Time-Series Anomaly Detection with Implicit Neural Representation

Kyeong-Joong Jeong [* 1]   **Yong-Min Shin** [* 1]

Abstract

Detecting anomalies in multivariate time-series data is essential in many real-world applications. Recently, various deep learning-based approaches have shown considerable improvements in time-series anomaly detection. However, existing methods still have several limitations, such as long training time due to their complex model designs or costly tuning procedures to find optimal hyperparameters (e.g., sliding window length) for a given dataset. In our paper, we propose a novel method called Implicit Neural Representation-based Anomaly Detection (INRAD). Specifically, we train a simple multi-layer perceptron that takes time as input and outputs corresponding values at that time. Then we utilize the representation error as an anomaly score for detecting anomalies. Experiments on five real-world datasets demonstrate that our proposed method outperforms other state-of-the-art methods in performance, training speed, and robustness.

1. Introduction

Time-series data is frequently used in various real-world systems, especially in multivariate scenarios such as server machines, water treatment plants, spacecraft, etc. Detecting an anomalous event in such time-series data is crucial to managing those systems (Su et al., 2019; Mathur & Tippenhauer, 2016; Hundman et al., 2018; Gupta et al., 2014; Blázquez-García et al., 2021). To solve this problem, several classical approaches have been developed in the past (Fox, 1972; Zhang et al., 2005; Ma & Perkins, 2003; Liu et al., 2008a). However, due to the limited capacity of their approaches, they could not fully capture complex, non-linear, and high-dimensional patterns in the time-series data.

---
*Equal contribution   [1]Computational Science and Engineering, Yonsei University, Seoul, South Korea. Kyeong-Joong Jeong <jeongkj@yonsei.ac.kr>, Yong-Min Shin <jordan3414@yonsei.ac.kr>.
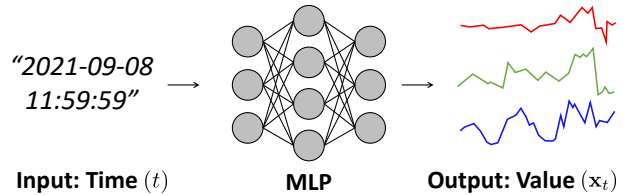
Figure 1: Implicit neural representation for multivariate time-series data.

Recently, various unsupervised approaches employing deep learning architectures have been proposed. Such works include adopting architectures such as recurrent neural networks (RNN) (Hundman et al., 2018), variational autoencoders (VAE) (Xu et al., 2018), generative adversarial networks (GAN) (Li et al., 2019), graph neural networks (GNN) (Deng & Hooi, 2021), and combined architectures (Zong et al., 2018; Su et al., 2019; Shen et al., 2020; Audibert et al., 2020; Park et al., 2018). These deep learning approaches have brought significant performance improvements in time-series anomaly detection. However, most deep learning-based methods have shown several downsides. First, they require a long training time due to complex calculations, hindering applications where fast and efficient training is needed. Second, they need a significant amount of effort to tune model hyperparameters (e.g., sliding window size) for a given dataset, which can be costly in real-world applications.

In our paper, we propose Implicit Neural Representation-based Anomaly Detection (INRAD), a novel approach that performs anomaly detection in multivariate time-series data by adopting implicit neural representation (INR). Figure 1 illustrates the approach of INR in the context of multivariate time-series data. Unlike conventional approaches where the values are passed as input to the model (usually processed via sliding window etc.), we directly input time to a multi-layer perceptron (MLP) model. Then the model tries to represent the values of that time, which is done by minimizing a mean-squared loss between the model output and the ground truth values. In other words, we train a MLP model to represent the time-series data itself. Based on our

observation that the INR represents abnormal data relatively poorly compared to normal data, we use the representation error as the anomaly score for anomaly detection. Adopting such a simple architecture design using MLP naturally results in a fast training time. Additionally, we propose a temporal encoding technique that improves efficiency for the model to represent time-series data, resulting in faster convergence time.

In summary, the main contributions of our work are:

- We propose INRAD, a novel time-series anomaly detection method that only uses a simple MLP which maps time into its corresponding value.

- We introduce a temporal encoding technique to represent time-series data efficiently.

- We conduct extensive experiments while using the same set of hyperparameters over all five real-world benchmark datasets. Our experimental results show that our proposed method outperforms previous state-of-the-art methods in terms of not only accuracy, but also training speed in a highly robust manner.

2. Related Work

In this section, we review previous works for time-series anomaly detection and implicit neural representation.

2.1. Time-Series Anomaly Detection

Since the first study on this topic was conducted by (Fox, 1972), time-series anomaly detection has been a topic of interest over the past decades (Gupta et al., 2014; Blázquez-García et al., 2021). Traditionally, various methods, including autoregressive moving average (ARMA) (Galeano et al., 2006) and autoregressive integrated moving average (ARIMA) model (Zhang et al., 2005)-based approaches, one-class support vector machine-based method (Ma & Perkins, 2003) and isolation-based method (Liu et al., 2008a) have been widely introduced for time-series anomaly detection. However, these classical methods either fail to capture complex and non-linear temporal characteristics or are very sensitive to noise, making them infeasible to be applied on real-world datasets.

