Mini-Project (ML for Time Series) - MVA 2025/2026

Stéphane EILLES-CHAN WAY stephane.eilles-chan-way@polytechnique.edu
Hugo PERCOT hugo.percot@polytechnique.edu

January 5, 2026

1 Introduction and contributions

In this report, we study Graph Deviation Network, a novel approach for anomaly detection in multivariate time series, leveraging graph structure and Graph Neural Network [3]. We highly rely on the original paper by Ailin Deng and Bryan Hooi [2]. This method takes as input a multivariate time series and learn a graph of the dependence relationships between sensors, and identifies and explains deviations from these relationships.

In this project, we reused the code for the model architecture and adapted the training and testing functions, while all other components were fully implemented by us. Furthermore, we apply GDN to two datasets not considered in the original paper, and all the results presented in this report arise from new experiments, which are summarized in Section 4.1.

Work repartition :

- **Stéphane** : train/test functions adaptations, point-wise anomaly detection functions, point-wise evaluation, data preprocessing (further analysis, detrending function), VAR, AR, paper configuration reproduction, mixed loss and score, varying window size on SMD, report and presentation writing
- **Hugo** : anomaly detection on sequences, sequence-wise evaluation, data preprocessing (sliding window creation, basic statistics), paper configuration reproduction, varying window size on SMAP, experiences harmonization, final notebook organisation and run, report and presentation writing

2 Method analysis

The Graph Deviation Network (GDN) method can be decomposed into several steps, establishing a training pipeline. While traditional Graph Neural Networks takes a graph as input, GDN has a wider scope. It acts on multivariate time series, generated by multiple sensors, without prior graph structure, learns relationships between them under a graph shape, and predicts the following time steps. Let's break down the method, summarized on Figure 1.

2.1 Method description

Sensor Embedding Let assume we have a multivariate time series of size N obtained from K sensors : $(\mathbf{X}[n])_n \in \mathbb{R}^{K \times N}$. First, the method introduces some sensor embedding vectors

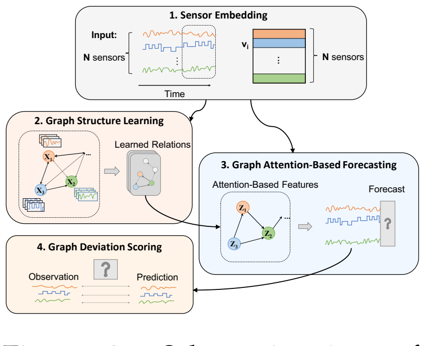


Figure 1: Schematic view of

$\mathbf{v}_k \in \mathbb{R}^d$, for $k \in \{1, 2, \ldots, K\}$. This allows to use a representation space $\mathbb{R}^d$ in which we expect sensor with similar behavior to be "close". Constructing the map from the time series to the representation space is done during training.

Graph Structure learning Given K sensors with embeddings, GDN build an undirected relational graph by considering each sensor as a node, and connecting each node to its κ nearest neighbor, i.e. with the highest cosine similarity w.r.t the embeddings. The adjacency matrix A is given by :

$$\begin{aligned} e_{j,i} = e_{i,j} &= \frac{\mathbf{v}_i^\top \mathbf{v}_j}{||\mathbf{v}_i|| \times ||\mathbf{v}_j||} \text{ for } j \in \{1, 2, , K\} \\ A_{j,i} = A_{i,j} &= \mathbb{1}\{j \in \mathrm{Top}_\kappa(e_{k,i} : k \in \{1, 2, , K\})\} \end{aligned}$$

Graph Attention-Based Forecasting: As many other methods, GDN produces predictions of the time series based on some previous samples to detect unexpected events, i.e. anomalies. For each time index t, the input is a sliding window of size w over the past sensor values:

$$\mathbf{x}[t] := [\mathbf{X}[t-w], \mathbf{X}[t-w+1], \ldots, \mathbf{X}[t-1]] \in \mathbb{R}^{K \times w}$$

The goal is to predict the current vector $\mathbf{X}[t]$ from $\mathbf{x}[t]$.

