Application of Implicit Neural Representation for Multi-Variate Time Series

Aamir Mohd[a], Alexis Huet[b], Alessandro Finamore[b], Giulio Franzese[a], Pietro Michiardi[a,*]

[a] *EURECOM, France*
[b] *Huawei Technologies, France*

Abstract

In this paper, we explore the effectiveness of Implicit Neural Representations (INRs) for multi-variate time series tasks. We introduce `MetaINAD`, a method that models time series data as functions, using a meta-learning approach to train a base INR model and to learn modulation weights to adapt it to data samples. We study the effectiveness of `MetaINAD` and compare it to the current state-of-the-art for multi-variate time series anomaly detection tasks, through an extensive experimental campaign on four public datasets. Our results demonstrate that `MetaINAD` is competitive, while offering inherent advantages in handling missing values without requiring explicit imputation techniques. Our analysis finally reveals important insights and limitations about how INRs perform with high-dimensional feature spaces, compared to their traditional uses in lower-dimensional domains. We believe this works offers a comprehensive view on the application of INRs to multi-variate time series anomaly detection, their benefits and challenges, as well as possible avenues for future research in this domain.

Keywords: implicit neural representation, time series, anomaly detection

1. Introduction

Time series Anomaly Detection (AD) methods are widely used in computer systems for monitoring and security purposes across various domains, e.g., water control industrial systems [14, 2], computing servers [30], IT system monitoring [3], or spacecraft telemetry [17]. Across recent literature , Implicit Neural Representation (INR) frameworks are worth of attention since they have shown significant capability to encode the functional relationship between a data sample and its coordinates through a simple Multi-Layer Perceptron (MLP) [28]. INR frameworks were initially applied to 3D graphics and rendering [29], but recently gained popularity for other domains like computer vision , audio [32], video [6], and time series [9, 8, 11, 25, 19], thanks to their capabilities to perform compression, super-resolution.

In this paper, we investigate the potential benefits and limitations of INRs when applied to multi-variate time series tasks. In such a scenario, the INR framework maps a function from the time coordinate t to the corresponding multi-variate signal $X(t)$. Our approach builds on COIN++ [9], an INR meta-learning scheme that initially trains a base neural network across multiple data points. Subsequently, it *modulates* a subset of the MLP weights through a few inner-loop gradient-descent steps, adapting them into a sample-specific neural network representation.

While COIN++ focuses only on data compression, we extend its capabilities to explore signal reconstruction and anomaly detection by incorporating reconstruction-based methods [7, 38], which identify anomalies through reconstruction errors. Intuitively, we expect normal instances to exhibit smaller reconstruction errors than anomalous ones. Our method is named `MetaINAD`.

To the best of our knowledge, only INRAD [19] has previously explored INR-based time series anomaly detection. However, INRAD employs a two-step learning process without meta-learning: it pre-trains on the entire dataset and then re-trains on the test set, which can introduce data leakage. In contrast, our approach mitigates this issue through meta-learning, where a shared initialization is learned from labeled normal instances and later adapted to each new instance using only a small set of modulation parameters avoiding data leakage. Other recent work on time series using INR have primarily focused on synthetic generation [11], forecasting [25], or imputation [4]. In contrast, `MetaINAD` specifically targets reconstruction and anomaly detection in real-world telemetry data from monitoring systems [2, 30, 14]. We conduct comprehensive comparisons with state-of-the-art methods [3, 38, 37, 21, 1], employing multiple F1-score metrics as recommended by [13] to ensure thorough evaluation.

We demonstrate the adaptability of our approach in real-world conditions, where data incompleteness and misalignment are common challenges. Finally, we conduct a critical examination of INRs for time series anomaly detection, exploring several key aspects that influence their effectiveness. Unlike image-based applications where INRs typically handle 2-3 dimensional inputs, our analysis investigates scenarios with significantly higher feature dimensionality and their impact on model performance. These investigations reveal important trade-offs and limitations that differ notably from INR applications in other domains, providing crucial insights for future applications.

*Corresponding author. Emails: mohd.aamir@eurecom.fr (A. Mohd), alexis.huet@huawei.com (A. Huet), alessandro.finamore@huawei.com (A. Finamore), giulio.franzese@eurecom.fr (G. Franzese), pietro.michiardi@eurecom.fr (P. Michiardi).

2. Background

INR preliminaries. An INR approximates the mapping from the *coordinate* space $\mathcal{T} \subset \mathbb{R}^n$ to the *signal* or *feature* space $\mathcal{X} \subset \mathbb{R}^m$ (for a single coordinate $t \in \mathcal{T}$) via an MLP f_θ with weights θ [15]. For example, in the image domain, the coordinate space $\mathcal{T}$ represents the set of pixel locations (a, b), while the signal space $\mathcal{X}$ corresponds to the RGB color values. An image X_i is therefore a mapping $\mathcal{T} \to \mathcal{X}$.

