machine learning & knowledge extraction

MDPI

Article

An Attention-Based ConvLSTM Autoencoder with Dynamic Thresholding for Unsupervised Anomaly Detection in Multivariate Time Series

Tareq Tayeh [1,*], Sulaiman Aburakhia [1], Ryan Myers [2] and Abdallah Shami [1]

[1] ECE Department, Western University, London, ON N6A 3K7, Canada; saburakh@uwo.ca (S.A.); abdallah.shami@uwo.ca (A.S.)
[2] National Research Council Canada, London, ON N6G 4X8, Canada; ryan.myers@nrc-cnrc.gc.ca
* Correspondence: ttayeh@uwo.ca

Abstract: As a substantial amount of multivariate time series data is being produced by the complex systems in smart manufacturing (SM), improved anomaly detection frameworks are needed to reduce the operational risks and the monitoring burden placed on the system operators. However, building such frameworks is challenging, as a sufficiently large amount of defective training data is often not available and frameworks are required to capture both the temporal and contextual dependencies across different time steps while being robust to noise. In this paper, we propose an unsupervised Attention-Based Convolutional Long Short-Term Memory (ConvLSTM) Autoencoder with Dynamic Thresholding (ACLAE-DT) framework for anomaly detection and diagnosis in multivariate time series. The framework starts by pre-processing and enriching the data, before constructing feature images to characterize the system statuses across different time steps by capturing the inter-correlations between pairs of time series. Afterwards, the constructed feature images are fed into an attention-based ConvLSTM autoencoder, which aims to encode the constructed feature images and capture the temporal behavior, followed by decoding the compressed knowledge representation to reconstruct the feature images' input. The reconstruction errors are then computed and subjected to a statistical-based, dynamic thresholding mechanism to detect and diagnose the anomalies. Evaluation results conducted on real-life manufacturing data demonstrate the performance strengths of the proposed approach over state-of-the-art methods under different experimental settings.

Keywords: anomaly detection; deep learning; unsupervised learning; Industrial Internet of Things; time series

Citation: Tayeh, T.; Aburakhia, S.; Myers, R.; Shami, A. An Attention-Based ConvLSTM Autoencoder with Dynamic Thresholding for Unsupervised Anomaly Detection in Multivariate Time Series. *Mach. Learn. Knowl. Extr.* **2022**, *4*, 350–370. https://doi.org/10.3390/make4020015

Academic Editor: Andreas Holzinger

Received: 1 February 2022
Accepted: 18 March 2022
Published: 2 April 2022

1. Introduction

Artificial intelligence (AI), predictive analytics, big data, and Internet of Things (IoT) have become widely recognized for equipping existing systems with highly complex intelligent technologies, smarter services, and advanced applications in various domains, such as data networks [1–5], wireless sensor networks [6], manufacturing [7], and ubiquitous environments [8]. More specifically, the manufacturing domain has advanced massively in recent years and has become more computerized, complex, and automated, driving the emergence of smart manufacturing (SM) [9]. SM is a technology-driven approach that harnesses sensor data and automation for improving the manufacturing performance. As quality control is an indispensable part of the production process in all manufacturing industries around the globe, SM enables higher-quality products to be manufactured, while reducing costs and improving safety [10].

SM encompasses complex systems of interconnected sensors and computer components with diverse types that generate a substantial amount of multivariate time series data. As a result, potential system or production failures might occur, rendering anomaly

detection at certain time periods a vital task in order for the operators to solve the underlying issues. Anomalies within a sequence of data are broadly defined as observations that are unusual and signify irregular behavior. These irregular and unusual events have a very low probability of occurring, meaning that manual detection processes are a meticulous assignment that often requires more manpower than is generally available. Broken, cracked, and other imperfect products may result in costly returns, imposing operational and financial difficulties [11]. It is estimated that a 1% productivity improvement across the manufacturing industry worldwide can result in an annual savings of USD $500 million, whereas a breakdown in the production line can cost up to USD $50 thousand per hour [12]. Furthermore, predicting anomalies on time can decrease the number of breakdowns by up to 70%, maintenance costs by up to 30%, and scheduled repairs by up to 12% [12]. As a result, machine learning (ML) has been utilized in recent years to detect anomalies [1,6,13], particularly deep learning (DL) algorithms [14–16].

