Machine Learning in Applications Detection of Anomalous Behaviour in Industrial Robot Project Report

Milad Zakhireh, Masoud Karimi, Zohreh Lahijaniamiri, SeyedOmid Mahdavi
Politecnico di Torino

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Machine Learning in Applications Detection of Anomalous Behaviour in Industrial Robot Project Report

***Abstract*—With the acceleration of system automation, system failures can now have a major impact on society. Therefore, it has become crucial to detect any anomalies in the system in order to prevent such failures. This field of study, called Anomaly Detection (AD), focuses on detecting abnormal states of a system. Using neural networks can be very helpful in addressing the issue of anomaly detection as they offer a robust platform to learn intricate patterns and associations within data. Autoencoders (AE) can contribute significantly to solving anomaly detection problems by providing a powerful framework for learning representations of normal data patterns. Once the autoencoder is trained, it can be used to reconstruct new data instances. Anomalies or outliers in the data tend to have higher reconstruction errors compared to normal data points. By setting an appropriate threshold on the reconstruction error, autoencoders can be used to detect anomalies. Adversarial Auto-Encoders (AAEs) also have contributed to solving this problem. The idea of using adversarial auto-encoders for anomaly detection builds upon the success of Generative Adversarial Networks (GANs) in generating realistic synthetic data. By using adversarial training, both the generator and the discriminator networks are improved, resulting in synthetic data that is increasingly more realistic. The AAE framework allows the autoencoder to learn a more robust and realistic representation of the input data by incorporating adversarial training.**

The aim of the project is to evaluate whether an adversarial component improves the performance of a traditional autoencoder to detect anomalies.

I. Introduction

The acceleration of system automation in Industry 4.0 has brought about significant societal repercussions in the event of system failures, as noted by (Lee [1]; Lee et al. [2]; Beheti et al. [3]). Consequently, detecting anomalous system states has become more crucial than ever in order to prevent such failures. Anomaly detection (AD), centers on precisely identifying and detecting anomalous system states. Additionally, deep learning has proven to be highly effective in modeling complex multivariate time-series data obtained from various sensors and actuators within large-scale systems (Chalapathy et al. [4]). As a result, a range of time-series AD (TAD) methods have embraced deep learning techniques and have individually showcased their superior performance compared to previous approaches. Industrial robots, which are integral components of modern manufacturing and production processes, are particularly subject to anomalies that can disrupt operations, compromise quality, or pose safety risks.

In recent years, the field of time-series anomaly detection has witnessed notable advancements with the advent of neural networks, which offer powerful tools for learning intricate patterns and relationships in complex data. Among these neural network architectures, normal autoencoders (AE) and adversarial Auto-Encoders $(AAEs)$ have emerged as a promising approach for anomaly detection. Autoencoders provide a powerful and flexible approach to anomaly detection by leveraging their ability to learn representations of normal data and identify instances that deviate from these learned patterns. On the other hand, combining the strengths of Auto-Encoders and Generative Adversarial Models (GANs), adversarial auto-encoders provide an effective framework for learning compact representations of data while incorporating the discriminative power of adversarial networks.

The primary objective of this project is to apply and evaluate the effectiveness of adversarial auto-encoders in detecting anomalous behavior in industrial robots. By leveraging their capabilities, we aim to enhance the detection accuracy and efficiency compared to traditional approaches, employing only auto-encoders.

We first train the auto-encoder to reconstruct the normal time series with the lowest reconstruction error. Then, alongside the autoencoder, a discriminator network will be employed where the discriminator learns to differentiate between real (normal) and reconstructed (fake) data, enhancing the model's ability to distinguish anomalies from normal behavior. This leads to a more robust encoding of normal patterns, making the model less prone to false positives or false negatives. Moreover, the incorporation of the discriminator helps refine the model's ability to distinguish subtle anomalies and capture complex patterns that may not be adequately captured by reconstruction-based methods alone.

In the subsequent sections, we will provide an overview of both normal autoencoders and adversarial autoencoders $(AAEs)$, and discuss their relevance in the context of anomaly detection. We will review the existing literature on anomaly detection techniques and their application to industrial robot behavior. Additionally, we will describe the methodology employed in our experiments, including the training and evaluation of both normal autoencoders and adversarial autoencoders. The results and analysis of our experiments will be presented, followed by a comprehensive assessment and comparison of the performance of these two approaches. Finally, we will conclude with a discussion of the implications and potential future directions for further enhancing anomaly detection in industrial robot systems.