In the multi-variate time-series setting, $\mathcal{T}$ represents the time indices, while the input $\mathcal{X}$ corresponds to the multi-variate feature space. A *window* X_i is then defined as the mapping $\mathcal{T} \to \mathcal{X}$, representing a "time slice" of the entire time series. The time series is segmented into continuous disjoint windows indexed by i, each of length J. Each window X_i consists of a set of coordinate-value pairs:

$$X_i = \{(j, X_i(j))\}_{j=1}^{J}, \tag{1}$$

where j is the time index within the window, and $X_i(j)$ is the corresponding multi-variate feature vector of dimension m. Here, m denotes the number of features in the time series, while J represents the number of time indices per window. In the following, we set $J = 10$. In the "vanilla" INR approach (e.g., [8]), each function X_i is modeled using a separate instance of f_{θ_i}, trained with a Mean Squared Error loss. The learned weights θ_i thus encode a compressed representation of the instance X_i.

Conversely, in a meta-learning setting, multiple instance-specific functions f_{θ_i} are aggregated into a shared initialization f_θ, which can be modulated during inference to adapt to individual test samples. Following MAML [10], this is achieved through a nested optimization strategy: an outer loop learns a generic initialization f_θ from normal training data, while an inner loop performs fine-tuning on each test instance X_i through a few gradient descent steps, using normal data at training time and both normal and anomalous data at inference.

In COIN++, the MLP f_θ with L layers employs additive modulated SiREN layers [28] where the modulations are additive terms β in each layer l:

$$h^{l+1} = \sin(\omega_0(W^l h^l + b^l + \beta^l)), \tag{2}$$

where $\omega_0 = 30$ [28], W, b are the weights of the base network, and h^l represents the activations at layer l. In addition, COIN++ employs a hyper-network g_ϕ to further compress the representation of each instance. In `MetaINAD` we removed this hyper-network component to simplify the overall scheme and reduce the number of hyper-parameters, focusing in the AD task.

Multi-variate time series reconstruction and anomaly detection. Traditional unsupervised methods for time series anomaly detection rely on the formulation of anomaly scores to capture outliers by investigating properties of the data, with techniques such as samples proximity (distance-based for kNN [27], density-based for LOF [5], clustering-based for MCOD [20]) ensembling (e.g., tree-based for Isolation Forest [24]), or subspace-based modeling approaches (e.g., Loda [26]).

The emergence of deep learning has led to more sophisticated methods using *reconstruction error*: first, a model is trained to learn an encoder-decoder transformation by means of *only normal samples* – the model learns to *compress* the data and reconstructs its original form and then, at inference, the model is applied on a test set, from which an anomaly is identified whenever the reconstruction error passes a certain threshold. The compression of the data can be realized through different deep learning methods, including deep autoencoders [38, 3], one-class neural networks [37], variational approaches [21], prototype learning [22], and attention-based models [1, 23]. Since these approaches have demonstrated superior performance compared to traditional methods on F1-scores and variants developed for time series data in the field [13], we focus our comparison against them.

Motivation and potential issues in using INR on multi-variate time series. INRs offer several promising capabilities for multi-variate time series analysis, while also presenting unique challenges that warrant careful consideration. A particularly compelling advantage of INRs is that it can naturally accommodate different sampling rates and temporal resolutions through their continuous representation.

However, several uncertainties emerge when applying INRs to multi-variate time series. While INRs have shown remarkable success in image processing tasks where they typically handle 2-3 dimensional inputs (e.g., RGB channels), their performance on higher-dimensional time series data is less understood. Notably, the relationship between the number of features and model performance is critical, as time series applications often involve dozens of variables, significantly departing from the low-dimensional settings where INRs have traditionally excelled.

Additionally, practical considerations around model tuning require attention, especially regarding the number of needed modulation steps. Unlike compression, where it's generally acceptable to simply compress and observe the reconstruction (with larger compression typically being better), in anomaly detection improving the reconstruction after modulation may not yield good anomaly detection.

3. Methodology

In this section we describe how INRs are adapted to the tasks of time series reconstruction and anomaly detection. We first discuss the general INR methodology, then its modulation process, and finally how we apply it to our specific tasks. A high-level overview of the methodology is provided in Fig. 1.

3.1. INR Methodology for Time Series

INRs encode signals as a continuous function mapping coordinates to values. INRs learn a function:

$$f_\theta : \mathcal{T} \subset \mathbb{R}^n \to \mathcal{X} \subset \mathbb{R}^m, \tag{3}$$

where $\mathcal{T}$ represents the coordinate space (time indices), and $\mathcal{X}$ represents the corresponding signal space. In a multi-variate time series setting, we define each time-series window as:

$$X_i = \{(j, X_i(j))\}_{j=1}^{J}, \tag{4}$$

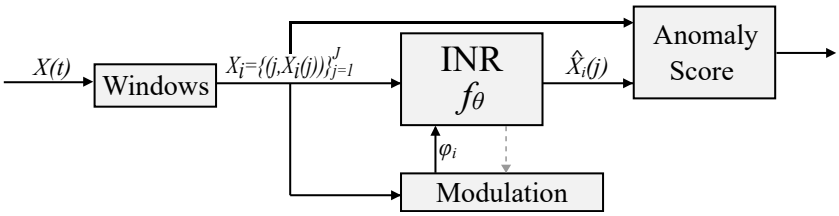

Figure 1: Overview of the proposed INR methodology with meta-learning MetaINAD. The multivariate time series $t \mapsto X(t)$ is segmented into disjoint windows of length J, forming instances $X_i = \{(j, X_i(j))\}_{j=1}^{J}$. At training stage, the base network f_θ undergoes modulation to adapt instance-specific parameters φ_i, following Equations 6 and 7. The outer loop updates θ using multiple instances for generalization. At inference stage, the learned f_θ is modulated using Equation 6, refining φ_i while keeping θ fixed. The modulated function reconstructs $\hat{X}_i(j) = f_\theta(j)$, and anomalies are detected by calculating the reconstruction error between $\hat{X}_i$ and X_i.