Automated anomaly detection algorithms leverage DL due to the latter's data-driven nature, efficiency, and ability to perform data analysis without explicitly programming the application [17]. Moreover, DL is able to learn higher-level features from data in a hierarchical fashion, which can aid in continuously improving the system's accuracy and automated decision-making processes. The aforementioned features make DL one of the main contributors to the fast growth of SM, as it reduces operating costs and improves operations visibility [18].

A basic requirement for building a supervised learning-based automated system to detect anomalies on a classification objective is the accessibility of a sufficient amount of training data for each class [19]. However, efficient supervised learning methods are often infeasible, as, with well-optimized processes, there is often an abundance of non-anomalous data and a relatively small or no amount of anomalous data. To address this data imbalance challenge, a substantial amount of unsupervised anomaly detection methods have been developed in recent years, as these methods are trained on unlabeled input data with no output variables. An additional advantage of this approach is potentially detecting anomalies in novel classes that are not part of the training data set, providing a general solution to the surface quality manufacturing process task [20].

Some previous approaches in the literature use defective samples for training [21,22], not solving the anomaly detection task described in this work. Other approaches utilize classical methods, such as probabilistic, distance-based, clustering, ensemble, and predictive approaches to detect anomalies in an unsupervised fashion, but all fail to capture complex structures in the data without the input of domain experts [20]. Unsupervised ML approaches were then proposed to capture these complex structures; however, they start failing to deal with the high dimensionality of the data feature space and the varying data aggregation as the data volume increases, requiring human expertise for feature extraction [18]. Various DL architectures, such as Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) neural networks, Convolutional LSTM (ConvLSTM) [23], neural networks, and autoencoders, have emerged in the recent literature to solve the aforementioned challenges. Furthermore, when a production failure is occurring, it is important to localize the anomaly root causes to plan adequate countermeasures and fix the production system [9]. This is done by pinpointing the sensors with anomalous readings, assisting the system operators to perform the system diagnosis and repair the system accordingly.

In this paper, we propose the ACLAE-DT framework, an unsupervised attention-based ConvLSTM autoencoder with dynamic thresholding to detect anomalies and identify anomalies' root cause in a manufacturing process multivariate time series. ACLAE-DT is a DL-based framework that is able to capture both the temporal and contextual dependencies across different time steps in the multivariate time series data, while being robust to noise. The framework starts by normalizing the input data via Min-Max scaling to scale the values to a fixed range. Post pre-processing, the data are then enriched using sliding windows to be able to capture more temporal behavior via the lagged values, followed

by the incorporation of contextual manufacturing process data via an embedding layer to capture the contextual information. Afterwards, ACLAE-DT constructs feature images based on the processed and enriched data, which are matrices of inner products between a pair of time series within the sliding window segments. Feature images characterize the system statuses across different time steps by capturing shape similarities and inter-correlations between two time series across different time steps, while being robust to noise. Subsequently, an attention-based ConvLSTM autoencoder is employed to encode the constructed feature images and capture the temporal behavior, followed by decoding the compressed knowledge representation to reconstruct the feature image input. The structures that exist in the data are learned and consequently leveraged when forcing the input through the autoencoder's bottleneck. Attention is added to the autoencoder to sustain a constant performance as the input time series sequences increase, reducing model errors [24]. Furthermore, several hyperparameters are optimized via random search in order to enhance the model's performance [25]. A dynamic thresholding mechanism is then utilized against the computed reconstruction errors to detect and diagnose the anomalies, where the threshold is dynamically updated based on statistical derivations from the reconstructed normal data errors. The intuition is that ACLAE-DT will not be able to reconstruct the feature images well if it has never observed similar system statuses before, resulting in anomalous processes to be flagged as it crosses the computed threshold. ACLAE-DT underwent rigorous empirical analysis, where results demonstrated the superior performance of the proposed approach over state-of-the-art methods.

