Visual Analytics for Process Monitoring: Leveraging Time-Series Imaging for Enhanced Interpretability

Ibrahim Yousef[1], Aditya Tulsyan[2], Sirish L. Shah[3], and R. Bhushan Gopaluni[1*]

[1]Department of Chemical and Biological Engineering, The University of British Columbia, Vancouver, Canada
[2]Process Systems Engineering Laboratory, Massachusetts Institute of Technology, Cambridge, USA
[3]Department of Chemical and Materials Engineering, University of Alberta, Edmonton, Canada
[*]Corresponding author: bhushan.gopaluni@ubc.ca

Abstract

In the era of big data driven by the advent of the Internet of Things (IoT), process industries face the challenge of analyzing massive and complex data to extract relevant information for effective process monitoring. Despite exploring various approaches, scalability and interpretability continue to present practical limitations. To address these limitations, we propose a new framework called *visual analytics*. Visual analytics offers a new perspective on solving process monitoring problems. It involves transforming historical process data into visual clues, thereby converting traditional fault detection problems into image classification problems. This approach allows process experts to visually analyze patterns and textures within the data, making interpretation much easier compared to traditional time domain analysis. Moreover, by treating process data as images, visual analytics can leverage a wide range of computer vision techniques, including convolutional neural networks (CNNs), to accurately classify and detect faults. By integrating human visual perception with advanced computer vision techniques, visual analytics enables the effective analysis of massive and complex process data. To empirically validate the proposed visual analytics approach, we conduct experiments on both the simulated continuous stirred tank heater (CSTH) benchmark and the industrial arc loss benchmark. The experimental results from both benchmarks demonstrate that the proposed visual analytics approach yields competitive performance in predicting process faults while enhancing interpretability by providing meaningful and informative visual representations.

Keywords— Fault detection and diagnosis (FDD), Deep learning, Image analysis, Time-Series imaging, Convolutional neural networks (CNNs), Applications.

1 Introduction

With the increasing automation of industries and advancement in connectivity, modern industries generate large volumes of data continuously [1, 2]. These industrial systems are equipped with a wide range of sensors strategically placed throughout the process, resulting in data being dispersed across multiple dimensions and time [3, 4]. This type of data, referred to as time-series data, encompasses sequential observations or measurements recorded over time, capturing the temporal aspect of system behavior [5, 6]. The classification of these time-series data is crucial in detecting faults, ensuring safety, and upholding high product quality standards [7]. In fact, process monitoring tasks, such as fault detection and diagnosis (FDD), can be configured as time-series classification (TSC) problems.

Traditionally, TSC methods have relied on manually extracting relevant features from the input data [8]. The goal of these methods is to identify important local or global patterns within the time-series that are associated with specific categories or classes [9]. A common feature-based method is the k-nearest neighbors (k-NN) classifier with handcrafted features [10]. In this approach, a set of relevant features is manually extracted from each time-series, such as statistical features (e.g., mean, standard deviation) or frequency domain characteristics (e.g., Fourier coefficients). These features capture key aspects of the time-series data that are assumed to be relevant for classification. The k-NN algorithm then classifies an unseen time-series by measuring the similarity between its extracted features and those of labeled examples in the training set.

However, traditional feature-based approaches have significant drawbacks. Firstly, their performance heavily relies on selecting relevant features, which can be labor-intensive and time-consuming [11]. Different problems may require distinct sets of discriminatory features, limiting the generalizability of these methods. Secondly, feature extraction can result in the loss of information present in the original data [12]. For example, fine-grained temporal changes that are important for understanding the behavior of a dynamic process can be smoothed out or overlooked during feature extraction. Moreover, the computational complexity of these methods evolves according to a power law with respect to the input data size. As industrial data scale up, the computational demands of these methods increase exponentially, rendering them impractical for managing the large-scale datasets prevalent in industrial applications [13].

More recently, deep learning (DL) has sparked numerous breakthroughs across various problem domains, including computer vision and natural language processing [14]. The core strength of DL models lies in their ability to directly learn high-level representations from input data, bypassing the tedious feature engineering process [15]. As a result, DL has garnered significant research interest in addressing TSC problems. For instance, Wang et al. [11] proposed deep neural networks such as multilayer perceptron (MLP) and long short-term memory (LSTM) networks as robust baseline approaches for TSC. These models achieved competitive performance in TSC tasks, effectively learning discriminative representations from raw time-series data, thereby highlighting the effectiveness of DL in TSC. In addition, Zerveas et al. [16] applied the transformer architecture, originally developed for natural language processing tasks, to the realm of multivariate time-series classification (MTSC). By leveraging self-attention mechanisms, transformers directly capture temporal dependencies in time-series data.

TSC presents unique challenges compared to traditional supervised learning for structured data, as algorithms must effectively handle and exploit the temporal information embedded within the signal [9]. Interestingly, there are striking parallels between TSC and computer vision problems like image classification and object localization [17]. In the latter, successful algorithms learn from the spatial information contained in images. Similarly, in TSC, the core problem is fundamentally the same, albeit with one fewer dimension [13]. Inspired by these observations, the opportunity for applying DL-based computer vision algorithms to TSC becomes apparent. One such example is using convolutional neural networks (CNNs), a powerful class of models commonly used for image classification. Although CNNs were originally designed to operate exclusively on spatial data, they can be adapted to handle time-series data by treating the temporal dimension as analogous to the spatial dimension in images [18]. Using the hierarchical feature extraction capabilities, CNNs have achieved competitive results in TSC and FDD tasks, often surpassing traditional approaches [19–22].

At the same time, many practical limitations remain, especially when dealing with industrial process data. First, process variables in data samples are typically arranged according to their order in the processing procedure. Although CNNs can effectively learn correlations between variables within the same local receptive field, they may not fully capture the various correlations among variables that are distant in the physical topological structure (i.e., variables order within the dataset) [20]. The limited scope of the receptive field in the first convolutional layers prevents the learning of correlations between process variables that are not in the same local receptive field. Consequently, higher-level representations inadequately capture these correlations. Furthermore, interpretability is crucial for understanding and explaining model predictions, particularly for FDD applications [23]. However, CNNs are black-box models, making it challenging to interpret their decisions and restricting their practical applicability in critical industrial settings [24].

To address the aforementioned gaps, we introduce a new paradigm for process monitoring called *visual analytics*. The main idea behind visual analytics involves the transformation of time-series data (i.e., process data) into visual images, thereby enabling the use of computer vision algorithms (e.g., CNNs) [25]. This approach capitalizes on the strengths of CNNs while acknowledging the inherent dissimilarities between time-series data and images. By representing the data as visual images, process operators can interact with the data more intuitively compared to time-series data analyzed in the time domain. In addition, this visual representation allows operators to develop a deeper understanding and intuition in relating different image patterns to different process operating modes.

The remainder of this paper proceeds as follows. In Section 2, we provide the necessary background and discuss related work. Section 3 presents a supervised learning-based visual analytics framework for FDD and describes the network architecture and its main building blocks. In Section 4, we describe the implementation details of our proposed approach with a simulation and industrial case studies. Finally, we conclude with closing remarks in Section 5, highlighting potential future prospects.

2 Background & Related Work

In this section, we begin by introducing the fundamental concepts and definitions relevant to our study. Following that, we provide an overview of TSC and delve into the foundational aspects of convolution operations, which serve as the primary component in CNNs. Lastly, we discuss two widely known time-series imaging tools.

2.1 Definitions

The focus of this work revolves around time-series data. A univariate time-series signal, denoted as $S = \{s_1, s_2, ..., s_L\}$, represents a sequence of L chronologically ordered observations recorded over time. Each observation is associated with a timestamp from the set $T = \{t_1, t_2, .., t_L\}$. When multiple time-series signals are recorded simultaneously by a set of p sensors, we refer to the data as a multivariate time-series (MTS) signal denoted as $X = \{S_1, S_2, ..., S_p\}$, where $S_i \in R^L$. We consider MTS with a fixed and synchronized sampling along all dimensions. As a result, we omit the time index from the MTS definition. In this work, we consider an MTS dataset $D = \{(X_1, Y_1), (X_2, Y_2), \dots (X_N, Y_N)\}$, containing a collection of paired samples (X_i, Y_i). Each sample consists of an MTS signal X_i with p dimensions and a length of L, and its corresponding label Y_i. The task of TSC involves training a classifier C on dataset D that maps an input X_i to its true label Y_i (i.e., $C : X \longrightarrow Y$). In the context of FDD, the labels represent the operating condition of a system, where $Y = 0$ denotes a normal condition and $Y = 1$ denotes a faulty condition. Therefore, the goal of TSC in FDD is to accurately classify the input MTS signals into their corresponding operating conditions.

2.2 Time-series classifiers

Broadly speaking, time-series classifiers can be grouped based on the algorithmic technique used into three categories: distance-based, feature-based, and deep learning approaches.

Distance-based approaches rely on distance measures to evaluate the similarity/ dissimilarity between pairs of time-series [26]. A significant research effort has been dedicated to the development of "elastic" distance measures that compensate for small misalignments between time-series [27]. These measures seek to account for variations in the alignment of time points, allowing for more robust and accurate similarity comparisons. Among these elastic measures, dynamic time warping (DTW) has emerged as one of the most widely used measures [28]. In fact, DTW, in combination with 1-nearest neighbor (DTW+1NN), has gained significant recognition in the field of TSC and has long been considered a benchmark method, often referred to as the gold standard [29].

Next, the feature-based category refers to a group of TSC algorithms that rely on extracting relevant features from the time-series data [30]. It can be further divided into two main families: interval-based and dictionary-based approaches. Interval-based approaches use subsequences from the time-series, extracting discriminatory features using statistical measurements. One popular algorithm within the interval-based family is the time-series forest (TSF) [31]. TSF constructs an ensemble of decision trees using randomly selected intervals from the time-series and their corresponding statistical feature values (e.g., mean, slope, and standard deviation).

Furthermore, dictionary-based approaches involve discretizing time-series into symbolic sequences, extracting words from these sequences using a sliding window, and quantifying the frequency of each word in a predefined dictionary [32]. The bag of symbolic Fourier approximation symbols (BOSS) algorithm is a prominent example in the dictionary-based family [33]. BOSS represents time-series as bags of words, where each word corresponds to a symbolic Fourier approximation (SFA) coefficient. SFA is a technique that converts time-series data into a compact symbolic representation by approximating their Fourier transforms [34]. BOSS constructs a dictionary of SFA words using a training dataset and maps each time-series into a histogram representation based on the frequency of the SFA words occurring within it. Class labels can then be assigned using similarity or distance measures between the histograms.

DL approaches for TSC involve using neural networks that learn hierarchical representations from the input time-series data [35]. A neural network is considered *deep* when it consists of more than one layer between the input and output layers. Specifically, a deep neural network (DNN) is composed of K layers (i.e., parametric functions), where each layer serves as a representation of the input domain [36]. The simplest architecture within DL models is MLP, also known as a fully connected network (FCN) [11]. In an MLP, each neuron in layer k_i is connected to every neuron in layers k_{i+1} and k_{i-1}, with $i \in [2, K-1]$. These connections are modeled by the weights within the neural network, enabling the network to capture complex relationships within the data. One impediment to adopting MLPs for TSC is that they do not exhibit any spatial invariance. In other words, each time stamp has its own weight, and the temporal information is lost. To address the unique characteristics of time-series data, recurrent neural networks (RNNs) were introduced [37, 38]. RNNs maintain a hidden state that can capture information from previous time steps, allowing them to detect patterns in sequential data (e.g., time-series data). However, vanilla RNNs suffer from vanishing and exploding gradients, limiting their ability to capture long-term dependencies [39]. LSTM is a variant of RNNs that overcomes the limitations of vanilla RNNs [40]. Its architecture includes a cell state, input gate, forget gate, and output gate, allowing it to selectively remember or forget information over long sequences while avoiding the vanishing gradient problem [41, 42]. This makes LSTM networks particularly effective for tasks involving sequential data, including TSC and FDD [43, 44].

In this work, we compare and validate our proposed visual analytics framework by benchmarking it against state-of-the-art models in each of the three categories of time-series classifiers: i) distance-based category: DTW+1NN; ii) feature-based category: TSF and BOSS; iii) DL category: LSTM. A conceptual comparison between the aforementioned time-series classifiers is presented in Figure 1, showing the distinct approaches used by each classifier.