Attention-based feature extractor. For each sensor (node) i, GDN aggregates information from its neighbors $\mathcal{N}(i) = \{j \mid A_{j,i} > 0\}$ using an attention mechanism incorporating sensor embeddings. The attention coefficients $\alpha_{i,j}$ are built as follows:

$$\begin{aligned} \mathbf{g}_i[t] &= \mathbf{v}_i \oplus W\mathbf{x}_i[t], \\ \pi(i,j) &= \mathrm{LeakyReLU}\big(\mathbf{a}^\top(\mathbf{g}_i[t] \oplus \mathbf{g}_j[t])\big), \\ \alpha_{i,j} &= \frac{\exp(\pi(i,j))}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(\pi(i,k))} \end{aligned}$$

where $\oplus$ denotes concatenation, $\mathbf{a}$ is a trainable attention vector, and $W \in \mathbb{R}^{d \times w}$ is a trainable weight matrix. The aggregated representation of the node i is then:

$$\mathbf{z}_i[t] = \mathrm{ReLU}\left(\alpha_{i,i} W\mathbf{x}_i[t] + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} W\mathbf{x}_j[t]\right)$$

Output layer and training loss. The node representations are elementwise multiplied by their embeddings and passed to a shared MLP f_θ to obtain the prediction:

$$\hat{\mathbf{X}}[t] = f_\theta\big([\mathbf{v}_1 \circ \mathbf{z}_1[t], \ldots, \mathbf{v}_K \circ \mathbf{z}_K[t]]\big) \in \mathbb{R}^K$$

where $\circ$ denotes elementwise multiplication. The model is trained by minimizing the mean squared error:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{T_{\text{train}} - w} \sum_{t=w+1}^{T_{\text{train}}} \|\hat{\mathbf{X}}[t] - \mathbf{X}[t]\|_2^2.$$

Graph Deviation Scoring: Once the model is trained, and able to forecast the multivariate time series, we want to compare prediction and true data to detect anomalies. To do so, we compute for each sensor the normalized error value, $a_i[t] = \frac{\text{Err}_i[t] - \tilde{\mu}_i}{\tilde{\sigma}_i}$ where $\text{Err}_i[t] = |\mathbf{X}_i[t] - \hat{\mathbf{X}}_i[t]|$ with $\tilde{\mu}_i$ and $\tilde{\sigma}_i$ are the median and inter-quartile range across time ticks of the Err_i. Define the overall anomalousness score $A[t] = \underset{i}{max}\ a_i[t]$. The original paper applies a simple moving average over time to obtain a smoothed score $A_s[t]$. A time index t is finally declared anomalous if $A_s[t]$ exceeds a chosen threshold, taken as the maximum score over a validation set (assumed to be normal).

2.2 Discussion

We would like to discuss several points about GDN, which may not reflect the original authors' opinion. First, as with many unsupervised settings in anomaly detection, the GDN training pipeline assumes the training data to be normal, i.e. without any anomaly. In a context where anomalies are frequent or challenging to detect humanly, this condition may not be fulfilled. An interesting complementary experiment would be to assess method robustness to anomalies in the training set.

In addition, we question the fixed choice of the sliding window size $w = 5$ in the original paper. Since w controls how much past context is used for forecasting, an interesting complementary experiment would be to vary w and observe its impact on detection performance and anomaly scores.

Finally, the choice to use mean square error $\mathcal{L}_{\text{MSE}}$ as loss function is classic in machine learning, and is here relevant as the prediction window has a size of 1. In the case where we would train on predicting larger windows, another interesting choice would be to use the soft-DTW [1]. However, we do not investigate this in the following work.