The work presented in this paper is able to capture both the temporal and contextual dependencies across different time steps in the multivariate time series data in a manufacturing process to detect anomalies and identify the anomalies' root cause. The main contributions of this paper include:

- A novel framework that consists of pre-processing and enriching the multivariate time series, constructing feature images, and an attention-based ConvLSTM network autoencoder to reconstruct the feature image input. Moreover, the framework consists of a dynamic thresholding mechanism to detect anomalies and identify anomalies' root cause in multivariate time series.
- A generic, unsupervised learning framework that utilizes state-of-the-art DL algorithms and can be applied for various different multivariate time series use cases in SM.
- An attention-based, time-distributed ConvLSTM encoder–decoder model that is capable of sustaining constant performance as the rate of input time series sequences from the manufacturing operations increases.
- A nonparametric and dynamic thresholding mechanism for evaluating reconstruction errors that addresses non-stationarity, noise, and diversity issues in the data.
- A robust framework evaluated on a real-life public manufacturing data set, where results demonstrate its performance strengths over state-of-the-art methods under various experimental settings.

The remainder of this paper is structured as follows. Section 2 presents the motivations behind the use of DL in SM and explores related work. Section 3 discusses the technical preliminaries of the key concepts used in this paper. Section 4 details the methodology and implementation of the ACLAE-DT framework. Section 5 describes the data set used, the different experimental setups, and the comparison benchmarks. Section 6 discusses the obtained results and performance evaluation. Finally, Section 7 concludes the paper and discusses opportunities for future work.

2. Motivation and Related Work

Robotic finishing is one of the most important applications in SM. The goal of robotic finishing is to manufacture products without any defects and with an adequate amount of surface roughness, in order to ensure smooth functional and financial operations. However, potential system or production failures might occur at any point in time, demonstrating

the importance of having an efficient anomaly detection algorithm in place to detect and mitigate any manufacturing malfunctions.

Let us consider a robotic finishing system designed and calibrated to produce a finished metal car part. Multiple sensors are mounted on the system to gather different types of readings, such as the feed rate, spindle rate, and torque. As the robotic machine is manufacturing the car part, sensor readings are being processed to monitor the system's performance. In the case of an imperfect finished car part or anomalous system behavior, the anomaly detection algorithm will trigger an alarm and ideally locate the anomalous system behavior. Subsequently, the triggered alarm would notify and enable the system operators to solve the underlying issues.

Many classical methods have been used to detect anomalies, such as probabilistic approaches [26], distance-based approaches [27], clustering approaches [28], and predictive approaches [29]. These approaches may be computationally non-complex; however, their performance is sub-optimal as they fail to capture complex structures in the data without the input of domain experts [20]. Furthermore, as the data volume increases, traditional approaches can fail to scale up as required to maintain their anomaly detection performances [20].

Moreover, ML algorithms were proposed to resolve the limitations in classical methods for anomaly detection. ML algorithms include K-Nearest Neighbors (K-NNs) [30], Support Vector Machines (SVMs) [31], and neural networks [32]. These algorithms are all supervised learning-based, meaning they rely on labeled normal and anomalous historical data for training [33]. However, collecting labeled anomalous data is often infeasible, as, with well-optimized processes, there is a often high imbalance in the training data due to the abundance of non-anomalous samples and a relatively small or no amount of anomalous samples. Furthermore, ML approaches often require human input for feature extraction, where they start to fail in dealing with the dimensionality and variety of the data as the data volume and velocity increase [18].

DL techniques have emerged in the recent literature to solve the aforementioned challenges. Although ML is a data-driven AI technique to model input and output relationships, DL has distinctive advantages over ML in terms of feature learning, model construction, and model training. DL integrates model construction and feature learning into a single model and trains the model's parameters jointly, creating an end-to-end learning structure with minimal human interference. Moreover, DL models have the capability to learn deep feature representations from the data, enabling discriminative features to be extracted and potentially eliminating the need for manual feature engineering by domain experts.

There has been an increase in available types of sensory data collected from various distinct aspects across the operational system in SM. As a result, data modeling and analysis are vital tasks in order to handle the large data volume increase and the real-time data processing to detect any system anomalies, tasks that DL excels in [34]. Different DL architectures were used in the literature to detect anomalies in SM, such as CNNs [35], LSTM neural networks [36], and autoencoders [37]. Leveraging the complementary strengths of CNNs and LSTMs, ConvLSTMs [23] emerged to accurately model the spatio-temporal information by having convolutional structures in both the input-to-state and state-to-state transitions. Furthermore, different DL approaches were used in the literature to detect anomalies in multivariate time series. In [38], a general anomaly detection framework was proposed via residual analysis, where time series observation residuals were used to identify and detect anomalies in an unsupervised manner and without prior knowledge of anomalies. In [39], an unsupervised DL model based on a variational autoencoder and a CNN architecture was proposed, allowing temporal correlations and spatial features to be captured in multivariate time series. In [40], a convolutional autoencoding memory network was proposed to characterize the spatial dependence of the data with a maximum mean discrepancy to distinguish between the noisy, normal, and abnormal data. Afterwards, a memory network consisting of an autoregressive model and a bidirectional LSTM with attention was used to capture temporal dependence from time series data and detect anomalies.