where j is the time index within a window, and $X_i(j) \in \mathbb{R}^m$ is the corresponding feature vector of dimension m (number of features in the time series). The objective of time-series anomaly detection is to predict an output sequence:

$$Y_i = \{y_j\}_{j=1}^{J}, \quad y_j \in \{0, 1\}, \tag{5}$$

where y_j indicates whether the time step j in the window X_i is anomalous ($y_j = 1$) or normal ($y_j = 0$).

3.2. Basic Methodology

Our approach builds on the COIN++ framework [9], which refines INR-based models through meta-learning.This is achieved through a two-step optimization process involving an inner-loop adaptation and an outer-loop meta-learning update.

Meta-learning for INR adaptation. During training, the INR model learns from multiple time-series windows. Each training instance corresponds to a time-series window X_i, which consists of J time steps. Instead of learning a separate model for each instance, the INR framework learns a shared base network f_θ, which is iteratively modulated for different instances. The adaptation for each instance is controlled by a modulation parameter φ_i, which is updated using:

$$\varphi_i \leftarrow \varphi_i - \alpha \nabla_\varphi L(\theta, \varphi, X_i), \tag{6}$$

$$\theta \leftarrow \theta - \beta \nabla_\theta L(\theta, \varphi, X_i), \tag{7}$$

where α and β are learning rates, L is the loss function to measure the reconstruction error, and X_i represents all the values for an individual time-series window. During training, both Equations (6) and (7) are applied: the inner-loop update (Equation (6)) is performed at each modulation step for individual instances, while the outer-loop update (Equation (7)) aggregates information across multiple instances to refine θ. At inference time, only Equation (6) is applied, ensuring that the learned initialization θ is not modified, and only the instance-specific modulations φ_i are updated based on the test data.

Modulation at training vs. inference. The number of modulation steps M during training determines how well the model adapts to different datasets. On the considered datasets, empirical analyses show that $M = 3$ provides good performance. At inference time, a separate modulation parameter N is used to control the number of fine-tuning steps applied to test instances. If $N = 0$, the base INR model f_θ is directly used for inference. If $N > 0$, additional gradient updates refine the modulation parameters φ_i, allowing the model to adapt to unseen patterns. In our case, we set $N = 3$, the inner-loop learning rate to $\alpha = 10^{-2}$ and the outer-loop learning rate β in the range of 10^{-6} to 3×10^{-6}, depending on the dataset. This fine-tuning process enables INRs to handle distribution shifts, adapt to new time-series structures, and improve both reconstruction and anomaly detection performance.

3.3. INRs Capabilities and Modulation

The inner-loop adaptation in INRs provides some advantages for time-series modeling. It captures temporal variations as INRs model time series as a continuous function, which makes them invariant to sampling rates which enables generalization across datasets with different time resolutions.

Unlike classical deep learning models, INRs do not rely on an external encoder-decoder structure. Instead, they directly represent the time series as a function f_θ, allowing better handling of missing values, fine-grained control over adaptation using modulation steps and strong generalization across different time-series structures. By leveraging meta-learning, INRs enable rapid adaptation to unseen data using only a few gradient steps. This provides a powerful framework for learning compact, generalizable representations for multi-variate time-series anomaly detection.

3.4. Application of INR

We consider two tasks. First, we apply INR for *anomaly detection*, where anomalies are identified based on reconstruction error. If INR fails to accurately reconstruct a time instance, it is flagged as an anomaly. Specifically, we compute the absolute reconstruction error:

$$E_i = \frac{1}{m} \sum_{f=1}^{m} \|X_i(f) - \hat{X}_i(f)\|^2 \tag{8}$$

where $X_i(f)$ represents the true value of feature f in time-series window i, and $\hat{X}_i(f)$ is the corresponding reconstructed output from INR. The Euclidean norm $\|.\|$ is used to measure the reconstruction error. A threshold is then set to classify anomalies based on this error. Then, we also consider a *imputation task* to investigate INRs natural capability to handle missing data without explicit imputation, as INRs trained f_θ outputs a continuous signal even if the input sample has missing values.

Datasets name	Samples train (test)	# datasets (# features)	% anomalies (# events)
SMD [30]	25k (25k)	28 (38)	4% (11 events)
SWAT [14]	500k (450k)	1 (51)	12% (35 events)
WADI [2]	1000k (200k)	1 (123)	6% (13 events)
Exathlon [18]	200 (40k)	10 (45)	1.3% (97 events)

Table 1: Datasets description. The last column indicates, for the test set, the percentage of anomalous samples and the number of events. For SMD and Exathlon, the numbers are per-dataset averages.

