The 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
18-21 September, 2019, Metz, France

Comparison of Deep Autoencoder Architectures for Real-time Acoustic Based Anomaly Detection in Assets

Maarten Meire, Peter Karsmakers
KU Leuven, Dpt. of Computer Science, TC CS-ADVISE, Kleinhoefstraat 4, B-2440 GEEL, Belgium,
maarten.meire@kuleuven.be, peter.karsmakers@kuleuven.be

***Abstract*—This paper compares several different Autoencoder architectures for unsupervised anomaly detection in acoustic signals. The goal of unsupervised anomaly detection in sound is to detect anomalies without having prior knowledge regarding potential anomalies. Use of autoencoders (AE) to learn a normal model is a state-of-the-art technique for unsupervised anomaly detection. However, the main focus is almost always to increase the difference between the reconstruction error of normal and anomalous data, without taking into account network architecture and speed. This is not a problem when enough computational power is available. However, if the aim is to bring this system to the edge, meaning implementing it on the sensor or close to the sensor, speed and amount of parameters of the network become more important. In this paper we will do a comparative study between different AE architectures. For this comparison both the detection accuracy and the computational complexity will be taken into account. Based on this information it can be decided which AE is most suited to be implemented on hardware for real-time applications.**

***Keywords*—Anomaly detection in sound, deep learning, autoencoder, real-time**

I. INTRODUCTION

There are 2 main types of maintenance schemes for machines, based on if maintenance is done before or after a failure. In the early days the 2nd type was mostly used, but later preventive maintenance, where maintenance is done based on the age of a machine, became more popular. This scheme was further improved by monitoring the health of the machine, this is known as condition-based monitoring [1]. To do this monitoring, machine parameters need to be captured and then analyzed. This analysis can be done either manually, by trained personel, or it can be (partially) automated. In this study we want to focus on automated analysis. A commonly used technique to monitor machine health is vibration analysis, due to the different vibration patterns based on machine condition [2] [3]. However, in recent years computational acoustic monitoring is receiving more attention [4] [5] [6] [7]. Using acoustic analysis provides certain benefits over the traditional vibration analysis. The main benefits are the contactless nature of the sensors and the ability of these sensors to monitor multiple components of a machine at once, by using an array of sensors localization of an anomaly can be done as well [8]. Computational acoustic monitoring has also been used in other areas, for instance, in-home monitoring [9], road surveillance [10].

In this paper we focus on the task of building an acoustic model that detects anomalous acoustic events. Prior knowledge regarding the characteristics of these events can either be well-defined and linked to a specific failure such as a broken rotor bar [4] or non-existing.

For the former, because the anomalies are predefined a dataset with both normal and anomalous data can be collected even though anomalous data is usually rarer than normal data. Given such data set supervised learning methods can be used to estimate the model parameters. In case of the latter only acoustic information collected using normal operation of the asset can be used to estimate the acoustic model parameters. For this purpose, an unsupervised learning strategy is required. Since it is not realistic to enforce anomalies in target machines in a production environment, and the amount of anomalous sounds that occur "naturally" is small making it difficult to collect examples, the task of detecting anomalies in machines is usually tackled as an unsupervised task [11].

Unsupervised anomaly detection is based on the assumption that the "normal" regions in the original or latent features space can be distinguished from "anomalous" regions [11]. A common way to do this unsupervised anomaly detection is outlier-detection. This technique builds a "normal" model which is trained with data from the "normal" regions. This model is then used to calculate the deviation between it and an observed sound. Based on this deviation and a pre-defined threshold, the observed sound is considered anomalous if the deviation is larger than the threshold. Recently different variants of autoencoders (AE) have been used for the unsupervised detection of anomalies in acoustics [5] [9] [7]. AE were chosen since they give good results and are easy to automate for different machines, since they do not need a feature extraction step, which would change based on the machine. However, if the aim is to implement these techniques on or close to the sensor, the model with the best accuracy, might not be the best model to implement on the hardware. For instance it might be better to have a slight

978-1-7281-4069-8/19/$31.00 ©2019 IEEE

loss in accuracy if it significantly speeds up the model and reduces the amount of parameters, hereby reducing the computational complexity of the considered model. For this reason, this paper will validate different AE in terms of both the detection accuracy and the computational complexity, using a real-life data set.