3. Technical Preliminaries

3.1. Convolutional Neural Network (CNN)

A CNN is a feedforward deep neural network that is most commonly applied to analyzing visual imagery [41], having a wide range of applications such as image and video recognition, image classification, and recommender systems. A CNN uses convolution in place of general matrix multiplication in at least one of the layers, and reduces the number of parameters very efficiently without compromising the quality of models.

The network continues learning new higher dimensionality and more complex features with every layer. The input data layer takes an input vector with shape ($N \times H \times V \times D$), where N is the number of images, H is the height, V is the width, and D is the number of channels. The input is then passed to the convolutional layer, which convolves the input and abstracts it to a feature map based on a set of weights.

3.2. Long Short-Term Memory (LSTM) Neural Network

An LSTM deep neural network is a special variant of Recurrent Neural Networks (RNNs) that excels in modeling temporal behavior, such as time series, language, audio, and text, due to the feedback loops used for learning and the extra parameter metric available for connections between time steps. An LSTM's main components are the memory cells and the input, forget, and output gates. These components allow the LSTM network to have connections from previous time steps and layers, where every output is influenced by the input as well as the historical inputs. Usually, there are multiple LSTM layers, where each layer comprises many LSTM units, and each unit comprises input, forget, and output gates. The equations for the input, forget, and output gates, as well as the candidate cell state, the cell state, and the LSTM cell output, are described as, respectively,

$$i_t = \sigma\left(W_{hi}h_{t-1} + W_{xi}x_t + b_i\right) \quad (1)$$

$$f_t = \sigma\left(W_{hf}h_{t-1} + W_{xf}x_t + b_f\right) \quad (2)$$

$$o_t = \sigma\left(W_{ho}h_{t-1} + W_{xo}x_t + b_o\right) \quad (3)$$

$$\tilde{C}_t = tanh\left(W_{hC}h_{t-1} + W_{xC}x_t + b_C\right) \quad (4)$$

$$C_t = f_tC_{t-1} + (1 - f_t)\tilde{C}_t \quad (5)$$

$$h_t = o_ttanh\left(C_t\right) \quad (6)$$

where i is the input gate, f is the forget gate, o is the output gate, $\tilde{C}_t$ is the candidate cell state, C is the cell state, h is the hidden state and cell output, σ denotes a logistic sigmoid function, W is the weight matrix, and b is the bias vector. Figure 1 visualizes the structure of an LSTM memory cell.

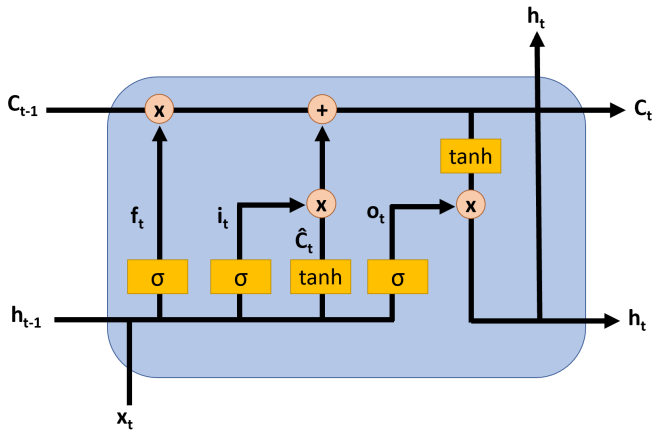

Figure 1. Structure of an LSTM memory cell.