4. Experimental Design

4.1. Datasets Description

We use four datasets for the evaluation: SMD [30] (Server Machine Dataset), SWaT [14] (Secure Water Treatment), WADI [2] (Water Distribution), and Exathlon [18]. SMD consists of telemetry data of a large Internet company. The dataset includes multi-variate sensor readings, capturing system metrics from multiple servers, with labeled anomalies. SWaT and WADI are datasets obtained from industrial water treatment and distribution testbeds. In both datasets, the distribution changes drastically in the second half of the test set and the evaluation on both of them are highly unrealiable [34]. Exathlon is a benchmark dataset constructed from real data traces collected during repeated executions of large-scale stream on Apache Spark cluster, and the anomalies are injected in the test set. The main characteristics of these datasets are summarized in Table 1.

4.2. Performance Metrics

We evaluate the effectiveness of our model using F1-score variants as primary metrics for anomaly detection, which allows for comparison against state-of-the-art methods. Specifically, we measure performance using the classic F1-score alongside its Point Adjusted (PA) [35], Composite (COM) [13], and Affiliation (AFF) [16] versions. These variants provide a comprehensive evaluation by considering different scenarios encountered in time-series anomaly detection. As in previous works [3, 35], we select the best F1-score over all possible threshold values. Additionally, we compute Precision-Recall Area Under the Curve (PRAUC) as an alternative anomaly detection metric.

For the reconstruction task instead we quantify performance by measuring the Mean Squared Error (MSE) [19]. In cases where missing values were introduced (as described in Sec. 4.3), the MSE is computed *only on the missing samples*, without being biased by observed values. Overall, the MSE of a single reconstructed window i is computed as:

$$\text{MSE} = \frac{1}{J' \cdot m} \sum_{j=1}^{J'} \sum_{f=1}^{m} (X_i(f) - \hat{X}_i(f))^2, \tag{9}$$

where $\hat{X}_i(f)$ represents the reconstructed value of feature f for time-series window i, $X_i(f)$ is the original value, m is the total number of features, and J' is the number of time steps used in the MSE computation. Specifically, when evaluating MSE on the full window, J' is the total number of time steps in the window, whereas when evaluating MSE only on missing values, J' is restricted to the subset of missing time steps.

For all experiments we report average values and related confidence intervals across 5 runs. For SMD and Exathlon, confidence intervals are derived from multiple datasets within the collection. For SWaT and WADI, since each dataset consists of a single instance, confidence intervals are computed across multiple experimental runs.

4.3. Evaluation settings

To systematically evaluate the performance of our models reconstruction and anomaly detection under different missing data scenarios, we define several controlled use cases by introducing missing values to the test set, removing features, or altering temporal alignments in the original datasets. Below, we describe each use case in detail.

4.3.1. Baseline

The original dataset remains intact, with no additional missing values introduced. This represents the baseline condition where the full time series is available for both training and inference. Anomaly detection is performed based on the reconstruction error computed by our model, with a predefined threshold distinguishing normal and anomalous data points.

4.3.2. Missing Timestamps (Imputation Use Case)

This use case simulates missing timestamps in the test set by randomly removing a given percentage of time indices across all features. Specifically, we evaluate three scenarios where 20%, 50%, and 80% of timestamps are missing at random, similar to [12]. `MetaINAD` reconstructs these missing timestamps and subsequently detects anomalies based on reconstruction errors. This experiment allows us to assess our model in handling incomplete time series data.

4.3.3. Missing Features (Sensor Removal)

In this use case, entire features (or sensors) are removed from the dataset instead of individual timestamps. This simulates scenarios where sensor failures or network issues lead to missing sensor data. We consider two distinct approaches [36]:

- Random Sensor Removal: A given percentage (20%, 50%, or 80%) of sensors is randomly selected and completely removed from the dataset. This means that all timestamps corresponding to the selected sensors are missing.
- Fixed Sensor Removal Based on Information Gain: Instead of randomly selecting sensors, we rank the sensors based on their information gain using a decision tree model. The top 20%, 50%, or 80% of the most informative sensors (as determined by the decision tree) are then removed. This scenario allows us to assess the impact of structured feature removal performance of our model.

5. Results

5.1. Anomaly Detection

To assess the performance of our method, we evaluate it in the original scenario, where all data is present, to allow a

Metric	F1 Point-Adjust [35]							F1 Composite [13]				F1 Affiliation [16]	AUC score		
Dataset	Stack-VAE-G	USAD	PUAD	NN-One-class	DAICS	MTAD-GAN	Ours	DAGMM	OmniAnomaly	OCAN	Ours	Ours	USAD	TranAD	Ours
SMD	0.95	0.94	**0.96**	-	-	-	$0.95_{\pm 0.13}$	0.02	0.50	0.46	$\mathbf{0.71}_{\pm 0.21}$	$\mathbf{0.90}_{\pm 0.06}$	0.94	0.94	$0.93_{\pm 0.09}$
SWAT	-	0.84	-	0.87	0.88	**0.89**	$0.86_{\pm 0.12}$	0.00	0.15	0.15	$\mathbf{0.33}_{\pm 0.03}$	$\mathbf{0.73}_{\pm 0.02}$	0.84	0.84	$\mathbf{0.85}_{\pm 0.08}$
WADI	-	0.42	-	-	**0.80**	0.79	$0.79_{\pm 0.05}$	0.03	0.24	0.00	$\mathbf{0.31}_{\pm 0.02}$	$\mathbf{0.68}_{\pm 0.01}$	0.87	**0.89**	$0.88_{\pm 0.05}$
Exathlon	-	-	-	-	-	-	$\mathbf{0.78}_{\pm 0.12}$	-	-	-	$\mathbf{0.39}_{\pm 0.18}$	$\mathbf{0.72}_{\pm 0.05}$	-	-	$\mathbf{0.71}_{\pm 0.11}$