3 Data

3.1 SMAP dataset

We use the SMAP dataset from the NASA anomaly benchmark, loaded through the `torch_timeseries` library. The multivariate time series contains 25 sensors (one continuous and 24 binary, see Figure ??), along with expert-labeled anomalies. The anomalies are all sequences of points of length going from 31 to 4218 time steps. The data shows no missing or infinite values, so no interpolation is needed. As in prior work, the training split is assumed to contain only normal behavior, and no outlier removal is required.

Detrending considerations: Sensors 1–24 are binary and clearly stationary (stable mean, short-range dependence, no visible low-frequency structure in the DFT). Sensor 0 is continuous and at first seemed to exhibit a slow trend based on autocorrelation (slow decay and stays high). However, further inspection showed that its dynamics are dominated by piecewise-constant regimes with abrupt jumps rather than a true smooth drift. Polynomial detrending systematically distorted the signal instead of improving stationarity (see Figure 5). We therefore do not apply detrending in our final pipeline.

Denoising: The binary sensors display ON/OFF patterns without noise, and sensor 0 does not exhibit high-frequency fluctuations characteristic of measurement noise (see Figure ??). No denoising step is applied.

3.2 SMD dataset

We also evaluate our methods on the SMD dataset from the Server Machine Dataset benchmark, which is automatically downloaded using the `torch_timeseries` library. The anomalies are all sequences of points of length going from 2 to 3161 time steps. The version provided by this library corresponds to a single long multivariate time series with 38 continuous sensors and expert-labelled anomalies (see Figure 6).

Missing values and preprocessing The loaded series contains no missing or infinite values, so no interpolation is required. As in the standard SMD protocol, the training split contains only normal behaviour, and the test set includes all labelled anomalies. Therefore, no outlier removal step is applied.

Detrending Unlike SMAP, all SMD sensors are continuous. We inspect several representative sensors using raw signal plots (see Figure 6), local variance, the autocorrelation function (see Figure 7)and the DFT (see Figure 8) spectrum. The signals display clear regime-switching dynamics with abrupt transitions but no evidence of a smooth low-frequency trend. The autocorrelation decays steadily without long-range dependence, and the spectral density is dominated by a peak near zero frequency, which is typical of piecewise-stationary behaviour rather than true non-stationarity. We do not apply detrending in our pipeline.

Denoising Visual inspection of the raw sensors does not reveal high-frequency fluctuations indicative of measurement noise. The signals appear structured rather than noisy, and denoising would risk removing meaningful dynamics. Consequently, no denoising step is applied.

4 Results

4.1 Experiments on both datasets

Evaluation metric As the method provides for each time step a binary classification, the strict evaluation is to compute classical metrics (AUC, F1-score, precision, recall) point-wise. Yet, in both datasets the anomalies are not isolated points but sequences, so we may also want to evaluate the model sequence-wise. To do so, points where the ground truth is 0 (no anomaly) stay unchanged, but, on each anomaly sequence, points are merged into one, keeping the highest anomalousness score on the interval. Then, we perform point-wise evaluation on this contracted time series. This last evaluation approach is more flexible and should show better performances. All performances are reported in Tables 1 and 2.

Baselines VAR(100) and AR(100): We evaluated two linear baselines: a multivariate VAR(100) fitted on all sensors (`VAR_100`) and an univariate AR(100) restricted to sensor 0 only (`AR_100`). The order 100 is motivated by a peak on the autocorrelation functions (see Figure **??**).

Reproduction of the original configuration: We first reproduced the exact architecture and hyperparameters reported in Deng and Hooi [2]. Results are reported under `paper_config`.

Mixed continuous/binary reconstruction loss (only on SMAP): To better reflect the sensors' heterogeneity, we modified the training loss so that sensor 0 is reconstructed using an MSE loss, whereas sensors 1–24 are reconstructed using a Binary Cross-Entropy loss applied to their logits. The anomaly scoring procedure was adapted accordingly: absolute prediction error for the continuous channel and sigmoid-transformed absolute error for the binary sensors. Results are reported under`mix_cont/bin`.