3.3. Convolutional LSTM (ConvLSTM)

ConvLSTMs [23] are a special kind of LSTMs that emerged to accurately model the spatio-temporal information, by leveraging the strengths of a CNN and an LSTM. Similar to the LSTM, the ConvLSTM is able to decide what information to discard or retain from the previous cell state in its present cell state. However, convolutional structures are utilized in both the input-to-state and state-to-state transitions, which basically exchanges the internal matrix multiplications with convolution operations. The input vector to the ConvLSTM is fed as a series of 2D or 3D images, as the convolution operations allows the data that flow through the ConvLSTM cells to keep their input dimension instead of being a 1D vector with features. To describe the ConvLSTM operations, Equations (1)–(6) are rewritten as:

$$i_t = \sigma\left(W_{Ci} \circ C_{t-1} + W_{hi} * h_{t-1} + W_{xi} * x_t + b_i\right) \tag{7}$$

$$f_t = \sigma\left(W_{Cf} \circ C_{t-1} + W_{hf} * h_{t-1} + W_{xf} * x_t + b_f\right) \tag{8}$$

$$o_t = \sigma\left(W_{Co} \circ C_t + W_{ho} * h_{t-1} + W_{xo} * x_t + b_o\right) \tag{9}$$

$$\tilde{C}_t = tanh\left(W_{hC} * h_{t-1} + W_{xC} * x_t + b_C\right) \tag{10}$$

$$C_t = f_t \circ C_{t-1} + (1 - f_t) \circ \tilde{C}_t \tag{11}$$

$$h_t = o_t \circ tanh\left(C_t\right) \tag{12}$$

where $\circ$ represents the Hadamard product, $*$ represents the convolutional operator, W_{Ci}, W_{hi}, W_{xi}, W_{Cf}, W_{hf}, W_{xf}, W_{Co}, W_{ho}, W_{xo}, W_{hC}, $W_{xC} \in \mathbb{R}^{n \times T}$ represent the convolutional kernels within the model, and b_i, b_f, b_o, b_C are the bias parameters. Figure 2 visualizes the structure of a ConvLSTM, where the red lines indicate the extra connections found in a ConvLSTM cell over an LSTM cell, which come from the current and previous cell states.

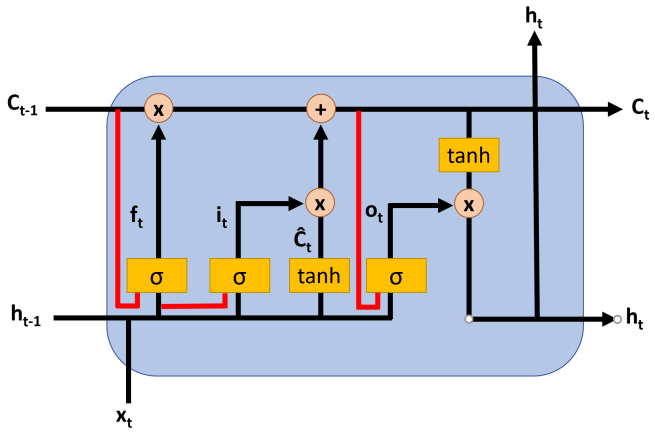

Figure 2. Structure of a ConvLSTM memory cell.

3.4. Autoencoder

An autoencoder is an unsupervised feedforward neural network that imposes a bottleneck in the network, forcing a compressed knowledge representation of the original input. More specifically, an autoencoder learns how to efficiently compress and encode data, before learning how to reconstruct the data back from the reduced encoded representation to an output representation that is as close to the original input as possible. An autoencoder consists of three main parts: the encoder, which learns how to reduce the input dimensions and compress the input data into an encoded compressed representation; the compressed representation itself; and the decoder, which learns how to reconstruct the compressed representation to be as close to the original input as possible. The network is trained to minimize the reconstruction error, $L(x, \hat{x})$, which measures the differences between the original input and the consequent reconstruction. By design, autoencoders reduce data dimensionality by learning to ignore noise in the data.

4. ACLAE-DT Framework

The following section details the ACLAE-DT framework's design, methodology, and implementation. First of all, the problem statement addressed in this work is discussed, before detailing each module in ACLAE-DT. The framework starts off by pre-processing and enriching the input raw time series, before constructing feature images across the different time steps. Afterwards, the attention-based ConvLSTM autoencoder aims to reconstruct the feature image input by minimizing the reconstruction errors. Hyperparameter optimization is conducted to improve the model's performance. Lastly, a dynamic thresholding mechanism is applied against the computed reconstruction errors for anomaly detection and diagnosis. Figure 3 visualizes a flowchart of the ACLAE-DT framework.