Recently, various unsupervised deep learning-based approaches have successfully improved performance in complex multivariate time-series anomaly detection tasks. As one of the well-known unsupervised models, autoencoder (AE)-based approaches (Sakurada & Yairi, 2014) capture the non-linearity between variables. Recurrent neural networks (RNNs) are a popular architecture choice used in various methods (Hundman et al., 2018; Malhotra et al., 2016) for capturing temporal dynamics of time series data. Generative

models are also used in the literature, namely generative adversarial networks (Li et al., 2019) and variational autoencoder (VAE)-based approaches (Xu et al., 2018). Graph neural network-based approach (Deng & Hooi, 2021) is also proposed to capture the complex relationship between variables in the multivariate setting. Furthermore, methodologies combining multiple architectures are also proposed, such as AE with the Gaussian mixture model (Zong et al., 2018) or AE with GANs (Audibert et al., 2020), stochastic RNN with a planar normalizing flow (Su et al., 2019), deep support vector data description (Ruff et al., 2018) with dilated RNN (Chang et al., 2017), and VAE with long short term memory (LSTM) networks (Park et al., 2018).

Despite remarkable improvements via those above deep learning-based approaches, most of the approaches produce good results at the expense of training speed and generalizability. Such long training time with costly hyperparameter tuning for each dataset results in difficulties applying to practical scenarios (Audibert et al., 2020).

2.2. Implicit Neural Representation

Recently, implicit neural representations (or coordinate-based representations) have gained popularity, mainly in 3D deep learning. Generally, it trains a MLP to represent a single data instance by mapping the coordinate (e.g., xyz-coordinates) to the corresponding values of the data. This approach has been proven to have expressive representation capability with memory efficiency. As one of the well-known approaches, occupancy networks (Mescheder et al., 2019) train a binary classifier to predict whether a point is inside or outside the data to represent. DeepSDF (Park et al., 2019) directly regresses a signed distance function that returns a signed distance to the closest surface when the position of a 3D point is given. Instead of occupancy networks or signed distance functions, NeRF (Mildenhall et al., 2020) proposes to map an MLP to the color and density of the scene to represent. SIREN (Sitzmann et al., 2020) proposes using sinusoidal activation functions in MLPs to facilitate high-resolution representations. Since then, various applications, including view synthesis (Martin-Brualla et al., 2021) and object appearance (Saito et al., 2019) have been widely studied.

However, the application of INR to time-series data has been relatively underdeveloped. Representation of time-varying 3D geometry has been explored (Niemeyer et al., 2019), but they do not investigate multivariate time-series data. Although SIREN (Sitzmann et al., 2020) showed the capability to represent audio, its focus was limited to the high-quality representation of the input signals. To the best of our knowledge, this is the first work to use INR to solve the problem of time-series anomaly detection.

3. INRAD Framework

In this section, we define the problem that we aim to solve, and then we present our proposed INRAD based on the architecture proposed by (Sitzmann et al., 2020). Next, we describe our newly designed temporal encoding technique in detail. Finally, we describe the loss function to make our model represent input time signals and describe the anomaly score used during the detection procedure. Figure 3 describes the overview of the proposed method.

3.1. Problem Statement

In this section, we formally state the problem of multivariate time-series anomaly detection as follows.

We first denote multivariate time-series data as $X = \{(t_1, \mathbf{x}_{t_1}), (t_2, \mathbf{x}_{t_2}), (t_3, \mathbf{x}_{t_3}), ..., (t_N, \mathbf{x}_{t_N})\}$, where t_i denotes a timestamp, $\mathbf{x}_{t_i}$ denotes corresponding values at the timestamp, and N denotes the number of observed values. As we focus on multivariate data, $\mathbf{x}_{t_i}$ is a d-dimensional vector representing multiple signals. The goal of time-series anomaly detection is to output a sequence, $Y = \{y_{t_1}, y_{t_2}, y_{t_3}, ..., y_{t_N}\}$, where $y_{t_i} \in \{0, 1\}$ denotes the abnormal or normal status at t_i. In general, 1 indicates the abnormal state while 0 indicates normal state.

3.2. Implicit Neural Representation of Time-Series Data

To represent a given time-series data, we adopt the architecture proposed by (Sitzmann et al., 2020), which leverages periodic functions as activation functions in the MLP model, resulting in a simple yet powerful model capable of representing various signals, including audio. After preprocessing the time coordinate input via an encoding function ϕ, our aim is to learn a function f that maps the encoded time $\phi(t_i)$ to its corresponding value $\mathbf{x}_{t_i}$ of the data.