Table 2: *Anomaly detection*: Comparison against state-of-the-art performance as reported in StackVAE-G [21], USAD [3], PUAD [22], NN-One-class [18], DAICS [1], MTAD-GAN [23], DAGMM [38], OmniAnomaly [31], OCAN [37], TranAD [33]. The best results are highlighted in bold.

Dataset	Metric	Reconstruction (MSE)				Anomaly detection (F1 AFF)				AUC score			
	Missing	Baseline	20%	50%	80%	Baseline	20%	50%	80%	Baseline	20%	50%	80%
SMD	Missing Timestamps	-	0.20 ±0.006	0.20 ±0.005	0.21 ±0.005	0.90 ±0.06	0.88 ±0.06	0.86 ±0.07	0.84 ±0.09	0.93 ±0.09	0.91 ±0.07	0.89 ±0.07	0.87 ±0.08
	Missing Features (random)	-	0.20 ±0.005	0.20 ±0.004	0.21 ±0.004	0.90 ±0.06	0.88 ±0.05	0.86 ±0.05	0.84 ±0.08	0.93 ±0.09	0.91 ±0.06	0.90 ±0.05	0.88 ±0.07
	Missing Features (fixed)	-	0.20 ±0.004	0.21 ±0.007	0.23 ±0.006	0.90 ±0.06	0.87 ±0.05	0.85 ±0.05	0.83 ±0.07	0.93 ±0.09	0.91 ±0.07	0.90 ±0.07	0.87 ±0.07
SWAT	Missing Timestamps	-	0.14 ±0.007	0.14 ±0.006	0.15 ±0.007	0.73 ±0.02	0.72 ±0.02	0.70 ±0.02	0.68 ±0.03	0.85 ±0.08	0.84 ±0.06	0.82 ±0.06	0.80 ±0.05
	Missing Features (random)	-	0.13 ±0.009	0.14 ±0.008	0.14 ±0.006	0.73 ±0.02	0.75 ±0.02	0.73 ±0.03	0.72 ±0.03	0.85 ±0.08	0.80 ±0.07	0.78 ±0.08	0.07 ±0.06
	Missing Features (fixed)	-	0.14 ±0.007	0.14 ±0.005	0.16 ±0.007	0.73 ±0.02	0.74 ±0.02	0.73 ±0.02	0.72 ±0.03	0.85 ±0.08	0.79 ±0.06	0.78 ±0.06	0.76 ±0.05
WADI	Missing Timestamps	-	0.22 ±0.003	0.22 ±0.002	0.23 ±0.002	0.68 ±0.01	0.68 ±0.02	0.66 ±0.03	0.61 ±0.02	0.88 ±0.05	0.87 ±0.05	0.85 ±0.04	0.82 ±0.04
	Missing Features (random)	-	0.22 ±0.003	0.22 ±0.003	0.23 ±0.003	0.68 ±0.01	0.68 ±0.03	0.65 ±0.04	0.60 ±0.02	0.88 ±0.05	0.87 ±0.04	0.84 ±0.05	0.81 ±0.04
	Missing Features (fixed)	-	0.22 ±0.004	0.23 ±0.003	0.24 ±0.004	0.68 ±0.01	0.69 ±0.03	0.67 ±0.05	0.61 ±0.04	0.88 ±0.05	0.87 ±0.04	0.85 ±0.06	0.82 ±0.05
Exathlon	Missing Timestamps	-	0.21 ±0.008	0.21 ±0.006	0.22 ±0.006	0.72 ±0.05	0.69 ±0.03	0.67 ±0.04	0.65 ±0.05	0.71 ±0.11	0.71 ±0.09	0.70 ±0.10	0.68 ±0.09
	Missing Features (random)	-	0.21 ±0.006	0.21 ±0.005	0.23 ±0.006	0.72 ±0.05	0.70 ±0.03	0.68 ±0.05	0.66 ±0.05	0.71 ±0.11	0.72 ±0.09	0.70 ±0.08	0.69 ±0.11
	Missing Features (fixed)	-	0.21 ±0.005	0.21 ±0.007	0.24 ±0.007	0.72 ±0.05	0.70 ±0.03	0.68 ±0.04	0.66 ±0.05	0.71 ±0.11	0.72 ±0.10	0.70 ±0.07	0.66 ±0.08

Table 3: INR Reconstruction, Anomaly Detection, and AUC Results

direct comparison with prior literature. We measure performance using different F1-scores and AUC (where available), following standard benchmarks in time-series anomaly detection.