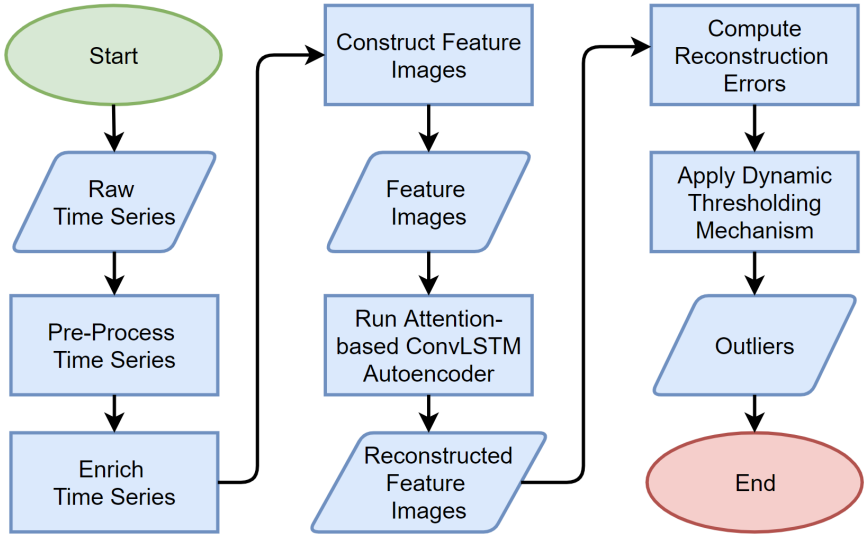

Figure 3. ACLAE-DT Framework.

4.1. Problem Statement

The historical raw multivariate time series is represented as

$$X = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^{n \times T} \tag{13}$$

where x_i is a single time series, n is the number of time series, T is the length of the time series, and X is the entire time series. Assuming that there are no anomalies in the training data, ACLAE-DT aims to detect anomalies by computing an anomaly score for each time step in x_i after T, such that a score outside of the threshold boundaries is flagged as an anomaly, indicating an anomalous time step. Moreover, given the anomaly detection results, ACLAE-DT aims to identify the anomalies' root cause by quantitatively and qualitatively analyzing the time series that are most likely to be the causes of the flagged anomalous time step.

4.2. Pre-Process Time Series

Each input raw multivariate time series, x_i, is normalized individually via Min-Max scaling to rescale their values between 0 and 1. Min-Max scaling normalization can be implemented as:

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \tag{14}$$

where x_i' is the normalized input time series value. As each time series is considered a feature, feature scaling eliminates large-scaled features to be dominating, while preserving

all relationships in the data [42]. Furthermore, it allows gradient descent to converge much faster when training the attention-based ConvLSTM autoencoder model [42].

4.3. Enrich Time Series

4.3.1. Utilizing Sliding Windows

Each pre-processed time series is then converted to a larger, more enriched time series of multivariate subsequences through the use of a sliding window. A sliding window indicates performing calculations on data within a specified window size, before sliding to the next window based on the step size specified, till the entire set of data has been covered in at least a single window. This means that data overlaps can occur across the different sliding windows when creating these extra sub-periods and incorporating the lagged values, which can assist the attention-based ConvLSTM autoencoder to extract richer features from the constructed feature images [43]. A more formal definition of a sliding window is specified as follows: given a time series x_i of length T and a user-defined subsequence length of d, all possible subsequences can be extracted by a sliding window of size d across x_i with a step size of $step$, and considering each subsequence s as $t - d$ to t, where t indicates the time position. The overlap rate is defined as $[(d - step)/d]$.

4.3.2. Embedding Contextual Information

Changes in a time series may be due to contextual changes. For example, a surface-finished material in a manufacturing process could end up being scratched or bent due to the clamp pressure used to hold the workpiece in the vise rather than the x, y, and z axis sensor values or the spindles. Therefore, taking contextual changes into consideration when detecting anomalies can enhance the anomaly detection performance [44].

Contextual information is usually represented as text or categorical variables. However, DL-based models are only able to process numerical values. One method to achieve the required numerical conversion includes the use of ordinal encoding, where each unique category or text value is assigned an integer value. Nonetheless, this assumes that integer values have a natural ordered relationship between each other, which is often not the case. To resolve this shortcoming, one-hot encoding can be applied to remove the integer-encoded variable and replace it with a new binary variable for each unique integer value. However, one-hot encoding does not scale well with respect to the number of categories, as the computation cost increases significantly as the categories increase, and it does not capture the similarities between categories.

