based anomaly detection approach, as in [13], since it supposes that we have prior information about the abnormal classes, and we intend to use only the information provided by the normal class to train the anomaly detection models. The spectrograms of anomalous classes are used only to evaluate the model.

The data set used to train those models consisted of 1440 spectrograms, which belonged to the normal class. The test set, on the other hand, contained 2160 spectrograms, in which 360 belonged to the normal class and the remaining ones were divided among the faulty classes, *i.e.* 200 spectrograms by class. Those spectrograms were randomly selected. We used this number of anomalous spectrograms not to distort the evaluation metrics since the number of anomalous images is much higher than the number of normal ones. 15 models were trained for 50 epochs in each scenario. The training process was performed on Google Colaboratory [20]. All scripts were written in Python programming language [21].

V. Results and Discussions

The first analysis was about the anomaly detection based on the reconstruction error. We used the model configuration illustrated in Fig. 5 to build the autoencoders. We considered 4 different sizes of dense layers: 32, 64, 96 and 128 neurons. The threshold adopted to distinguish between normal and abnormal samples was the 95th percentile of the reconstruction error distribution, which was obtained from the training data. Each model had its own threshold. The average classification metrics for 60 trained models (15 in each scenario, *i.e.* size of dense layer) are listed in Table II, in

which n means the number of neurons in the dense layer. Those results regard only test samples.

Table II – Average classification metrics for the anomaly detection based on the reconstruction error

n	Accuracy	Precision	Recall	F1-Score	AUC
32	**0.518**	**0.251**	**0.950**	**0.397**	**0.614**
64	0.488	0.239	0.947	0.382	0.607
96	0.477	0.236	0.945	0.377	0.604
128	0.451	0.227	0.941	0.365	0.597

We observe that the best performance was achieved by the autoencoders with 32 neurons in the dense layer. However, the results of accuracy, precision, F1-Score and AUC were inferior for all configurations, *e.g.* for n = 32 the models classified correctly just over half of the test samples. The recall and precision scores help to understand what happened. The high recall shows that the model has good results regarding the normal class, while the low precision shows a high occurrence of false positives concerning the number of true positives, suggesting that many abnormal samples were incorrectly classified as normal. Fig. 6 shows the average accuracy when the autoencoders have dense layers with 32 neurons, regarding each class individually. Although the anomaly detection regards only normal and abnormal classes, the damage levels are considered in this analysis only to understand the strengths and weaknesses of each approach better.

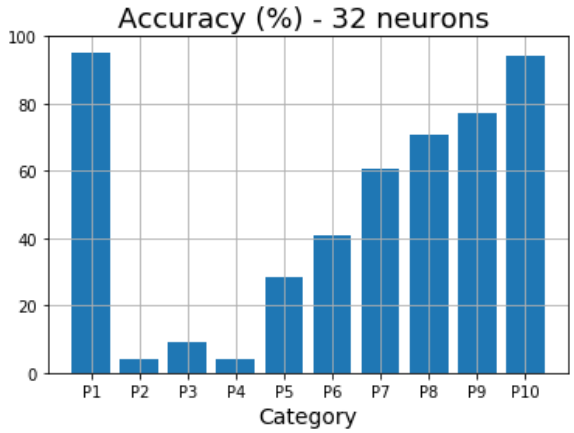

Fig. 6. Average accuracy of anomaly detectors based on the reconstruction error, considering 32 neurons on the dense layer, and regarding the normal class (P1) and all damage classes (P2 to P10).

According to Fig. 6, the accuracy tends to increase as the tooth damage level raises for the anomalous classes. It suggests that small damages on gears did not result in significant changes in the patterns of reconstructed anomalous spectrograms. This way, the anomaly detection based on reconstruction error presented the worst results in those scenarios. On the other hand, more severe damages resulted in higher hit rates. Figs. 7 and 8 help to understand this problem. They show the reconstruction error histograms regarding the normal class (P1), and the anomalous classes P2 and P10, *i.e.*, the minimum and maximum damage levels. We observe overlapping distributions in Fig.7, making it difficult to define normal and abnormal data by setting a threshold. Fig. 8 shows that this problem is smaller regarding the abnormal class P10.

