2022 IEEE 10th Jubilee International Conference on Computational Cybernetics and Cyber-Medical Systems (ICCC 2022) | 978-1-6654-8177-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCC202255925.2022.9922678

Predictive maintenance of pumping system bearings with evolved autoencoders

Sašo Pavlič
Faculty of Electrical Engineering and Computer Science
University of Maribor
Koroška cesta 46, 2000 Maribor
Slovenia
Email: saso.pavlic@student.um.si

Nicholas Young
Madison, WI
United States
Email: nyoung0@gmail.com

Sašo Karakatič
Faculty of Electrical Engineering and Computer Science
University of Maribor
Koroška cesta 46, 2000 Maribor
Slovenia
Email: saso.karakatic@um.si

***Abstract*—Industry 4.0 introduced the application of various remote sensing and intelligent or autonomous decision-making to the manufacturing floors. One of the most important aspects is the maintenance of the machinery and equipment preemptively so that their breakage does as minimal damage (in downtime, safety, etc.) as possible. Analyzing the machinery data with machine algorithms (especially artificial neural networks (ANN)) plays a crucial role. However, designing and training ANN algorithms is still a time-consuming and complex process with various remaining issues, including generalization, the necessity for large datasets, and high time complexity. As a result, a few decades ago, neuroevolution was developed to improve the ANN construction process. Recently, it has gained considerable attention alongside a rise in computational power. In this paper, we use the power of neuroevolution to design a specific type of ANN called an autoencoder (AE) which can be used for a particular type of task, such as anomaly detection (AD). Our proposed method, has shown promising results. Our approach achieved a peak test accuracy of 75% in 0.4 quantiles and an area under the curve (AUC) of 73% in AD on the predictive maintenance dataset. Our proposed method can serve as a starting point for further research.**

***Index Terms*—Predictive maintenance, neuroevolution, autoencoder, anomaly detection.**

I. Introduction

Industrial processes are heavily reliant on the working condition of the machinery and equipment which, in turn, influence work conditions, the safety of workers, and profits. With the rise of Industry 4.0, where continuous monitoring of the machinery and equipment is available with various sensors, the optimization of industrial processes has become one of the main pillars of an efficient and safe industry [1]. Thus, when the sensor data indicates that the machine or equipment is on a trajectory towards failure, preemptive maintenance should be done. This preemptive step minimizes downtime of the industrial process, prevents total breakdown of the machine or equipment, and enables optimization of the whole industrial process. This process is called predictive maintenance [1]–[3].

Unfortunately, data from machinery sensors is not enough, as the patterns of failure are normally too hard to recognize by any human expert. Thus, the application of machine learning methods on this sensor data has seen wide adoption in the industry and gained traction in the research community [4], [5]. Even though machine learning approaches are already being widely used for predictive maintenance, there is still a lack of serious research done on how these approaches should be optimized to work without bias and as efficiently as possible [5], [6]. Various nature-inspired meta-heuristic optimization approaches have been previously used to tackle this problem [7]–[9], yet there is still no serious application or research utilizing these nature-inspired optimization approaches for predictive maintenance.

To address the mentioned issues and potential solutions, we propose a method utilizing a modified version of Neuroevolution of Augmenting Topologies (NEAT) [10] called autoencoders-neat for anomaly detection (ANAD). ANAD utilizes the power of neuroevolution to construct novel autoencoder neural network architectures that can be used for unsupervised anomaly detection. We summarise our contributions as follows:

- We present a modified version of the NEAT algorithm for constructing the autoencoder architectures.
- We present a new evaluation technique for measuring the performance of these autoencoder architectures.
- We apply the ANAD method on real life predictive maintenance data from pumping system bearings located in a sewerage treatment plant.

II. Anomaly detection for predictive maintenance

Predictive maintenance of machinery sensors is usually treated as a supervised machine learning problem, as the machinery breakdown events are labelled [11]. However, in cases where there is no existing data of machinery breakdown, and the implementation of a predictive maintenance system prevents the long-term collection of such data, another approach has to be taken. In these cases, unsupervised anomaly detection (sometimes called novelty detection or outlier detection) has to be adopted, where preventive maintenance is commenced when an unusual operation state of the machinery is reached.