Anomalies during training: One of the main assumptions of the GDN method is that the training time series are normal. We train the model with the paper configuration but on data with anomalies to see how it degrades the detection afterward. To do so, we split the test time series (the only one that has anomalies) into a training and testing set and run the same experiment as the previous reproduction. Results are shown under `anom_in_train`.

Effect of the window size: To better understand the influence of temporal context on GDN, we evaluated the model on SMD for several context window sizes, $w \in \{5, 10, 20, 50, 75, 100, 150, 200\}$, training each configuration for 50 epochs and reporting the AUC ROC in Appendix 4.3

4.2 Results on SMAP

	Point-wise evaluation				Sequence-wise evaluation			
Method	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC
`VAR_100`	0.8867	0.0272	0.0528	0.4536	0.0821	0.2537	0.1241	0.9632
`AR_100`	0.1663	0.040	0.0078	0.5027	0.0153	0.2537	0.0289	0.9760
`paper_config`	0.8824	0.0005	0.0011	0.4035	0.6364	0.1045	0.1795	0.9326
`mix_cont/bin`	0.8582	0.0022	0.0044	0.4528	0.4286	0.2239	0.2941	0.9219
`anom_in_train`	0.4595	0.0013	0.0025	0.3979	0.2593	0.2857	0.2718	0.9417

Table 1: Results on SMAP dataset

Looking at the strict point-wise evaluation, it appears that baselines and all GND variations perform badly (worse than random on AUC on SMAP. This suggests that the main limitation lies in the SMAP dataset itself, notably in its main binary nature, which is challenging to handle when doing forecasting-based anomaly detection. The best-performing model is AR(100) on the continuous sensor alone, showing that SMAP's binary channels degrade rather than enhance anomaly separability. As expected, the sequence-wise evaluation yields better F1-score and AUC (and worse precision), but GDN methods stay below baselines on this difficult dataset. We also notice that putting an anomaly in the training time series significantly affects the precision of the model, suggesting that the normal training hypothesis is crucial. The effect of mixing the loss function according to the sensors' nature seems to increase performance. Finally, looking at figure 9, it appears that the smaller the context window size, the better the model is. This reinforces the idea that GDN is not adapted to SMAP binary nature and giving it more context is useless.

4.3 Results on SMD

Looking only at the AUC metric, we observe that all model variants achieve reasonably good discrimination performance on SMD. The best-performing configuration is VAR(100), followed closely by GDN and AR(100). The `anom_in_train` setup leads to the lowest AUC, confirming

	Point-wise evaluation				Sequence-wise evaluation			
Method	Precision	Recall	F1-score	AUC	Precision	Recall	F1-score	AUC
`VAR_100`	0.9848	0.0022	0.0044	0.7079	0.9286	0.0398	0.0762	0.9853
`AR_100`	0.8971	0.0021	0.0041	0.6464	0.8000	0.0856	0.1547	0.8820
`paper_config`	1.0000	0.0000	0.0001	0.6512	1.0000	0.0031	0.0061	0.9508
`anom_in_train`	0.6875	0.0005	0.0009	0.6061	0.5455	0.0275	0.0524	0.9267

Table 2: Results on SMD dataset

once again that injecting anomalies into the training data degrades the model's ability to learn normal patterns and reduces separability between normal and abnormal behaviour.

Overall, the AUC values on SMD are noticeably higher than on SMAP, suggesting that GDN adapts more effectively to the continuous multivariate nature of SMD. In contrast to SMAP's binary and more irregular structure, SMD provides richer temporal dependencies that the model can capture, resulting in better anomaly discrimination. We also observe that, contrary to SMAP, VAR achieves better AUC than the simpler AR model, indicating that additional sensors contribute useful contextual information rather than introducing noise.