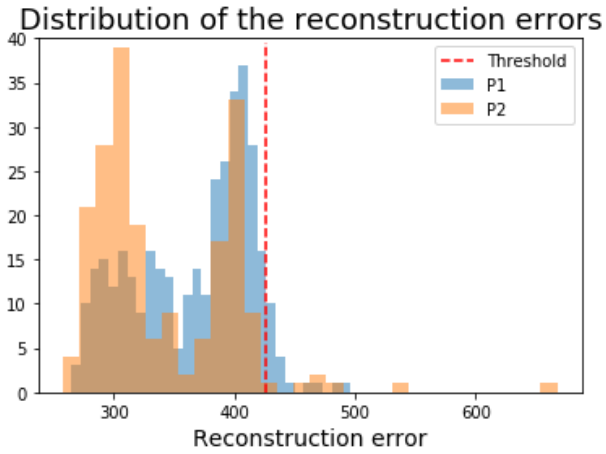

Fig. 7. Distribution of the reconstruction errors regarding classes P1 (normal) and P2 (anomaly).

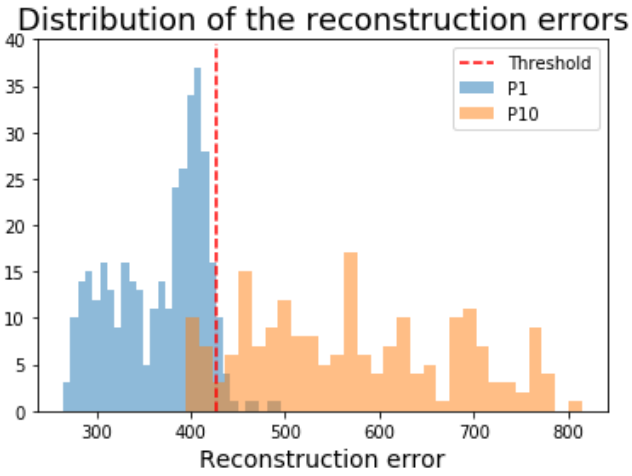

Fig. 8. Distribution of the reconstruction errors regarding classes P1 (normal) and P10 (anomaly).

The second analysis regarded the use of encoders as features extractors. These features were used by the OCSVM to perform anomaly detection. We used the encoders of the same autoencoders employed in the previous analysis. The results, *i.e.* classification metrics, are listed in Table III. The OCSVM was trained to classify 95% of the training data as normal, while the 5% remaining were classified as the anomaly.

Table III – Average classification metrics for the anomaly detection based on encoders + OCSVM.

n	Accuracy	Precision	Recall	F1-Score	AUC
32	0.844	0.558	0.949	0.691	0.773
64	0.865	0.580	0.954	0.715	0.785
96	0.851	0.598	**0.961**	0.723	0.794
128	**0.902**	**0.656**	0.959	**0.773**	**0.823**

This combination presented significantly better results regarding the ones based on the reconstruction error, *e.g.* the average accuracy for n=32 in this new scenario was about 63% higher regarding the previous one. However, the best results were provided by the encoders with 128 output neurons, *i.e.* the autoencoders with 128 neurons in the dense layers. It suggests that a higher amount of information (128 inputs) improved the OCSVM capability of defining a suitable decision border. Also, the scenario with 96 output neurons presented the highest average recall, but the remaining metrics were lower regarding the scenario with 128 neurons. It means that systems for n=96 have less false negatives, *i.e.* less false alarm rates, although their capability of detecting anomaly is not as good as the one of systems for n=128.

Fig. 9. shows the same information presented in Fig. 6, *i.e.*, the average accuracy regarding each class individually. As we observe, the main improvement provided by using encoders and OCSVM is a higher accuracy regarding the classes related to smaller damages. It means that a more substantial amount of information, *i.e.* 128 features instead of just the reconstruction error, allows the system to identify more relevant differences among closer classes such as P1 and P2, for example. It helps to understand the advantages of our proposal over the one based on the reconstruction error.

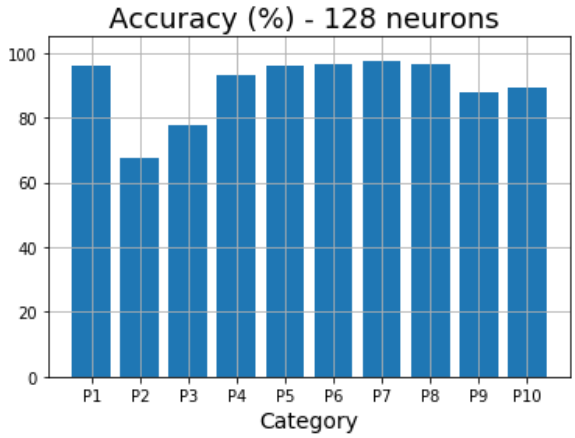

Fig. 9. Average accuracy of anomaly detectors based on encoders and OCSVM, considering 128 neurons on the output layer, and regarding the normal class (P1) and all damage classes (P2 to P10).