This research was funded by the Slovenian Research Agency (research core funding No. P2-0057)

The strategies employed for a given anomaly detection challenge are determined by the presence of labels in the dataset. In some circumstances, future projections are based on prior observations, whilst in others, they are primarily dynamic and must be recognized ad hoc. Three techniques for anomaly detection are primarily used [11]: (1) the supervised technique, where labels on anomalous and normal data instances are given; (2) a semi-supervised technique where only normal instances are annotated with labels and others are indicating a pre-captured spectrum of anomalies; (3) An unsupervised technique where we do not have any information on which instance is normal or anomalous.

When dealing with machinery sensor data without any labels, the last technique is considered. In this case, the labels do not play any role during the training of the anomaly detection model. In other words, the anomaly detection model learns to distinguish normal data from unusual data, and hopefully that unusual data is the data that we should perform preventive maintenance on.

A. Autoencoders for anomaly detection

In a paper published by Pierre Baldi [12], the author defined AEs as simple circuits capable of transforming inputs into outputs with as little distortion as possible. They first appeared in academic papers in the 1980s to solve the issue of backpropagation in the topology of conventional neural networks (NNs). The idea behind an AE is that the input data serves as a teacher for the final output. In this way, we cut down on the need for any additional system which would rate our NN output.

Typically, an AE is composed of three parts. The *encoder*, which is a feedforward NN responsible for compressing the input into a smaller vector representation. Then comes an abstract concept known as the *latent space*, in which just the most important characteristics are preserved. Practically speaking, this is not a component of the NN. Rather, it is an encoder output. This compressed data is eventually sent to a *decoder*, which is responsible for restoring the data to its original dimensions with as little reconstruction loss as possible. The encoder and decoder could be mirrored if the AE is symmetrical (in NN depth and layers), but this is not necessarily the case. An example of such an AE is described in Figure 1.

Aside from the numerous applications of AEs in the industry [13]–[15], they can also be used effectively to discover anomalies in datasets. The learning process of an AE is designed in such a way that it learns the latent space better for normal versus anomalous samples during training. The key is that normal data represents the great majority of samples during training. As a result, it will have a low reconstruction error on normal data and a high reconstruction error on anomalies. The learned NN model can then be used to detect anomalies.

Using deep hidden layers, as is common with all types of NNs, can result in many advantages. The same holds for encoders and decoders. Deeper AEs outperform shallow AEs in terms of data compression [15], but choosing the deep

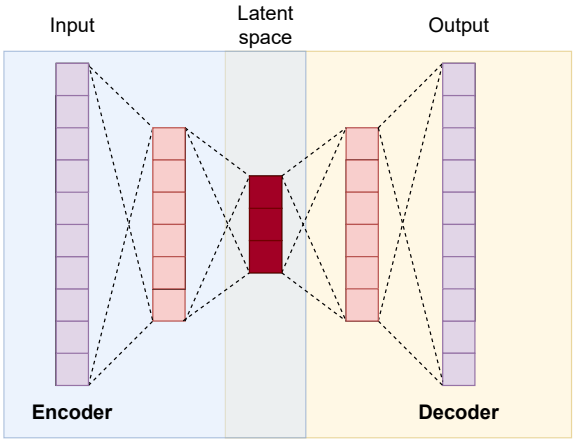

Fig. 1. Example of symmetrical AE components.

architecture is tedious work, which is usually done by manual selection by a machine learning practitioner. Thus, other techniques have to be used to solve the problem of NN architecture selection. There is already a wide variety of research done on architecture building with meta-heuristic nature-inspired optimization techniques, but they haven't been applied to the problem of predictive maintenance with autoencoders.

III. Neuroevolution of Augmenting Topologies

Back in the 2000s, there were theories on how neuroevolution (NE) could contribute to designing the topology of neural networks, since at this time there were only recent studies on weight evolving artificial neural networks. At this time, fixed-topology NE had already outperformed topology-evolving systems [15] on the pole balancing task, which was and still is a good benchmark for NE or reinforcement learning. This suggested that there is an open question of whether Topology and Weight Evolving Artificial Neural Networks (TWEANNs) can enhance performance. When striving to outperform fixed-topology NE, the NEAT [10] authors came up with three important solutions. One is to utilize historical markings to align genes of the same origin. This is important when performing crossover between genomes in a meaningful way. The second challenge is to ensure that topological innovation does not disappear from the population prematurely. This is accomplished by dividing each innovation into separate species. The third challenge is minimizing the topology over time without the use of a sophisticated fitness function. The authors' answer was elegant and simple: start with a minimal structure and expand only when necessary.