II. Background

Various types of anomalies exist in TAD dataset (Choi et al. [5]). Several pieces of research work have been conducted across the world contributing to the improvement of time series

anomaly detection. A typical AD setting assumes that only normal data are accessible during the training time (Kim et al. [6]). Therefore, an unsupervised method is one of the most appropriate approaches for TAD, which trains a model to learn shared patterns in only normal signals. The final objective is to assign different anomaly scores to inputs depending on the degree of their abnormality, i.e., low and high anomaly scores for normal and abnormal inputs, respectively. Recent unsupervised TAD methods can be categorized into three types: reconstruction-based, forecasting-based, and others.

Reconstruction-based AD method trains a model to minimize the distance between a normal input and its reconstruction. Anomalous input at the test time results in large distance as it is difficult to reconstruct. The distance, or reconstruction error, serves as an anomaly score.

Forecasting-based AD method is similar to the reconstruction-based methods, except that it predicts a signal for the future time steps. The distance between the predicted and ground truth signal is considered an anomaly score.

Others include various attempts to model the normal data distribution. For instance, one-class classification-based approaches (Ma and Perkins [7]; Shen et al. [8]) measured the similarity between the hidden representations of the normal and abnormal signals.

In some scenarios, detecting anomalies becomes challenging using standard approaches based on mathematical models that rely on stationarity, or prediction models that utilize prediction errors to detect anomalies. It is often due to external factors or variables which are not captured by sensors leading to time-series which are inherently unpredictable. Malhotra et al. [9] propose a Long Short Term Memory Networks based Encoder-Decoder scheme for Anomaly Detection (EncDec-AD) that learns to reconstruct 'normal' time-series behavior, and thereafter uses reconstruction error to detect anomalies. EncDec-AD uses only the normal sequences for training. This is particularly useful in scenarios when anomalous data is not available or is sparse, making it difficult to learn a classification model over the normal and anomalous sequences. They show that EncDec-AD is robust and can detect anomalies from predictable, unpredictable, periodic, aperiodic, and quasi-periodic time-series. Further, they show that EncDec-AD is able to detect anomalies from short time-series (length as small as 30) as well as long time-series (length as large as 500). Li et al. [10] believe that conventional threshold-based anomaly detection methods are inadequate for many of cyber-physical systems (CPSs) due to their dynamic complexities, while supervised machine learning methods are unable to exploit the large amounts of data due to the lack of labeled data. They propose an unsupervised multivariate anomaly detection method based on Generative Adversarial Networks (GANs), using the Long-Short-Term-Memory Recurrent Neural Networks (LSTM-RNN) as the base models (namely, the generator and discriminator) in the GAN framework to capture the temporal correlation of time series distributions. Their proposed Multivariate Anomaly Detection with GAN (MAD-GAN) framework considers the entire variable set concurrently to capture the latent interactions amongst the variables instead of treating each data stream independently. Zhou et al. [11] also conduct a study to address the challenge of detecting anomalous beats from electrocardiogram (ECG) readings. Their objective is to develop an effective and efficient approach for detecting anomalies in large-scale rhythmic time series data, primarily consisting of normal data segments or 'beats.' They observe that existing methods for anomaly detection in ECG data either require a significant amount of labeled and balanced data for classification or rely on less regularized reconstructions, resulting in lower accuracy. Therefore, they propose BeatGAN, an unsupervised anomaly detection algorithm for time series data. BeatGAN outputs explainable results to pinpoint the anomalous time ticks of an input beat, by comparing them to adversarially generated beats. Their experiments show that BeatGAN accurately and efficiently detects anomalous beats in ECG time series, and routes doctors' attention to anomalous time ticks, achieving accuracy of nearly 0.95 AUC, and very fast inference (2.6 ms per beat).

III. Materials and methods

A. MODELS

A.1 Autoencoder An autoencoder for time series is a type of neural network architecture designed to capture and reconstruct temporal patterns and features within sequential data. The network consists of an encoder and a decoder, with the encoder compressing the input time series into a lower-dimensional latent representation and the decoder reconstructing the original input from this compressed representation. We use CNNs because many studies show that CNN networks are more robust than LSTMs for time series (Rajpurkar et al. [12]). Using autoencoders as a framework for anomaly detection based on reconstruction error comprises two main components: optimizing the reconstruction (or generation) model and scoring anomalies based on the reconstructed data.