Table 2 summarizes our results. Overall, our method achieves comparable performance to state-of-the-art approaches, with competitive F1 and AUC scores across datasets and in some cases even slightly better. Specifically our approach performs on par with existing methods [22] [23] in terms of F1-score and AUC, confirming its effectiveness in detecting anomalies. While our method slightly under-perfoms considering F1 PA (-0.01 to -0.05 range) compared to some prior work, it demonstrates better generalization in F1 COM (+0.11 to +0.21 range) compared to [31] [37] [38], making it more robust in different anomaly detection settings. Furthermore, our approach achieves high AUC scores across all datasets, often outperforming previous methods [3, 33]. This further supports the reliability of our method in distinguishing anomalies effectively and consistently across different datasets.

These results indicate that our INR-based anomaly detection framework is a competitive alternative to existing methods while providing additional flexibility in handling missing data, as explored in the next section.

5.2. Data Imputation

To handle missing values, traditional approaches typically rely on explicit imputation methods, such as replacing missing values with the mean, median, or interpolation. However, these can introduce bias and/or fail to capture the underlying structure of the data. To investigate our model capabilities, we conducted experiments introducing missing values according to the three scenarios discussed in Sec. 4.3).

Table 3 summarizes the results. In missing timestamps, the results show that even when 80% of the timestamps are missing, `MetaINAD`'s reconstruction still maintains a high AUC score, with a maximum drop of 0.04 across datasets compared to cases with full data availability. Notably, SMD and SWAT datasets maintain strong anomaly detection performance. The case of missing features (random) presents a more challenging scenario since missing values are not uniformly distributed across time. Here, AUC scores see a maximum drop of 0.05 in extreme cases (80% missing), but for SMD and SWAT, the performance remains robust, with only minor variations. These results suggest that our model can still infer missing feature dependencies. When considering missing features (fixed), it proves a little more challenging as the model shows a slightly higher drop, with a maximum AUC drop of 0.06 at 80% missing values. However, for SMD and WADI, the results remain more stable, reinforcing our model's ability to generalize well even under high missingness.

Across all scenarios, INR-based imputation maintains competitive performance, even in extreme cases where 80% of data is missing. Comparing the results of fully available data to missing-value cases, we observe that our model still achieves comparable anomaly detection performance. These findings confirm `MetaINAD` 's ability to reconstruct missing values from its learned representation maintains both data quality and anomaly detection performance.

5.3. Critical View / Ablation Study

To better understand the behavior of our model in multivariate time series, we conduct an ablation study analyzing different aspects of the model's reconstruction and anomaly detection capabilities.

5.3.1. Constant MSE during Reconstruction

A key observation in our experiments is that the reconstruction performance, measured by MSE, remains nearly constant across different scenarios. This can be attributed to the structure of time-series datasets, where many features exhibit low variance over long time periods.

In real-world multi-variate time-series applications such as SMD, SWAT, WADI, and Exathlon, most features capture rela-

tively stable system states. Therefore, even when a significant portion of the data is missing, our model reconstructs these missing values with minimal deviation from its normal behavior. Additionally, the relatively low proportion of anomalies in these datasets means that their effect on overall MSE is limited. To further investigate this phenomenon, we compare INR-based reconstruction against mean-based imputation in Table 4. The results show that replacing missing values with the mean yields nearly identical MSE results to INR reconstruction. This suggests that in datasets where many features exhibit low variance, mean imputation and INR-based reconstruction perform similarly.

Dataset	Use case	Reconstruction (MSE): missing / mean		
	Missing %	20%	50%	80%
SMD	Missing Timestamps	0.20 / 0.21	0.20 / 0.21	0.21 / 0.22
	Missing Features (random)	0.20 / 0.20	0.20 / 0.21	0.21 / 0.22
	Missing Features (fixed)	0.20 / 0.21	0.21 / 0.23	0.23 / 0.25

Table 4: INR reconstruction with missing values left unattended vs. Mean Imputation before INR.

5.3.2. *Effect of Dimensionality on Reconstruction Performance Performance*

INRs have been widely used in low-dimensional tasks such as image super-resolution (e.g., RGB images with only three channels), where they have shown exceptional ability to reconstruct fine-grained details. However, moving to multi-variate time-series data introduces significant complexity due to the much larger number of features.

To analyze this effect, we conducted an experiment on the SMD dataset, comparing INR's reconstruction ability when trained on all 38 features versus only 1, 2, or 3 features. The most informative features were chosen based on the information gain using a decision tree, as discussed in Sec. 4.3. In our test, we found that INR achieves near-perfect reconstruction when using only 1–3 features (MSE = 0.001–0.002). However, as the number of features increases, MSE rises significantly (MSE = 0.2 for 38 features), indicating that INR struggles with high-dimensional data (Table 5). The degradation in high-dimensional settings occurs because INR must learn complex dependencies across many features, making it harder to capture smooth representations. These results highlight a fundamental limitation of INR-based reconstruction; while highly effective in low-dimensional settings, performance deteriorates as the feature dimensionality increases.

Dataset		Reconstruction (MSE)			
	Number of features	1	2	3	38
SMD	w/o missing values	0.001	0.001	0.002	0.2

Table 5: Reconstruction performance based on the number of features.