In NEAT, during the evolution process, the hidden nodes of the architecture are evolved. Hidden nodes can be added or deleted as the topology of the network grows. Connection genes specify in and out nodes, the weight of the connection, an enabled parameter indicating if the connection is alive, and a parameter innovation number indicating when in the evolutionary process the gene emerged.

As with every other NE system, a *mutation* might appear in a connection weight in NEAT. However, structural mutations

can occur in two ways: by adding/deleting a connection or by adding/deleting a node in the topology. When a new connection is added between a start and end node, it is assigned a weight at random. When a new node is added, it is placed between two existing connected nodes. This causes the prior connection to be deactivated. This previous connection and its weight are now linked to the new node, and the new node is linked to the previous end node with a weight of 1. Both types of mutation are described in Figure 2

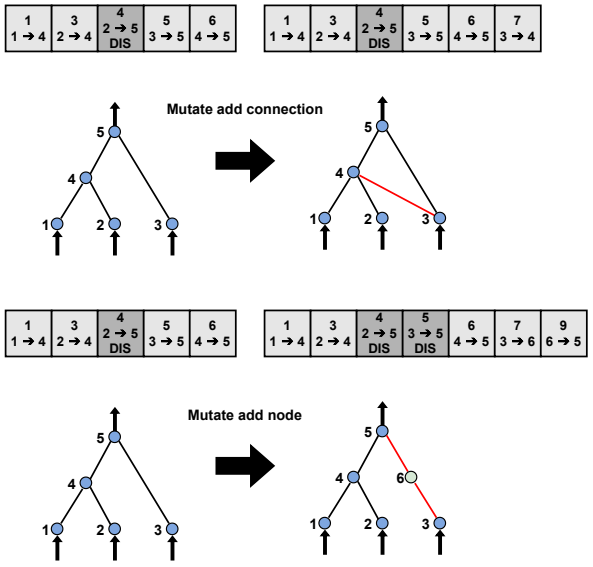

Fig. 2. The two types of structural mutations in NEAT.

Before performing *crossover*, the algorithm must be able to tell which individuals in the population are similar, since blindly choosing individuals can result in non-functional models. Having multiple ways to express a solution to a weight optimisation problem with a NN is referred to in the literature as a *Competing Convention Problem*. Crossover is likely to result in injured offspring when genomes represent the same solution but do not have the same encoding (each of them is a permutation of the other).

The important solution to get around this problem in NEAT is determining which two genes have the same historical origin (they represent the same structure), as they are both descended from the same ancestor gene at some point in the past. As a result, all the system needs to do is keep track of the historical origins of each gene in the population. When it comes to crossover operations, each gene can be aligned and potentially crossed over, making it less probable that two incompatible individuals will be chosen, resulting in an offspring that isn't functional. Each time a structural mutation happens, such as adding a new node or connection, a *global innovation number* is incremented and assigned to this gene, allowing easy alignment when it comes to breeding two individuals. As the authors of NEAT stated, this operation requires very little computation and is simple, without the need for any topological analysis.

The concept of *speciation* in NEAT stems from the fact that adding new structure to a network usually results in a drop in fitness, which increases the likelihood of it being eliminated from the population before it has had a chance to optimize. As a result, the mechanism for preserving topological innovation is to divide the population into species. By doing so, we can ensure that individuals compete within their own niche (similar topologies) rather than with the entire population. This enables individuals to optimize under softer conditions. We may say that NEAT supports talented/unique/unconventional topologies before they must compete with other niches in the entire population.

NEAT goes even further by introducing explicit fitness sharing. Individuals share their performance across species, promoting higher-performing species while enabling other species to experiment with structural optimization before being out-evolved.