The main objective of autoencoders is to reconstruct time series data x in order to closely resemble the original input. The encoder $GE(x)$ transforms the input x into a hidden vector z, capturing essential features. Subsequently, the decoder $GD(z)$ utilizes z to generate a reconstructed time series x'. The reconstruction process can be summarized as $G(x) = GD(GE(x))$. However, explicit regularization is not present in AE-based methods. The main goal of an autoencoder is to minimize the reconstruction error:

$$\mathcal{A}(x) = \|x - G(x)\|_2^2 \tag{1}$$

In terms of their structures, the encoder $GE(\cdot)$ is designed as a mirrored version of the decoder $GD(\cdot)$, ensuring symmetry. Within our architecture, we integrate the components proposed by Radford et al. [13].

- Replace any pooling layers with strided convolutions (encoder) and fractional-strided convolutions (decoder).
- Use batchnorm in both the encoder and the decoder.
- Remove fully connected hidden layers.
- Use ReLU activation in decoder for all layers except for the output, which uses Tanh.

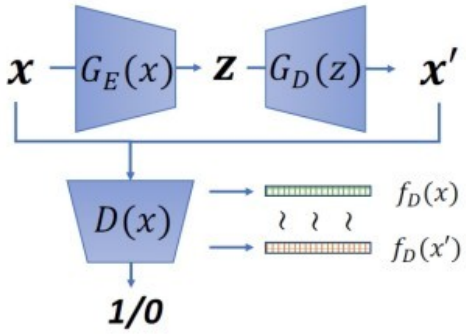

Fig. 1. Illustration of our network structure.

Algorithm 1 Training algorithm

1: $\theta_G, \theta_D \leftarrow$ initialize network parameters
2: **for** number of training iterations **do**
3: Sample $\{x_1, ..., x_m\} \sim$ a batch from the normal data
4: Generate $\{x'_1, ..., x'_m\}$ by G_E and G_D
5: Compute L_D by Eq (7)
6: $\theta_D \longleftarrow \theta_D + \alpha \nabla_{\theta_D}(L_D)$ // ∇ is the gradient
7: Compute L_G by Eq (6)
8: $\theta_G \longleftarrow \theta_G + \alpha \nabla_{\theta_G}(L_G)$
9: **end for**

Fig. 2. Adversarial Autoencoder Training Algorithm

- Use LeakyReLU activation in the encoder for all layers.

A.2 Adversarial Autoencoder To enforce regularization, we integrate an adversarial framework into our model Fig. 1. This framework involves a generator (G) and a discriminator (D) participating in a two-player minimax game. The discriminator's objective is to accurately classify real samples (x) and synthesized samples (x') with distinct class labels 1 and 0, respectively. On the other hand, the generator strives to minimize the loss function in such a way that the generated samples are indistinguishable by the discriminator (i.e., close to class label 1). To achieve this, we employ a pairwise feature matching loss $(\mathcal{L}_{pfm})$, which aims to minimize the statistical differences between the original and generated time series in the hidden layers of the discriminator $(D(\cdot))$.

$$\mathcal{L}_{pfm} = \|f_D(x) - f_D(x')\|_2^2 \tag{2}$$

Overall, the objective of reconstruction with adversarial regularization is to minimize the following loss function:

$$\mathcal{L}_G = \|x - x'\|_2^2 + \lambda \|f_D(x) - f_D(x')\|_2^2 \tag{3}$$

where $x' = G(x)$, and λ is the weighting parameter adjusting the impact of the adversarial regularization. Meanwhile, the objective of the discriminator is to maximize the following loss function:

$$\mathcal{L}_D = \frac{1}{N} \sum_{i}^{N} [\log D(x_i) + \log(1 - D(x'_i))] \tag{4}$$

Both the encoder $GE(\cdot)$ and decoder $GD(\cdot)$ networks employ a similar structure. Additionally, the discriminator D shares the same architectural details as the encoder $GE(\cdot)$. This symmetry in the network structures facilitates the learning and optimization process within the GAN-based regularization framework.

