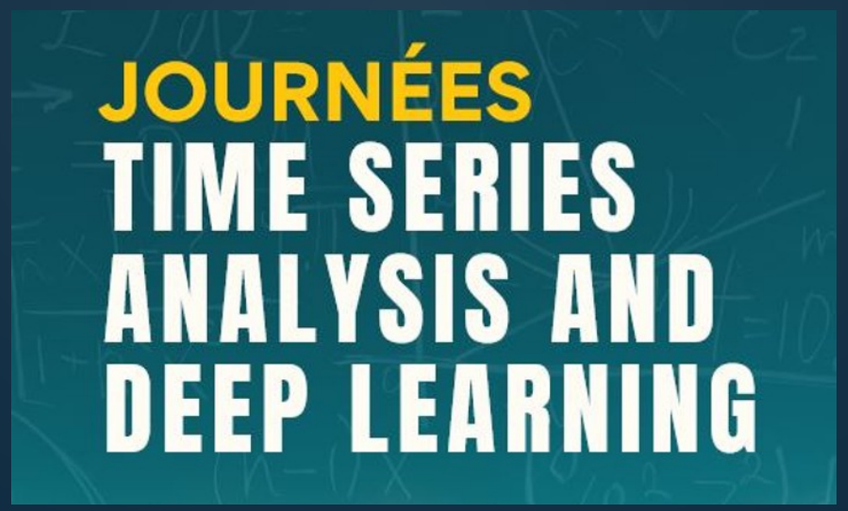
PatchTrAD
A Patch-Based Transformer focusing on Patch-Wise Reconstruction Error for Time Series Anomaly Detection

Mokhtar Z. Alaya



June 8-9, 2026, Institut Camille Jordan, Saint-Etienne

ALLIANCE SORBONNE UNIVERSITÉ
utc Recherche
LMAC
donnons un sens à l'innovation
ALLIANCE SORBONNE UNIVERSITÉ
utc

Joint work with

Samy-Melwan Vilhes
LITIS, INSA Rouen

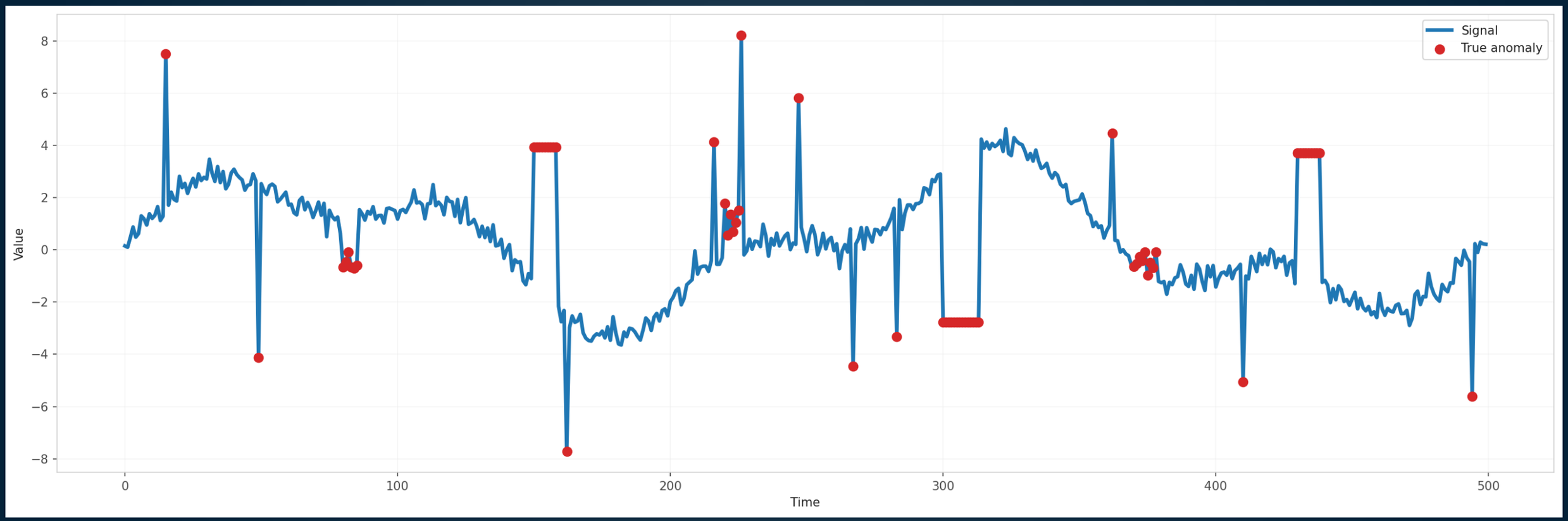
Gilles Gasso
LITIS, INSA Rouen

Eusipco2025
SIGNAL PROCESSING IN THE LAND OF ART, CULTURE AND BEAUTY

1. Motivations

Time Series Anomaly Detection (TSAD)

TSAD identifies observations that deviate from normal behavior in streaming data, and it is important for early monitoring and mitigation in many domains.



Anomaly: rare point or sequence (of a given length) potentially non-desired.

TSAD: real applications

- **System failures and malfunctions:** In industrial settings, anomalies in sensor readings can signal equipment breakdown, requiring predictive maintenance.

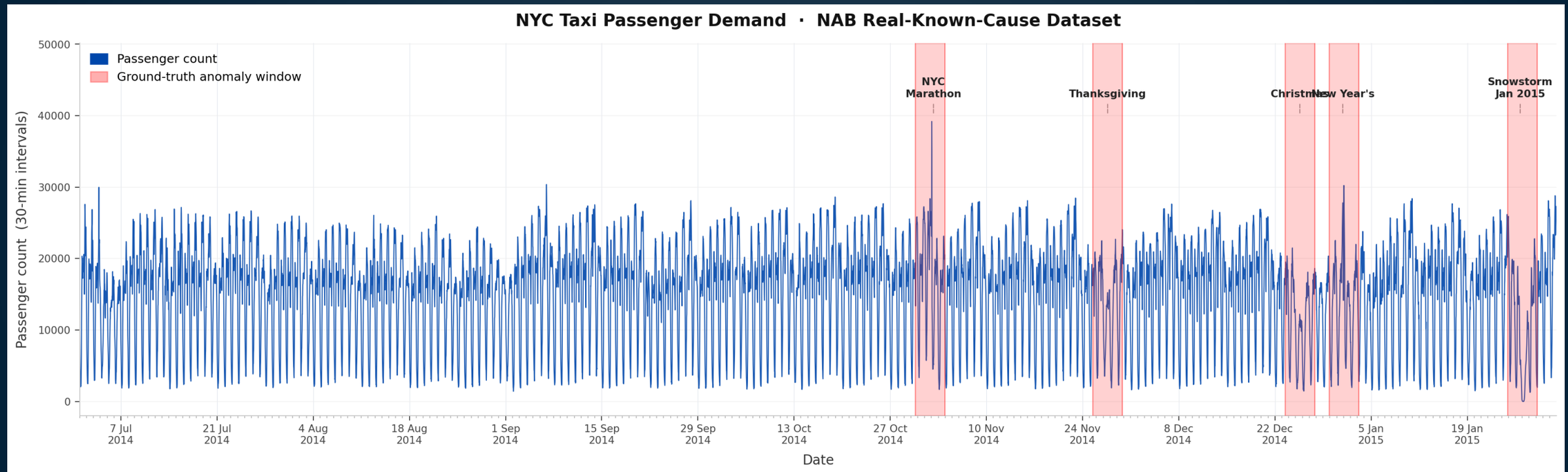
- **Fraud detection:** Unusual patterns in financial transactions or network traffic can point to fraudulent activities or cyberattacks.