Entity embeddings [45] resolve the aforementioned limitations by using an additional mapping operation that transforms and represents each category onto a low-dimensional space. The entity embedding vector or layer is a matrix of weights represented as $W_{embedding} \in \mathbb{R}^{q \times p}$, where q indicates the number of categories and p indicates the number of dimensions in the low-dimensional space. In this work, given a particular category v, a one-hot representation method $onehot(v) \in \mathbb{R}^{q \times 1}$ is used. Afterwards, an embedded representation method $embedded(v) = onehot(v) \times W_{embedding}$ is used for each category. It is important to set $p < q$ to ensure that as the number of categories increases, the dimensional value limits the computational cost increase. As a result, the embedded representation is much smaller than the one-hot representation and is able to capture similarities between the categories.

4.4. Construct Feature Images

In order to characterize the manufacturing system status accurately, it is critical to calculate and pinpoint the correlations between the different pairs of time series [46]. Acting as an extension to the work in [40], feature images are constructed in this paper to represent the inter-correlations between the different pairs of sensor values and contextual entity embedding time series in a multivariate time series. More specifically, an $(n + p) \times (n + p)$ feature image M^t is constructed for each sliding window segment s based upon the pairwise

inner product of two time series within this segment. The inter-correlation between two time series in a single segment is calculated as:

$$m_{ij}^{t} = \frac{\sum_{\delta=0}^{w} x_i^{t-\delta} x_j^{t-\delta}}{s} \in M^t \tag{15}$$

where i and j indicate two time series features, and δ indicates every single step in the segment. A feature image matrix is produced for each experiment or multivariate time series of length T, which consists of a feature image M^t for each segment s. In addition to representing the inter-correlations and shape similarities between the pairs of time series, feature images are robust to input noise, as instabilities at certain time steps have small consequences. Figure 4 visualizes a single feature image example M^t.

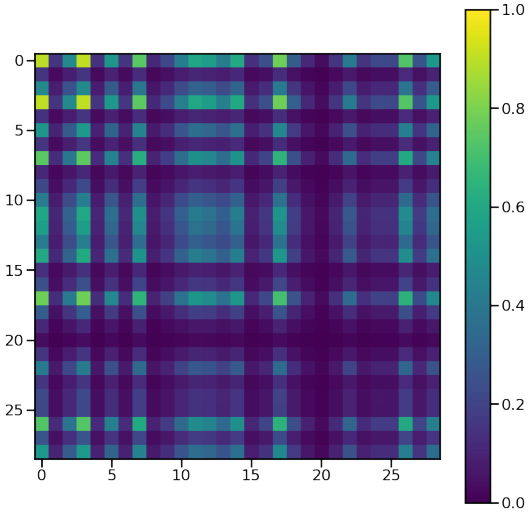

Figure 4. Feature image visualization.

4.5. Attention-Based ConvLSTM Autoencoder Model

Once all feature images have been constructed, they are input into the ConvLSTM autoencoder for reconstruction. More specifically, the autoencoder first encodes the constructed feature images and captures the temporal behavior via three alternating ConvLSTM encoding layers and MaxPool layers. Afterwards, the autoencoder decodes the compressed knowledge representation to reconstruct the original feature images via three alternating ConvLSTM decoding layers and UpSample layers. All MaxPool and UpSample layers have a size of (2×2), whereas all the ConvLSTM layers have a kernel size of (3×3) and the 'same' padding. Moreover, all the ConvLSTM layers have 64 filters, except for the last ConvLSTM encoder layer and the first ConvLSTM decoder layer, which have 32 filters. Figure 5 visualizes the employed ConvLSTM autoencoder architecture.

Although ConvLSTMs have been developed to accurately model the spatio-temporal information in a sequence, their performance may deteriorate as the sequence length increases. To overcome this issue, the Bahdanau attention [24] is added to the model, which can adaptively select relevant hidden states across different time steps and aggregate the representations of the informative feature maps to form a refined output of feature maps.