Figure 10 shows that larger windows tend to improve performance, with a clear upward trend beyond window sizes of 50. The maximum AUC is obtained at $w = 200$, indicating that providing more temporal context helps the model better understand the dynamics of SMD sensors. This contrasts with SMAP, where additional context did not translate into performance gains. On SMD, a larger window appears beneficial.

However, the curve presents a surprisingly high AUC at $w = 10$, despite neighbouring values yielding poorer performance. This behaviour suggests that the model may be aligning with an intrinsic periodicity of the system around 10 time steps. This hypothesis is supported by the autocorrelation analysis of several sensors, such as sensor 10 (Figure 11), which exhibits clear peaks every 10 lags.

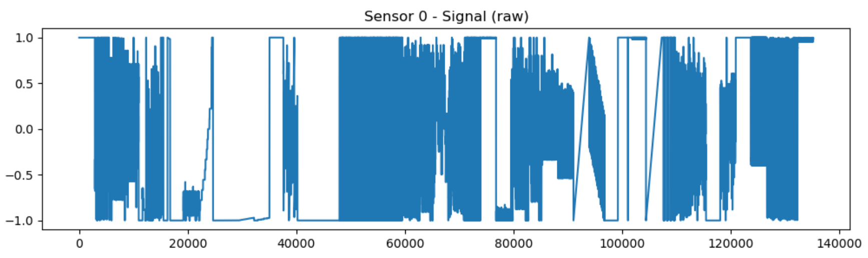
References

[1] Marco Cuturi and Mathieu Blondel. *Soft-DTW: a Differentiable Loss Function for Time-Series*. 2018. arXiv: 1703.01541 [stat.ML]. URL: https://arxiv.org/abs/1703.01541.

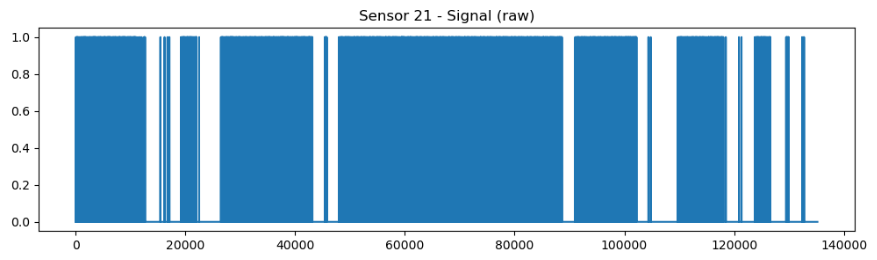
[2] Ailin Deng and Bryan Hooi. *Graph Neural Network-Based Anomaly Detection in Multivariate Time Series*. 2021. arXiv: 2106.06947 [cs.LG]. URL: https://arxiv.org/abs/2106.06947.

[3] Franco Scarselli et al. “The Graph Neural Network Model”. In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605.