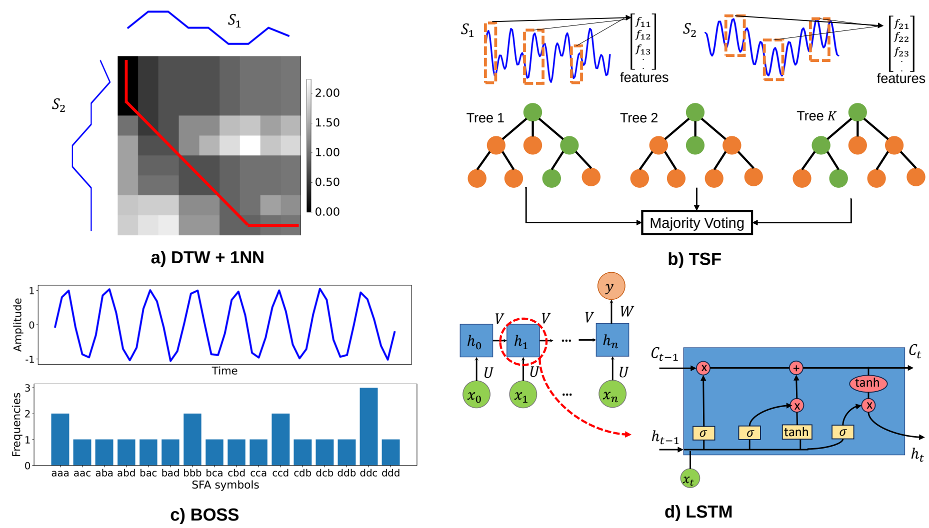

Figure 1: A conceptual illustration of state-of-the-art time-series classifiers: a) DTW+1NN: measures the similarity between time-series using DTW; b) TSF: uses random forest on features extracted from raw time-series data; c) BOSS: constructs a histogram-based representation of time-series using SFA; and d) LSTM: captures temporal dependencies in time-series data through its memory state.

2.3 Convolution operations

Convolution operations are the key functional operations that CNNs use to extract features from the input data [45]. In essence, a convolution operation involves an input signal and a kernel (an operator function). One can think of convolution as a mathematical operation that seeks to transform the input data to uncover and extract relevant features [46]. Convolution can be applied to signals of varying dimensions, such as 1D, 2D, or 3D. Notably, the dimensions of the kernel must align with those of the input. In practical applications, 1D convolution finds common use in processing audio signals [47, 48], while 2D convolution is widely used for image analysis tasks [49, 50]. Similarly, 3D convolution plays a role in video processing applications [51]. For the purpose of this paper, the discussion is narrowed to 1D and 2D convolutions, as our work does not involve video data.

Convolution operations involve sliding a kernel over input data and performing a dot product to extract features, which are referred to as feature maps [52]. The kernel, also known as a filter or a convolutional filter, is a set of weights that defines the transformation applied to the input. The specific design of the kernel determines the nature of the extracted features. These features encode various hidden aspects of the data, such as trends and variations. The kernel is typically smaller in size than the input data, reducing computational complexity. In 1D convolutions, the kernel is a weight vector, and the resulting feature map is obtained by adding a bias term to the sliding dot product between the 1D input data and kernel weights [53]. Figure 2 demonstrates a 1D convolution operation with a gradient kernel on a time-series signal. On the other hand, 2D convolutions are applied to 2D data (e.g., images), and the kernel takes the form of a matrix of weights. The feature maps produced by convolutions indicate the degree of similarity between the input and the kernel pattern. Convolution operations share similarities with feature-based methods, as both approaches rely on the identification and frequency of occurrence of specific patterns or motifs in the input data. In convolution operations, multiple kernels with different weights are used to capture a wide range of patterns and variations within the data. This combination of kernels allows for the detection of complex and discriminative features. The success of CNNs for TSC and image classification demonstrates the effectiveness of convolutional kernels as the foundation for extracting informative features from input data.

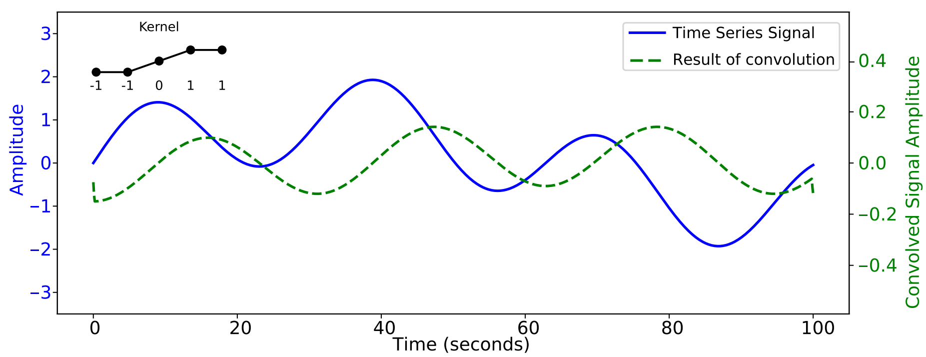

Figure 2: An illustration of a 1D convolution operation. The gradient kernel, [-1, -1, 0, 1, 1], is applied to the 1D signal to extract changes in amplitude or slope. The resulting convolved signal highlights regions where the input signal exhibits positive and negative gradients. The convolved signal can provide insights into the overall trend and direction of changes in the original time-series data.

2.4 Encoding time-series into images

Understanding and analyzing complex systems in the time domain poses a significant challenge in many scientific and engineering domains. Traditional time-series analysis methods often extract features that fail to capture the temporal evolution and dynamics of such systems. Consequently, researchers in signal processing and computer science have been exploring methods to represent temporal characteristics of time-series signals visually, using 2D images. Such methods provide a visual framework to capture, interpret, and extract meaningful temporal information from dynamic processes. Additionally, imaging tools transform raw time-series data into visual representations, facilitating the use of a wide range of image analysis algorithms (e.g., CNNs). Two prominent imaging techniques are the Gramian Angular Field (GAF) and the Recurrence Plot (RP).

A GAF is a 2D visual representation of a univariate time-series, originally introduced by Wang and Oates, which captures information about the static behavior of the time-series [54]. Figure 3 illustrates the step-by-step instructions for encoding a univariate time-series as a GAF image. First, the scaled time-series $\hat{S} = \{\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_L\}$ is transformed from the space coordinate system to polar coordinates. The time step t_i is encoded as the radius r_i and the scaled value $\hat{s}_i$ of the time-series is encoded as the angular cosine θ_i, given by:

$$\begin{aligned} r_i &= \tfrac{t_i}{L}; && i \in L \\ \theta_i &= \cos^{-1}(\hat{s}_i); && \hat{s}_i \in [0,1] \end{aligned} \tag{1}$$

where L represents the time-series length. Once $\hat{S}$ is transformed into polar coordinates, the square GAF matrix is constructed. In the GAF matrix, each entry represents the pairwise cosine distance between two angles, i.e.:

$$\text{GAF}\ [i,j] = \cos(\theta_i + \theta_j); \qquad i,j = 1,2,...,L \tag{2}$$

GAF offers several key advantages, making it a valuable tool for capturing temporal dynamics. Firstly, GAF preserves the temporal order of the original time-series, ensuring that the sequential nature of the data is maintained in the resulting image representation. In addition, GAF is invariant to monotonic transformations, meaning that it can capture the same underlying patterns regardless of scaling or shifting of the time-series values.

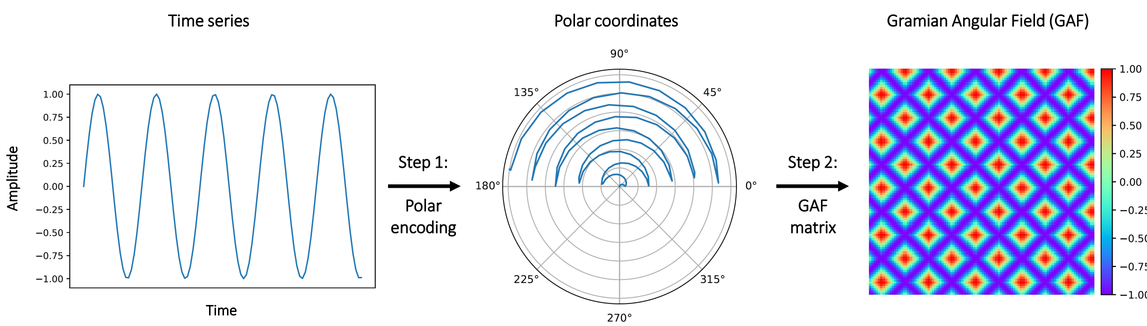

Figure 3: An illustration of the transformation from a sinusoidal time-series signal into a 2D GAF representation. Left: a scaled 1D time-series $\hat{S} = \{\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_{20}\}$ with $L = 20$ time steps. Middle: the first step is to represent $\hat{S}$ in polar coordinates using Equation 1. Right: the GAF image, a square matrix of size 20×20, is obtained using Equation 2.

Furthermore, dynamic nonlinear systems often exhibit recurrent behavior (e.g., periodicities and oscillations) that can be challenging to observe in the time domain. To tackle this challenge, Eckmann et al. introduced RP, a 2D visual representation of higher dimensional phase space trajectories [55]. RP is a square matrix that reveals at which points the m-dimensional phase space trajectory revisits a previously visited state. In this work, we consider the non-binarized version of RP, as proposed by [56], to avoid information loss when the matrix is binarized. In practice, one performs two steps to obtain an RP image from a univariate time-series. First, an embedding dimension m is chosen. The m-dimensional phase space $\overrightarrow{S}$ is constructed from the scaled time-series $\hat{S} = \{\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_L\}$ using the time-delay embedding method (i.e., $\overrightarrow{s_i} = (\hat{s}_i, \hat{s}_{i+1}, ..., \hat{s}_{i+m-1})$. Next, the RP matrix is calculated as follows:

$$\text{RP}\ [i,j] = \|\overrightarrow{s_i} - \overrightarrow{s_j}\|; \qquad i,j = 1,2,...,R \tag{3}$$

where R is the total number of considered states $\overrightarrow{s}$ (i.e., $R = L - m + 1$) and $\|.\|$ is the Euclidean norm.

Each pixel in RP denotes the Euclidean distance of two states in the m-dimensional phase space. The full procedure for encoding a univariate time-series as an RP image is shown in Figure 4. The colors in the RP image indicate the closeness of the states in the 2D phase space according to the corresponding color bar. As shown in Figure 5, GAF and RP, both display texture and typology, which provide hints about the *static* and *recurrent* behaviors of the 1D time-series, respectively.

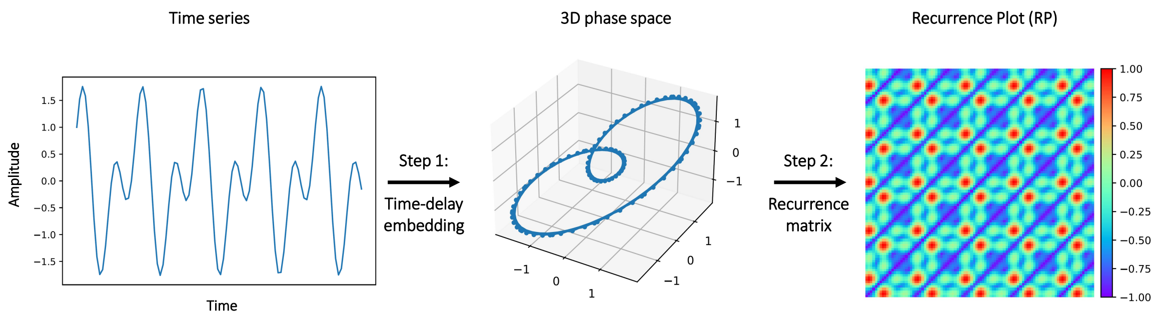

Figure 4: An illustration of the encoding map from a raw 1D time-series signal to a 2D RP image. Left: a scaled 1D time-series $\hat{S} = \{\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_{20}\}$ with $L = 20$ time steps. Middle: the m-dimensional phase space trajectory is constructed from X using the time-delay embedding. In this examples, $m = 3$; hence, the states, represented in dots, $\overrightarrow{s_i} = (\hat{s}_i, \hat{s}_{i+1}, \hat{s}_{i+2})$. Right: the RP image, an 18×18 matrix, is obtained using Equation 3.

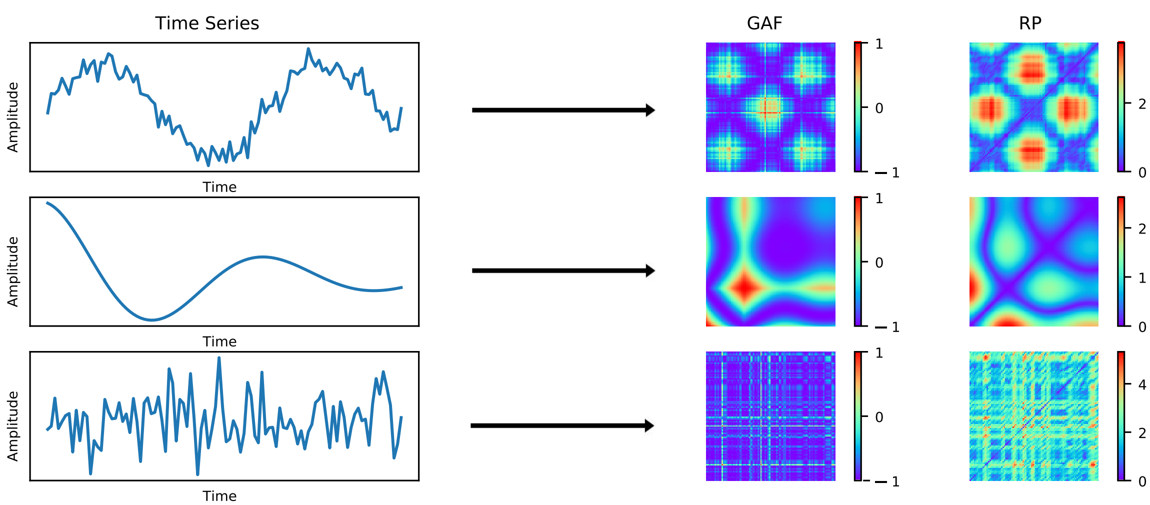

Figure 5: An illustration of the qualitative interpretations of GAF and RP representations for three distinct time-series signals.