In the literature few examples of semi-supervised learning strategies exist that next to the large amount of unlabeled data also exploit label information in the anomaly detection task such as that in [12] but these are not considered in this article and are left for future research.

The rest of this paper is organized as follows. Section II describes the different anomaly detection methods that will be compared. The dataset and the results of the various methods will be presented in Section III. Finally we conclude this paper in Section IV.

II. Methods

This section briefly summarizes the methods that are being used in the experimental section. All of them are employing an unsupervised learning scheme to fit a model that can detect anomalous acoustics. First, a baseline method will be introduced. Then, the use of AE in anomaly detection will be revisited.

A. Baseline Method Using OC-SVM

To detect anomalous sounds in an unsupervised method, a model that represents normal operation needs to be trained. It has been shown that One-Class Support Vector Machines (OC-SVM) are capable of succesfully doing this [13].

OC-SVM aims at separating all the data points from the origin by means of a hyperplane and tries to maximize the distance of this hyperplane to the origin. It does this by solving

$$\min_{w,\xi,\rho} \frac{1}{2} w^T w + \frac{1}{\nu m} \sum_{j=1}^{m} \xi_j - \rho, \qquad (1)$$

subject to (for j = 1,…,m)

$$w^T \varphi(x_j) \geq \rho - \xi_j, \xi_j \geq 0, \qquad (2)$$

where φ is a function that maps the input data x_j to a feature space enabling non-linear decision boundaries in the input space. ν is the fraction of data that are allowed to be on the wrong side of the margin, w is the parameter vector perpendicular to the hyperplane and ρ is the bias of the hyperplane [13].

B. Autoencoders

Compared to OC-SVM the goal of an AE is to use an encoding network (Ɛ) to create a compact representation and then a decoding network (Ɗ) to reconstruct the original signal from this representation. The parameters of AE are learned using normal data letting them define a transformation function that compresses the normal data while retaining the information that is needed for a proper reconstruction. In case of anomalous data the error of this reconstruction is expected to be higher compared to that in case of normal data. Hence based on the reconstruction error a prediction can be made whether the input is normal or anomalous. This process is shown in Figure 1.

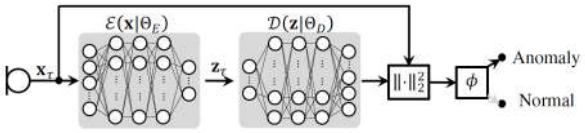

Figure 1. Basic autoencoder structure. Network Ɛ maps the input to a compact representation, then network Ɗ tries to reconstruct the input from this representation. Since these networks are trained on normal data, if x_τ is normal data the reconstruction error should be small. If this reconstruction error is larger than a threshold Φ, the input is considered anomalous [7].

AE have the capability to automatically learn features from low-level signals while OC-SVM requires handcrafted feature engineering to preprocess the low-level signals to a format that can serve as an input to OC-SVM.

The different AE compared in this study are based on 1D-CRNN [7], 2D-CNN [5] and MSCRED with the attention layer removed [14]. 2D-CNN and MSCRED have the same architecture as in the original papers. 1D-CRNN was adjusted to 1D-CNN, where the LSTM layer was removed for increased speed.

III. Experiments

A. Dataset

To compare the previously mentioned techniques we use a dataset that was generated by Siemens Industry Software [15]. This data set contains acoustic data from both healthy and faulty bearings. The data was recorded using a fault simulator test rig from SpectraQuest. Using this simulator data was collected in 4 different scenario's each having different bearing conditions: healthy, inner raceway fault, outer raceway fault and ball defect. In this study we consider the 3 faulty conditions as anomalous and the healthy condition as normal. These conditions were measured for a speed range between 300 and 2700 rpm with an interval of 120 rpm, for a duration of 30.5s for each interval. These experiments were done twice, adding up to 61s for each interval, totalling up to 2562s of normal data and 7686s of anomalous data. This ratio of normal to anomalous data is not standard for anomaly detection, however this is attributed to the fact that this dataset was originally created for classification, while we used it to validate anomaly detection algorithms. The bearings under analysis were ER-16k and the microphone used to capture the data was a GRAS 40 PH. A schematic of the simulator is shown in Figure 2.