To keep reproduction within the species, the original authors needed to define a rule for what makes two individuals compatible. In nature, a species is defined as having the ability to reproduce solely within its species and not with individuals from other species. In NEAT, the same set of rules apply, using the historical markings to track the roots of individuals. When observing individuals, if they have too many excess and disjoint genes, they are probably incompatible with each other. The following equation (1) describes how compatibility distance is calculated. When the distance between two individuals is less than a specified threshold, they are regarded as belonging to the same species.

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \overline{W} \tag{1}$$

where E and D denote the number of excess and disjoint genes; N is a total number of genes $\overline{W}$ is representing the average connection weight difference; Coefficients are defined by the user for altering the significance of these factors.

A. Minimal Structure

As noted in the original NEAT paper, they intended to avoid the problems associated with randomly creating a complicated topology that performs well on a specific task and then pruning to find a simpler topology. Since the initial search space is larger than we may require, there is no guarantee that a smaller topology would perform as well. Instead, the goal is to start with the simplest topology possible and evolve it over time, only if it is beneficial through fitness evolution.

It starts with a topology using no hidden nodes. Each individual in the initial population is nothing more than a set of input and output nodes connected by many connection genes. This would be insignificant on its own, but when combined with the concept of speciation, it becomes a strong technique for building on top of minimal topologies and achieving high performance for a given complexity.

IV. Proposed Method

In the previous section, we discussed the fundamental concepts of the NEAT algorithm. In this section, we will show

how we can impose a specific NN topology on NEAT. The reason for this is that we want to design the topologies (nodes and connections) of NNs that will represent AEs. This implies that NEAT must create separate encoder and decoder genomes that are compatible with one another. The middle layer, often known as a bottleneck layer [16], is used to ensure compatibility between them (latent space). This prevents NEAT from directly connecting nodes in the encoder to nodes in the decoder. Instead, it connects the layers as follows: encoder, bottleneck, decoder. By doing this, information is forced to flow through the bottleneck.

Thus, the AE genotype holds two separate genomes: one for the encoder and one for the decoder. Each of these is evolved separately to ensure true exploration of the search space. Regardless of that, they are still compatible through the bottleneck layer.

a) Individual initialisation: NEAT constructs the genotype based on the supplied genome configuration during the AE genome creation step. This configuration method determines if connections to hidden nodes are permitted, as they must be connected in this case for the encoder and decoder supplied by the configuration parameter.

b) Crossover: Steps in crossover operations remain the same as they are in the original source code. As in the section on *individual initialisation*, the steps here are duplicated for the encoder and decoder. Therefore, the crossover method is inheriting the connection and node genes separately for the encoder and decoder. In crossover operations, excess and disjoint genes are inherited by the fittest parent, while homologous are combined from both parents.

c) Mutation: Separate structural mutations are made on the encoder and decoder genomes. This means that a node or connection in the topology can be added or removed independently on one side of the AE without requiring the other side to perform the same operation. That algorithm does not constrain AE topologies to being symmetrical.

d) Distance: Measuring the distance between the two AE genotypes is done by first measuring the distance between encoders and decoders separately and then summing up the values to get the actual difference between the two AE models. The result of this method is used to calculate genome compatibility in the context of speciation.

A. Anomaly detection with ANAD

In our predictive maintenance scenario, we perform unsupervised anomaly detection, which means that AE topologies need to be trained on a dataset that contains both normal and anomalous data instances. The idea is that models learn relatively well to process the majority of data (normal instances) but poorly to process the minority of data (anomalous instances). The data instances that account for more than half of all data instances make up the majority. The higher the ratio of normal to anomalous instances, the more accurate the ANAD method will be in making predictions.

Distances or densities are commonly used to determine what is normal and what is an outlier [17]. It is worth noting that NEAT's NN models are not trained, but rather built from the ground up. As a result, at the end of the evolution phase, the algorithm returns the best performing individual. Simply put, the individual is the autoencoder model that achieved the highest metric AUC score. The metric score was calculated by running the anomaly detection process on a testing dataset for each individual (genome) separately. General steps in the neuroevolution process are described in Algorithm 1.