3 Proposed Approach

In this section, we present a new end-to-end visual analytics framework for industrial fault detection using 1D and 2D convolution operations. The proposed approach obtains visual data representations from input MTS signals in a *supervised* manner, i.e., the proposed model takes an annotated MTS dataset to learn visual representations. The main objective of the proposed network is to maximize the visual distinction between the visual representation of samples across different classes. The overall architecture of the proposed visual analytics framework is shown in Figure 6.

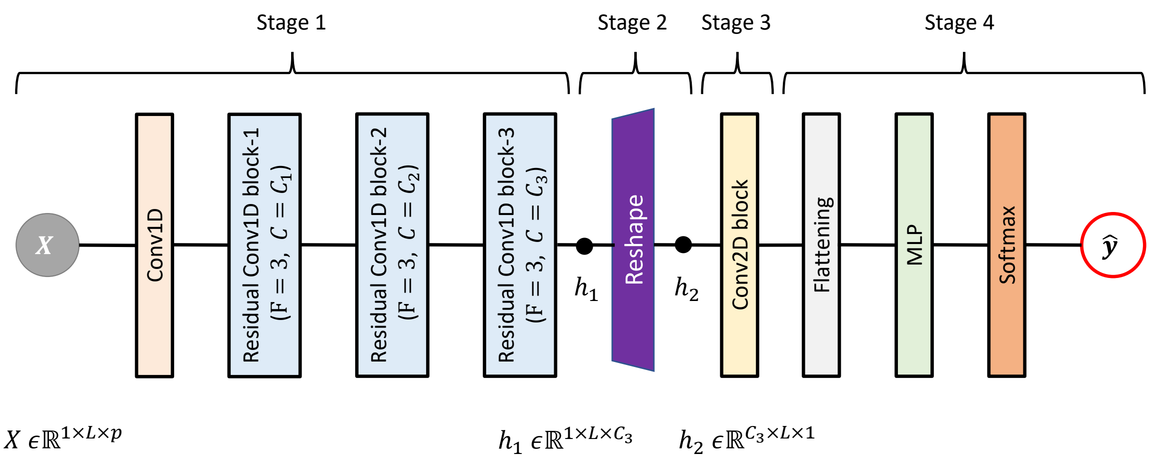

Figure 6: The proposed network architecture. The residual block configuration is presented in Figure 7. Abbreviations: F - kernel size, C - number of kernels, L - the size of the input signal (i,e, the number of time steps), p - number of variables, $\hat{y}$ - predicted class.

The proposed network is inspired based on two influential architectures: AlexNet [57] and ResNet [50]. Like AlexNet, the proposed network uses MLP for making predictions. Also, we implement dropout as a regularization technique to prevent overfitting. By randomly dropping out units during training, we enhance the network's ability to generalize and avoid relying too heavily on specific neurons [58]. Furthermore, the proposed network adopts the concept of residual blocks from ResNet. Using residual connections, we address the vanishing gradient problem and facilitate the training of deeper networks. The residual blocks enable the network to capture residual information, making it easier to propagate gradients through the network and effectively learn both shallow and deep features. This improves network performance, allowing for better representation learning and enhanced prediction performance.

Stage 1 of the network focuses on the initial processing and feature extraction from the input data X. This stage comprises a 1D convolutional layer followed by a series of residual blocks. The primary purpose of the initial 1D convolutional layer is to transform the input data X into a suitable format for subsequent layers while extracting low-level features from the data. Next, we use a series of "bottleneck" 1D residual blocks, consisting of three convolutional layers: a 1×1 convolution, a 1×3 convolution, and another 1×1 convolution (refer to Figure 7). This design effectively reduces the number of trainable parameters, resulting in a computationally efficient network. In addition, we increment the number of 1D kernels after each residual block to enable the network to capture increasingly higher-level features as the information propagates through the layers. The number of residual blocks, as well as the number of kernels within each residual block, are hyperparameters that need to be prespecified based on the complexity of the problem and data requirements. We show three blocks in Figure 6 for illustration purposes. Note that we omit pooling layers in the network to preserve valuable spatial information. We also apply "same" padding and a stride size set to 1. Overall, stage 1 of the network seeks to learn discriminative visual representations via 1D convolution operations.

The output of stage 1 is a combination of C_3 feature maps, corresponding to the number of 1D kernels

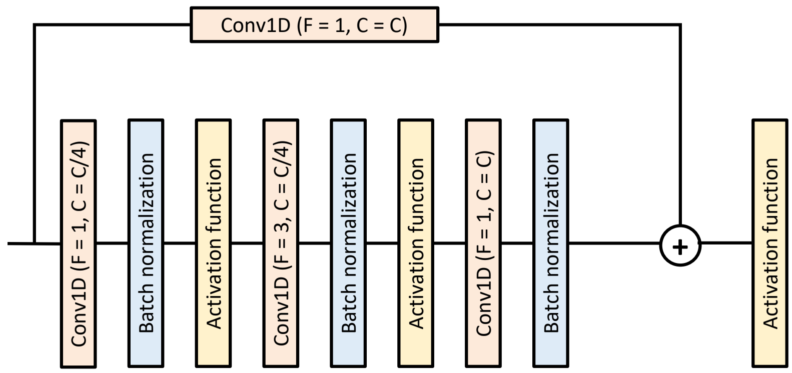

Figure 7: The residual block configuration. Abbreviations: F - kernel size, C - number of kernels.

in the last residual block of size L. Each feature map captures the activation level of its corresponding kernel, indicating the response of that specific kernel pattern across the features of the input X. To visualize these features and gain a deeper understanding of the network's representations, we perform a reshaping step (stage 2). This step involves vertically stacking the 1D feature maps to form a 2D-like matrix with dimensions $C_3 \times L \times 1$ (height $\times$ width $\times$ channels). This reshaping enables the visualization of the learned features using color mapping techniques. Each pixel in the resulting image represents the activation level of a particular feature map at a specific spatial location. Figure 12 presents six samples of the images obtained in Stage 2.

Next, the one-channel image obtained from stage 2 is "visually" recognized, and visual features and patterns are learned via 2D convolution operations. Stage 3 consists of stacked 2D convolutional blocks. Each 2D convolution block contains a 2D convolutional layer followed by a batch normalization layer and a non-linear activation layer. The number of 2D convolution blocks used may vary depending on the application. For demonstration purposes, we use a single 2D convolution block. The visual features are extracted using a stride size set to 1, and we use "valid" padding to reduce the dimension of the feature maps, thus reducing the computational cost. The 2D convolutional blocks enable the network to effectively capture spatial information and extract discriminative visual features from the input data.

In the last stage of the network, the feature maps obtained from the last 2D convolutional layer are flattened before being fed into an MLP network. Flattening is the process of converting the multidimensional feature maps into a one-dimensional vector. The MLP network consists of multiple dense layers that map the extracted visual features into a scalar, representing a class label. The output layer of the MLP network uses the softmax activation function, which produces a probability distribution over the classes. The number of neurons in the output layer corresponds to the number of classes. For binary classification problems, the number of output neurons is two. The class label associated with the highest probability is considered the predicted class for a given input. During training, the network is trained to minimize the loss function. Specifically, we use the binary cross-entropy loss for our case (i.e., binary classification problems). Next, the proposed network is trained by minimizing the loss function using an optimization algorithm. Therefore, the model learns to assign high probabilities to the correct class and lower probabilities to the other classes. The optimization algorithm and its arguments are hyperparameters to be specified before training.

4 Applications of the Proposed Approach

In this section, we use two benchmarks to assess the effectiveness of our proposed visual analytics approach. We start with the simulated continuous stirred tank heater (CSTH) benchmark to better demonstrate the applicability and benefits of our approach in a controlled setting. By analyzing the simulated data, we show the interpretability and insights provided by our visual analytics approach. Next, we move on to the industrial arc loss benchmark, which serves as a large-scale scenario to test the scalability and robustness of our proposed approach. The proposed approach will be compared with DTW+1NN, TSF, BOSS, LSTM, GAF followed by CNN, and RP followed by CNN models.

4.1 A simulation case study: the CSTH system

The CSTH system is a dynamic nonlinear system reported in [59]. Figure 8 shows the feedback control system used for the CSTH system. In this system, hot water (HW) and cold water (CW) are mixed, heated by steam flowing through a heating coil, and eventually drained from the tank. To ensure system stability, a closed-loop control system is implemented to regulate the tank's temperature, level, and CW flow. The input signals for the system consist of steam, HW, and CW valve openings, while the controlled variables are the CW flow, tank level, and temperature. The CSTH model can be classified as a semi-empirical model, combining first principles equations and algebraic equations derived from experimental data. The left-hand column of Figure 9 illustrates measurements of the tank's level, temperature, and CW flows acquired under normal operating conditions. To formulate a binary fault classification problem, we are considering three scenarios resulting from instrumentation faults and errors. These scenarios are as follows: i) an abrupt pulse change introduced into the level transmitter signals (Figure 9II), ii) a malfunction in the temperature controller characterized by a random change in the controller parameters (Figure 9III), and iii) a random sinusoidal noise introduced into the CW flow controller output signals (Figure 9IV). Table 1 provides a comprehensive overview of the simulated dataset. It is worth noting that this case study specifically addresses a binary classification problem, where discrete outputs are used: normal data inputs are represented by $Y = 0$ and faulty data inputs by $Y = 1$.

Table 1: CSTH data summary

Number of total samples (N)	8000
Training: validation: testing ratio	70:10:20
Number of variables (p)	3
Sampling frequency	1 sec
Signal length (L)	200
Number of classes	2
Class ratio	50:50 (balanced)

4.1.1 Simulation study setup

For model evaluation, we use a hold-out strategy. The entire CSTH dataset is split randomly into three balanced subsets: i) a training set for training the models, ii) a validation set for optimizing the models' hyperparameters, and iii) a testing set for testing the models on unseen data. We systematically explore different hyperparameter combinations using a random search to identify well-performing model configurations. This involves searching over a manually predefined search space, where various hyperparameter combina-

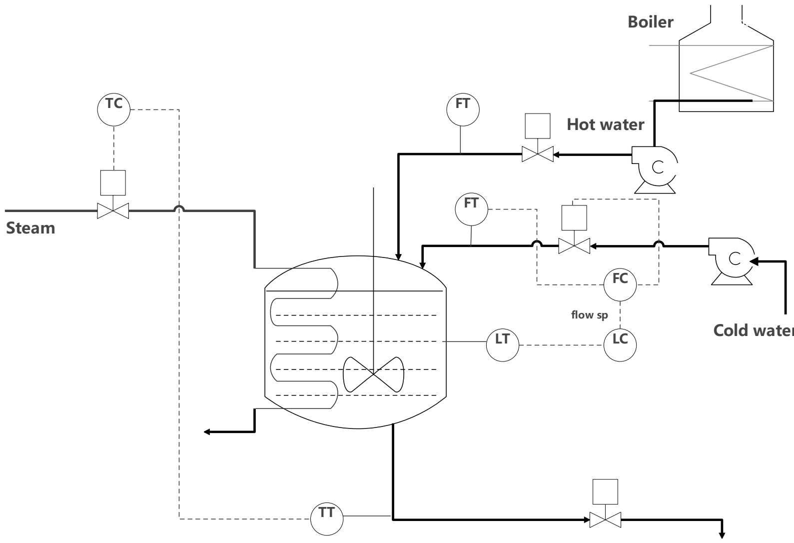

Figure 8: The CSTH feedback control system.