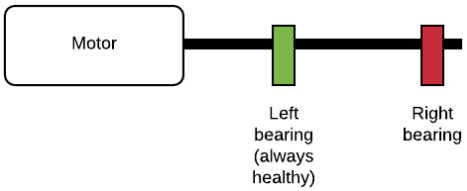

Figure 2. Schematic of the fault simulator test rig

The left bearing is always healthy, while the faults were introduced in the right bearing. Both bearings had a microphone placed close to them, resulting in 2 datasets, one with sound captured close and one with sound captured further from the faulty bearing. In the experiments both data sets are used to compare the algorithms. The data set was split in 2 ways. For the different AE the normal data was split into 70% training, 20% validation and 10% test and for the OC-SVM the normal data was split into 90% training and 10% test. For both cases, the same test set containing the entire set of anomolous examples was used.

B. Experimental Conditions

Prior to be processed by the anomaly detection models the time-domain acoustic signals were first transformed to Mel-Frequency Cepstral Coefficients (MFCC). These features were chosen over regular Short-Time Fourier Transform (STFT) because in [6] it is shown that the increased dimension of STFT is not appropriate for satisfying (near) real-time constraints. The MFCC were obtained using a STFT window length of 2048, a hop length of 512, a sampling rate of 25.6KHz and 64 mel-filterbanks. Before using these as input to the networks they were normalized to have zero mean and unit standard deviation, based on the training set.

In Figure 3 the mel-spectrograms for the different conditions are shown. These mel-spectrograms were using the first 10 seconds of data captured for a speed of 1020 rpm with the microphone close to the healthy bearing.

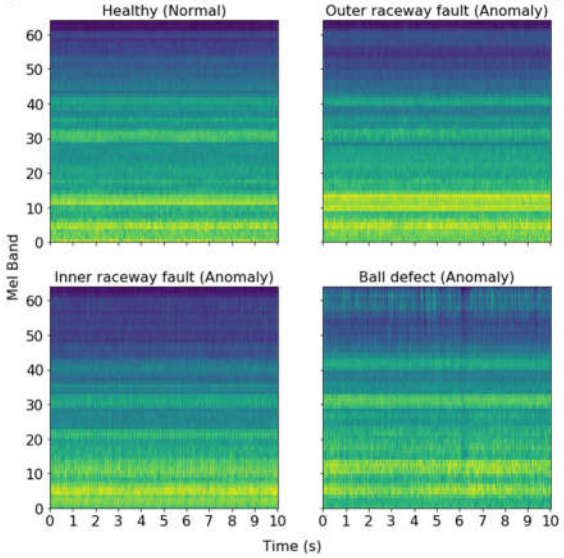

Figure 3. Mel-spectrograms of the 4 different conditions, 1 normal condition and 3 anomaly conditions. On the images the brighter (yellow) color means there is more energy present in that mel band.

For 1D-CNN a single MFCC feature vector was used as input, while for 2D-CNN and MSCRED a segment size of 72 vectors was used. This number is based on a formula and premise in [5], the premise is that it takes an expert roughly 2s to determine machine state based on sound and the machine needs about 1.5s. The number of vectors needed can then be calculated using (3), where S_p is the period of time in seconds, ws is the window size in samples used to compute a single feature vector, ov is the amount of overlap in samples, sr is the sampling rate of the signal and w is the number of vectors,

$$w = \frac{S_p * sr - ws}{ws - ov} + 1. \quad (3)$$

All networks were trained on minibatches of 8, 72×8 for 1D-CNN, samples using the Adam method with an initial learning rate of 10^{-3} and if the validation loss did not decrease for five consecutive epochs this was halved. To avoid overfitting dropout was used in the encoding networks with a 50% drop rate and L2 regularization was used with a factor of 10^{-4} in all networks. All AE were implemented using tensorflow and OC-SVM used the scikit-learn implementation.