The last analysis regarded the proposed solution, *i.e.*, a feature extractor trained to learn a predefined and randomly chosen pattern. We used the configuration illustrated in Fig. 4, with 4 different sizes of output layers: 32, 64, 96 and 128 neurons. The feature extractor based on a CNN was trained to learn features with random values between 0 and 10. We have tested random values with different upper bounds, *e.g.* 1, 2.5, 5 and 10, and the best results were achieved using 10. We did not test higher values for the upper bound. However, as the results improved from 1 to 10, we have reasons to believe that higher values of upper bounds can improve the system performance even more, but it will be analyzed on future work. The high range of expected output values, *i.e.* up to 10, was the reason to adopt the ReLU activation function also for the output layer.

The average results of the 60 trained models (15 in each scenario) are listed in Table IV, in which n means the number of neurons in the dense layer. The OCSVM was also trained to classify 95% of the training data as normal. Those results regard only test samples.

Table IV – Average classification metrics for the anomaly detection based on CNN + OCSVM.

N	Accuracy	Precision	Recall	F1-Score	AUC
32	0.940	0.764	0.937	0.841	0.875
64	0.944	0.774	0.941	0.849	0.881
96	**0.948**	**0.789**	0.946	**0.860**	**0.890**
128	0.946	0.780	**0.947**	0.855	0.884

Table IV shows that the best performance was achieved by the models with output layers with 96 neurons. However, the scenario with 128 neurons presented the highest recall values. It suggests that models in this scenario, *i.e.*, 128 neurons, presented lower false alarm rates. On the other hand, they have more chances to misclassify anomalous examples. Also, we observe that all scenarios presented results very close to each other.

Fig. 10 shows the average accuracy for models that have output layers with 96 neurons, regarding each class individually. As in previous approaches, we observe that the best results were achieved by those classes with higher gear damages, although the overall accuracy values regarding all classes were higher with respect to the other anomaly detection approaches.

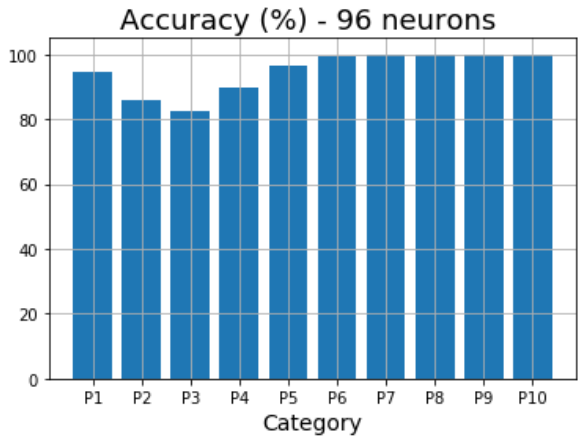

Fig. 10. Average accuracy of anomaly detectors based on CNNs and OCSVM, considering 96 neurons on the output layer, and regarding the normal class (P1) and all damage classes (P2 to P10).

In addition, the results achieved by combining the CNN-based feature extractor with the OCSVM were superior to the ones presented by the previous solutions, *e.g.*, the worst average accuracy value of the proposed solution was 4.2% higher than the best solution provided by the scenario regarding the encoder + OCSVM approach, and more than 81% higher than the best solution provided by the approach based on reconstruction error. Regarding the higher average accuracy achieved by the proposed solution, those improvements were 5.1% and 83%, respectively. Table V shows the best results achieved by all the techniques we assessed. We observe that the proposed solution presented the best performance regarding 4 out of 5 metrics. The exception was the recall, suggesting that our anomaly detection system performs better at identifying anomalies. On the other hand, the false alarm rate is higher regarding the other techniques, but the recall values are close to each other.

Table V – Best average classification metrics regarding the assessed techniques

Approach	Accuracy	Precision	Recall	F1-Score	AUC
Reconstruction Error	0.518	0.251	0.950	0.397	0.614
Encoder + OCSVM	0.902	0.656	**0.959**	0.773	0.823
CNN + OCSVM	**0.948**	**0.789**	0.946	**0.860**	**0.890**

Those results show that training the feature extractor to learn a specific distribution may be better than training models in unsupervised way to extract features, or to use the reconstruction error. In unsupervised training, the model learns the average characteristics of a given data set. Regarding the anomaly detection problem, such characteristics may or not belong to abnormal samples as well, what may decrease the system performance.

On the other hand, by training a model to learn a specific distribution, we can create relations among input data characteristics and output features. Those relations are built in such a way that, when we provide an input with different characteristics regarding the inputs used to train the model, the output features are different from the ones expected. This way, an algorithm such the OCSVM use these features and identify normal and abnormal behaviors.