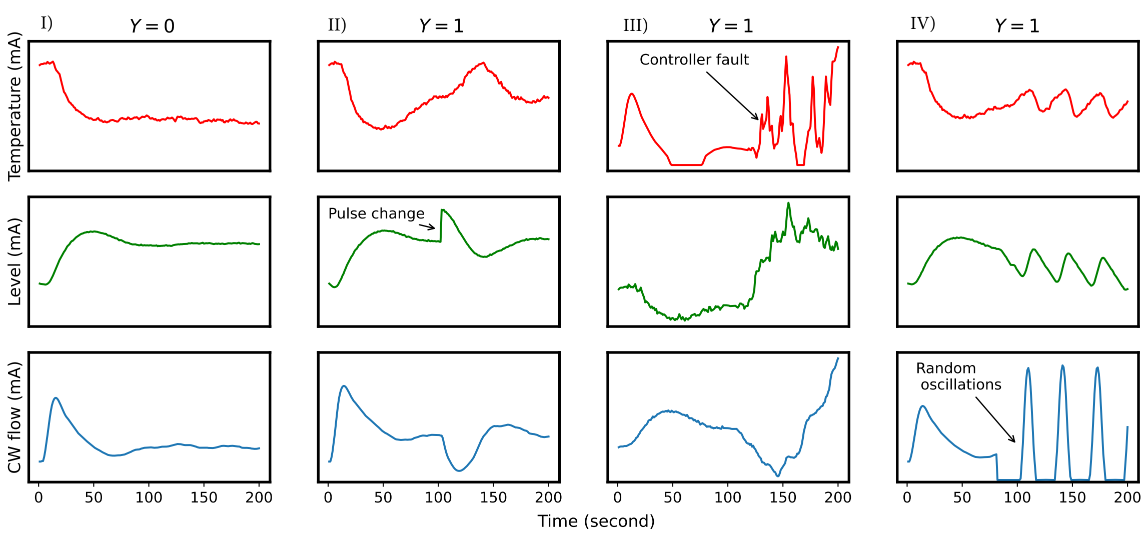

Figure 9: A visual comparison between a smooth operation ($Y = 0$) and a faulty operation ($Y = 1$).

tions are sampled. For each sampled set of hyperparameters, we train the models on the training set and estimate their performance using the validation set. After conducting multiple trials, which are manually set with consideration for the size of the hyperparameter search space and computational limitations, we select the models with the hyperparameter configurations that yield the best results on the validation set. These selected models are then retrained using the entire training set. To ensure effective model training, we implement an early stopping criterion that prevents overfitting of the data. Specifically, if the validation loss does not exhibit improvement for a consecutive number of epochs, the training process is halted. A patience value of 10 epochs is selected for the early stopping criterion. Subsequently, the retrained models are evaluated on the unseen testing set to provide reliable and unbiased performance metrics.

To assess the classification performance of our proposed visual analytics approach, we use carefully tuned implementations of the following models:

- **DTW + 1NN:** {implementation = fast DTW, max warping window = 400}
- **TSF:** {number of classifiers = 3, ensemble method = voting, number of trees = 200, minimum interval length = 3, number of attributes = 3 (mean, slope, and standard deviation)}
- **BOSS:** {number of classifiers = 3, ensemble method = voting, word size= 7, number of bins= 20, window size= 10, window step= 2}
- **LSTM:** {LSTM layers = 2, LSTM units = 32, recurrent activation function = hard sigmoid, hidden layers = 1, hidden neurons = 128, hidden activation function = ELU, batch normalization = True, dropout coefficient = 0.3, L_2 regularization penalty = 0.01, optimizer = RMSprop, learning rate = 0.001, batch size = 64}
- **GAF + CNN:** {GAF method = summation, convolution layers = 1, convolution kernels = 16, size of kernels = (3, 3), padding = "same", pooling = max, pool size = (2, 2), convolution activation function = SELU, hidden layers = 1, hidden neurons = 64, hidden activation function = ELU, batch normalization = True, L_2 regularization penalty = 0.01, optimizer = AdaGrad, learning rate = 0.001, batch size = 16}
- **RP + CNN:** {phase space dimensions $m = 2$, convolution layers = 2, convolution kernels = {8, 16}, size of kernels = (5, 5), padding = "same", pooling = max, pool size = (2, 2), convolution activation function = tanh, hidden layers = 1, hidden neurons = 16, hidden activation function = ReLU, batch normalization = True, L_2 regularization penalty = 0.01, optimizer = AdaGrad, learning rate = 0.001, batch size = 32}
- **Proposed:** {1D residual blocks = 1, 1D convolution kernels = 32, size of 1D kernels = 3, 2D convolution layers = 2, 2D convolution kernels = 16, size of 2D kernels = (3, 3), convolution activation function = SELU, hidden layers = 3, hidden neurons = 32, hidden activation function = ReLU, batch normalization = True, dropout coefficient = 0.2, L_2 regularization penalty = 0.0001, optimizer = SGD, learning rate = 0.001, batch size = 16}

4.1.2 Classification performance

The experimental results, summarized in Table 2, provide insights into the performance of each model on the final testing set. Six evaluation metrics are used to comprehensively evaluate the performance. Accuracy is calculated as the ratio of correctly predicted samples to the total number of testing samples. Precision measures the percentage of accurately predicted faulty samples out of all faulty predictions, while recall (i.e., true positive rate (TPR)) quantifies the proportion of correctly predicted faulty samples compared to the total number of faulty samples. The false positive rate (FPR) represents the ratio of false positive predictions to the total number of normal samples. Next, the F_1 score captures the harmonic mean of precision and recall, providing a balanced assessment of model performance. Finally, training time (TT) is used to compare models' computational complexities. The TT denotes the time required to train the model on the training set. Note that the reported TTs for deep learning models correspond to the time taken to complete 100 training epochs. To ensure a fair comparison, all models considered in this study are trained on an NVIDIA A100 GPU with 51GB of virtual RAM (VRAM). The proposed approach achieved the highest score in five out of the six key metrics, demonstrating its competitive classification performance and computational efficiency.

Table 2: Simulation results summary

	Accuracy	Precision	Recall	1-FPR	F_1	TT (min)
DTW+1NN	0.9569	0.9816	0.9315	0.9824	0.9559	112.15*
TSF	0.9775	0.9773	0.9773	0.9777	0.9773	2.80
BOSS	0.9413	0.9917	0.8904	0.9925	0.9383	**0.56**
LSTM	0.9338	0.9861	0.8804	0.9875	0.9303	370.11
GAF+CNN	0.9825	0.9936	0.9714	0.9937	0.9824	5.65**
RP+CNN	0.9788	0.9949	0.96264	0.9950	0.9785	9.12**
Proposed	**0.9881**	**0.9987**	**0.9776**	**0.9987**	**0.9880**	7.06

* TT corresponds to the time required to compute the DTW similarity matrix.
** TT includes the time required to encode time-series data into images.

4.1.3 Visual representations comparative analysis

In this subsection, we illustrate how our visual analytics approach reconciles performance and visual interpretability. We demonstrate how our approach enables process operators to visually validate and analyze the correlation between the model's predictions and the underlying process. While visual interpretability cannot be quantified by a metric, we use a qualitative approach to analyze the visual representations derived from GAF encoding, RP encoding, and our proposed approach.

Figure 10 displays two negative MTS samples extracted from the CSTH dataset and their corresponding GAF and RP representations. The GAF and RP encodings are visualized as RGB images, with red, green, and blue channels denoting temperature, level, and CW flow measurements, respectively. Analyzing the GAF representations, we identify distinct “L-shaped” patterns, which serve as indicators of changes in the system's set points. Moreover, we observe homogeneous patterns, represented by uniform boxes, in the upper-right section of the GAF images. These consistent patterns symbolize steady-state behavior, where the system maintains a stable operational regime over a certain period. The extent of this uniformity directly corresponds to the duration of stationarity. The presence of “L-shaped” transitions followed by homogeneous patterns in the GAF images reveals a distinct sequence in the underlying process: a change in set point followed by a period of stationarity. This sequence is a hallmark of normal operating regimes within the CSTH system. Contrastingly, the RP images appear mostly blank, indicating the absence of recurrent behaviors. In other words, the system lacks repeating patterns or periodic behaviors that RP is designed to highlight. Next, Figure 11 provides a different perspective as it showcases two positive MTS samples alongside their corresponding GAF and RP images. The presence of cyclicities within the system is reflected in the RP and GAF images, which exhibit distinctive periodic patterns. These patterns offer insights into the cyclic behavior of the underlying process, with the time distance between the patterns representing the

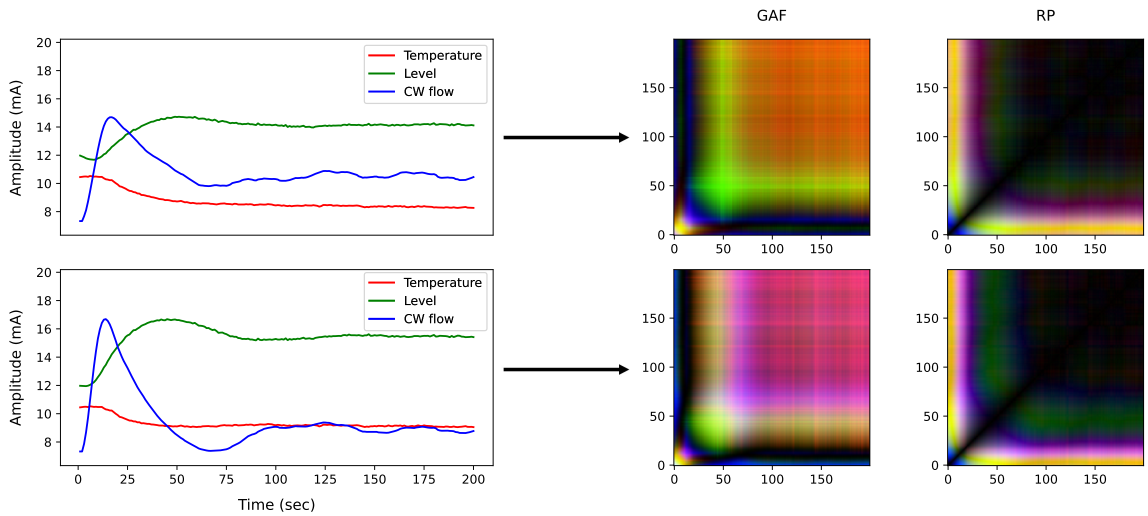

Figure 10: Two normal CSTH samples ($Y = 0$) alongside their corresponding GAF and RP encodings, revealing meaningful insights: i) A greenish-yellow (green + red) "L-shaped" pattern emerged around the 52-second mark in the upper GAF implies shifts in both level and temperature set points. Given the greater shift in the level set point (green) compared to the temperature set point (red), the resulting color leans more towards green. ii) The appearance of a reddish-white (red + green + blue) homogeneous box in the top GAF image represents stationarity in temperature, level, and CW flow measurement. Because temperature measurements exhibit greater stationarity, the red color is more dominant. iii) The bottom GAF image reveals a reddish-purple (red + blue) homogeneous box, indicating stable temperature and CW flow measurements. iv) Blank RP images denote the absence of recurrent patterns in the underlying system.

period of this recurring behavior. It is important to note that the construction of the simulated CSTH dataset emphasizes that periodic patterns are a strong indication of faulty operations. Thus, the periodicity observed in the RP and GAF images serves as a valuable indicator for identifying faulty operating regimes.

Figure 12 offers a visual comparison of the visual representations of three normal and three faulty MTS signals obtained using the proposed visual analytics framework. Unlike RP and GAF encodings, our proposed approach generates a single-channel matrix that can be visualized via color mapping. The color mapping assigns colors to matrix values, following a selected color map, which facilitates enhanced visual interpretation. The visual representations of normal MTS samples are visually distinct from those of faulty samples. Specifically, the visual representations for normal samples exhibit an even texture, whereas the visual representations for faulty samples display an irregular texture. This clear visual distinction is achieved through the application of 1D convolution operations. By applying 1D convolutional kernels, our approach effectively captures discriminative patterns present in the raw time-series data. As a result, normal and faulty MTS signals respond differently when convolved with these kernels. This differential response contributes to the observed visual distinctions between normal and faulty MTS samples in the visual representations produced by our framework.

4.1.4 Fault magnitude sensitivity analysis

While previous subsections have empirically demonstrated the effectiveness of the proposed approach in distinguishing normal and faulty MTS signals using distinctive image representations, a pivotal question

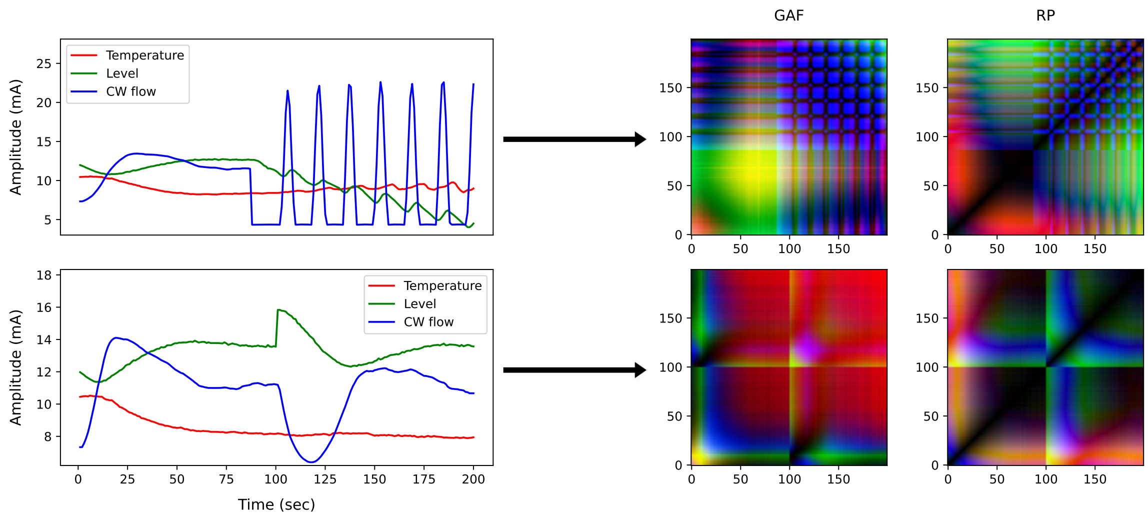

Figure 11: Two faulty CSTH signals ($Y = 1$) and their corresponding GAF and RP representations. Several observations and their qualitative interpretations can be made: i) The presence of cyclic behavior in the system results in distinct periodic patterns within the GAF and RP images. The temporal interval between these patterns corresponds directly to the oscillation period of the system. ii) Bright corners in the GAF images (e.g., the bright red upper corner in the lower GAF image) indicate shifts in trends—either decreasing or increasing.