We can describe the MLP f by first describing each fully-connected layer and stacking those layers to get the final architecture. Formally, the lth fully-connected layer f_l with hidden dimension m_l can be generally described as $f_l(\mathbf{h}_{l-1}) = \sigma(\mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l)$, where $\mathbf{h}_{l-1} \in \mathbb{R}^{m_{l-1}}$ represents the output of the previous layer f_{l-1}, $\mathbf{W}_l \in \mathbb{R}^{m_l \times m_{l-1}}$ and $\mathbf{b}_l \in \mathbb{R}^{m_l}$ are learnable weights and biases, respectively, and σ is a non-linear activation function. Here, sine functions are used as σ, which enables accurate representation capabilities of various signals. In practice, a scalar ω_0 is multiplied such that the lth layer is $f_l = \sin(\omega_0 \cdot \mathbf{W}_l \mathbf{h}_{l-1} + \mathbf{b}_l)$, in order for the input to span multiple periods of the sine function.

Finally, by stacking a total of L layers with an additional linear transformation at the end, we now have our model $f(\phi(t_i)) = \mathbf{W}(f_L \circ f_{L-1} \circ \cdots \circ f_1)(\phi(t_i)) + \mathbf{b}$ which maps the input t_i to the output $f(\phi(t_i)) \in \mathbb{R}^d$.

3.3. Temporal Encoding

As INR has been mainly developed to represent 2D or 3D graphical data, encoding time coordinate for INR has rarely been studied. Compared to graphical data, which the number of points in each dimension is fairly limited to around thousands, the number of timestamps is generally much larger and varies among different datasets. Also, training and test data need to be considered regarding their chronological order (training data usually comes first). These observations with a real-world time-series data motivate us to design a new encoding strategy such that 1) the difference between $\phi(t_i)$ and $\phi(t_{i+1})$ is not affected by the length of the time sequence 2) chronological order between train and test data is preserved after encoding 3) timestamps from real-world data are naturally represented using its standard time scale rather than relying on the sequential index of time-series data. These desired properties are not satisfied with the encoding strategy applied in (Sitzmann et al., 2020) (which we call vanilla encoding), where it normalizes coordinates in the range $[-1, 1]$.

We now describe our temporal encoding, a simple yet effective method which satisfies conditions mentioned above. The key idea is to directly utilize the timestamp data present in the time-series data (we can assign arbitrary timestamps if none is given). We first represent t_i into a 6-dimensional vector $\mathbf{k} = [k_{yr}, k_{mon}, k_{day}, k_{hr}, k_{min}, k_{sec}] \in \mathbb{R}^6$, each dimension representing year, month, day, hour, minute, and second respectively. Here, $k_{yr}, k_{mon}, k_{day}, k_{hr}, k_{min}$ are all positive integers, while $k_{sec} \in [0, 60)$. Note that this can flexibly change depending on the dataset. For instance, if the timestamp does not include minute and second information, we use a 4-dimensional vector (i.e., $[k_{yr}, k_{mon}, k_{day}, k_{hr}]$ and $k_{hr} \in [0, 24)$).

Next, we normalize the vectorized time information. With a pre-defined year k'_{yr}, we first set $[k'_{yr}, 1, 1, 0, 0, 0]$ (January 1st 00:00:00 at year k'_{yr}) as [-1,-1,-1,-1,-1,-1]. Now, let us represent the current timestamp of interest as $\mathbf{k}^{curr}$. We normalize the j-th dimension of the current timestamp $\mathbf{k}^{curr}$ by the following linear equation:

$$n_j^{curr} = -1 + \frac{1 - (-1)}{N_j - 1} \times (k_j^{curr} - \mathbb{I}(j = 1)k'_{yr}) \quad (1)$$