For each AE we started from existing architectures, as mentioned in Section II.B, and tried a different number of layers, each with different amounts of neurons for each layer. We tested sizes from 1 to 4 layers in the encoder and decoder with 16-32-64-128 and 32-64-128-256 neurons in the respective layers. For 2D-CNN we kept a fixed stride of 2 in the convolutional layer to reduce the size of each dimension by half at each layer. For 1D-CNN the neurons per layer are in reversed order and a Fully Connected (FC) layer with 64 neurons is used as the bottleneck layer, the decoding layer starts with a FC layer which has the same amount of neurons as the total shape of the output of the last encoding layer, to reconstruct the output of that encoding layer. All networks were trained on a Geforce RTX 2080 TI GPU, feature extraction and OC-SVM computation was done on a I-XEON E7-4850v2 12-CORE CPU.

As explained earlier OC-SVM requires the data to first be processed to a set of hand-crafted features. The features used in this study are summed up in TABLE I. .

TABLE I. FEATURES USED FOR OC-SVM

Domain	Description	Ref.
Time	Variance, kurtosis, skewness, mean value, peak to peak, crest factor, RMS, 75th percentile, entropy, impulse, margin and shape factors, M6A, M8A	[16]
Time	Fourth order figure of merit	[17]
Frequency	3 first harmonics of the 3 fault types	[18]

C. Results

To determine the ranking of the AE in our comparison, we need to determine an evaluation criterion, for this we used a self-made score, made for this specific study, that combines the Area under Curve (AUC), prediction time (t_p) compared to the timeframe of the prediction (t) and the amount of parameters (P) in the network,

$$S = AUC^3 + \left(1 - \frac{t_p}{t}\right) + \frac{(10 - \log_{10} P)}{8}. \quad (4)$$

In this score we decided to punish worse performance by cubing the AUC, the ratio of prediction time compared to the timeframe of the data is used to see how fast the

network is and the $\log_{10}$ is used for parameters since they measure between 300 and 3×10^6, this number is divided by 8 so it is roughly between 0 and 1. We did not take training time into account since we assume we will be doing the training offline. However, we will still present the training time of the AE with the rest of the parameters.

Based on this score, we will show the 2 best network configurations for each AE and the OC-SVM for both the close and far dataset. TABLE II. and TABLE III. show the results and the individual variables of the different networks for respectively the microphone close to and far from the faulty bearing. AUC is the Area Under Curve (AUC), t_p is the prediction time in ms, t is the timeframe of the prediction in s, t_{tr} is the training time in s, P is the amount of parameters in the network and S is the result. For OC-SVM the time for feature extraction will be added to the time for prediction, both will be shown separately, and the amount of support vectors will be taken as the amount of parameters. The time added for feature extraction is about 0.41 times the timeframe we are looking at, e.g. if the timeframe is 1.5s, the added time is 0.615s. This added time is based on our implementation of the feature extraction which did not specifically exploit parallelization opportunities. Hence, there is room to optimize the implementation to gain in execution speed. Both the results with and without the added time are shown, in this order.