5.3.3. *Effect of Inner Step Modulation on Anomaly Detection Performance*

We analyze the sensitivity of anomaly detection performance with respect to the number of inner steps used during modulation. The results demonstrate that while N=0 already provides a solid starting point, different datasets exhibit varying behaviors as the modulation steps increases. For the SMD dataset, performance improves significantly from 0.85 (N=0) to 0.93 (N=3), after which it stabilizes at 0.94 for larger values of N, suggesting that a small number of inner steps is sufficient to reach optimal performance. For SWAT, performance slightly improves from 0.82 (N=0) to 0.85 (N=3) and remains stable at 0.85 (N=10), but further increasing the modulation to N=20 leads to a sharp drop to 0.69. Similarly, for WADI, we observe an initial improvement going from N=0 to N=3, followed by a drop in performance.

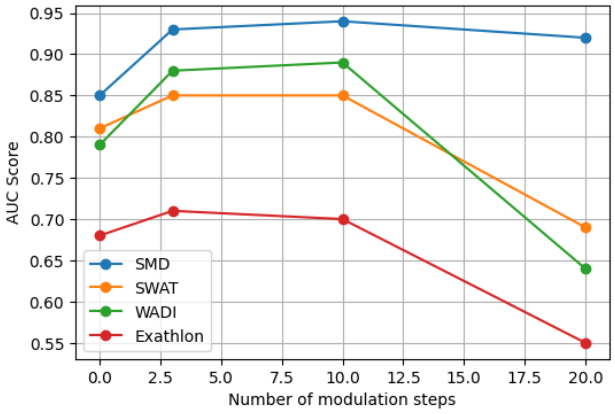

Figure 2: Effect of number of modulation steps on anomaly detection performance across different datasets.

The reason for this decline at high N values is likely due to overfitting: INR over-adapts to normal patterns, making it harder to distinguish anomalies, and excessive modulation distorts the learned representation, reducing generalization.

6. Limitations and conclusion

In this paper, we investigated the application of Implicit Neural Representations (INRs) trained via a meta-learning process for multi-variate time series reconstruction & anomaly detection. Our results demonstrate that `MetaINAD` achieves competitive performance, often matching state-of-the-art methods in anomaly detection while inherently handling missing values. The meta-learned models, even when trained only on normal data, generalize effectively to unseen instances, requiring minimal adaptation at inference time. Despite these promising results, certain limitations must be acknowledged. The scalability of INRs remains a challenge, as reconstruction performance deteriorates when applied to datasets with large feature spaces. Additionally, while the inner-loop modulation improves adaptability, excessive meta-learning steps can lead to overfitting; furthermore, modulation can increase inference-time computational requirements, making real-time deployment of `MetaINAD` demanding. Finally, our analysis primarily focused on system monitoring datasets, leaving open questions about the generalization of our approach to other domains, such as healthcare or finance, where time series exhibit different characteristics.

Avenues for future work include exploring architectural optimizations to enhance INR scalability for high-dimensional time series, as well as benchmarking INR-based imputation against more advanced missing-data handling techniques. Expanding the evaluation to broader real-world scenarios and investigating hybrid approaches that combine INRs with sequence models could further refine their effectiveness. Nonetheless, our findings highlight the potential of INR-based representations for anomaly detection, offering a robust alternative to conventional methods while leveraging their unique ability to model continuous signals.

Aknowledgments

We thank Alan Collet and Antonio Mastropietro for their inputs on this research.

References

[1] M. Abdelaty, R. Doriguzzi-Corin, and D. Siracusa. DAICS: A deep learning solution for anomaly detection in industrial control systems. *IEEE Trans. Emerg. Top. Comput.*, 10(2):1117–1129, 2021.

[2] C.M. Ahmed, V.R. Palleti, and A.P. Mathur. Wadi: A water distribution testbed for research in the design of secure cyber physical systems. In *Proc. 3rd Int. Workshop Cyber-Physical Syst. Smart Water Networks*, pages 25–28, 2017.

[3] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M.A. Zuluaga. USAD: Unsupervised anomaly detection on multivariate time series. In *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pages 3395–3404, 2020.

[4] T. Bamford, E. Fons, Y. El-Laham, and S. Vyetrenko. Mads: Modulated auto-decoding siren for time series imputation. *arXiv preprint arXiv:2307.00868*, 2023.

[5] M.M. Breunig, H.-P. Kriegel, R.T. Ng, and J. Sander. Lof: Identifying density-based local outliers. In *Proc. 2000 ACM SIGMOD Int. Conf. Manag. Data*, pages 93–104, 2000.

[6] H. Chen, B. He, H. Wang, Y. Ren, S.N. Lim, and A. Shrivastava. Nerv: Neural representations for videos. *Adv. Neural Inf. Process. Syst.*, 34:21557–21568, 2021.

[7] J. Chen, S. Sathe, C.C. Aggarwal, and D.S. Turaga. Outlier detection with autoencoder ensembles. In *Proc. 2017 SIAM Int. Conf. Data Min.*, pages 90–98, 2017.

[8] E. Dupont, A. Golinski, M. Alizadeh, Y.W. Teh, and A. Doucet. COIN: Compression with implicit neural representations. In *Neural Compression Workshop @ ICLR*, 2021.

[9] E. Dupont, H. Loya, M. Alizadeh, A. Golinski, Y.W. Teh, and A. Doucet. COIN++: Neural compression across modalities. *arXiv preprint arXiv:2201.12904*, 2022.