Appendix A — Figures examples for SMAP Dataset analysis

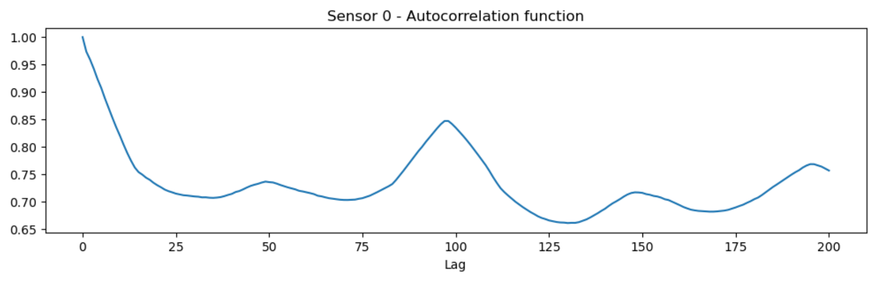


(a) Sensor 0

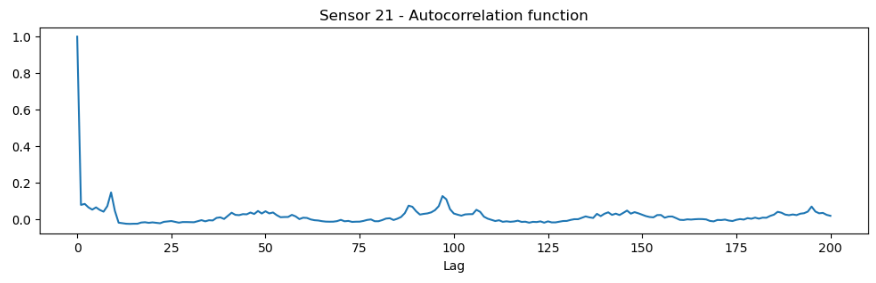


(b) Sensor 21

Figure 2: Raw signal visualization

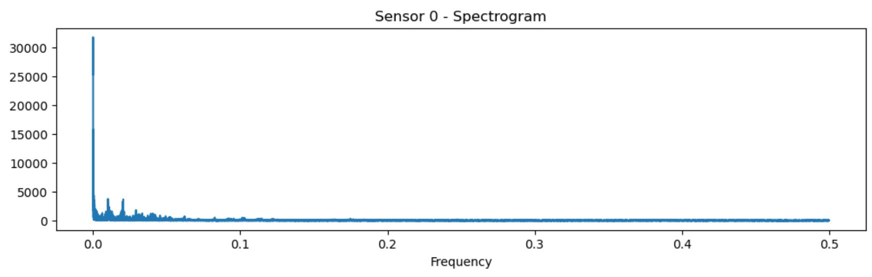


(a) Sensor 0

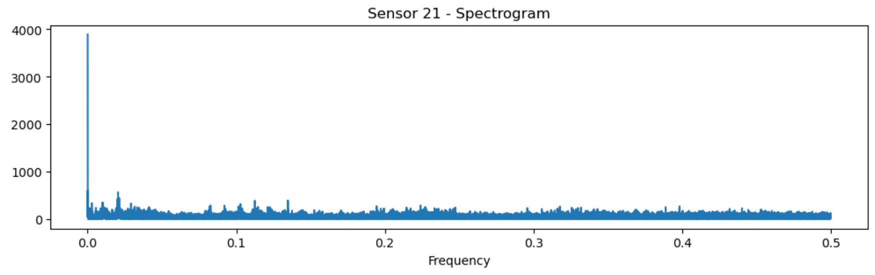


(b) Sensor 21

Figure 3: Autocorrelation functions



(a) Sensor 0



(b) Sensor 21

Figure 4: DFT spectrum

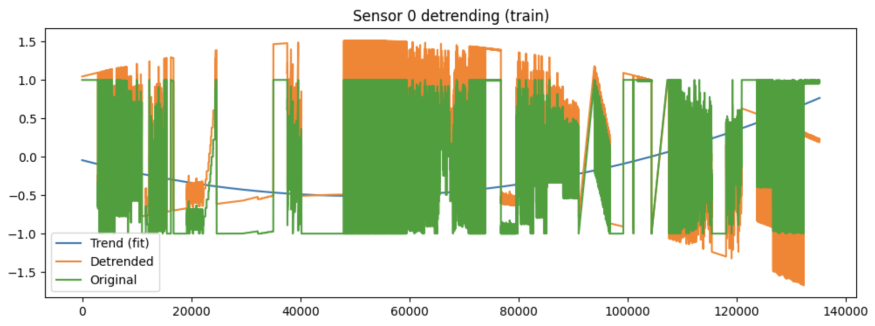


Figure 5: Raw signal and detrending (order 2) overview for Sensor 0.