where n_i^{curr} is the jth dimension of the normalized vector $\mathbf{n}^{curr} \in \mathbb{R}^6$ and $\mathbb{I}$ is an indicator function. For the values of N_i, we set $N_2 = 12, N_3 = 31, N_4 = 24, N_5 = 60, N_6 = 60$ to match the standard clock system. We assume that N_1 is pre-defined by the user. In short, we define a temporal encoding function ϕ that transforms a scalar t into $\mathbf{n}_i$ ($\phi : t_i \mapsto \mathbf{n}_i$). In our method, we will by default use this temporal encoding unless otherwise stated.
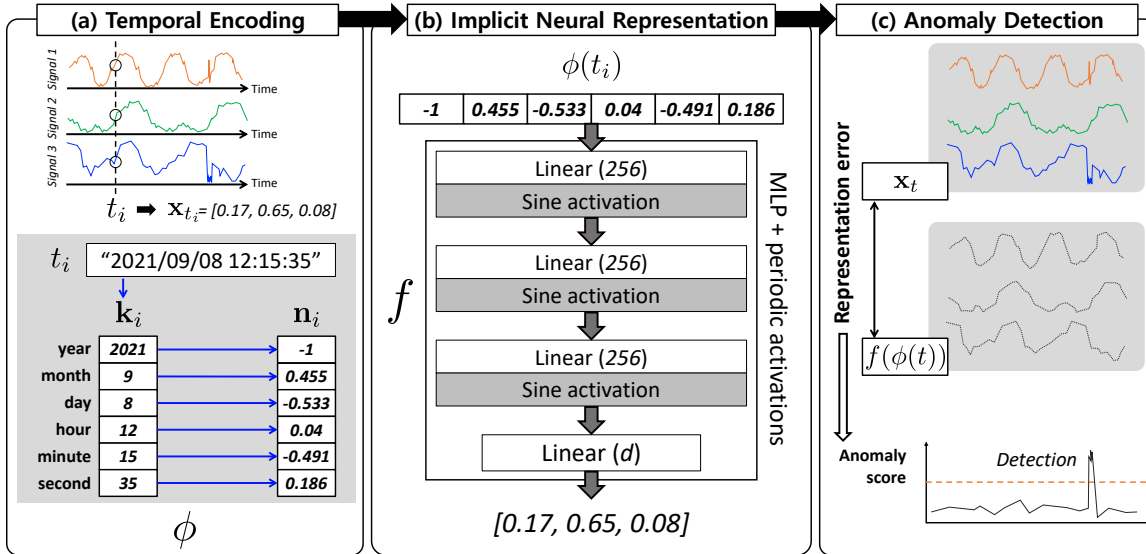
Figure 2: The overview of the proposed Implicit Neural Representation-based Anomaly Detection (INRAD). (a) From the given time-series data, we perform temporal encoding and represent time as a real-valued vector. (b) An MLP using periodic activation functions represents the given data by mapping the time processed by temporal encoding to the corresponding values. (c) After the model converges, we calculate the representation error and use this as the anomaly score for detection.

3.4. Loss Function

As we aim the model to represent the input time-series data, we compare the predicted value at each timestamp t_i to its ground-truth value $\mathbf{x}_{t_i}$. Therefore, we minimize the following loss function:

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} ||\mathbf{x}_{t_i} - f(\phi(t_i))||^2 \tag{2}$$

where $|| \cdot ||$ indicates the l2 norm of a vector.

3.5. Anomaly Score and Detection Procedure

Our proposed representation error-based anomaly detection strategy is built on the observation that values at an anomalous time are difficult to represent, resulting in relatively high representation error. By our approach described above, the given data sequence X is represented by an MLP function f. We now perform anomaly detection with this functional representation by defining the representation error as the anomaly score. Formally, the anomaly score a_{t_i} at a specific timestamp t_i is defined as $a_{t_i} = |\mathbf{x}_{t_i} - f(\phi(t_i))|$, where $| \cdot |$ indicates the l1 norm of a vector. Anomalies can be detected by comparing the anomaly score a_{t_i} with the pre-defined threshold τ.

In our approach, we first use the training data to pre-train our model f and then re-train the model to represent the given test data to obtain the representation error as an anomaly score for the detection.

Datasets	Train	Test	Features	Anomalies
SMD	708405	708420	28×38	4.16 (%)
SMAP	135183	427617	55×25	13.13 (%)
MSL	58317	73729	27×55	10.72 (%)
SWaT	496800	449919	51	11.98 (%)
WADI	1048571	172801	123	5.99 (%)

Table 1: Statistics of the datasets used in our experiments.

4. Experiments

In this section, we perform various experiments to answer the following research questions:

- **RQ1:** Does our method outperform various state-of-the-art methods, even with a fixed hyperparameter setting?

- **RQ2:** How does our proposed temporal encoding affect the performance and convergence time?

- **RQ3:** Does our method outperform various state-of-the-art methods in terms of training speed?

- **RQ4:** How does our method behave in different hyperparameter settings?

4.1. Dataset

We use five real-world benchmark datasets, SMD (Su et al., 2019), SMAP & MSL (Hundman et al., 2018), SWaT & WADI (Mathur & Tippenhauer, 2016), for anomaly detection for multivariate time-series data, which contain ground-truth anomalies as labels. Table 1 summarizes the statistics of each dataset, which we further describe its detail in the supplementary material.

In our experiments, we directly use timestamps included in the dataset for SWAT and WADI. We arbitrarily assign timestamps for the other three datasets since no timestamps representing actual-time information are given.

4.2. Baseline methods

We demonstrate the performance of our proposed method, INRAD, by comparing with the following six anomaly detection methods:

- **IF** (Liu et al., 2008a): Isolation forests (IF) is the most well-known isolation-based anomaly detection method, which focuses on isolating abnormal instances rather than profiling normal instances.

- **LSTM-VAE** (Park et al., 2018): LSTM-VAE uses a series of connected variational autoencoders and long-short-term-memory layers for anomaly detection.

- **DAGMM** (Zong et al., 2018): DAGMM is an unsupervised anomaly detection model which utilizes an autoencoder and the Gaussian mixture model in an end-to-end training manner.

- **OmniAnomaly** (Su et al., 2019): OmniAnomaly employs a stochastic recurrent neural network for multivariate time-series anomaly detection to learn robust representations with a stochastic variable connection and planar normalizing flow.