- **Healthcare monitoring:** Anomalies in patient vital signs or medical sensor data can alert healthcare providers to critical health events.

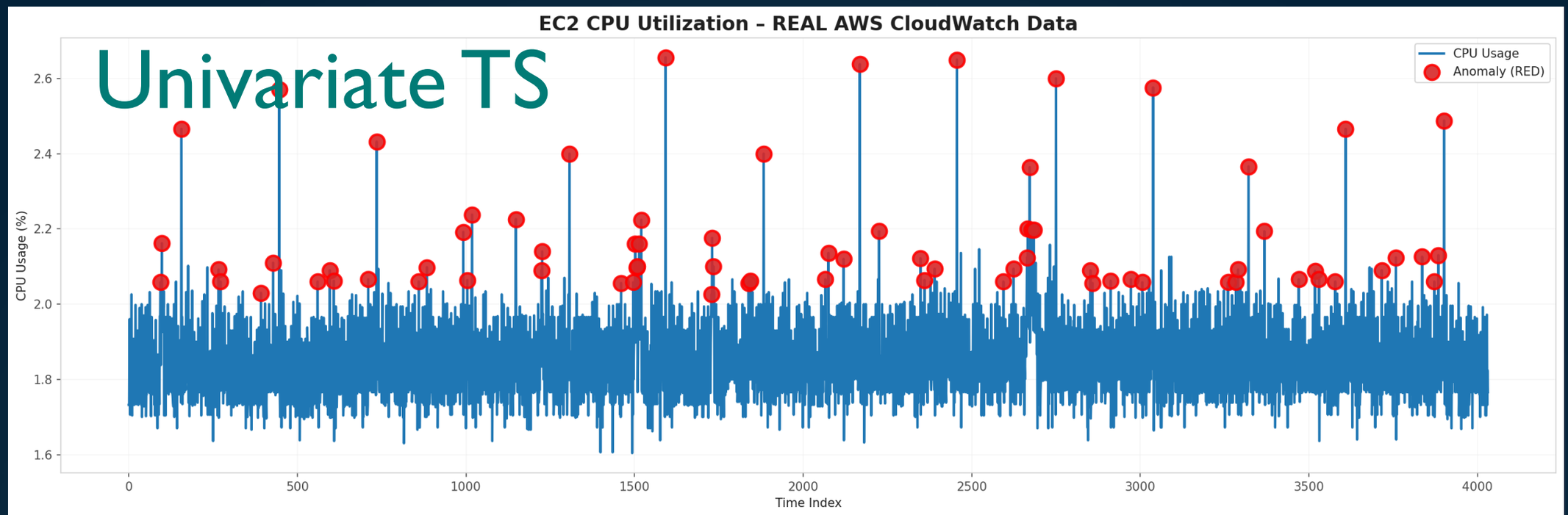
- **Environmental monitoring:** Detecting abnormal changes in climate data or pollution levels can help identify environmental crises.

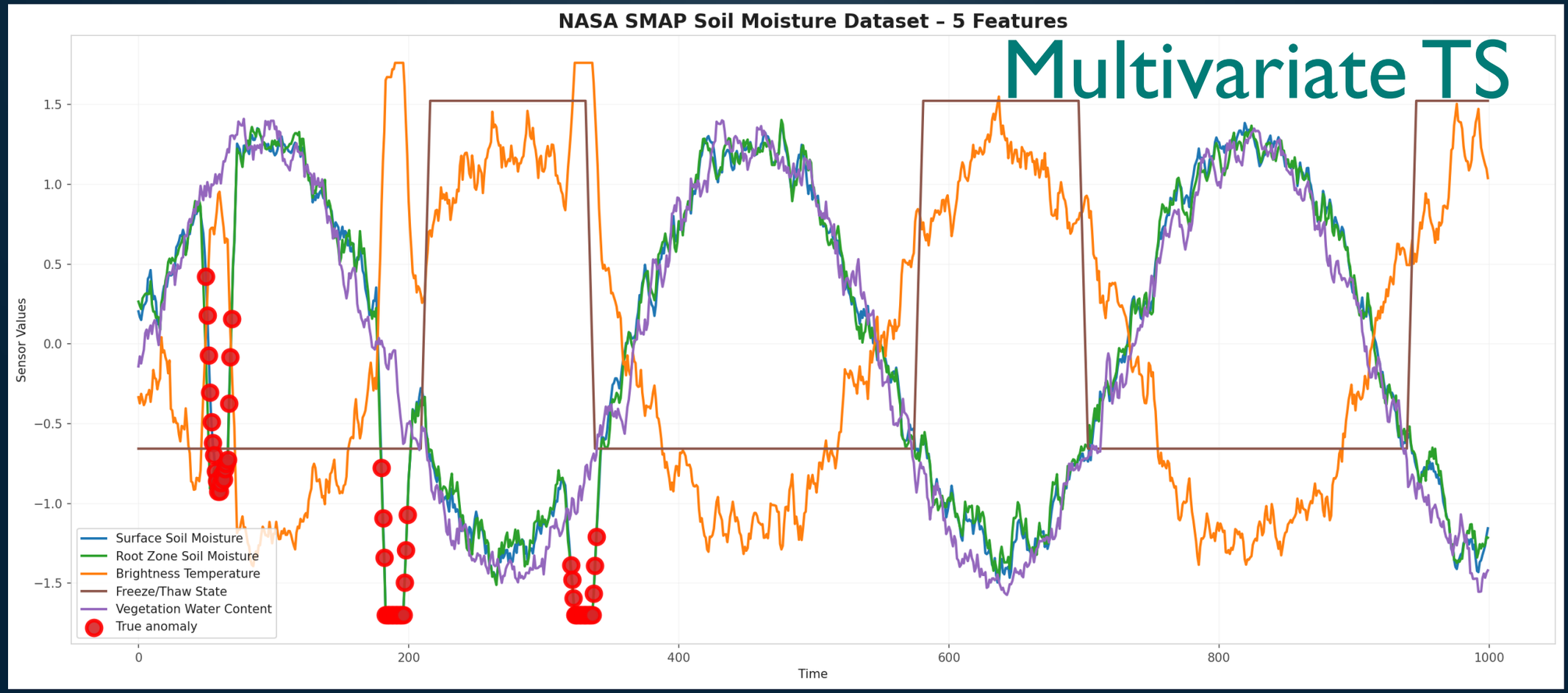
- **Urban planning:** Identifying unusual spikes or drops in public transport usage (like taxi passengers) can inform urban planning and emergency response.