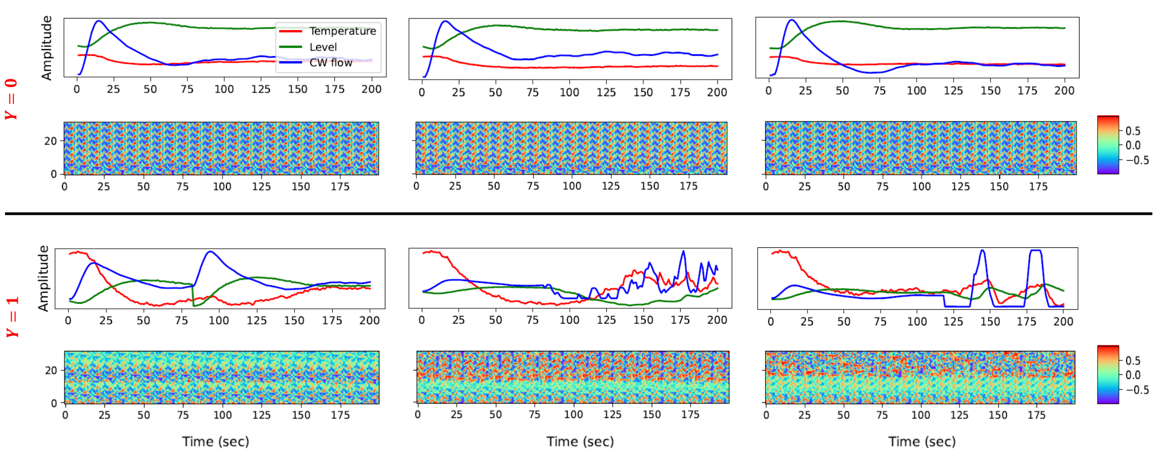

Figure 12: A visual comparison of the representations of three normal (top) and three faulty MTS signals (bottom) obtained using our proposed visual analytics framework. Homogenous texture indicates normal operating conditions, whereas uneven texture represents faulty operations. Normal and faulty signals respond to the learned 1D convolutional kernels differently. This dissimilarity in response leads to visually distinct representations between normal and faulty signals.

arises: Does the proposed approach have the requisite sensitivity to detect variations in fault magnitudes, specifically those of small ones? In this subsection, we explore the detection capabilities of the proposed approach across various fault sizes. The objective is to demonstrate that the proposed approach is not only able to detect substantial deviations (i.e., faults with a great magnitude) but also to capture minor faults within its image representations. Similar to subsection 4.1.3, we compare the proposed approach against the GAF and RP tools.

In pursuit of our objective, we introduce faults of varying magnitudes (small and large) into a normal CSTH sample and then assess how the proposed approach, GAF, and RP representations respond. Figure 13 displays the progression of GAF and RP encodings for different fault magnitudes alongside the underlying MTS in the time domain. The encodings are presented for a normal MTS, a faulty MTS with a small fault magnitude (equivalent to 5% of steady-state value), and a faulty MTS with a large fault magnitude (equivalent to 30% of steady-state value). Moreover, Figure 14 shows the visual representations of the same aforementioned MTS obtained using the proposed approach. The visual encodings generated for the normal MTS and the faulty MTS of large magnitude (30%) using GAF, RP, and the proposed approach exhibit perceptible disparities. However, the proposed approach better distinguishes between the normal MTS and the faulty MTS of a small magnitude (5%). Notably, for the proposed approach, the extent of textural irregularity in the visual representations directly corresponds to the magnitude of the fault.

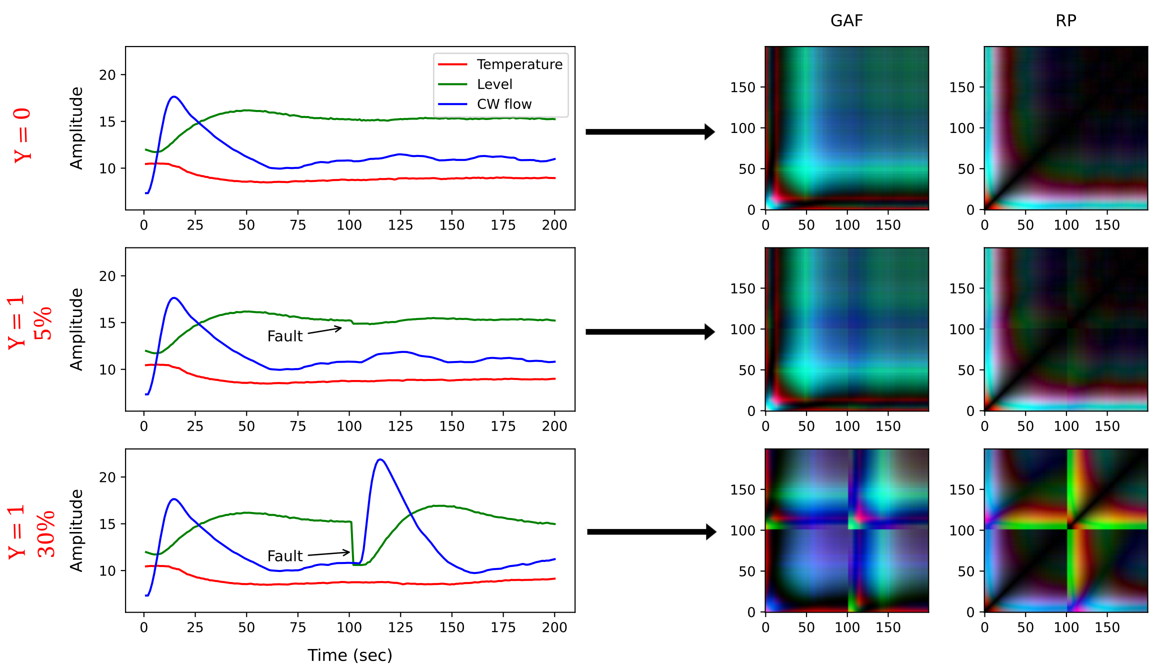

Figure 13: Responses of GAF and RP encodings across different fault magnitudes. Left: Original time-series plots. Middle: GAF encodings. Right: RP encodings. The visual encodings are shown for three scenarios: the top row shows a baseline normal CSTH sample, the middle row shows a faulty sample with a minor fault magnitude (5% deviation from steady-state value), and the bottom row represents a faulty sample with a large fault magnitude (30% deviation from steady-state value).

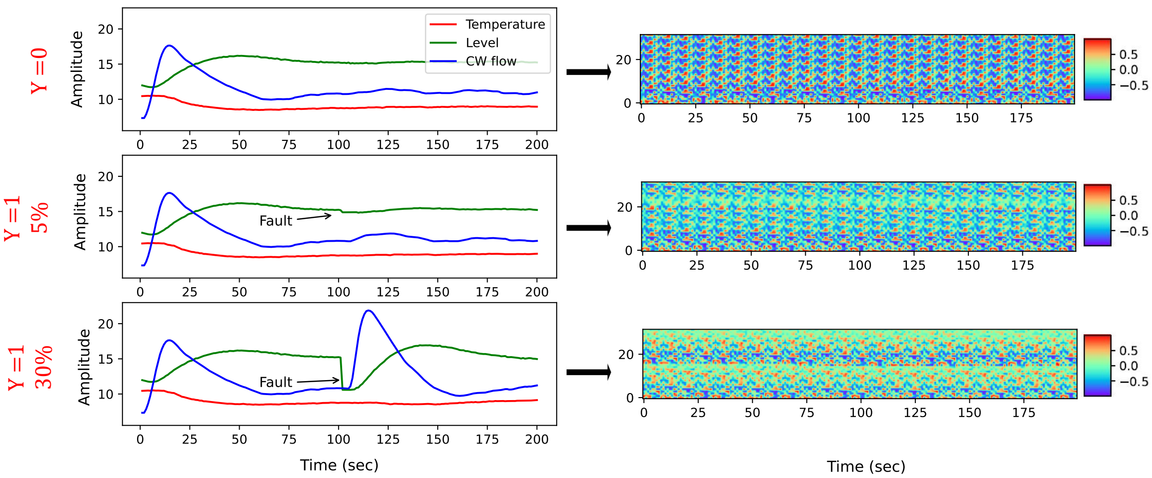

Figure 14: A visual comparison of the representations of a normal CSTH sample (top), a faulty CSTH sample of size 5% (middle), and a faulty CSTH sample of size 30% (bottom) obtained using our proposed visual analytics framework.

4.2 An industrial case study: the arc loss benchmark

The arc loss benchmark [60] was proposed as a real-world data benchmark for developing and validating data-driven process monitoring workflows. It provides a large-scale dataset obtained from an industrial pyrometallurgical smelting process. In this process, high-grade oxidized ore deposits are converted into refined base metals. Figure 15 illustrates the high-level pyrometallurgical processing. First, ore deposits extracted from open pits undergo grinding and drying operations using hammer mill flash dryers, resulting in fine ores with low moisture content. The dried ores are then dehydrated and deoxidized through a series of flash calciners and fluidized bed reducers. Subsequently, the processed ores are fed into a direct current electric arc furnace (DC EAF) unit to obtain base metals. The base metals are later processed by shotting and packaging units before being shipped to customers. An in-depth description of the process and its schematics can be found in Yousef et al. [60].

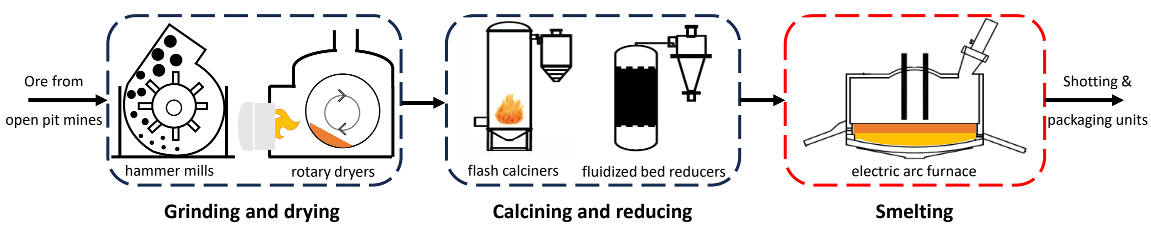

Figure 15: The broader pyrometallurgical processes.

The DC EAF unit, shown in Figure 16, is a high-temperature furnace that converts the electrical energy attained from the DC power supply into thermal energy by means of two plasma arcs. The two plasma arcs serve as the main heating component in the DC EAF unit, facilitating the separation of the base metals from other components of the ore (i.e., slag). Ensuring stable DC EAF operation is crucial for maximizing production efficiency and stability. However, the DC EAF unit faces a persistent fault known as *the arc loss fault*. The arc loss fault refers to the intermittent or sudden disruptions of the plasma arcs within the furnace.

These disruptions cause undesirable fluctuations in temperature and hinder the smelting efficiency, leading to potential production loss and variations in product quality. In this case study, the arc loss benchmark dataset is used to validate the proposed visual analytics framework in predicting the onset of arc loss in the DC EAF unit.

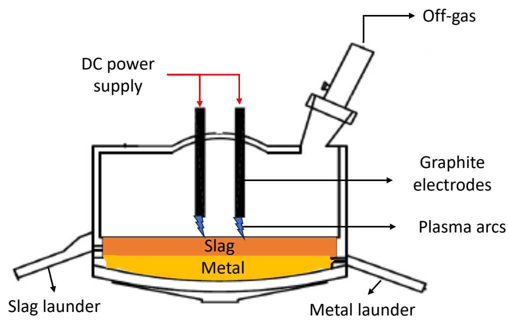

Figure 16: An illustration of the DC EAF unit.