Appendix B — Figures example for the SMD Dataset analysis

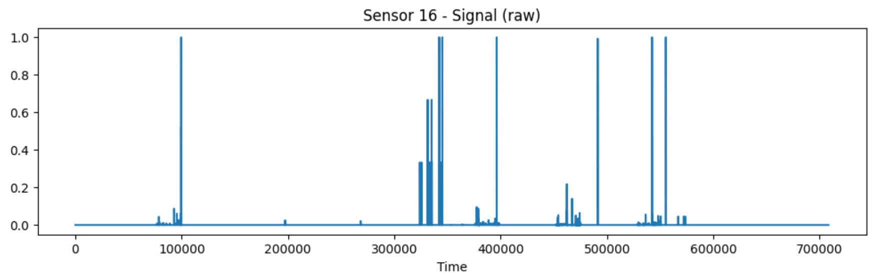


Figure 6: Raw signal for Sensor 16

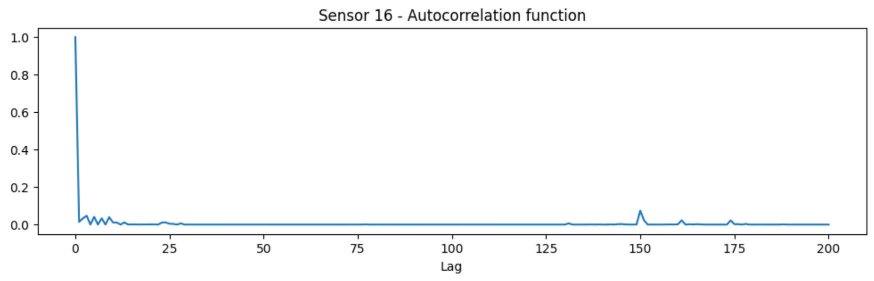


Figure 7: Autocorrelation function for Sensor 16

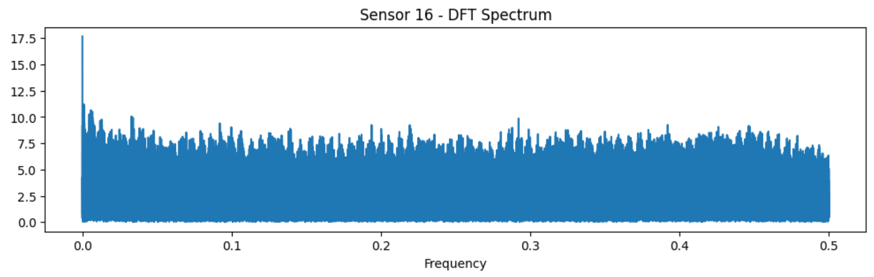


Figure 8: DFT spectrum of Sensor 16