TSAD: New York Taxi Data



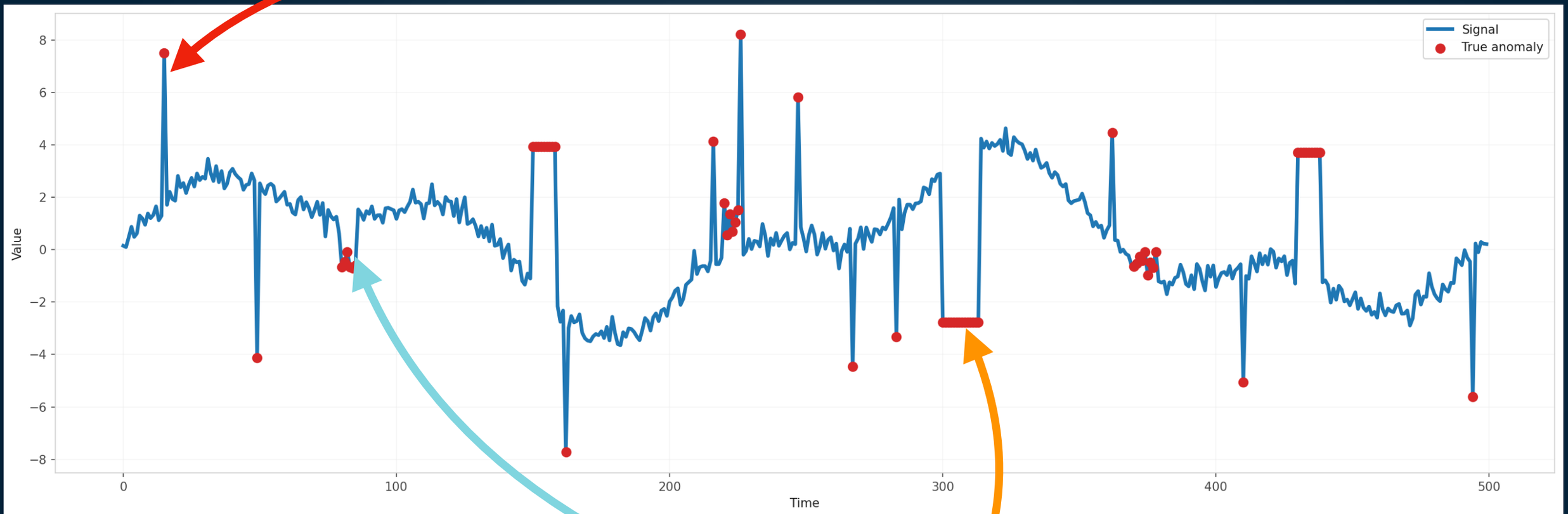
Type of Time Series





Type of anomalies

Point anomaly: observation differs significantly from the overall behavior of the signal.



Contextual anomaly: observation appears normal overall, but deviates from its local context.

Collective anomaly: series of observations forms an abnormal pattern, whether these are isolated anomalies or recurring. **contextual**

Semi-supervised learning for TSAD

- Anomalies are rare events. We train the model exclusively on normal (clean) samples.
- At inference time, when anomalies may occur, the model outputs an anomaly score.
- Higher scores indicate higher likelihood of abnormal behaviour.

$$\textbf{Score}\left(x_t = \begin{pmatrix} x_t^{(1)}\,(\text{Feature 1}) \\ \vdots \\ x_t^{(M)}\,(\text{Feature M}) \end{pmatrix}\right)$$

Semi-supervised learning for TSAD

$$x_t = \begin{pmatrix} x_t^{(1)} \text{(Feature 1)} \\ \vdots \\ x_t^{(M)} \text{(Feature M)} \end{pmatrix} \in \mathbb{R}^M$$

Model

- **Statistical model:** Z-scores, Isolation Forest, KNN, …
- **Classical ML model:** One Class SVM, …
- **Deep learning model:** neural networks

Anomaly Score

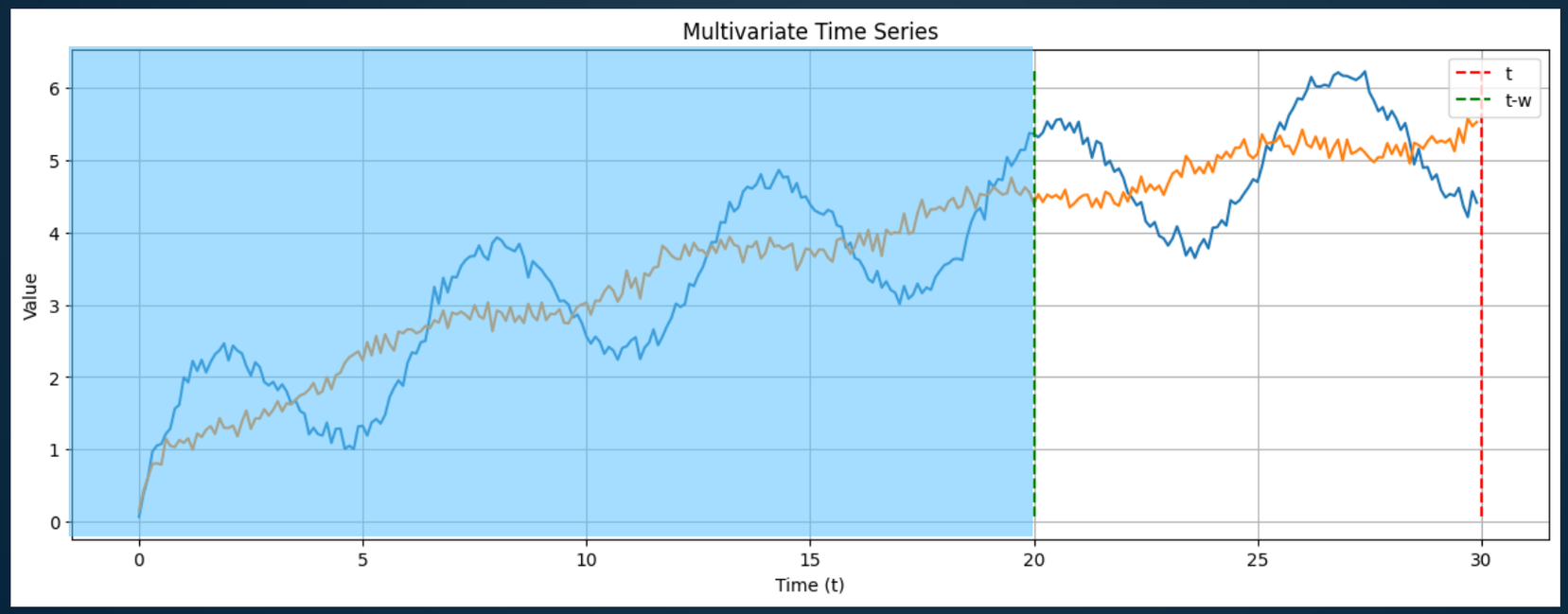
Anomalous

Normal

- In the following, our attention is focused on deep learning–based models.

Deep Learning for TSAD: Sliding window

- In practice, one uses a sliding window of a predefined size ω, .e., one relies on the the most recent $x_{t-\omega+1:t} \in \mathbb{R}^{\omega \times M}$ to infer the normality of x_{t+1}.