- **USAD** (Audibert et al., 2020): USAD utilizes an encoder-decoder architecture with an adversely training framework inspired by generative adversarial networks.

- **THOC** (Shen et al., 2020): THOC combines a dilated recurrent neural network (Chang et al., 2017) for extracting multi-scale temporal features with the deep support vector data description (Ruff et al., 2018).

4.3. Evaluation Metrics

We use precision (P), recall (R), F1-score (F1) for evaluating time-series anomaly detection methods. Since these performance measures depend on the way threshold is set on the anomaly scores, previous works proposed a strategy

such as applying extreme value theory (Siffer et al., 2017), using a dynamic error over a time window (Hundman et al., 2018). However, not all methodologies develop a mechanism to select a threshold in different settings, and many previous works (Audibert et al., 2020; Su et al., 2019; Xu et al., 2018) adopt the best F1 score for performance comparison, where the optimal global threshold is chosen by trying out all possible thresholds on detection results. We also use the point-adjust approach (Xu et al., 2018), widely used in evaluation (Audibert et al., 2020; Su et al., 2019; Shen et al., 2020). Specifically, if any point in an anomalous segment is correctly detected, other observations in the segment inside the ground truth are also regarded as correctly detected.

Therefore, we adopt the best F1-score (short F1 score hereafter) and the point-adjust approach for evaluating the anomaly detection performance to directly compare with the aforementioned state-of-the-art methods.

4.4. Hyperparameters and Experimental Setup

To show the robustness of our proposed method, we conduct experiments using the same hyperparameter setting for all benchmark datasets.

The details of the experimental setting are as follows. For the model architecture, we use a 3-layer MLP with sinusoidal activations with 256 hidden dimensions each (refer to Figure 3(b)). Following (Sitzmann et al., 2020), we set $\omega_0 = 30$ except for the first layer, which is set to 3000. During training, we use the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.0001 and $(\beta_1, \beta_2) = (0.9, 0.99)$. Additionally, we use early stopping with patience 30. Our code and data are released at https://github.com/KyeongJoong/INRAD

4.5. RQ 1. Performance Comparison

Table 2 shows the performance comparison results of our proposed method INRAD_{temp} and its variants, along with other baseline approaches on five benchmark datasets. We use the reported accuracy values of baselines (except THOC (Shen et al., 2020)) from the previous work (Audibert et al., 2020), which achieves state-of-the-art performance in the identical experimental setting with ours, such as datasets, train/test split, and evaluation metrics. Note that results of THOC on the WADI dataset are omitted due to an out-of-memory issue.

Overall, our proposed INRAD_{temp} consistently achieves the highest F1 scores over all datasets. Especially, the performance improvement over the next best method achieves 0.32 in terms of F1 score on the WADI dataset, where most other approaches show relatively low performance. On other datasets, we still outperform the second-best performance of other baselines by 0.02 to 0.12. Considering that the single

Method	SMD			SMAP			MSL			SWaT			WADI		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
IF	59.4	85.3	0.70	44.2	51.1	0.47	56.8	67.4	0.62	96.2	73.1	0.83	62.4	61.5	0.62
LSTM-VAE	87.0	78.8	0.83	71.6	98.8	0.83	86.0	97.6	0.91	71.2	92.6	0.80	46.3	32.2	0.38
DAGMM	67.3	84.5	0.75	63.3	99.8	0.78	75.6	98.0	0.85	82.9	76.7	0.80	22.3	19.8	0.21
OmniAnomaly	98.1	94.4	0.96	75.9	97.6	0.85	91.4	88.9	0.90	72.2	98.3	0.83	26.5	98.0	0.41
USAD	93.1	94.4	0.96	77.0	98.3	0.86	88.1	97.9	0.93	98.7	74.0	0.85	64.5	32.2	0.43
THOC	73.2	78.8	0.76	79.2	99.0	0.88	78.9	97.4	0.87	98.0	70.6	0.82	-	-	-
INRAD$^c_{van}$	94.7	97.8	0.96	80.0	99.3	0.89	93.6	98.1	**0.96**	96.9	88.7	0.93	60.2	67.0	0.63
INRAD$^c_{temp}$	98.0	98.3	**0.98**	83.2	99.1	0.90	92.1	99.0	0.95	93.0	96.3	0.95	78.4	99.9	0.88
INRAD$_{van}$	98.0	98.6	**0.98**	84.0	99.4	0.91	90.4	99.0	0.95	96.4	91.7	0.94	77.1	66.5	0.71
INRAD$_{van*}$	95.0	96.4	0.95	82.6	99.3	0.90	91.7	98.7	0.95	84.2	84.7	0.84	72.4	72.8	0.73
INRAD$_{temp}$	98.2	97.5	**0.98**	85.8	99.5	**0.92**	93.3	99.0	**0.96**	95.6	98.8	**0.97**	88.9	99.1	**0.94**

Table 2: Anomaly detection accuracy results in terms of precision(%), recall(%), and F1-score, on five real-world benchmark datasets. INRAD$_{van}$, INRAD$_{van*}$, and INRAD$_{temp}$ adopts the vanilla, vanilla*, and temporal encoding, respectively. Also, INRAD$^c_{van}$ and INRAD$^c_{temp}$ indicates that the experiment was run on the cold-start setting with each encoding.