The arc loss benchmark dataset comprises one year of high-frequency operating data collected from various pyrometallurgical process units. However, the raw data contains problematic artifacts, including outliers, irrelevant data, missing data, and class imbalance. To address the aforementioned issues, we processed the raw data, resulting in a balanced, clean dataset containing 3226 MTS samples. Each sample corresponds to 55 consecutive minutes of 96 different process measurements taken during either a smooth or faulty operating period (i.e., arc loss period). In the interest of brevity, we refrain from detailing the data preprocessing steps in this paper. Instead, we refer readers to [61] for comprehensive insights into the preprocessing techniques performed. Table 3 provides an overview of the processed arc loss data. The primary objective of this case study is to validate and compare the performance of our visual analytics workflow in accurately predicting whether a given MTS signal belongs to a smooth ($Y = 0$) or a faulty ($Y = 1$) operation.

Table 3: The processed arc loss data summary

Number of total samples (N)	3226
Training: validation: testing ratio	70:10:20
Number of variables (p)	96
Sampling frequency	3 sec
Signal length (L)	1100
Number of classes	2
Class ratio	50:50 (balanced)

4.2.1 Classification performance

We adopt a parallel approach to that used in the simulation case study, using consistent strategies for model evaluation and hyperparameter optimization. Notably, in the context of the industrial case study, we encounter a substantial difference in data scale compared to the simulated scenario. This difference requires a tailored preprocessing step to reduce data dimensionality, mitigating computational complexity. Specifically, the computational demands of imaging tools (i.e., GAF and RP) are directly proportional to the length of the time-series data, following a power-law relationship. Given the limitations of our hardware in terms of memory and processing capacity, imaging a full-length MTS sample extracted from the arc loss data becomes challenging. To tackle this challenge, we incorporate techniques like piecewise aggregate approximation (PAA) [62] prior to encoding MTS data into GAF and RP images. PAA segments the time-series into non-overlapping windows and computes the mean value for each window, reducing the temporal complexity.

We consider the following models, accompanied by their respective tuned implementations:

- **DTW + 1NN:** {implementation = fast DTW, max warping window = 1000}
- **TSF:** {number of classifiers = 96, ensemble method = voting, number of trees = 300, minimum interval length = 3, number of attributes = 3 (mean, slope, and standard deviation)}
- **BOSS:** {number of classifiers = 96, ensemble method = voting, word size= 5, number of bins= 20, window size= 10, window step= 3}
- **LSTM:** {LSTM layers = 3, LSTM units = 16, recurrent activation function = hard sigmoid, hidden layers = 2, hidden neurons = 16, hidden activation function = SELU, batch normalization = True, dropout coefficient = 0.1, L_2 regularization penalty = 0.01, optimizer = Adam, learning rate = 0.001, batch size = 32}
- **GAF + CNN:** {PAA window size = 10, GAF method = summation, convolution layers = 2, convolution kernels = 32, size of kernels = (3, 3), padding = "same", pooling = max, pool size = (2, 2), convolution activation function = ELU, hidden layers = 2, hidden neurons = 16, hidden activation function = ReLU, batch normalization = True, L_2 regularization penalty = 0.001, optimizer = AdaMax, learning rate = 0.001, batch size = 32}
- **RP + CNN:** {PAA window size = 10, RP phase space dimensions $m = 2$, convolution layers = 3, convolution kernels = 64, size of kernels = (5, 5), padding = "same", pooling = max, pool size = (2, 2), convolution activation function = tanh, hidden layers = 1, hidden neurons = 32, hidden activation function = ReLU, batch normalization = True, L_2 regularization penalty = 0.1, optimizer = RMSprop, learning rate = 0.001, batch size = 32}
- **Proposed:** {1D residual blocks = 3, 1D convolution kernels = {64, 128, 256}, size of 1D kernels = 3, 2D convolution layers = 2, 2D convolution kernels = {16, 32}, size of 2D kernels = (3, 3), convolution activation function = ReLU, hidden layers = 3, hidden neurons = 16, hidden activation function = SELU, batch normalization = True, dropout coefficient = 0.3, L_2 regularization penalty = 0.1, optimizer = Adam, learning rate = 0.0001, batch size = 16}

Table 4 summarizes the classification performance of each model on the held-out testing set. The proposed approach achieved the highest score in five out of six key performance metrics, whereas GAF + CNN was the best configuration with respect to recall. Notably, non-deep learning models, such as DTW + 1NN, TSF, and BOSS, exhibit significantly longer training times. This prolonged training duration can be attributed to the high dimensionality of the data and the intricate computations required by these models. Additionally, LSTM, owing to its recurrent nature and the long signal length (i.e., 1100-time steps for each signal), also incurs a substantial training time. On the other hand, the performance of GAF + CNN and RP + CNN models has been adversely impacted by the use of PAA for data preprocessing. PAA introduces a loss of temporal information, which is essential for these convolutional models to capture meaningful patterns. As a result, their classification performance, while still competitive, is somewhat compromised. In contrast, our proposed approach scales efficiently, with relatively short training times. This efficiency is attributed to the use of 1D convolutions, which are computationally efficient and well-suited for processing large-scale industrial time-series data (e.g., the arc loss benchmark dataset). This scalability makes the proposed approach an appealing choice for real-world applications where computational resources are limited.

Table 4: Industrial case study results summary

	Accuracy	Precision	Recall	1-FPR	F_1	TT (min)
DTW+1NN	0.6388	0.6486	0.6227	0.6552	0.6354	1372.27*
TSF	0.6977	0.8499	0.6787	0.7389	0.7547	79.62
BOSS	0.6682	0.6474	0.7546	0.5799	0.6969	15.68
LSTM	0.7519	0.7280	0.8129	0.6897	0.7681	203.33
GAF+CNN	0.6915	0.6377	**0.9018**	0.4765	0.7471	8.05**
RP+CNN	0.7147	0.6830	0.8129	0.6144	0.7423	9.12**
Proposed	**0.7721**	**0.8650**	0.7325	**0.8308**	**0.7932**	**3.33**

* TT corresponds to the time required to compute the DTW similarity matrix.
** TT includes the time required to encode time-series data into images.

4.2.2 Visual representations comparative analysis

Similar to the simulation case study, in this subsection, we analyze the visual patterns within the visual representations obtained using GAF, RP, and our proposed approach. However, this industrial case study offers a distinctive perspective as it focuses on how the visual interpretability of these representations is influenced by the substantial differences in data dimensionality compared to the simulation case.

Firstly, both GAF and RP encodings operate on a single dimension, specifically designed for univariate time-series data. Consequently, when applied to a multivariate system like the arc loss benchmark, with 96 process variables, these encoding techniques generate visual representations comprising 96 channels. The enormity of this dimensionality poses a significant challenge for visualization, as conventional computers struggle to render images with more than three channels effectively. To overcome this limitation, we carefully select three pivotal variables for imaging: total power (TP), furnace feed (FF), and furnace off-gas temperature (FOGET). This selection is based on process knowledge, as these variables exhibit strong correlations that are important in predicting arc loss faults. Figure 17 shows the GAF and RP encodings as RGB images for FOGET, FF, and TP measurements recorded during two distinct normal operating periods. As previously discussed, we use the PAA method to reduce the raw time-series size, mitigating the computational complexity associated with these imaging tools. Specifically, we use a PAA window size of 10, reducing the length of the time-series by a factor of 10. Figure 18 further illustrates the application of GAF and RP encodings as RGB images, this time focusing on FOGET, FF, and TP measurements during two distinct faulty operating periods.

While these encodings reveal valuable visual patterns, it is important to acknowledge their limitations, particularly when applied to large-scale industrial datasets. One major challenge arises from the process of selecting the optimal set of three variables for imaging. This task can be time-consuming and labor-intensive, particularly in high-dimensional systems where process knowledge is limited, such as in industrial settings. Additionally, the use of the PAA method, although essential for managing the computational demands of these imaging techniques, introduces its own set of limitations. PAA inherently smooths the original time-series data, which can have unintended consequences. For instance, singular abnormal outlier measurements may get averaged out in the reduced time-series. Consequently, the resulting RP or GAF images may not capture these anomalous data points, potentially affecting the accuracy of FDD. Furthermore, it is important

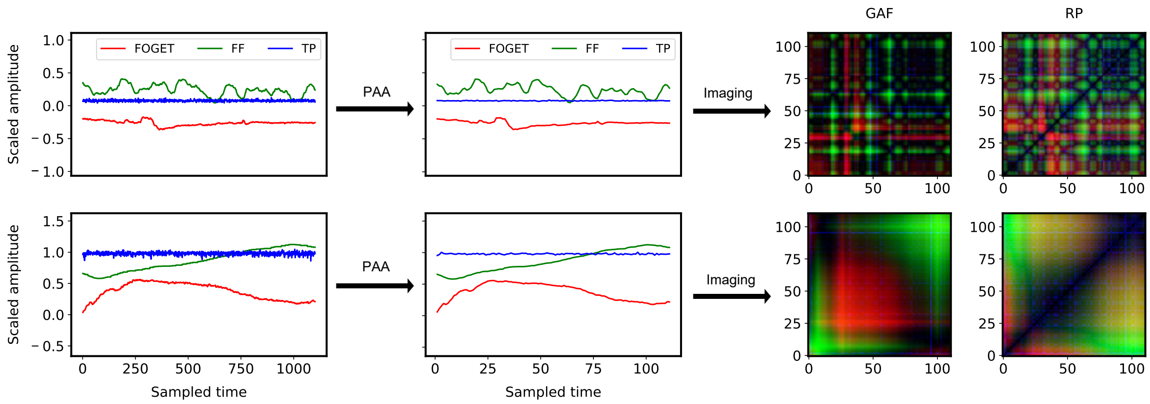

Figure 17: An illustration of two normal arc loss samples ($Y = 0$) and their step-by-step encoding into GAF and RP images. Firstly, a set of three variables (FOGET, TP, and FF) was carefully chosen for imaging. Subsequently, the PAA method is used to reduce the dimensionality of the raw signals, using a window size of 10. Finally, the RP and GAF encodings are applied to the reduced time-series. The resulting GAF and RP representations are displayed as RGB images, where the red, green, and blue channels represent FOGET, FF, and TP measurements, respectively. Qualitative interpretations are as follows: In the first sample, uniform patterns indicate stationary measurements, while red vertical/horizontal lines mark the time length in which a disturbance in FOGET measurements is introduced. In the second sample, vibrant green and red corners highlight increasing and decreasing trends in FF and FOGET measurements, respectively.

to note that the time distance represented in the GAF and RP encodings corresponds to the reduced time-series length resulting from PAA. In other words, each pixel in the GAF and RP representations represents the average of two-time windows of size 10 in the underlying system. This temporal aggregation may mask certain transient dynamics that occur within shorter time frames, which can be critical for understanding the dynamics of industrial processes, especially during fault conditions.

Next, Figure 19 compares two normal arc loss samples ($Y = 0$) along with their visual representations obtained using the proposed approach versus two faulty arc loss samples ($Y = 1$) and their visual representations. The size of the visual representation is $256 \times 1100 \times 1$. In other words, the width of the one-channeled image corresponds to the length of the input signals (i.e., 1100-time steps), while the height represents the number of representation dimensions, which equals the number of 1D kernels in the last 1D residual block. Although the global texture is differentiable between the normal and faulty visual representations, the low-level details are not as pronounced as in RP and GAF representations due to the high resolution of the images. As shown in Figure 19, the normal visual representations exhibit regular patterns, while the faulty visual representations display irregular textures.

The proposed approach offers several distinctive advantages when compared to existing imaging tools like RP and GAF. Firstly, the proposed method operates directly on MTS signals and produces single-channel visual representations, simplifying the interpretation process. This simplicity in representation can enhance the ease of understanding and analysis. Furthermore, the proposed approach handles MTS signals of any size directly without the need for dimensionality reduction techniques such as PAA. This advantage stems from the computational efficiency of 1D convolution operations, which scale linearly with the input size. As a result, the proposed approach can efficiently handle large-scale industrial datasets without the preprocessing steps that might introduce data loss and compromise temporal resolution.

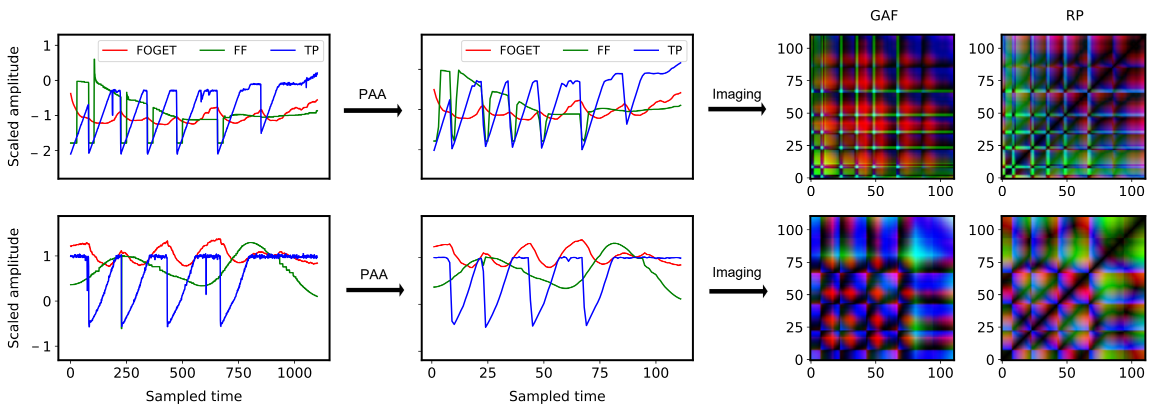

Figure 18: GAF and RP encodings presented as RGB images for FOGET, FF, and TP measurements recorded during two different faulty operating periods ($Y = 1$). The presence of a periodic checkerboard structure in both GAF and RP images suggests underlying fluctuations in FOGET, TP, and FF measurements. Unexpected power drops, leading to pronounced variations in FF and FOGET measurements, define an arc loss event.