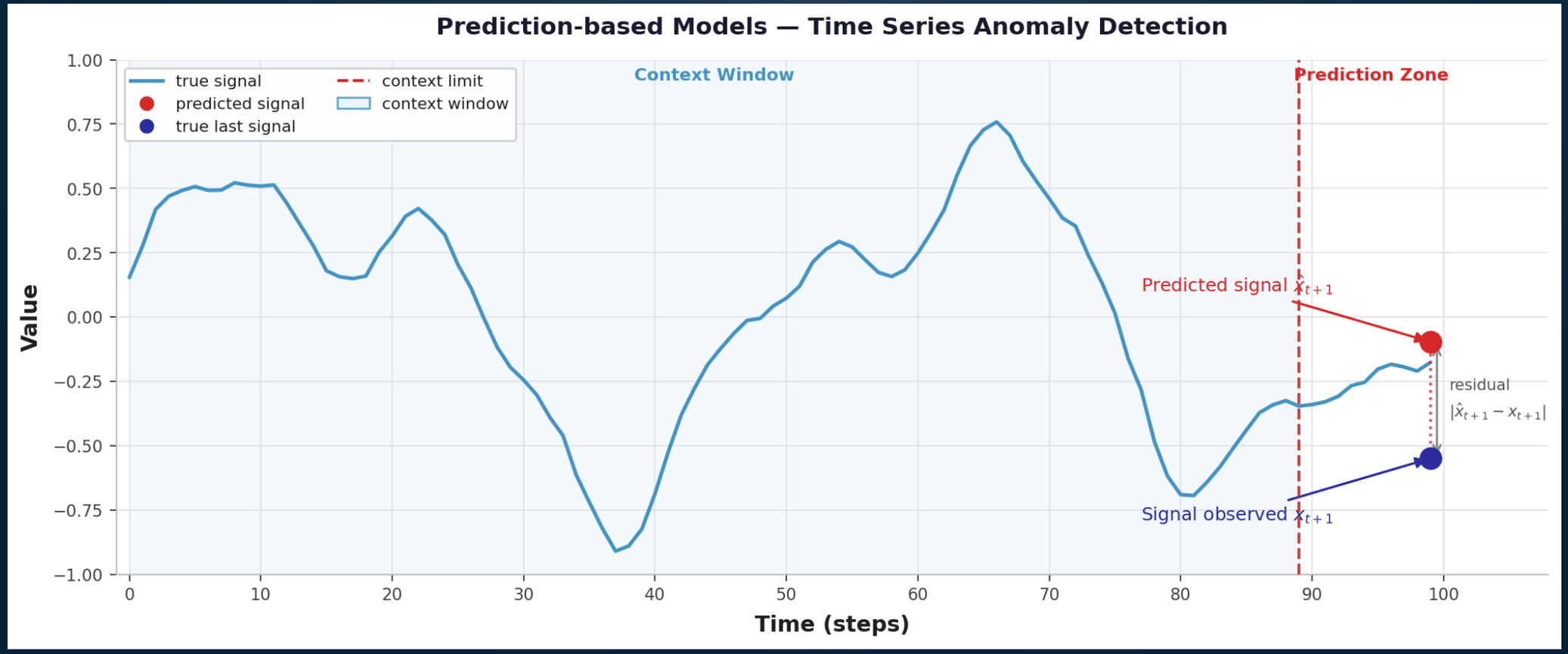
$$M = 2, t = 30, \omega = 10$$

- $x_{t-\omega+1:t}$: signals from time $t - \omega + 1$ to t.

Deep Learning for TSAD: Prediction error-based anomaly

- If the prediction error exceeds a predefined threshold, x_{t+1} is deemed anomalous otherwise, it is considered normal.
  - Model's prediction of the true signal x_{t+1}

$$\hat{x}_{t+1} = f_\theta(x_{t-\omega+1:t}) \in \mathbb{R}^M$$



Deep Learning for TSAD: Prediction error-based anomaly

- The anomaly score of prediction-based models corresponds to the prediction error given by

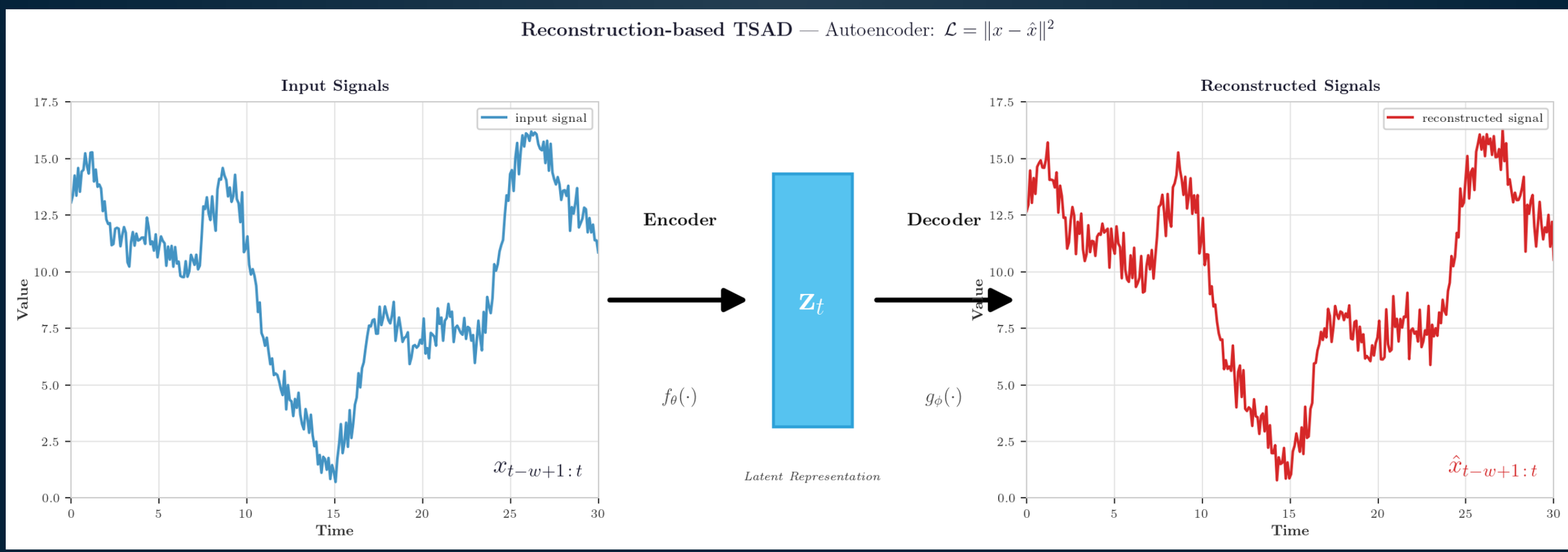
$$\text{anomaly score} = \|\hat{x}_{t+1} - x_{t+1}\|^2$$

- According to the model, the higher the anomaly score, the more likely x_{t+1} is abnormal.

- Typical models include **LSTM-based model**, **Transformer-based model** [Vaswani et al. NeuriPS '17] , **PatchTST** [Ni et al. ICLR'23]

Deep Learning for TSAD: Reconstruction-based models

- Reconstruction models aim to reconstruct the input window.
- These models commonly learn a latent representation space in an autoencoder manner based on windowed inputs $x_{t-w+1:t}$.

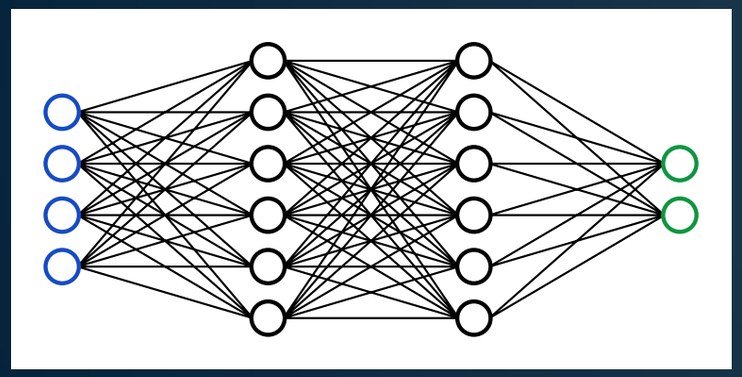


Deep Learning for TSAD: Reconstruction-based models

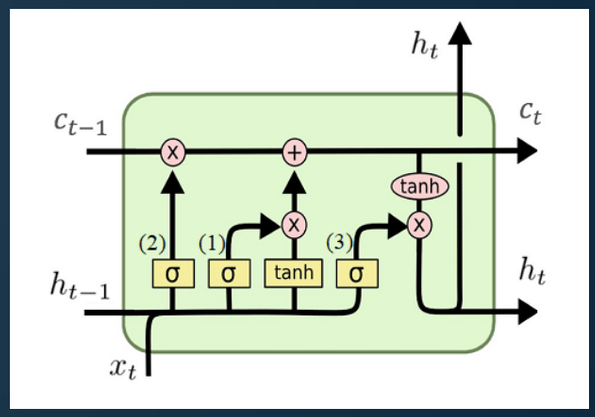
$$\hat{x}_{t-w+1:t} = \text{Decoder}(\text{Encoder}(x_{t-w+1:t})) \in \mathbb{R}^{\omega \times M}$$