TABLE II. RESULTS MICROPHONE CLOSE TO THE FAULTY BEARING

Model	AUC	t_p (ms)	t (s)	t_{tr} (s)	P	S
OC-SVM	90.1%	1.94 + 615	1.5	1.90 + 472.6	208	2.690 / 2.280
1D-CNN	100%	15.9	0.08	6.66	165281	2.398
1D-CNN	100%	14.5	0.08	5.70	224353	2.399
2D-CNN	94.8%	9.17	1.5	1.909	9569	2.598
2D-CNN	100%	10.9	1.5	2.597	46529	2.659
MSCRED	89.4%	36.9	1.5	13.35	406881	2.239
MSCRED	88.4%	28.5	1.5	8.72	102065	2.297

The best results for 1D-CNN were achieved with a network consisting of 3 1D convolution layers followed by a FC layer in the encoder and the reverse with deconvolution layers in the decoder and another with 4 1D convolution layers instead of 3. Both networks started with 128 neurons in the first layer and halved each layer. The network with 3 1D convolution layers achieved the best result. The best results for 2D-CNN were achieved with a network consisting of 2 2D convolution layers in the encoder and 2 deconvolution layers in the decoder and another with 3 layers instead of 2. Both networks started with 16 neurons in the first layer and doubled each subsequent layer. The network with 3 layers achieved the best result. The best results for MSCRED were achieved with networks with 2 2D convolution layers in the encoder and 2 2D deconvolution layers in the decoder, the rest of the architecture follows the one described in [14] without the attention layer. One network started with 16 neurons in the first layer and doubled each subsequent layer, the other one started with 32 neurons in the first layer. The network starting with 16 neurons achieved the best result. An example of the difference in the sum of reconstruction errors (residuals) for normal and anomalous data can be seen in Figure 4. These reconstruction errors were gathered from the results of the best 2D-CNN model.

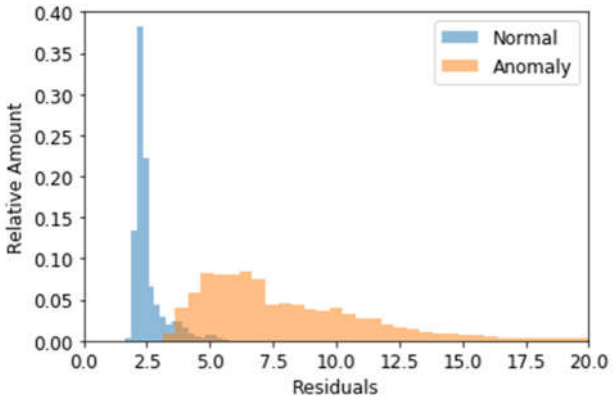

Figure 4. Histogram of the sum of the reconstruction errors (residuals) for normal and anomalous data. The height of the histogram has been normalized, such that the bins for each class sum to 1, meaning the height equals the percentage of samples in each class for that residual value.

While there is some overlap between the residuals of normal and anomalous data, a clear difference in values can be seen between both groups.

TABLE III. RESULTS MICROPHONE FAR FROM THE FAULTY BEARING

Model	AUC	t_p (ms)	t (s)	t_{tr} (s)	P	S
OC-SVM	90.0%	2.31 + 615	1.5	2.30 + 472.6	206	2.689 / 2.279
1D-CNN	100%	13.6	0.08	11.25	468545	2.371
1D-CNN	100%	16.8	0.08	5.85	224353	2.372
2D-CNN	99.9%	14.1	1.5	4.50	194177	2.579
2D-CNN	99.7%	9.47	1.5	2.71	46529	2.650
MSCRED	91.6%	29.1	1.5	8.57	102065	2.374
MSCRED	95.2%	38.7	1.5	13.5	406881	2.385

The general structure of each AE remains the same, so for these results we will only mention the amount of convolutional layers in the encoder and the amount of starting neurons in each network. The best results for 1D-CNN were achieved using a 1 and 3 layer network, both starting with 128 neurons in the first layer, the network with 3 layers achieved the best results. The best results for 2D-CNN were achieved using a 3 and 4 layer network, both starting with 16 layers in the first layer, the network with 3 layers achieved the best result. The best results for MSCRED were achieved with a 2 layer network, one with 16 neurons in the first layer and one with 32, the network with 32 neurons in the first layer achieved the best results.