Method	SMD	SMAP	MSL
LSTM-VAE	3.807	0.987	0.674
OmniAnomaly	77.32	16.66	15.55
USAD	0.278	0.034	0.029
THOC	0.299	0.07	0.066
INRAD	0.243	0.024	0.020

Table 3: Comparison of training time (sec) per epoch.

hyperparameter setting restriction is only applied for our method, this shows that our approach can provide superior performance in a highly robust manner in various datasets.

As we adopt the representation error-based detection strategy, it is possible that our method detects anomalies without training data by directly representing the test set. We hypothesize that the test set already contains an overwhelming portion of normal samples from which the model can still learn temporal dynamics of normal patterns in the given data without any complex model architectures (e.g., RNN and its variants). To distinguish from the original method, we denote this variant as INRAD$^c_\phi$ (ϕ = 'van' or 'temp' for vanilla and temporal encoding, respectively), which we also investigate its performance. We observe that both INRAD$^c_{van}$ and INRAD$^c_{temp}$ generally achieves slightly inferior performance to INRAD$_{temp}$ except for WADI. This result shows that utilizing the training dataset has performance benefits, especially when the training data is much longer than the test data.

4.6. RQ 2. Effect of temporal encoding

We study the effect of our temporal encoding method by comparing it with two encoding methods: vanilla and its variant, vanilla*. Vanilla encoding normalizes the indices

$[1, 2, \cdots, M]$ of the training data to $[-1, 1]$, and the indices of the test data are also mapped to $[-1, 1]$. On the other hand, vanilla* encoding is derived from vanilla encoding to preserve chronological order and unit interval of training and test data after encoding by mapping indices of test data to the range $[1, \infty)$, while keeping the difference between neighboring encoding consistent with the training data. We denote the variant using vanilla and vanilla* as INRAD$_{van}$ and INRAD$_{van*}$, respectively.

Table 2 shows that INRAD$_{temp}$ achieves slightly superior performance in general compared to INRAD$_{van}$ and INRAD$_{van*}$. However, when the length of the dataset becomes long, the performance of vanilla and vanilla* degrades significantly while temporal encoding remains at 0.94, as shown in the case of WADI. The performance gap in WADI becomes even more significant in the case of cold-start settings, which is 0.25.

Also, Figure 3 compares the convergence time for representation of test data between INRAD$_{van}$, INRAD$_{van*}$, and INRAD$_{temp}$ using MSL and SMAP dataset. The vanilla encoding shows the slowest convergence time, and the vanilla* and our temporal encoding shows competitive results. This result shows that representation of test data is learned faster when time coordinates in training and test data are encoded while preserving the chronological order. Overall, our temporal encoding strategy achieves superior performance and fast convergence compared to the vanilla encoding strategy.

4.7. RQ 3. Training speed comparison

Here, we study the training speed of INRAD$_{temp}$ and compare it to the other four baselines that show good performance. Table 3 summarizes the results the training time per epoch for INRAD$_{temp}$ along with other baseline methods on three benchmark datasets. Specifically, the reported time is
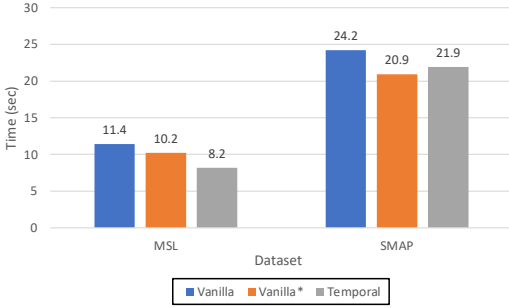
Figure 3: Convergence time (sec) comparison for different encoding techniques.

(a) Effect of patience   (b) Effect of hidden dimension

(c) Effect on ω_0 in first layer   (d) Effect of number of layers

Figure 4: Performance of $INRAD_{temp}$ for MSL under various hyperparameter settings.

the average time across all entities within each dataset (i.e., 28 entities for SMD, 55 for SMAP, and 27 for MSL). The results show that our method achieves the fastest training time, mainly because our method only uses a simple MLP for training without any additional complex modules (e.g., RNNs).

4.8. RQ 4. Hyperparameter sensitivity

In Figure 4, we test the robustness of $INRAD_{temp}$ by varying different hyperparameters settings using the MSL dataset. In our experiment, we change the patience in early stopping in range $\{30, 60, 90, 120, 150\}$, size of hidden dimension in range $\{32, 64, 128, 256, 512\}$, ω_0 of the first layer in range $\{30, 300, 3000, 30000, 300000\}$, and the number of layers in range $\{1, 2, 3, 4, 5\}$. Figure 4a, 4b, and 4d shows that $INRAD_{temp}$ achieves high robustness with varying hyperparameter settings. We see that the choice of ω_0 also minimally impacts $INRAD_{temp}$ as in Figure 4c. This results suggests that the MLP struggles to differentiate neighboring inputs in the case where ω_0 is extremely low.