- The anomaly score of reconstruction-based models corresponds to the reconstruction error given by

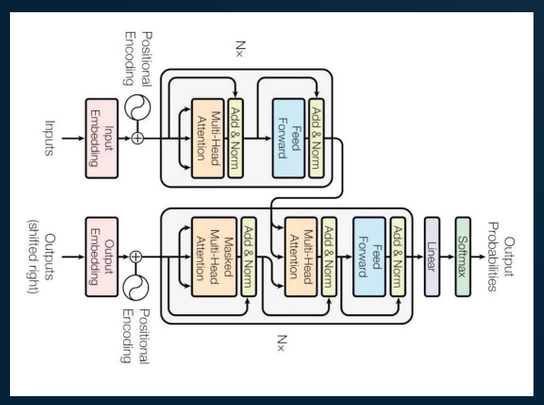
$$\text{anomaly score} = \|\hat{x}_{t-w+1:t} - x_{t-w+1:t}\|^2$$



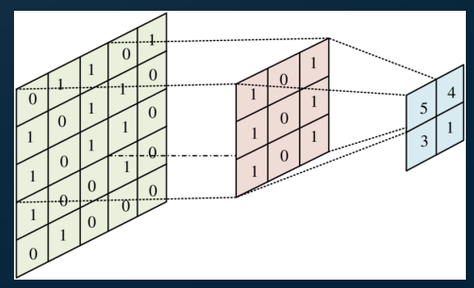
MLP (**PatchAD**) [Zhong et al. '24]



LSTM-based autoencoder [Provotar et al., ATIT'19]



Transformer (**TranAD**) [Tulie et al. '22]



CNN [Ismail Fawaz et al. '20]

2. PatchTrAD

Patching (1)

- **Patching mechanism**: divides time series into smaller, overlapping segments called "patches," treating these as tokens for the Transformer.
- This helps in extracting local semantic meaning and reduces the sequence length, making the Transformer more efficient.

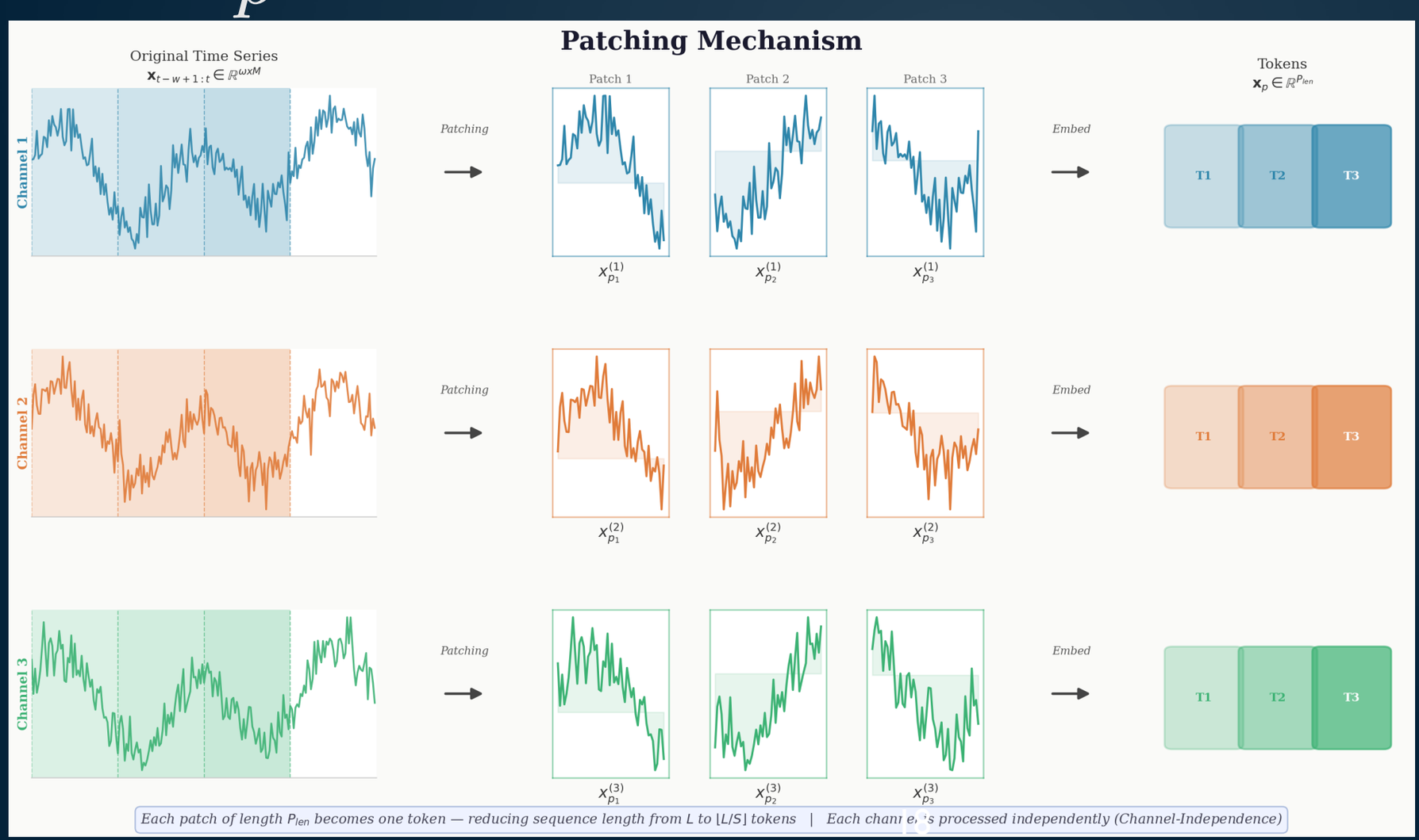
- The input data is a window $x_{t-\omega+1:t}$.
- For a given stream of a modality m, denoted $x^{(m)}_{t-w+1:t}$ its patch transformation is determined by the patch length P_{len} and the stride S.
- Before patching, we pad the m-th stream by repeating S times the last/test observation $x^{(m)}_t$.
- Patches can overlap and $x^{(m)}_t$ belongs to the last patch.