For both cases a 2D-CNN based AE produced the best results of the AE variants. Even though 1D-CNN achieved higher performance, due to the increased prediction time, compared to the timeframe, and higher amount of parameters needed for this performance it received lower scores. OC-SVM receives the highest scores overall, if we assume no time is needed for feature extraction. This is due to a big advantage in the amount of parameters it has and a very fast prediction time if the features are already

extracted. However, if we add the time needed for feature extraction, with our implementation, the score it receives is severely lowered. Even if the feature extraction can be further optimized, the difference between the best AE and the OC-SVM is already relatively small, meaning it would have to be reduced to several milliseconds to stay ahead, which seems implausible.

IV. Conclusion and Future Work

In this study we compared a set of AE with a baseline OC-SVM to see which is the most suited to be implemented on hardware for real-time applications, based on the performance, speed and amount of parameters of the algorithm. From this study we can conclude that for this comparison the best AE for real-time use on hardware is a 2D-CNN based network. OC-SVM could be a competitor due to the large increase in speed and reduced parameters, if the feature extraction can be highly optimized.

In future work the use of semi-supervised learning techniques and other criteria to make the comparison between models, such as energy consumption, will be examined.

References

[1] S. Mostafa, S.-h. Lee and et al., "Lean thinking for a maintenance process," *Production & Manufacturing Research,* 2015, pp. 236-272, 2015.

[2] O. Janssens, V. Slavkovikj and et al., "Convolutional neural network based fault detection," *Journal of Sound and Vibration,* no. 377, pp. 331-345, 2016.

[3] F. R. Gomaa, M. A. Eissa and et al., "Fault diagnosis of rotating machinery based on vibration analysis," *International Journal of Advanced Engineering and Global Technology*, pp. 1571-1586, January 2016.

[4] A. Glowacz, "Acoustic based fault diagnosis of three-phase induction motor," *Applied Acoustics,* no. 137, pp. 82-89, 2018.

[5] D. Y. Oh and I. D. Yun, "Residual error based anomaly detection using auto-encoder in SMD machine sound," *Sensors,* vol. 18, issue 5, paper 1308, 2018.

[6] Y. Park and I. D. Yun, "Fast adaptive RNN encoder-decoder for anomaly detection in SMD assembly machine," *Sensors,* vol. 18, paper 3573, 2018.

[7] Y. Koizumi, S. Saito and et al., "Unsupervised Detection of Anomalous Sound based on Deep Learning and the Neyman-Pearson Lemma," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, issue 1, pp. 212-224, January 2019.

[8] P. Becht, E. Deckers and et al., "Experimental application of TR-music for loose bolt detection in complex assemblies," in *SHM-NDT*, 2018.

[9] E. Marchi, F. Vesperini and et al., "A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional LSTM neural networks," *Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 1996-2000.

[10] P. Foggia, N. Petkov and et al., "Audio surveillance of roads: A system for detecting anomalous sounds," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, issue 1, pp. 279-288, 2016.

[11] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019. [Online]. Available: http://arxiv.org/abs/1901.03407.

[12] V. Vercruyssen, W. Meert and et al., "Semi-supervised anomaly detection with an application to water analytics," 2018.

[13] A. Rabaoui, H. Kadri and et al., "One-class SVMs challenges in audio detection and classification applications," *EURASIP Journal on Advances in Signal Processing, Springer Open,* vol. 2008, paper 834973, Dec. 2008.

[14] C. Zhang, D. Song and et al., "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," *CoRR,* 2018.

[15] A. Maurício, C. Freitas, and et al., "Condition monitoring of gears under medium rotational speed," *Proceedings of the ICSV24*, London, 2017.

[16] P. J. N. Morias, *Condition Monitoring of Bearings under Low and Medium Speed Rotation*, Master's Thesis, Faculdade de Engenharia da Universidade do Porto, 2016.

[17] A. Maurício, *Condition Monitoring of Gearbox Faults with Acoustic and Vibration Signals*, Master's Thesis, Faculdade de Engenharia da Universidade do Porto, 2017.

[18] R. B. Randall and J. Antoni, "Rolling element bearing diagnostics – A tutorial," *Mechanical Systems and Signal Processing,* no. 25, pp. 485-520, 2011.