Appendix C — Context window size variations

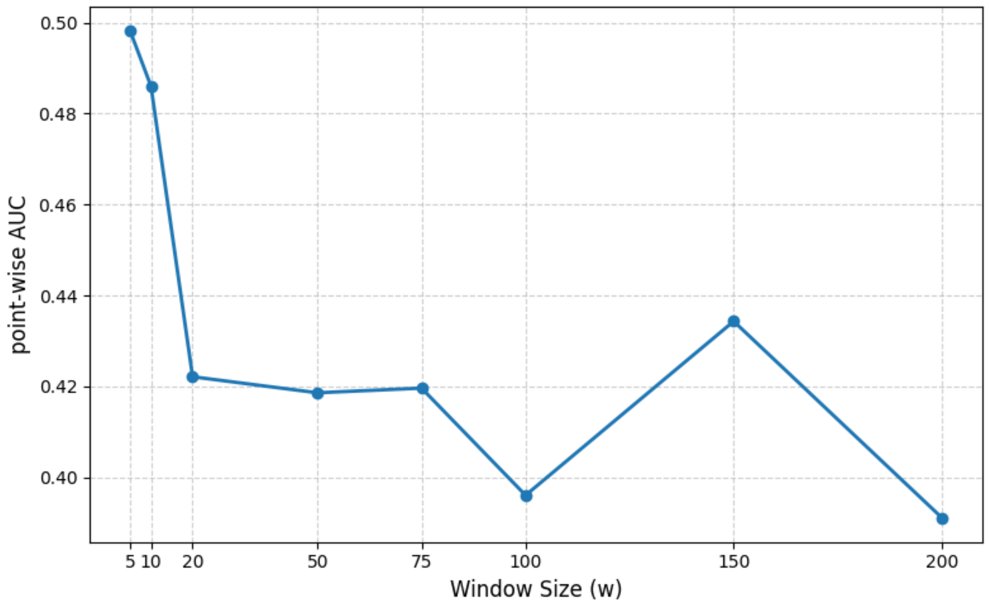


Figure 9: AUC against context window size for SMAP dataset

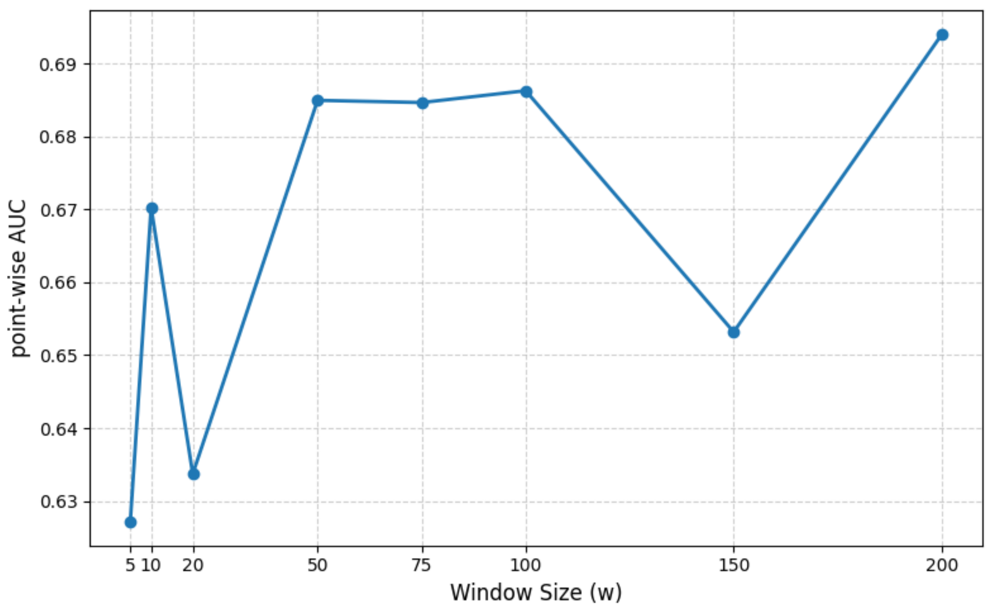


Figure 10: AUC against context window size for SMD dataset

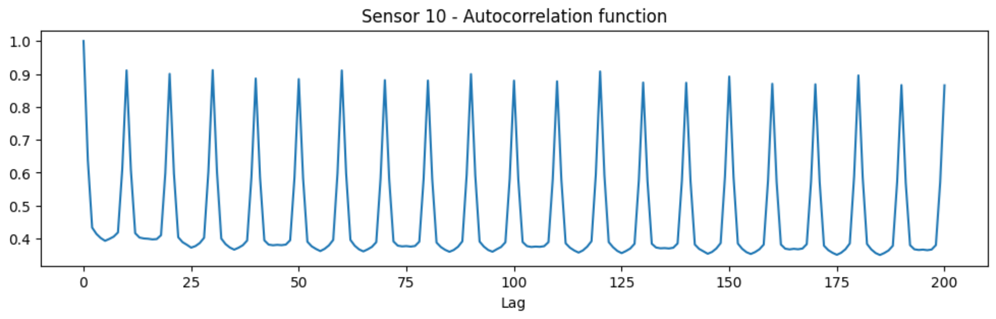


Figure 11: Autocorrelation function for Sensor 10