Patching (2)

- The number of patches si given by

$$P_{\text{num}} = \left\lfloor \frac{(w - P_{\text{len}})}{S} \right\rfloor + 2.$$

- We transform the input window $x_{t-w+1:t} \in \mathbb{R}^{M \times \omega}$ into $x_p \in \mathbb{R}^{M \times P_{\text{num}} \times P_{\text{len}}}$



$$x_p^{(m)} \in \mathbb{R}^{P_{\text{num}} \times P_{\text{len}}}$$

set of patches for the m-th modality extracted from x_p

$$x_{p_i}^{(m)} \in \mathbb{R}^{P_{\text{len}}}$$

i-th patch for the m-th modality, with $i \in \{1, \ldots, P_{\text{num}}\}$

Transformer attention mechanism

- We project x_p using a learnable $W_{\text{proj}} \in \mathbb{R}^{P_{\text{len}} \times D_{\text{model}}}$ and add a fixed positional encoding $W_{\text{pe}} \in \mathbb{R}^{P_{\text{num}} \times D_{\text{model}}}$

$$\tilde{x}_p = x_p W_{\text{proj}} + W_{\text{pe}} \in \mathbb{R}^{M \times P_{\text{num}} \times D_{\text{model}}}$$

- The single-head attention block for one layer is defined by $W_Q \in \mathbb{R}^{D_{\text{model}} \times D_k}$, $W_K \in \mathbb{R}^{D_{\text{model}} \times D_k}$, $W_V \in \mathbb{R}^{D_{\text{model}} \times D_v}$ and $W_{\text{out}} \in \mathbb{R}^{D_v \times D_{\text{model}}}$ (only one head and one layer presented, with D_k, D_v hidden dimensions).

$$Q = \tilde{x}_p W_Q \in \mathbb{R}^{M \times P_{\text{num}} \times D_k},$$

$$K = \tilde{x}_p W_K \in \mathbb{R}^{M \times P_{\text{num}} \times D_k},$$

Patch head

$$V = \tilde{x}_p W_V \in \mathbb{R}^{M \times P_{\text{num}} \times D_v},$$
$$h = \text{Softmax}\left(\frac{QK^\top}{\sqrt{D_{\text{model}}}}\right) V \in \mathbb{R}^{M \times P_{\text{num}} \times D_v},$$
$$z = hW_{\text{out}} \in \mathbb{R}^{M \times P_{\text{num}} \times D_{\text{model}}}.$$

- The patch head takes as input the output of the encoder $z \in \mathbb{R}^{M \times P_{\text{num}} \times D_{\text{model}}}$. It projects each $z^{(m)} \in \mathbb{R}^{P_{\text{num}} \times D_{\text{model}}}$ back to the patch length size using M learnable linear functions $W_{\text{out}}^m \in \mathbb{R}^{D_{\text{model}} \times P_{\text{len}}}$. Hence we have

$$\tilde{x}_p^{(m)} = z^{(m)} W_{\text{out}}^m \quad \in \mathbb{R}^{P_{\text{num}} \times P_{\text{len}}},$$
$$\tilde{x}_p = \text{concat}(\tilde{x}_p^{(1)}, \ldots, \tilde{x}_p^{(M)}) \quad \in \mathbb{R}^{M \times P_{\text{num}} \times P_{\text{len}}}.$$

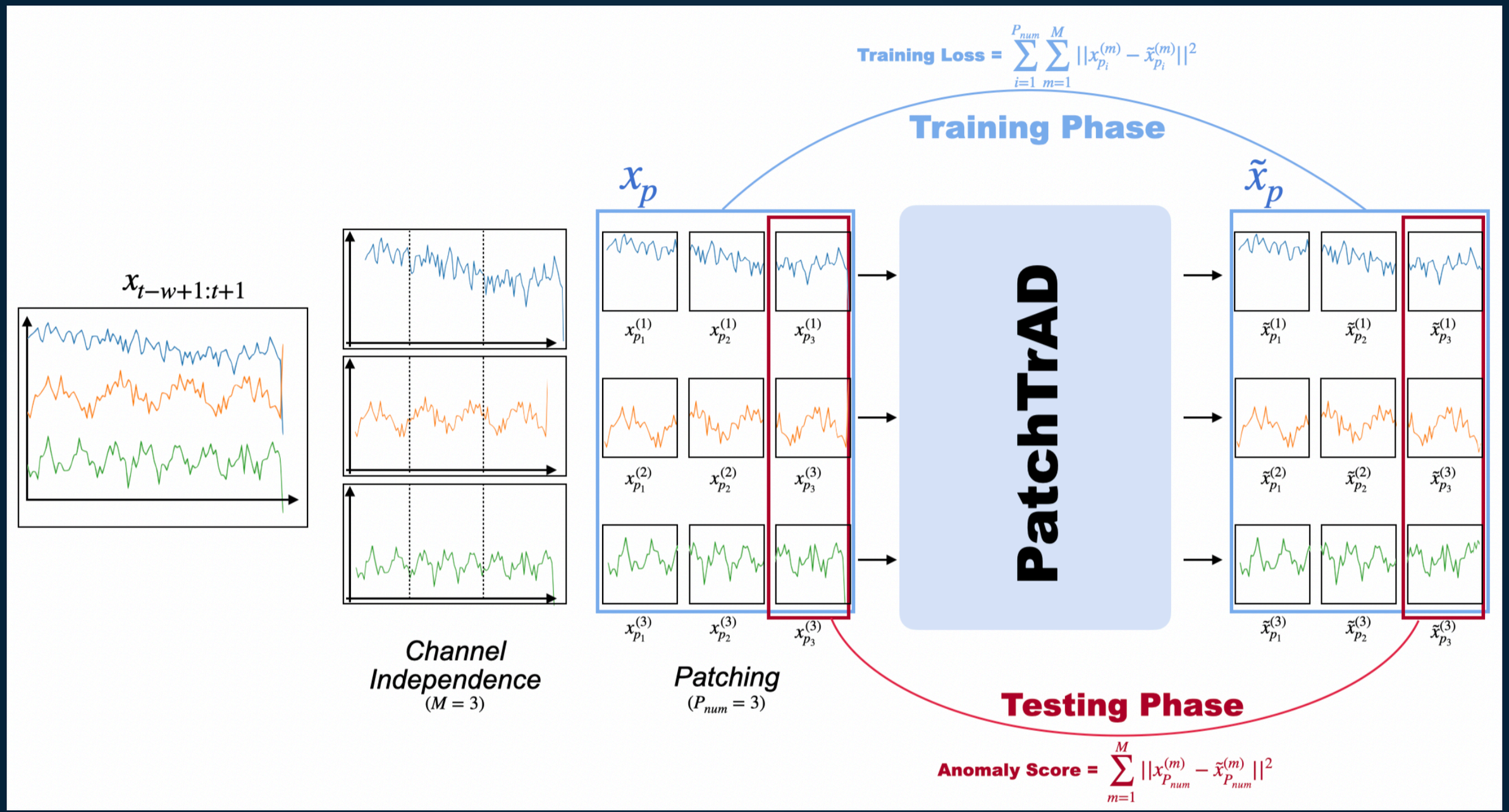
Patching Transformer

Channel independence

- **Channel independence**: refers to the scenario where each input patch contains information from only a single modality.
- Each channel is patched, embedded, and passed through the same Transformer encoder independently.

- This reduces complexity, increases the number of training samples, and helps the model focus on temporal patterns inside each variable.
- Empirical studies show that this design improves model robustness while maintaining performance (Han, Ye, and Zhan 2023).

PatchTrAD : Overview



By construction, the test observation x_t always belongs to the last patch of each modality. Therefore, during inference, we focus on the error of this final patch.

PatchTrAD: Benchmarks

- We train models on a training set containing only normal observations.
- We evaluate its performance on a test set, which includes both normal and anomalous samples, using the ROC-AUC metric.