Algorithm 1 Proposed method **ANAD**

Input: $dataset, iterMax, fitThresh$
Output: The best autoencoder

population ← *initialPopulation()*
species ← *speciation(population)*
iter ← 0
bestFitness ← *evaluate(population, dataset)*
while *iter* < *iterMax* **AND** *bestFitness* < *fitThresh* **do**
  parents ← *selection(population, species)*
  population ← *crossover(parents)*
  population ← *mutate(population)*
  bestFitness ← *evaluate(population, dataset)*
  species ← *speciation(population)*
  iter ← *iter* + 1
end while
bestGenome ← *best(population, dataset)*
return *bestGenome*

In summary, Algorithm 1 begins with the simplest possible AE topology (input layer, latent space, output layer) and then allows NEAT to evolve the NNs until the evolution stops, which is defined by a condition. Each genome is evaluated based on the accuracy of anomaly detection in the testing dataset. The chosen metric for evaluation is the receiver operating characteristic (ROC) curve and AUC score. When the condition is met, the best performing genome (AE model) is returned.

V. Experiments and Results

In this paper we focus on a real-world predictive maintenance case of pumping system bearings. The data was gathered from a sewerage treatment plant and consists of sensor data acquired by monitoring the bearing movement, such as type, temperature and vibration of the Driving End (DE) and Non-Driving End (NDE) bearings of a pump. The dataset's feature values are represented solely by numerical values, with no missing values and a weak correlation (average correlation of features is 0.215). We standardised the data for each feature before using it, which enables the NN to be generated with weights that are more similar across the features, resulting in more uniform topologies.

A. Environment setup

The experiment was carried out using the Python programming language with the libraries: NEAT-Python [18] for NN topology construction, Scikit-learn [19] for evaluating NN models, NumPy [20] for working with arrays and NEAT based

autoencoders [21]. The following computational resources were used for the development and training environment: Razer Blade 15 Advanced (RZ09-036) with Intel i7-10875H CPU, Nvidia GeForce RTX 3080 with 8 GB GDDR6 memory and 6144 CUDA cores GPU, and 32 GB DDR4 RAM. Running on Ubuntu 20.04.2 LTS.

B. Evaluation of neuroevolution

First, we obtained the fitness function results throughout the neuroevolutionary process. This helped us to determine if the fitness of each individual is increasing from generation to generation. Since our proposed method employs an evolutionary process, it is expected that individual fitness capabilities should improve over time. Results are displayed in Figure 3, where we can see how in the first 200 generations fitness values increased drastically from 0.35 to 0.65 on the ROC curve. This drastic jump is related to the quick transfer of good genes into new generations. Here we can assume that in the beginning, the neuroevolution algorithm is not aware of what makes a good AE for a given dataset. As we proceed through the generations, we can see how the growth in fitness value levelled out and then continued to develop steadily. This is valid until the 350th generation, after which the fitness value's development becomes less steep. This happens due to a local optimum problem, which means crossover and mutation between individuals in the population does not produce a significantly better individual in terms of fitness value.

Another observation we can make from Figure 3 is a red line (the best individual in the population) and upper green line (+1 standard deviation) which shows how better-performing individuals can boost the entire population's performance. This impact is caused by the fact that nature-inspired algorithms favour superior individuals to better exploit a local search space. Another mechanism for overcoming this problem is integrated into NEAT. As indicated in section III, the formation of multiple species softens the impact of this problem. As a result, the average fitness level of individuals is increasing.

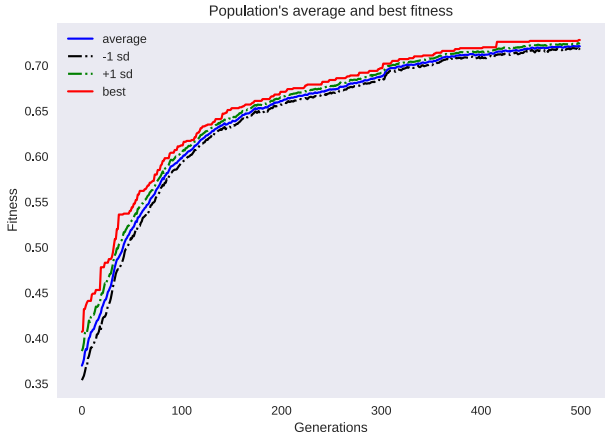

Fig. 3. Progression of solutions with neuroevolution through the generations.