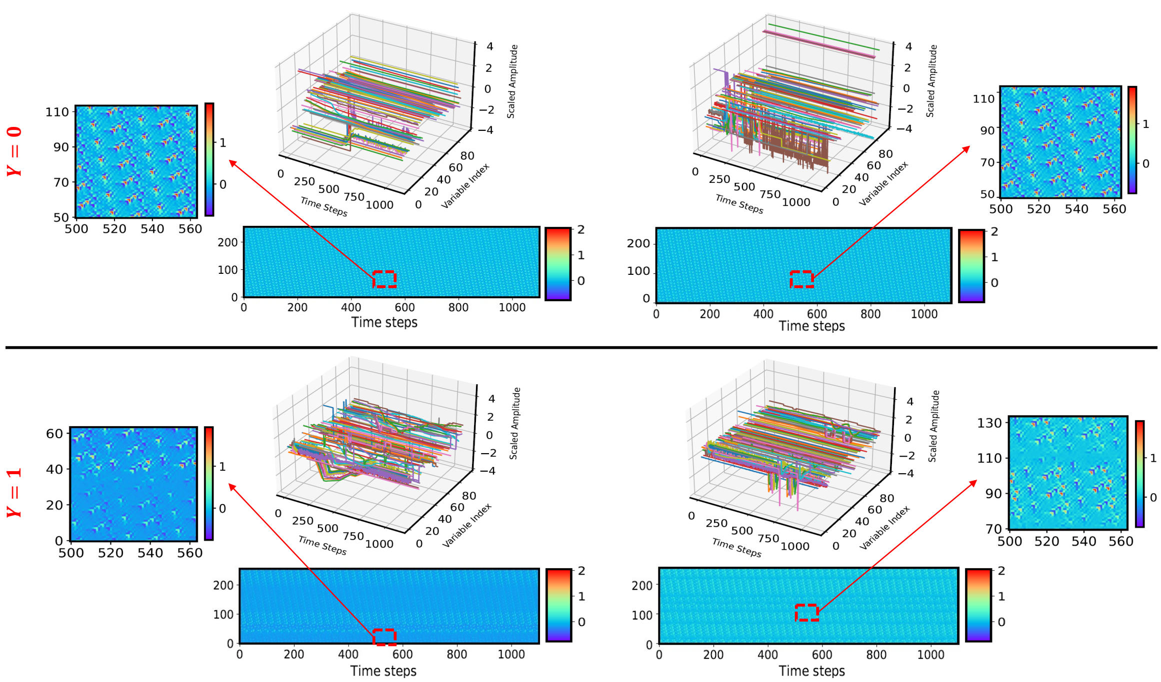

Figure 19: A visual comparison of two normal ($Y = 0$) and two faulty ($Y = 1$) arc loss samples using our proposed visual analytics framework. Normal samples exhibit homogeneous textures, while faulty samples display unsmooth textures. Zoomed-in patches highlight local patterns.

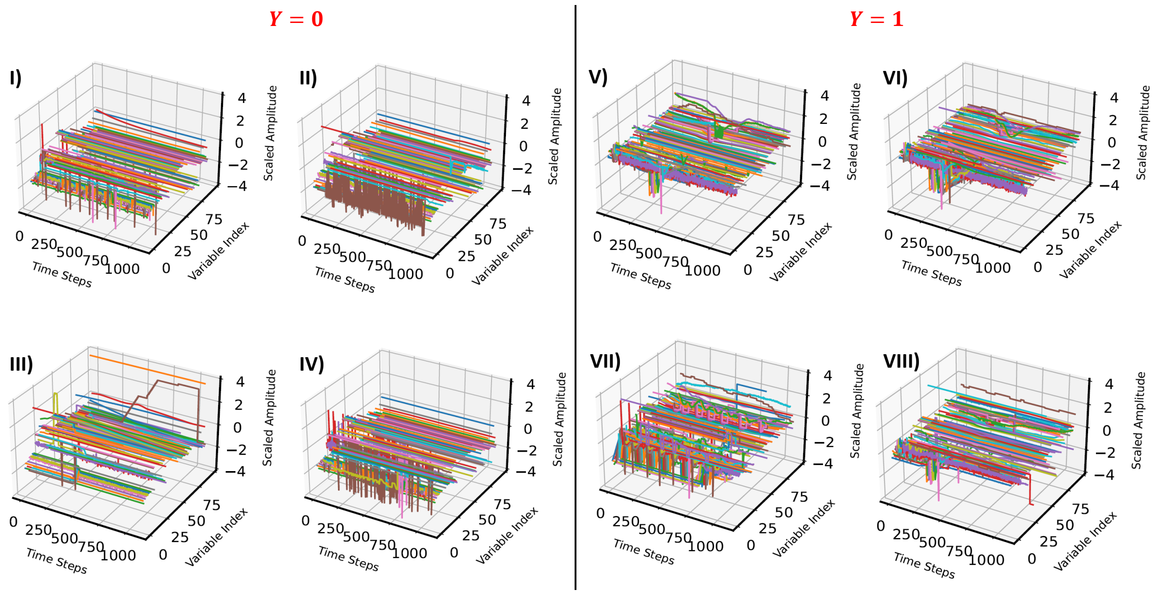

Figure 20: time domain data comparison: normal ($Y = 0$) vs. faulty ($Y = 1$) operating conditions. Readers are referred to Figure 21 for a more detailed view of some of the most important features, as identified by process experts.

4.2.3 Why visual analytics for industrial process monitoring?

In the realm of process monitoring, visual analytics involves transforming historical process data in the form of MTS into visually meaningful representations. The visual nature of the representations allows process experts to manually analyze patterns and textures within the images, facilitating easier interpretation and identification of significant insights. This is done by enabling process experts to develop intuition in relating different patterns in the visual representations to distinct process operating modes. In this work, we propose a visual analytics framework based on supervised learning, which converts MTS into pixelated images with patterns directly associated with various operating conditions.

Once the proposed network is trained on the training set, the trained stages 1 and 2 of the network can be used to convert input MTS signals into two-dimensional images. Figure 20 shows a visual comparison of four normal ($Y = 0$) and four faulty ($Y = 1$) arc loss samples in the time domain, while Figure 22 illustrates their corresponding visual representations obtained using the proposed network. In the time domain, the data from both normal and faulty operations may appear as intricate, overlapping patterns, making it difficult for process operators to discern meaningful insights. However, when using the visual analytics approach, the corresponding images reveal a stark contrast, revealing the underlying nature of the operating conditions. During normal operation, these images exhibit regular and smooth patterns, reflecting the consistent and expected behavior of the process. In contrast, under faulty conditions, the images display a distinctly unsmooth texture. This intuitive visual distinction allows process operators to quickly identify and understand the operating condition of the data without the need for extensive data analysis. The integration of human visual perception and computer vision algorithms in visual analytics offers a powerful tool for enhancing process monitoring and responding effectively to the challenges posed by the era of big data in process industries.

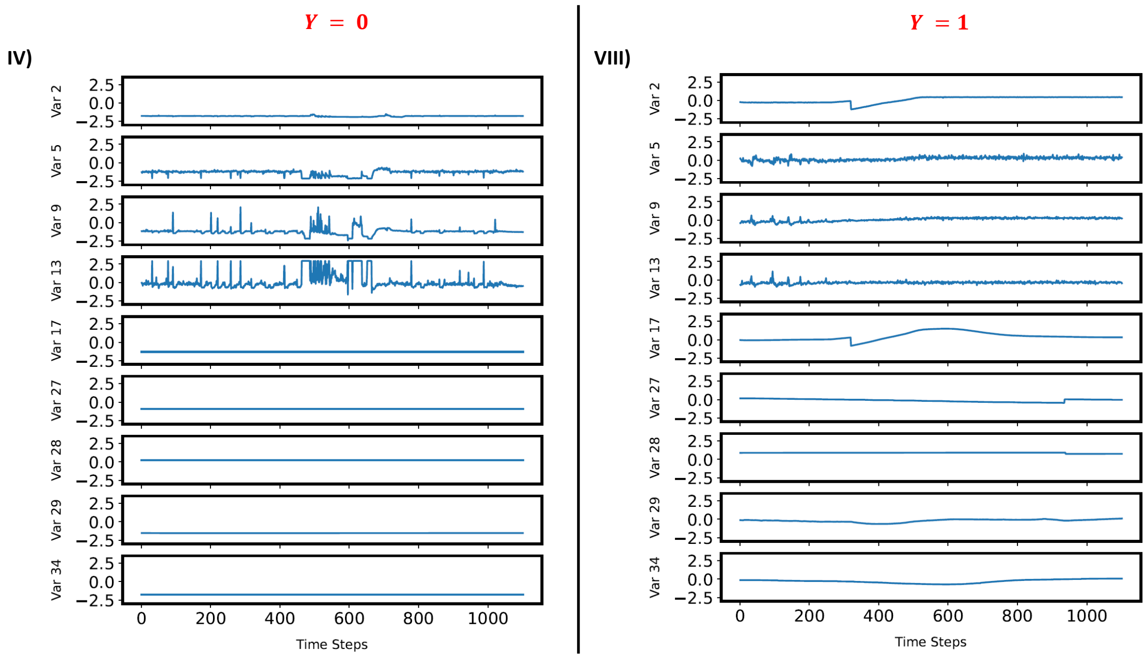

Figure 21: A visual demonstration of nine process variable measurements for normal ($Y = 0$) and faulty ($Y = 1$) operating conditions. Discriminative patterns are non-evident in the time domain. Latin numbers match Figure 20.

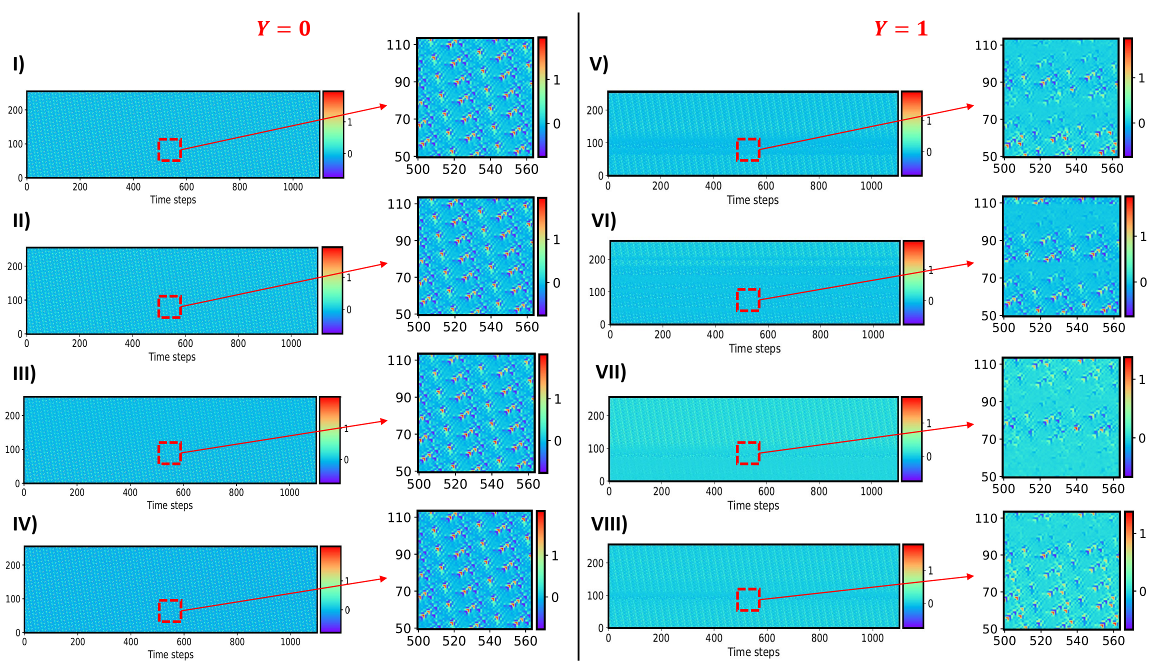

Figure 22: Visual representations, obtained using the proposed approach, comparison: normal ($Y = 0$) vs. faulty ($Y = 1$) operating conditions. Latin numbers match Figure 20.