5. Conclusion

In this paper, we proposed INRAD, a novel implicit neural representation-based method for multivariate time-series anomaly detection, along with a temporal encoding technique. Adopting a simple MLP, which takes time as input and outputs corresponding values to represent a given time-series data, our method detects anomalies by using the representation error as anomaly score. Various experiments on five real-world datasets show that our proposed method achieves state-of-the-art performance in terms of accuracy and training speed while using the same set of hyperparameters. For future work, we can consider additional strategies for online training in order to improve applicability in an environment where prompt anomaly detection is needed.
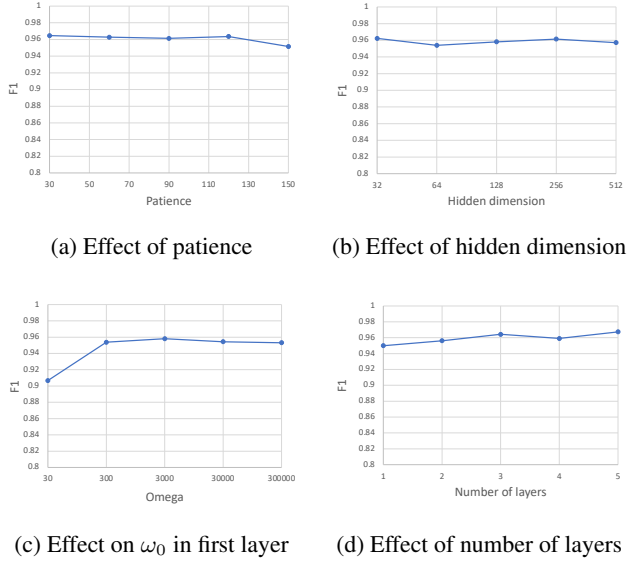
References

Audibert, J., Michiardi, P., Guyard, F., Marti, S., and Zuluaga, M. A. USAD: UnSupervised anomaly detection on multivariate time series. In *KDD*, pp. 3395–3404, 2020.

Blázquez-García, A., Conde, A., Mori, U., and Lozano, J. A. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys (CSUR)*, 54(3):1–33, Apr. 2021.

Chang, S., Zhang, Y., Han, W., Yu, M., Guo, X., Tan, W., Cui, X., Witbrock, M., Hasegawa-Johnson, M. A., and Huang, T. S. Dilated recurrent neural networks. In *NeurIPS*, pp. 76–86, 2017.

Deng, A. and Hooi, B. Graph neural network-based anomaly detection in multivariate time series. In *AAAI*, pp. 4027–4035, 2021.

Fox, A. J. Outliers in time series. *J. Roy. Statist. Soc. B Methodol.*, 34(3):350–363, July 1972.

Galeano, P., Peña, D., and Tsay, R. S. Outlier detection in multivariate time series by projection pursuit. *J. Amer. Statist. Assoc.*, 101(474):654–669, June 2006.

Gupta, M., Gao, J., Aggarwal, C. C., and Han, J. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 2014.

Hundman, K., Constantinou, V., Laporte, C., Colwell, I., and Söderström, T. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *KDD*, pp. 387–395, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Li, D., Chen, D., Jin, B., Shi, L., Goh, J., and Ng, S.-K. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In *ICANN*, pp. 703–716, Munich, Germany, September 2019.

Liu, F. T., Ting, K. M., and Zhou, Z. Isolation forest. In *ICDM*, pp. 413–422, 2008a.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. Isolation forest. In *ICDM*, pp. 413–422, 2008b.

Ma, J. and Perkins, S. Time-series novelty detection using one-class support vector machines. In *IJCNN*, pp. 1741–1745, Portland, OR, July 2003.

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., and Shroff, G. LSTM-based encoder-decoder for multi-sensor anomaly detection. In *ICML, Anomaly Detection Workshop*, pp. 1–5, 2016.

Martin-Brualla, R., Radwan, N., Sajjadi, M. S. M., Barron, J. T., Dosovitskiy, A., and Duckworth, D. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, pp. 7210–7219, 2021.

Mathur, A. P. and Tippenhauer, N. O. Swat: a water treatment testbed for research and training on ics security. In *CySWater@CPSWeek 2016*, pp. 31–36, 2016.

Mescheder, L. M., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, pp. 4460–4470, 2019.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, pp. 405–421, 2020.

Niemeyer, M., Mescheder, L. M., Oechsle, M., and Geiger, A. Occupancy flow: 4d reconstruction by learning particle dynamics. In *ICCV*, pp. 5378–5388, 2019.

Park, D., Hoshi, Y., and Kemp, C. C. A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. *IEEE Robot. Autom. Lett.*, 3(3): 1544–1551, 2018.