The implemented training algorithm, referred to as shown in Fig. 2, is based on the principles and concepts introduced in BeatGAN (Zhou et al. [11]). It serves as the foundation for the training process in the current context.

Finally, we utilize the Adam optimization algorithm for training and learning the reconstruction model. The Adam optimizer is a popular choice for its adaptive learning rate and efficient parameter updates, which help in optimizing the reconstruction model effectively.

For anomaly detection, we employ the trained reconstruction model to reconstruct a time series x' based on the given input x. By comparing the difference or the reconstruction error between x' and x, we evaluate the anomalousness score. This score serves as a measure of how much the reconstructed time series deviates from the original input, indicating the likelihood of an anomaly in the data. Afterwards, we compare the score with threshold to decide if the signal is anomaly or not.

IV. Results and discussion

A. DATASET

The dataset consists of time-series data collected from the sensors of an industrial robot named Kuka. Each sample in the dataset consists of 86 features, which encompass various information such as joint angle positions, velocity, current, and power usage values. The dataset is comprised of 233,792 instances of normal data and 41,538 instances of abnormal data.

B. EVALUATION METRICS

To validate our trained model using the validation set, we utilized the Average Precision metric (AP), which was computed using Eq. 5. Afterward, we calculated the maximum F1 score $(F1_{max})$ and identified the corresponding threshold value for the validation set. This threshold was then saved within the model. By employing this approach, we aimed to select an optimal threshold that achieved the best balance between precision and recall, ultimately ensuring the model's optimal performance on the validation set and transferring this performance to the test set. Subsequently, when evaluating the model's performance on the test set, the saved threshold was used to assign a label to each window-shaped sample and calculate the recall metric (R).

$$\mathcal{AP} = \sum_{n} (R_n - R_{n-1}) P_n \tag{5}$$

C. DATA PRE-PROCESSING

In our project, we undertook a data processing procedure that involved multiple steps. Firstly, we normalized each

sample in the dataset, ensuring that all values fell within the range of 0 to 1. This step allows for easier comparison and processing of different sensor readings. Subsequently, we divided the data into fixed-size windows of length (W). For the normal data, these windows were created with an overlap, specifically using a stride of $(W/3)$. This overlapping strategy helps capture temporal dependencies and patterns present in the normal data. In contrast, the abnormal data windows were constructed without any overlap. To evaluate our models, we designated 10% of both the normal and abnormal data for validation purposes. The remaining normal data was employed for training the models, while the remaining abnormal data served as the test set.

D. EXPERIMENTS

In this section, we present the experimental setup and results of our study we describe the experimental methodology, present the obtained results, and provide an analysis of the comparative performance between the normal autoencoder and adversarial autoencoder.

Firstly, we focused on analyzing the performance of an autoencoder model for anomaly detection. We conducted several experiments by varying the hyperparameters, specifically the window size (W) and the latent space dimensionality (L).

The architecture of the decoder in our autoencoder model consists of five 1D transposed convolutional layers, each followed by batch normalization and ReLU activation. The configuration of the transposed convolutional layers is as follows: $L(10/1)-16W(4/2)-8W(4/2)-4W(4/2)-2W(7/1)$: e.g. $L(10/1)$ means that the number of filters is L, the size of filter is 10 and the stride is 1. The encoder's structure is a mirrored version of the decoder.

The model is trained for 5000 iterations. For optimization, we employ the Adam optimizer with an initial learning rate (lr) set to 0.001. These characteristics are consistent across all our experiments to maintain a fair comparison.

Table I presents the results obtained from different configurations. In this table, the threshold value corresponding to the best F1 score for each experiment is saved within each model. This threshold is determined using the validation set. It is then used as the final threshold to compute the recall score. The configuration $W = 30, L = 30$ achieved the highest average precision of $\mathbf{79.43}\%$. The maximum F1 score of $\mathbf{75.25}\%$ and recall of $\mathbf{77.53}\%$ further confirm the effectiveness of the model. Increasing the window size to 50 and 70 does not yield significant changes or improvements in the results. However, when the window size is set to 100 and the latent space dimensionality is set to 30, it yields the highest recall score compared to all other configurations. Although the average precision and maximum F1 score are not as high as those of the model with $W = 30$ and $L = 30$, they still fall within an acceptable range.