5 Conclusion

This paper introduces a new paradigm for process monitoring called visual analytics. Visual analytics framework provides a powerful solution for industrial process monitoring by integrating the human visual system and computer vision algorithms. In this context, we propose a new end-to-end visual analytics pipeline for industrial fault detection, using both 1D and 2D convolution operations. Our approach begins with a series of 1D convolutions to capture relevant temporal information from the input MTS data. Subsequently, the extracted features are transformed into a 2D matrix, facilitating analysis and interpretation in the image domain, which proves to be more intuitive than the traditional time domain. By leveraging 2D convolution operations, our framework enables the network to visually recognize and classify these extracted features. To validate the effectiveness and interpretability of our approach, we conduct a simulated case study using the CSTH benchmark and an industrial case study using the arc loss benchmark. Experimental results demonstrate the superiority of our proposed visual analytics approach over state-of-the-art algorithms, providing not only improved performance but also meaningful and informative visual representations that enhance interpretability. Future works include examining the application of visual analytics in multiple fault scenarios. Furthermore, we plan to investigate the potential of visual analytics to determine the time of fault occurrence, which is an important aspect of process monitoring and FDD.

Acknowledgement

We gratefully acknowledge the financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] V. Alcácer and V. Cruz-Machado, "Scanning the industry 4.0: A literature review on technologies for manufacturing systems," *Engineering Science and Technology, an International Journal*, vol. 22, no. 3, pp. 899–919, 2019, ISSN: 2215-0986.

[2] F. Yang and S. Gu, "Industry 4.0, a revolution that requires technology and national strategies," *Complex & Intelligent Systems*, vol. 7, pp. 1311–1325, 2021.

[3] C. Li, Y. Chen, and Y. Shang, "A review of industrial big data for decision making in intelligent manufacturing," *Engineering Science and Technology, an International Journal*, vol. 29, p. 101 021, 2022, ISSN: 2215-0986. DOI: `https://doi.org/10.1016/j.jestch.2021.06.001`.

[4] R. B. Gopaluni *et al.*, "Modern machine learning tools for monitoring and control of industrial processes: A survey," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 218–229, 2020, 21st IFAC World Congress, ISSN: 2405-8963. DOI: `https://doi.org/10.1016/j.ifacol.2020.12.126`.

[5] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: A review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, Mar. 2019. DOI: `10.1007/s10618-019-00619-1`.

[6] P. Esling and C. Agon, "Time-series data mining," *ACM Computing Surveys*, vol. 45, no. 1, p. 12, 2012. DOI: `10.1145/2379776.2379788`.

[7] Y.-J. Park, S.-K. S. Fan, and C.-Y. Hsu, "A review on fault detection and process diagnostics in industrial processes," *Processes*, vol. 8, no. 9, p. 1123, Sep. 2020, ISSN: 2227-9717. DOI: 10.3390/pr8091123.

[8] J. Faouzi, "Time Series Classification: A review of Algorithms and Implementations," in *Machine Learning (Emerging Trends and Applications)*, K. Kotecha, Ed., Proud Pen, 2022.

[9] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 31, pp. 606–660, May 2017. DOI: 10.1007/s10618-016-0483-9.

[10] Y.-H. Lee, C.-P. Wei, T.-H. Cheng, and C.-T. Yang, "Nearest-neighbor-based approach to time-series classification," *Decision Support Systems*, vol. 53, no. 1, pp. 207–217, 2012, ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2011.12.014.

[11] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," May 2017, pp. 1578–1585. DOI: 10.1109/IJCNN.2017.7966039.

[12] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *Web-Age Information Management*, Springer International Publishing, 2014, pp. 298–310, ISBN: 978-3-319-08010-9.

[13] A. Dempster, F. Petitjean, and G. I. Webb, "ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1454–1495, Jul. 2020. DOI: 10.1007/s10618-020-00701-z.

[14] I. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions," *SN Computer Science*, vol. 2, p. 420, Aug. 2021. DOI: 10.1007/s42979-021-00815-1.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[16] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery Data Mining*, ser. KDD '21, Virtual Event, Singapore: Association for Computing Machinery, 2021, pp. 2114–2124, ISBN: 9781450383325. DOI: 10.1145/3447548.3467401.

[17] H. Ismail Fawaz *et al.*, "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, pp. 1–27, Nov. 2020. DOI: 10.1007/s10618-020-00710-y.

[18] S. Jiang and V. M. Zavala, "Convolutional neural nets in chemical engineering: Foundations, computations, and applications," *AIChE Journal*, vol. 67, no. 9, e17282, 2021. DOI: https://doi.org/10.1002/aic.17282.

[19] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. Zhao, "Time series classification using multi-channels deep convolutional neural networks," *WAIM 2014. LNCS*, vol. 8485, pp. 298–310, Jan. 2014.

[20] X. Yuan, S. Qi, Y. A. Shardt, Y. Wang, C. Yang, and W. Gui, "Soft sensor model for dynamic processes based on multichannel convolutional neural network," *Chemometrics and Intelligent Laboratory Systems*, vol. 203, p. 104 050, 2020, ISSN: 0169-7439. DOI: `https://doi.org/10.1016/j.chemolab.2020.104050`.

[21] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017. DOI: `10.21629/JSEE.2017.01.18`.

[22] K. Fauvel, T. Lin, V. Masson, É. Fromont, and A. Termier, "Xcm: An explainable convolutional neural network for multivariate time series classification," *Mathematics*, vol. 9, no. 23, p. 3137, Dec. 2021, ISSN: 2227-7390. DOI: `10.3390/math9233137`.

[23] L. Brito, G. A. Susto, J. Brito, and M. Duarte, "An explainable artificial intelligence approach for unsupervised fault detection and diagnosis in rotating machinery," *Mechanical Systems and Signal Processing*, vol. 163, p. 108 105, Jan. 2022. DOI: `10.1016/j.ymssp.2021.108105`.

[24] J. Grezmak, P. Wang, C. Sun, and R. X. Gao, "Explainable convolutional neural network for gearbox fault diagnosis," *Procedia CIRP*, vol. 80, pp. 476–481, 2019, 26th CIRP Conference on Life Cycle Engineering (LCE) Purdue University, West Lafayette, IN, USA May 7-9, 2019, ISSN: 2212-8271. DOI: `https://doi.org/10.1016/j.procir.2018.12.008`.

[25] I. Yousef, S. L. Shah, and R. B. Gopaluni, "Visual analytics: A new paradigm for process monitoring," *IFAC-PapersOnLine*, vol. 55, no. 7, pp. 376–383, 2022, 13th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems DYCOPS 2022, ISSN: 2405-8963. DOI: `https://doi.org/10.1016/j.ifacol.2022.07.473`.

[26] A. Abanda, U. Mori, and J. Lozano, "A review on distance based time series classification," *Data Mining and Knowledge Discovery*, vol. 33, pp. 378–412, Mar. 2019. DOI: `10.1007/s10618-018-0596-4`.

[27] J. Lines, S. Taylor, and A. Bagnall, "Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 1041–1046. DOI: `10.1109/ICDM.2016.0133`.

[28] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978. DOI: `10.1109/TASSP.1978.1163055`.

[29] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," English, *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 565–592, May 2015, ISSN: 1384-5810. DOI: `10.1007/s10618-014-0361-2`.

[30] Y. Lei and Z. Wu, "Time series classification based on statistical features," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, p. 46, Feb. 2020. DOI: `10.1186/s13638-020-1661-4`.

[31] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Information Sciences*, vol. 239, pp. 142–153, 2013, ISSN: 0020-0255. DOI: `https://doi.org/10.1016/j.ins.2013.02.030`.

[32] J. Large, A. Bagnall, S. Malinowski, and R. Tavenard, "On time series classification with dictionary-based classifiers," *Intelligent Data Analysis*, vol. 23, pp. 1073–1089, Oct. 2019. DOI: 10.3233/IDA-184333.

[33] P. Schäfer, "The boss is concerned with time series classification in the presence of noise," *Data Mining and Knowledge Discovery*, vol. 29, pp. 1505–1530, Nov. 2015. DOI: 10.1007/s10618-014-0377-7.

[34] P. Schäfer and M. Högqvist, "Sfa: A symbolic fourier approximation and index for similarity search in high dimensional datasets," in *Proceedings of the 15th International Conference on Extending Database Technology*, ser. EDBT '12, Berlin, Germany: Association for Computing Machinery, 2012, pp. 516–527, ISBN: 9781450307901. DOI: 10.1145/2247596.2247656.

[35] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015, ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2014.09.003.

[36] N. Papernot and P. McDaniel, *Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning*, 2018. arXiv: 1803.04765 [cs.LG].

[37] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132 306, Mar. 2020. DOI: 10.1016/j.physd.2019.132306.

[38] R. M. Schmidt, *Recurrent neural networks (rnns): A gentle introduction and overview*, 2019. arXiv: 1912.05911 [cs.LG].

[39] R. Pascanu, T. Mikolov, and Y. Bengio, *On the difficulty of training recurrent neural networks*, 2013. arXiv: 1211.5063 [cs.LG].

[40] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[41] G. Van Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5929–5955, Dec. 2020, ISSN: 0269-2821. DOI: 10.1007/s10462-020-09838-1.

[42] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, 2019. DOI: 10.1162/neco_a_01199.

[43] R. Yang, M. Huang, Q. Lu, and M. Zhong, "Rotating machinery fault diagnosis using long-short-term memory recurrent neural network," *IFAC-PapersOnLine*, vol. 51, no. 24, pp. 228–232, 2018, 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2018, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2018.09.582.

[44] S. Belagoune, N. Bali, A. Bakdi, B. Baadji, and K. Atif, "Deep learning through lstm classification and regression for transmission line fault detection, diagnosis and location in large-scale multi-machine power systems," *Measurement*, vol. 177, p. 109 330, 2021, ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2021.109330.

[45] R. Yamashita, M. Nishio, R. Do, and K. Togashi, "Convolutional neural networks: An overview and application in radiology," *Insights into Imaging*, vol. 9, pp. 611–629, Jun. 2018. DOI: 10.1007/s13244-018-0639-9.

[46] A. Ajit, K. Acharya, and A. Samanta, "A review of convolutional neural networks," in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, 2020, pp. 1–5. DOI: 10.1109/ic-ETITE47903.2020.049.

[47] F. Li *et al.*, "Feature extraction and classification of heart sound using 1D convolutional neural networks," *EURASIP Journal on Applied Signal Processing*, vol. 2019, no. 1, 59, p. 59, 2019. DOI: 10.1186/s13634-019-0651-3.

[48] S. Abdoli, P. Cardinal, and A. Lameiras Koerich, "End-to-end environmental sound classification using a 1d convolutional neural network," *Expert Systems with Applications*, vol. 136, pp. 252–263, 2019, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2019.06.040.

[49] C. Szegedy *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[51] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732. DOI: 10.1109/CVPR.2014.223.

[52] K. O'Shea and R. Nash, *An introduction to convolutional neural networks*, 2015. arXiv: 1511.08458 [cs.NE].

[53] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1d convolutional neural networks and applications: A survey," *Mechanical Systems and Signal Processing*, vol. 151, p. 107 398, 2021, ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2020.107398.

[54] Z. Wang and T. Oates, *Imaging time-series to improve classification and imputation*, 2015. arXiv: 1506.00327 [cs.LG].

[55] J.-P. Eckmann, S. O. Kamphorst, and D. Ruelle, "Recurrence plots of dynamical systems," *Europhysics Letters (EPL)*, vol. 4, no. 9, pp. 973–977, Nov. 1987.

[56] V. M. Souza, D. F. Silva, and G. E. Batista, "Extracting texture features for time series classification," in *2014 22nd International Conference on Pattern Recognition*, 2014, pp. 1425–1430.

[57] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012.

[58] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[59] N. F. Thornhill, S. C. Patwardhan, and S. L. Shah, "A continuous stirred tank heater simulation model with applications," *Journal of Process Control*, vol. 18, no. 3, pp. 347–360, 2008, Festschrift honouring Professor Dale Seborg, ISSN: 0959-1524.

[60] I. Yousef, L. D. Rippon, C. Prévost, S. L. Shah, and R. B. Gopaluni, "The arc loss challenge: A novel industrial benchmark for process analytics and machine learning," *Journal of Process Control*, vol. 128, p. 103 023, 2023, ISSN: 0959-1524. DOI: `https://doi.org/10.1016/j.jprocont.2023.103023`.

[61] L. Rippon *et al.*, "Representation learning and predictive classification: Application with an electric arc furnace," *Computers Chemical Engineering*, vol. 150, p. 107 304, 2021, ISSN: 0098-1354. DOI: `https://doi.org/10.1016/j.compchemeng.2021.107304`.

[62] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and Information Systems*, vol. 3, no. 3, pp. 263–286, Aug. 2001, ISSN: 0219-1377. DOI: `10.1007/PL00011669`.