C. Best performing topology

At the end of neuroevolution, the best performing individual is selected. This is done by comparing the fitness values, and the individual with the highest value is considered to be the best performing. The topology of the winner model is divided between the encoder and decoder, as seen in Figure 4.

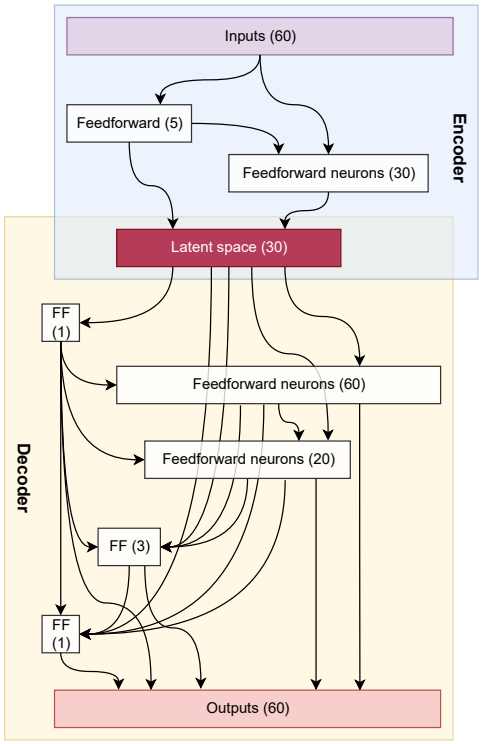

Fig. 4. Best performing topology (simplified).

Together they are forming the AE model which can accept input data and return the output. The winning topology consists of an encoder side of 35 nodes, of which 5 are hidden, and a decoder side of 85 nodes, of which 25 are hidden. Together, they are forming a sum of 120 nodes in the AE model. This AE model's accuracy (reconstruction of data in comparison to ground truth) reaches a peak value of 0.75 at 0.4 quantile and accuracy of 0.58 around a 0.99 quantile. In the AUC metric, the model reached 0.73 area under the ROC curve. This implies that the AE model was able to achieve relatively good results.

Note that the resulting NN topology from ANAD (Figure 4) is not a structure that we expect from manually built NNs – there are no hidden layers, just hidden neurons without forming any layer structure. Rather, a topology that appears to be biologically inspired. For the sake of simplicity, the topology of the best performing NN is depicted in the form of layers, which include neurons with connections from and to the same set of neurons.

VI. Conclusion

In this paper, we presented a proposed method ANAD. This method is a modification of the existing NEAT algorithm,

where we joined together the existing technologies to detect anomalies. The idea was to use a neuroevolution approach when building up NN topologies. Since many applications in industry are using autoencoders for anomaly detection, we set the objective of a neuroevolution algorithm to design an AE-like NN topology.

Evaluation of the AE model was done on a testing subset derived from the predictive maintenance dataset. Each data instance inputted into the AE model was processed and returned as an output, where we measured the difference between the input and output by calculating the MSE. The next step was to calculate the effectiveness of AD within various quantiles of the MSE results. When a data instance's reconstruction error exceeds a specified threshold, it is considered an anomaly. Once we had evaluation metrics in place, we continue with the neuroevolution process, where each model was evaluated with the fitness function. The result given by the fitness function indicates the performance on anomaly detection measured by the ROC-AUC metric. The higher the AUC-ROC score, the more successful the model is when identifying true positives and false positives. The AE model with the highest fitness result was selected as the winning model.

The selected model was further tested to yield multiple results. We determined, based on the metrics, that our proposed method was proven to achieve surprisingly good results. With ANAD, it is possible to detect more than 93.3% of true positives and 40% of false positives. Nevertheless, we also need to point out that in our work we used a fairly simple metric to measure the model's fitness score and only 500 generations in evolution. The primary goal of this study was to put the proposed method to the test and see if it was useful before moving forward with research in this field.

Our contribution to the field is that, from this point forward, our suggested method can be used for any size and length dataset as long as the parameters in the configuration file are modified accordingly. This has the potential to open many new doors in this field of study and mainly to reduce the human time required when designing AE topologies. In the future, we hope to apply our method to many more industry-specific datasets and a much more computationally powerful computer system, where we would like to run experiments comparing current domain methods to our ANAD.