To assess the impact of adding a discriminator to the previous model, we implement an adversarial autoencoder by incorporating a discriminator with a similar architecture as the encoder. The purpose is to evaluate whether this addition improves the performance of the normal autoencoder and yields better results. We specifically focus on the best configurations identified from the previous autoencoder experiment and conducted experiments with various values of the regularization weight (λ). In order to ensure proper model fitting, the number of iterations is increased to 7000, considering the larger number of parameters in this experiment.

TABLE I
AUTOENCODER EXPERIMENTS

W	L	Validation Metrics		Test Metrics
		$AP(\%)$	$F1_{max}(\%)$	$Recall(\%)$
30	30	$\mathbf{79.43}$	$\mathbf{75.25}$	77.53
	50	72.48	69.25	79.94
	70	55.79	63.24	58.91
50	30	70.96	69.04	74.70
	50	64.76	69.35	70.55
	70	68.20	70.21	66.27
70	30	73.39	64.18	73.03
	50	76.53	64.23	76.03
	70	68.46	65.65	74.34
100	30	76.52	73.39	$\mathbf{88.24}$
	50	76.37	71.58	74.33
	70	74.53	73.21	83.96

TABLE II
ADVERSARIAL AUTOENCODER EXPERIMENTS

W	L	λ	Validation Metrics		Test Metrics
			$AP(\%)$	$F1_{max}(\%)$	$Recall(\%)$
30	30	0.0001	77.33	70.92	72.55
		0.001	$\mathbf{83.56}$	$\mathbf{80.81}$	81.46
		0.01	73.36	69.48	77.77
		0.1	67.69	65.78	67.82
100	30	0.0001	76.32	75.00	77.01
		0.001	77.08	75.73	85.83
		0.01	74.74	70.18	$\mathbf{86.63}$
		0.1	61.07	64.10	52.41

After conducting the training process and evaluating the outcomes, it became evident that the mere inclusion of a discriminator into the existing model did not yield significant improvements. Instead, the discriminator gradually converged to assigning the same label to all data samples, resulting in stagnant results compared to the autoencoder experiment. To overcome this issue, we employed the R1 regularization technique, as introduced by Mescheder et al. [14]. R1 regularization, also known as gradient penalty, is a clever strategy aimed at enhancing training stability and overall model performance. By incorporating this technique, we encouraged the discriminator to establish smooth decision boundaries and avoid overconfidence in its predictions. Specifically, we penalized the gradients of the discriminator concerning real samples. As a result, at the end of the training, the discriminator demonstrated improved differentiation between real and fake inputs.

The experimental results are summarized in Table II. For the configuration with a window size of 30 and latent space dimensionality of 30, the best recall score of $\mathbf{81.46}\%$ is achieved when setting the regularization weight to 0.001. However, the improvement in recall score compared to the baseline autoencoder experiment is not as significant as anticipated, despite the extended training duration and the inclusion of the discriminator. In this case, the best average precision and maximum F1 score among all the experiments are attained,

reaching values of $\mathbf{83.56}\%$ and $\mathbf{80.81}\%$ respectively. Similarly, for the configuration with a window size of 100 and latent space dimensionality of 30, the highest recall score of $\mathbf{86.63}\%$ is achieved with a regularization weight of 0.01, representing the highest performance among all the adversarial autoencoder experiments. Notably, although the average precision displayed a slight enhancement and the model effectively identified anomalies in the dataset, the expected performance boost in the test stage was not fully realized. This suggests that further investigation and fine-tuning of the model may be necessary to achieve the desired performance boost in anomaly detection.