Dataset	# Feats	Train Size	Test Size	% Anomaly Test
NYC Taxi	1	5570	4750	0.11
EC2	1	1984	2049	0.15
SWAT	51	495000	449919	12.13
SMD	38	708405	708420	4.16
MSL	55	58317	73729	10.48
SMAP	25	140825	444035	12.85

Datasets Statistics:

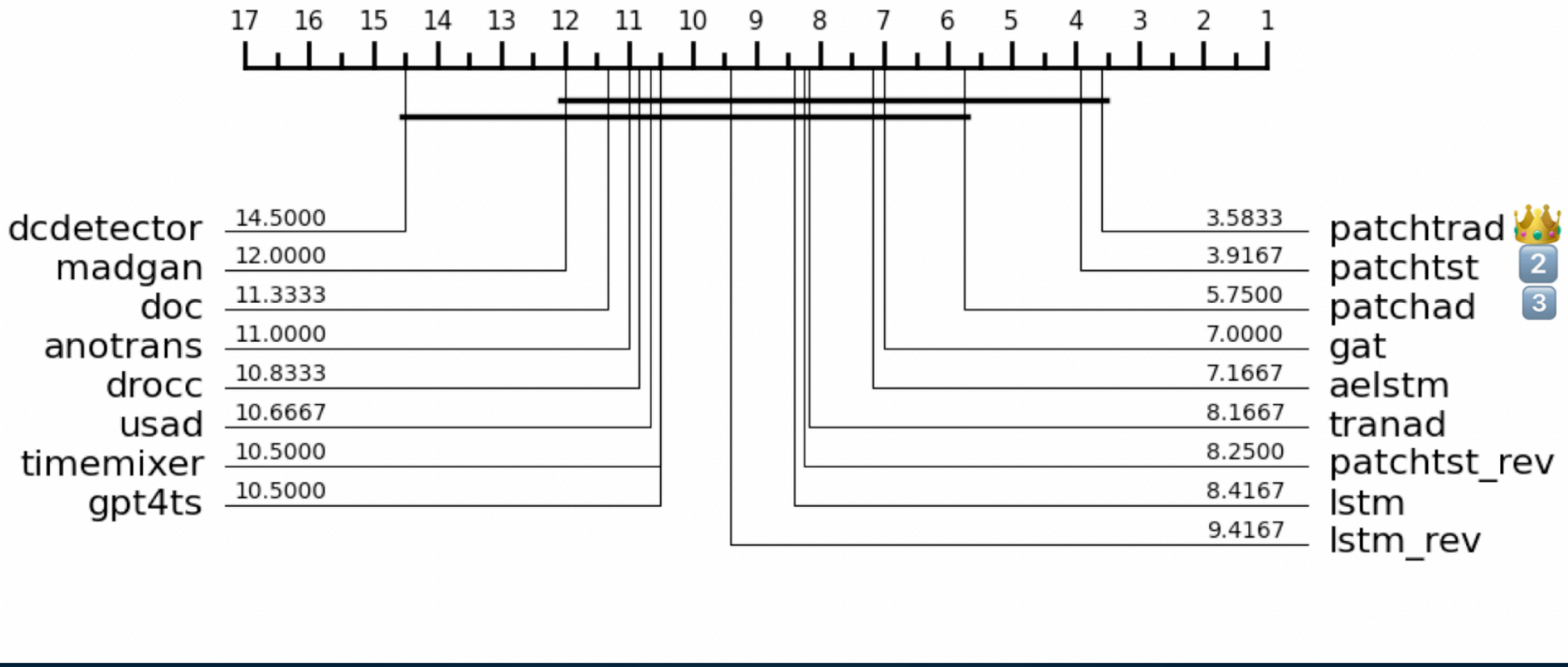
Results: ROC-AUC

ROC-AUC scores (red: first, blue: second, green: third)

	Dataset	NYC-Taxi	EC2	MSL	SWaT	SMAP	SMD
Category	**Model**				**ROC-AUC**		
Others	DC-Detector	0.498	0.827	0.537	0.435	0.566	0.530
	AnomalyTransformer	0.491	0.994	0.553	0.819	0.621	0.678
	DOC	0.704	0.804	0.538	0.404	0.634	0.766
	DROCC	0.529	0.886	0.593	0.751	0.705	0.638
Predictions-based	PatchTST-revin	0.552	0.999	0.626	0.233	0.537	0.873
	LSTM-revin	0.646	0.998	0.627	0.238	0.586	0.858
	LSTM	0.511	0.999	0.595	0.842	0.604	0.833
	GAT	0.689	0.999	0.617	0.816	0.646	0.820
	PatchTST	0.696	0.999	0.626	0.843	0.622	0.882
Reconstruction-based	MADGAN	0.782	0.011	0.460	0.791	0.568	0.708
	USAD	0.669	0.977	0.684	0.255	0.547	0.605
	TimeMixer	0.523	0.942	0.681	0.235	0.536	0.900
	GPT4TS	0.272	0.954	0.699	0.235	0.544	0.890
	TranAD	0.551	0.967	0.644	0.815	0.581	0.884
	AE-LSTM	0.716	0.998	0.612	0.840	0.618	0.828
	PatchAD	0.972	0.998	0.622	0.822	0.671	0.818
	PatchTrAD (ours)	0.922	0.999	0.661	0.845	0.660	0.869