[10] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep. In *Int. Conf. Mach. Learn.*, 2017.

[11] E. Fons, A. Sztrajman, Y. El-Laham, A. Iosifidis, and S. Vyetrenko. Hypertime: Implicit neural representation for time series. *arXiv preprint arXiv:2208.05836*, 2022.

[12] E. Fons, A. Sztrajman, Y. El-Laham, A. Iosifidis, and S. Vyetrenko. Hypertime: Implicit neural representations for time series generation. *arXiv preprint arXiv:2208.05836*, 2023.

[13] A. Garg, W. Zhang, J. Samaran, R. Savitha, and C.-S. Foo. An evaluation of anomaly detection and diagnosis in multivariate time series. *IEEE Trans. Neural Netw. Learn. Syst.*, 33(6):2508–2517, 2021.

[14] J. Goh, S. Adepu, K.N. Junejo, and A. Mathur. A dataset to support research in the design of secure water treatment systems. In *Crit. Inf. Infrastructures Security: 11th Int. Conf. CRITIS*, pages 88–99, 2017.

[15] D. Grattarola and P. Vandergheynst. Generalised implicit neural representations. *Adv. Neural Inf. Process. Syst.*, 35:30446–30458, 2022.

[16] A. Huet, J.M. Navarro, and D. Rossi. Local evaluation of time series anomaly detection algorithms. In *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Min.*, pages 635–645, 2022.

[17] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pages 387–395, 2018.

[18] V. Jacob, F. Song, A. Stiegler, B. Rad, Y. Diao, and N. Tatbul. Exathlon: A benchmark for explainable anomaly detection over time series. *arXiv preprint arXiv:2010.05073*, 2020.

[19] K.-J. Jeong and Y.-M. Shin. Time-series anomaly detection with implicit neural representation. *arXiv preprint arXiv:2201.11950*, 2022.

[20] M. Kontaki, A. Gounaris, A.N. Papadopoulos, K. Tsichlas, and Y. Manolopoulos. Continuous monitoring of distance-based outliers over data streams. In *IEEE 27th Int. Conf. Data Eng.*, pages 135–146, 2011.

[21] W. Li, W. Hu, T. Chen, N. Chen, and C. Feng. Stackvae-g: An efficient and interpretable model for time series anomaly detection. *AI Open*, 2022.

[22] Y. Li, W. Chen, B. Chen, D. Wang, L. Tian, and M. Zhou. Prototype-oriented unsupervised anomaly detection for multivariate time series. In *Proc. 40th Int. Conf. Mach. Learn.*, volume 202 of *Proc. Mach. Learn. Res.*, pages 19407–19424, 2023.

[23] Y. Lian, Y. Geng, and T. Tian. Anomaly detection method for multivariate time series data of oil and gas stations based on digital twin and mtad-gan. *Appl. Sci.*, 13(3):1891, 2023.

[24] F.T. Liu, K.M. Ting, and Z.-H. Zhou. Isolation forest. In *Proc. 2008 IEEE 8th Int. Conf. Data Min.*, pages 413–422, 2008.

[25] E. Le Naour, L. Serrano, L. Migus, Y. Yin, G. Agoua, N. Baskiotis, and V. Guigue. Time series continuous modeling for imputation and forecasting with implicit neural representations. *arXiv preprint arXiv:2306.05880*, 2023.

[26] T. Pevnỳ. Loda: Lightweight on-line detector of anomalies. *Mach. Learn.*, 102:275–304, 2016.

[27] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proc. 2000 ACM SIGMOD Int. Conf. Manag. Data*, pages 427–438, 2000.

[28] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *Adv. Neural Inf. Process. Syst.*, 33:7462–7473, 2020.

[29] V. Sitzmann, M. Zollhöfer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Adv. Neural Inf. Process. Syst.*, 32, 2019.

[30] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pages 2828–2837, 2019.

[31] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pages 2828–2837, 2019.

[32] F. Szatkowski, K.J. Piczak, P. Spurek, J. Tabor, and T. Trzciński. Hypersound: Generating implicit neural representations of audio signals with hypernetworks. *arXiv preprint arXiv:2211.01839*, 2022.

[33] S. Tuli, G. Casale, and N.R. Jennings. Tranad: Deep transformer networks for anomaly detection in multivariate time series data. *Proc. VLDB Endow.*, 15(11):3118–3130, 2022.

[34] D. Wagner, T. Michels, F.C.F. Schulz, M. Rudolph, and M. Kloft. Timesead: Benchmarking deep time-series anomaly detection. *arXiv preprint arXiv:2209.12345*, 2022.

[35] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, and Y. Feng. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proc. WWW*, pages 187–196, 2018.

[36] X. Zhang, M. Zeman, T. Tsiligkaridis, and M. Zitnik. Graph-guided network for irregularly sampled multivariate time series. *arXiv preprint arXiv:2110.05357*, 2021.

[37] P. Zheng, S. Yuan, X. Wu, J. Li, and A. Lu. One-class adversarial nets for fraud detection. In *Proc. AAAI Conf. Artif. Intell.*, volume 33, pages 1286–1293, 2019.

[38] B. Zong, Q. Song, M.R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *Int. Conf. Learn. Represent.*, 2018.