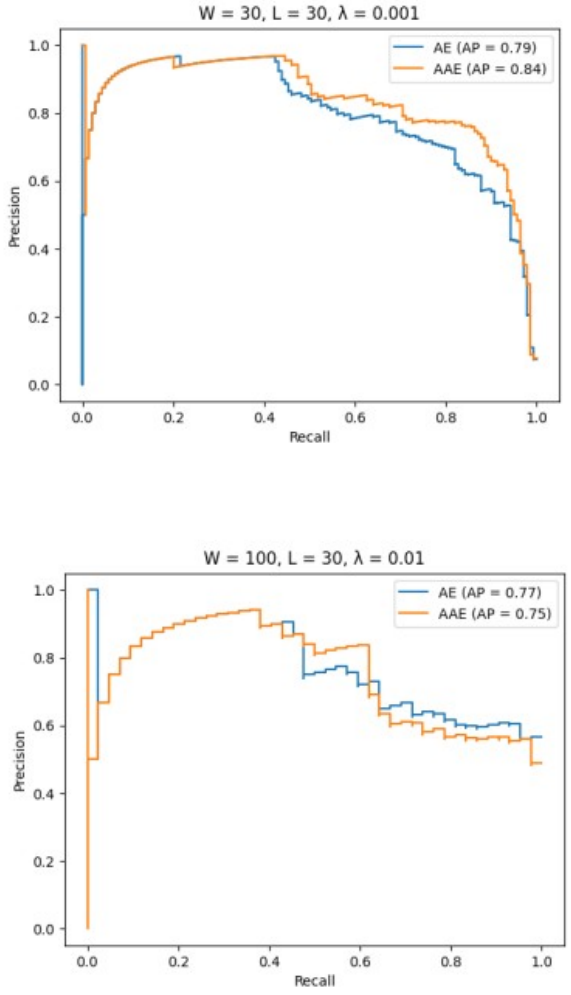

Fig. 3. Precision-recall curves for the validation set.

Fig. 3 displays the precision-recall curves for both the autoencoder (AE) and adversarial autoencoder (AAE) models. Upon examining the plots, it is evident that the performance of both models is quite similar. It is important to note that the presence of outliers in the validation set is noticeable, which can impact the overall performance of the models.

V. Conclusions and Future Works

In our study, we aimed to evaluate the performance of both autoencoder and adversarial autoencoder models in anomaly detection. Our findings indicate that the autoencoder model demonstrated promising results in detecting anomalies. However, the addition of a discriminator in the adversarial autoencoder did not yield significant improvements in performance. Furthermore, due to the relatively high computational cost and time required, utilizing the adversarial autoencoder instead of a regular autoencoder seems to be disadvantageous.

Based on our observations, it can be tentatively concluded that for datasets with a larger number of samples and higher dimensionality, the performance of the autoencoder model alone may not be satisfactory. In such cases, incorporating an adversarial component, such as a discriminator, could potentially enhance the detection of anomalies.

Further research and experimentation are necessary to validate this hypothesis and explore the performance of these models on datasets with different characteristics. Additionally, optimization techniques and parameter tuning may also contribute to improving the overall performance of the models in anomaly detection tasks.

References

[1] E. A. Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC)*. IEEE, 2008, pp. 363–369.

[2] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing letters*, vol. 3, pp. 18–23, 2015.

[3] R. Baheti and H. Gill, "Cyber-physical systems," *The impact of control technology*, vol. 12, no. 1, pp. 161–166, 2011.

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

[5] K. Choi, J. Yi, C. Park, and S. Yoon, "Deep learning for anomaly detection in time-series data: review, analysis, and guidelines," *IEEE Access*, vol. 9, pp. 120 043–120 065, 2021.

[6] S. Kim, K. Choi, H.-S. Choi, B. Lee, and S. Yoon, "Towards a rigorous evaluation of time-series anomaly detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 7194–7201.

[7] J. Ma and S. Perkins, "Time-series novelty detection using one-class support vector machines," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 3. IEEE, 2003, pp. 1741–1745.

[8] L. Shen, Z. Li, and J. Kwok, "Timeseries anomaly detection using temporal hierarchical one-class network," *Advances in Neural Information Processing Systems*, vol. 33, pp. 13 016–13 026, 2020.

[9] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.

[10] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *Artificial Neural Networks and Machine Learning–ICANN 2019: Text and Time Series: 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17–19, 2019, Proceedings, Part IV*. Springer, 2019, pp. 703–716.

[11] B. Zhou, S. Liu, B. Hooi, X. Cheng, and J. Ye, "Beatgan: Anomalous rhythm detection using adversarially generated time series." in *IJCAI*, vol. 2019, 2019, pp. 4433–4439.

[12] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng, "Cardiologist-level arrhythmia detection with convolutional neural networks," *arXiv preprint arXiv:1707.01836*, 2017.

[13] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[14] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for gans do actually converge?" in *International conference on machine learning*. PMLR, 2018, pp. 3481–3490.