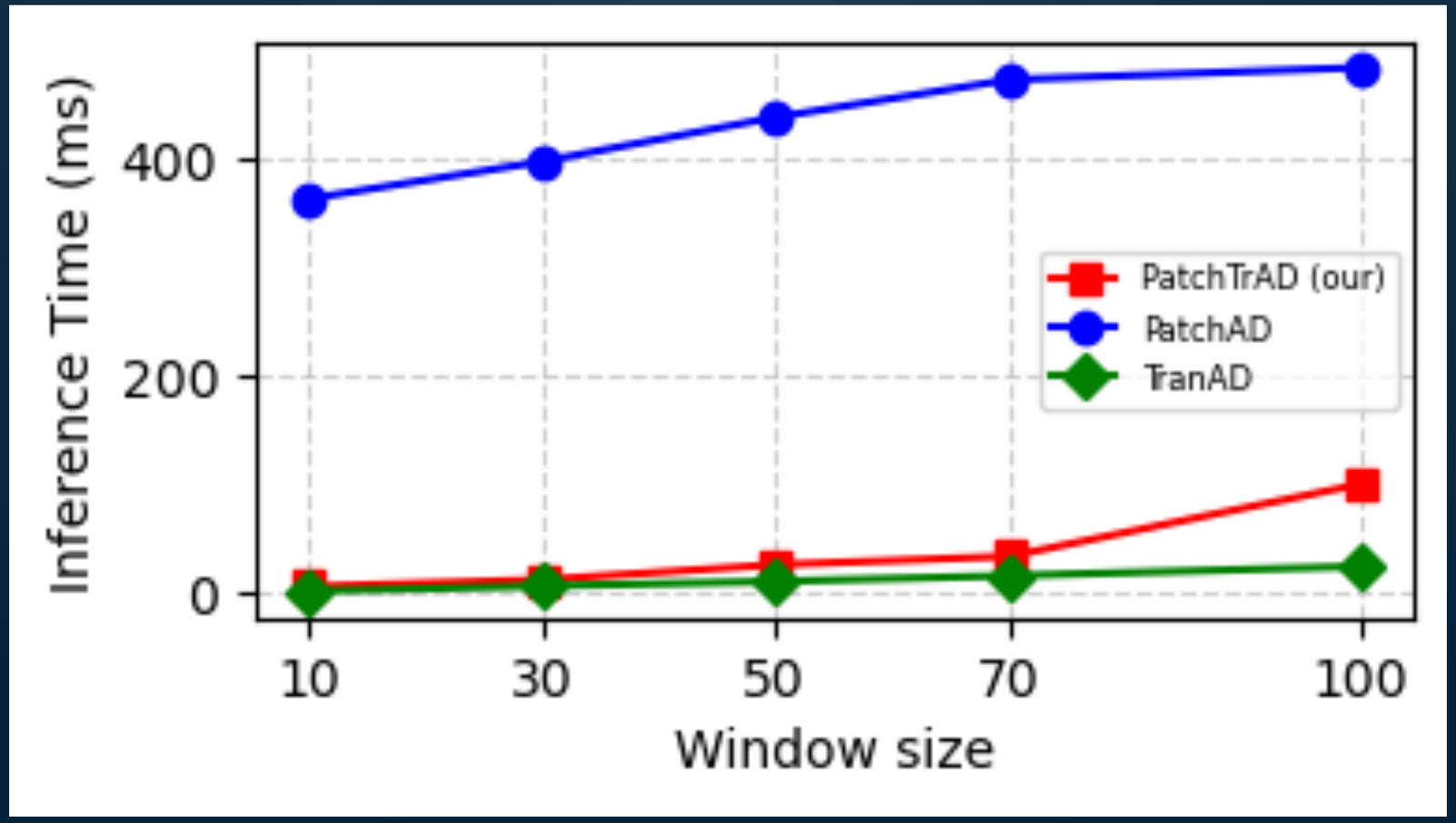
Results: Nemenyi test

Critical difference diagram for ROC-AUC scores using the post-hoc Nemenyi test with $\alpha = 5\%$, where better-ranked methods appear on the upper right.



Inference-time computation

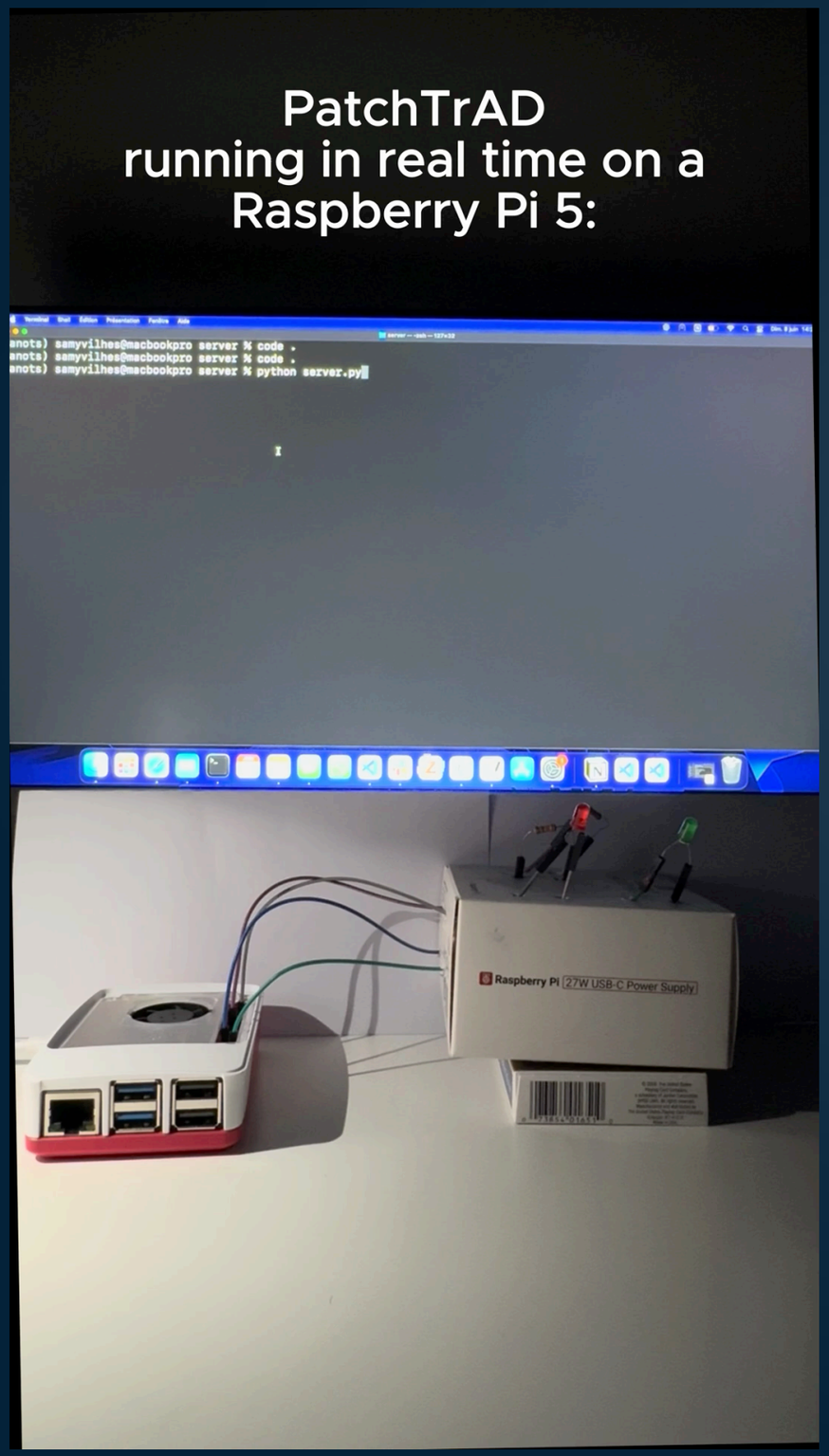
- Since we focus on real-time anomaly detection, the models under consideration should be both fast and efficient during inference.
- PatchAD: most time-consuming and PatchTrAD is more efficient than PatchAD.
- PatchTrAD is still behind TranAD, and this gap becomes more noticeable as the window size increases.



Take home meassage

- We introduced a Patch-Based Transformer leveraging reconstruction error, that combines the strengths of patch-based transformers with channel independence ,and reconstruction-based approaches for TSAD.
- Effective for both univariate and multivariate signal monitoring,
- Efficient and lightweight at inference,
- Achieves competitive performance compared to SOTA methods,
- PatchTrAD shows strong potential for addressing future industrial TSAD challenges.

Samy's investment:
A formal closing point !



- PatchTrAD running locally on a Raspberry Pi for real-time anomaly detection on time series data.
- Observations are send every 0.05 seconds from a simulated sensor stream, and the model processes this data in real-time.
- The system uses a green LED to indicate normal behaviour and switches to a red LED when an anomaly is detected.

Code available at:
https://github.com/vilhess/PatchTrAD

Thank you !

Ablation Study

- PatchTrAD's architecture is determined by the patch size and stride, together define the number of patches.
- We analyze how these parameters impact the final score by evaluating the model exclusively on NYC Taxi Demand and SWaT datasets.

Dataset		NYC Taxi ($w = 32$)	SWaT ($w = 100$)
P_{len}	S	**ROC-AUC**	
3	3	0.776	0.839
5	3	0.904	0.839
5	5	0.832	0.839
6	6	0.872	0.842
8	3	0.838	**0.846**
8	5	0.801	0.844
8	6	**0.922**	<u>0.845</u>
8	8	<u>0.917</u>	<u>0.845</u>
16	12	0.536	0.821
16	16	0.801	0.820
28	22	0.890	0.822
28	28	0.544	0.823
32	28	0.549	0.829
32	32	0.568	0.825

- If the patch size and stride are too large, performance decreases.
- Conversely, if they are too small, the model does not achieve its best results.
- Based on our experiments, **a patch length of 8 and a stride of 6** yield the best detection performances.