Park, J. J., Florence, P., Straub, J., Newcombe, R. A., and Lovegrove, S. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, pp. 165–174, 2019.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., and Kloft, M. Deep one-class classification. In *ICML*, pp. 4393–4402, Stockholm, Sweden, July 2018.

Saito, S., Huang, Z., Natsume, R., Morishima, S., Li, H., and Kanazawa, A. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *ICCV*, pp. 2304–2314, 2019.

Sakurada, M. and Yairi, T. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *MLSDA Workshop*, pp. 4–11, Gold Coast, Australia, December 2014.

Shen, L., Li, Z., and Kwok, J. T. Timeseries anomaly detection using temporal hierarchical one-class network. In *NeurIPS*, 2020.

Siffer, A., Fouque, P.-A., Termier, A., and Largouet, C. Anomaly detection in streams with extreme value theory. In *KDD*, pp. 1067–1075, 2017.

Sitzmann, V., Martel, J. N. P., Bergman, A. W., Lindell, D. B., and Wetzstein, G. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020.

Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., and Pei, D. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *KDD*, pp. 2828–2837, 2019.

Xu, H., Chen, W., Zhao, N., Li, Z., Bu, J., Li, Z., Liu, Y., Zhao, Y., Pei, D., Feng, Y., Chen, J., Wang, Z., and Qiao, H. Unsupervised anomaly detection via variational autoencoder for seasonal KPIs in web applications. In *WWW*, pp. 187–196, Lyon, France, April 2018.

Zhang, Y., Ge, Z., Greenberg, A., and Roughan, M. Network anomography. In *IMC*, pp. 317–330, Berkeley, CA, October 2005.

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., and Chen, H. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *ICLR*, 2018.

A. Baseline Implementation

We describe the implementation of the baseline methods used in our paper. Isolation forest (IF) (Liu et al., 2008b) is implemented using the scikit-learn library, and we use the source code of THOC (Shen et al., 2020) given by authors. The other baselines are downloaded from the following links:

- LSTM-VAE (Park et al., 2018):
  https://github.com/Danyleb/Variational-Lstm-Autoencoder

- DAGMM (Zong et al., 2018):
  https://github.com/tnakae/DAGMM

- OmniAnomaly (Su et al., 2019):
  https://github.com/NetManAIOps/OmniAnomaly

- USAD (Audibert et al., 2020):
  https://github.com/robustml-eurecom/usad

B. Vanilla Encoding

In this section, we describe the encoding strategy in (Sitzmann et al., 2020) which we call vanilla encoding in our main paper. Let us assume that the given dataset has N timestamps with index i (i.e., $X = \{(t_i, \mathbf{x}_{t_i})\}_{i=1}^N$). Also, denote the vector of indices as $\mathbf{i} = [1, 2, \cdots, N]$, and $\mathbf{1} \in \mathbb{R}^{1 \times N}$ as the one vector. Vanilla encoding plainly normalizes each indices i to the range $[-1, 1]$ by $\mathbf{i}_{naive} = (2/N) \times \mathbf{i} - \mathbf{1}$.

C. Detailed Setting of Temporal Encoding

In the case of SWaT and WADI datasets, we use the actual timestamps given in each dataset. On the other hand, in the case of SMD, MSL, SMAP dataset that does not contain such information, we arbitrarily set the timestamps for each sample starting from "2021-01-01 00:00:00" with one-minute intervals. We assume that the test data is directly after the end of the timestamp in the training set.

Now we describe the details of the pre-defined k'_{year} and N_1. We set k'_{year} as the year of the first timestamp in the training set as we assume that our model will not encounter past information before the first sample in the training set. In the case of SWAT and WADI, we set k'_{year} as 2015 and 2017, respectively, following the given timestamp information as we stated earlier. For the other datasets, we set k'_{year} as 2021 as we set the first timestamp in the training set as "2021-01-01 00:00:00". Next, we set N_1 by $\gamma + 1$, where γ indicates the difference between the year of the earliest and latest observed data. We note that, however, these settings can be flexibly chosen for various settings.

D. Detailed Description of the Datasets

SMD (Su et al., 2019) is a 5-week-long public dataset collected from a large Internet company containing data from 28 server machines, each one monitored by 33 metrics. It is divided into two subsets of equal size, where the first half is the training set and the second half is the testing set. SMAP and MSL (Hundman et al., 2018) are two real-world public datasets, expert-labeled datasets from NASA. SMAP contains the data from 55 entities monitored by 25 metrics, and MSL contains the data from 27 entities monitored by 55 metrics. SWaT (Mathur & Tippenhauer, 2016) is collected from a scaled-down real-world industrial water treatment plant that produces filtered water. In SWaT, operational data under normal circumstances are collected for 7 days, and operational data with attack scenarios are collected under 4 days. In WADI (Mathur & Tippenhauer, 2016), operational data under normal circumstances are collected for 14 days, and operational data with attack scenarios are collected under 2 days. This dataset is collected from the WADI testbed, an extension of the SWaT testbed.