VI. Conclusion

This paper presented a new solution based on hybrid systems for the anomaly detection problem. The proposed approach combined feature extraction with one-class classifier. Our contribution focused on the feature extractor, which was a convolution neural network trained for a regression task to learn a predefined distribution. Their outputs fed the one-class support vector machine, which performed the anomaly detection.

We assessed the performance of the proposed system on a gearbox diagnosis data set. We also compared the performance of the proposed approach with anomaly detection solutions commonly found in the literature. After performing a few analyzes, we observed that the proposed anomaly detection technique provided better results regarding the other solutions. We believe that training the feature extractor allowed the model to create more specific relations among input data characteristics and output features, instead of just learning general attributes of the training data, as unsupervised trained usually do.

As a future work, we could perform a deeper investigation about the influence of the target distribution on the results achieved by the feature extractor with OCSVM. Also, we could look for ways to determine the target distribution autonomously, aiming to improve the system performance. Other anomaly detection algorithms could be assessed as well, *e.g.* k-nearest neighbors.

VII. Aknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

VIII. Bibliography

[1] M. Raza and U. Qayyum, "Classical and Deep Learning Classifiers for Anomaly Detection," in *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, Islamabad, 2019.

[2] Q. Tian, J. Li and H. Liu, "A Method for Guaranteeing

Wireless Communication Based on a Combination of Deep and Shallow Learning," *IEEE Access*, vol. 7, pp. 38688-38695, 2019.

[3] V. Chandola, A. Banerjee and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[4] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey.," *arXiv preprint arXiv:1901.03407*, 2019.

[5] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.

[6] O. M. Ezeme, Q. H. Mahmoud and A. Azim, "Dream: deep recursive attentive model for anomaly detection in kernel events," *IEEE Access*, vol. 7, pp. 18860-18870, 2019.

[7] W. Chu, H. Xue, C. Yao and D. Cai, "Sparse coding guided spatiotemporal feature learning for abnormal event detection in large videos," *IEEE Transactions on Multimedia*, vol. 21, no. 1, pp. 246-255, 2018.

[8] S. Garg, K. Kaur, N. Kumar and J. J. Rodrigues, "Hybrid Deep-Learning-Based Anomaly Detection Scheme for Suspicious Flow Detection in SDN: A Social Multimedia Perspective," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 566-578, 2019.

[9] W. Lu, Y. Cheng, C. Xiao, S. Chang, S. Huang, B. Liang and T. Huang, "Unsupervised sequential outlier detection with deep architectures," *IEEE transactions on image processing*, vol. 26, no. 9, pp. 4321-4330, 2017.

[10] S. Xu, H. Wu and R. Bie, "CXNet-m1: Anomaly detection on chest X-rays with image-based deep learning," *IEEE Access*, vol. 7, pp. 4466-4477, 2018.

[11] Z. Li, A. L. G. Rios, G. Xu and L. Trajković, "Machine Learning Techniques for Classifying Network Anomalies and Intrusions," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, Sapporo, 2019.

[12] M. B. Fadhel and K. Nyarko, "GAN Augmented Text Anomaly Detection with Sequences of Deep Statistics," in *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, Baltimore, 2019.

[13] R. Monteiro, C. Bastos-Filho, M. Cerrada, D. Cabrera and R. V. Sánchez, "Convolutional neural networks using fourier transform spectrogram to classify the severity of gear tooth breakage," in *2018 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*, Xi'an, 2018.

[14] R. P. Monteiro, M. Cerrada, D. R. Cabrera, R. V. Sánchez and C. J. Bastos-Filho, "Using a Support Vector Machine Based Decision Stage to Improve the Fault Diagnosis on Gearboxes," *Computational Intelligence and Neuroscience*, 2019.

[15] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352-2449, 2017.

[16] S. M. Abbas and S. N. Singh, "Region-based object detection and classification using faster R-CNN," in *2018 4th International Conference on Computational Intelligence & Communication Technology (CICT)*, Ghaziabad, 2018.

[17] J. Patterson and A. Gibson, Deep learning: A practitioner's approach, Sebastopol: O'Reilly Media, Inc., 2017.

[18] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, Bellevue, 2012.

[19] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor and J. C. Platt, "Support vector method for novelty detection.," in *Advances*, Denver, 2000.

[20] Carneiro, T., Da Nóbrega, R. V. M., Nepomuceno, T., Bian, G. B., De Albuquerque, V. H. C., & Reboucas Filho, P. P. (2018). Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access*, *6*, 61677-61685.

[21] Van Rossum, G., & Drake, F. L. (2011). *The python language reference manual*. Network Theory Ltd.