References

[1] T. Zonta, C. A. da Costa, R. da Rosa Righi, M. J. de Lima, E. S. da Trindade, and G. P. Li, "Predictive maintenance in the industry 4.0: A systematic literature review," *Computers & Industrial Engineering*, vol. 150, p. 106889, 2020.

[2] H. M. Hashemian, "State-of-the-art predictive maintenance techniques," *IEEE Transactions on Instrumentation and measurement*, vol. 60, no. 1, pp. 226–236, 2010.

[3] R. K. Mobley, *An introduction to predictive maintenance*. Elsevier, 2002.

[4] T. P. Carvalho, F. A. Soares, R. Vita, R. d. P. Francisco, J. P. Basto, and S. G. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," *Computers & Industrial Engineering*, vol. 137, p. 106024, 2019.

[5] J. Dalzochio, R. Kunst, E. Pignaton, A. Binotto, S. Sanyal, J. Favilla, and J. Barbosa, "Machine learning and reasoning for predictive maintenance in industry 4.0: Current status and challenges," *Computers in Industry*, vol. 123, p. 103298, 2020.

[6] W. Zhang, D. Yang, and H. Wang, "Data-driven methods for predictive maintenance of industrial equipment: A survey," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2213–2227, 2019.

[7] L. B. Bosman, W. D. Leon-Salas, W. Hutzel, and E. A. Soto, "Pv system predictive maintenance: Challenges, current approaches, and opportunities," *Energies*, vol. 13, no. 6, p. 1398, 2020.

[8] Z. M. Çınar, A. Abdussalam Nuhu, Q. Zeeshan, O. Korhan, M. Asmael, and B. Safaei, "Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0," *Sustainability*, vol. 12, no. 19, p. 8211, 2020.

[9] B. Shaheen, A. Abu Hanieh, and I. Németh, "Fault Detection of a Wind Turbine's Gearbox, based on Power Curve Modeling and an on-line Statistical Change Detection Algorithm," May 2021, accepted: 2021-06-21T05:41:00Z Publisher: Acta Polytechnica Hugarica (Journal of Applied Sciences Hungary). [Online]. Available: https://fada.birzeit.edu/handle/20.500.11889/6756

[10] K. Stanley and R. Miikkulainen, "Efficient evolution of neural network topologies," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, vol. 2. Honolulu, HI, USA: IEEE, 2002, pp. 1757–1762. [Online]. Available: http://ieeexplore.ieee.org/document/1004508/

[11] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[12] P. Baldi, "Autoencoders, Unsupervised Learning, and Deep Architectures," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. JMLR Workshop and Conference Proceedings, Jun. 2012, pp. 37–49, iSSN: 1938-7228. [Online]. Available: http://proceedings.mlr.press/v27/baldi12a.html

[13] L. Gondara, "Medical image denoising using convolutional denoising autoencoders," *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp. 241–246, Dec. 2016, arXiv: 1608.04667. [Online]. Available: http://arxiv.org/abs/1608.04667

[14] T.-H. Song, V. Sanchez, H. EIDaly, and N. Rajpoot, "Hybrid deep autoencoder with Curvature Gaussian for detection of various types of cells in bone marrow trephine biopsy images," *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*, 2017.

[15] A. Radhakrishnan, K. Yang, M. Belkin, and C. Uhler, "Memorization in Overparameterized Autoencoders," *arXiv:1810.10333 [cs, stat]*, Sep. 2019, arXiv: 1810.10333. [Online]. Available: http://arxiv.org/abs/1810.10333

[16] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv:2003.05991 [cs, stat]*, Apr. 2021, arXiv: 2003.05991. [Online]. Available: http://arxiv.org/abs/2003.05991

[17] M. Goldstein and S. Uchida, "A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data," *PLOS ONE*, vol. 11, no. 4, p. e0152173, Apr. 2016, publisher: Public Library of Science. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0152173

[18] CodeReclaimers, "CodeReclaimers/neat-python," Mar. 2022, original-date: 2015-09-26T22:59:53Z. [Online]. Available: https://github.com/CodeReclaimers/neat-python

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[20] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[21] N. Young, "palmettos/neat-autoencoders," Feb. 2022, original-date: 2020-09-08T07:32:54Z. [Online]. Available: https://github.com/palmettos/neat-autoencoders