Constant model performance is achieved and model errors are reduced as the input time series sequences increase. The additive Bahdanau attention is described as:

$$c^t = \sum_{t'=1}^{T_x} \alpha^{t,t'} h^{t'} \tag{16}$$

$$\alpha^{t,t'} = \frac{exp(u^{t,t'})}{\sum_{k=1}^{T_x} exp(u^{t,k})} \tag{17}$$

$$u^{t,t'} = a(s^{t-1}, h^{t'}) \tag{18}$$

where c is the context vector for the sequence of hidden state annotations and α denotes the weights of each annotation. u is the alignment model of the feedforward neural network described by function a. The function attempts to capture the alignment between the attention-based ConvLSTM hidden state s of time step $t-1$ and the t'-th annotation from the hidden state h of the input sequence.

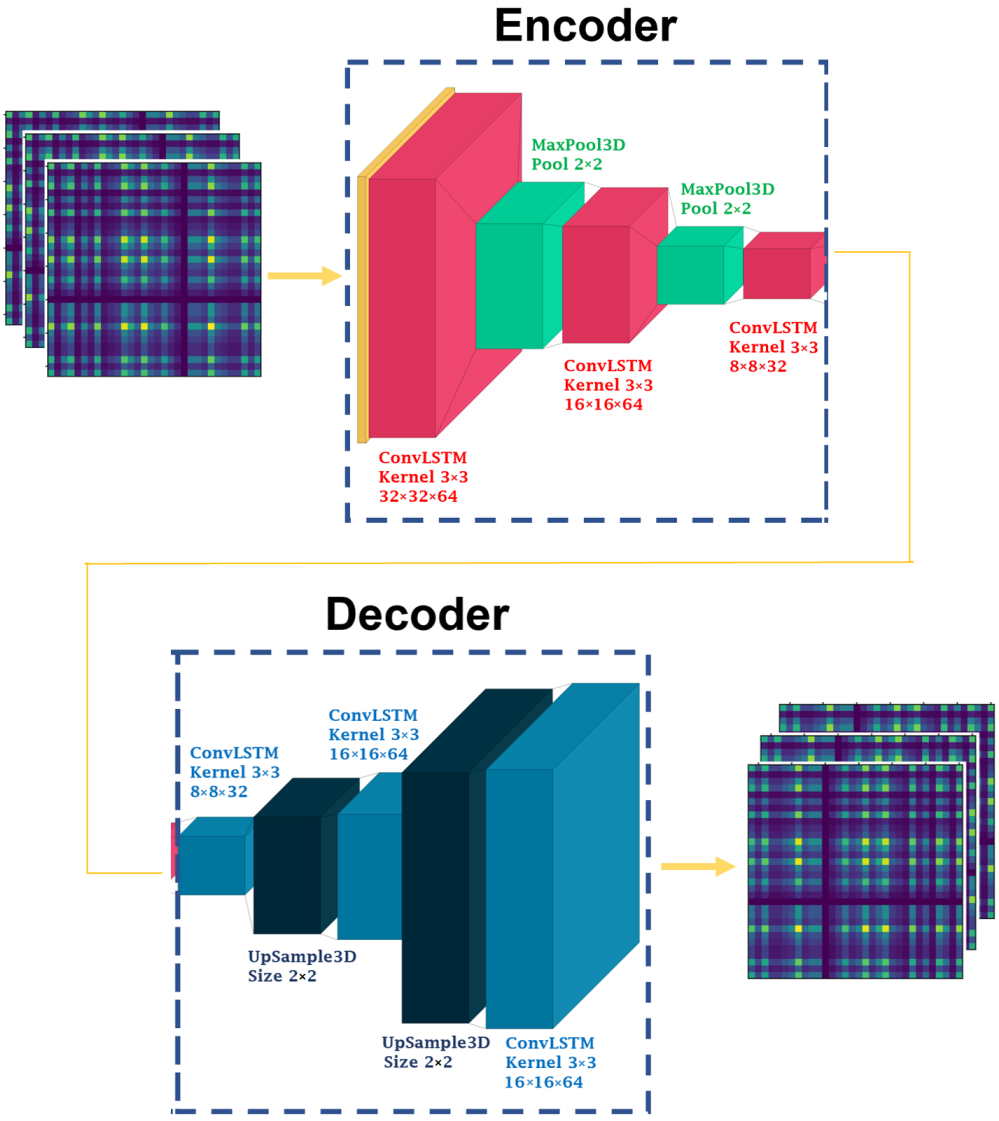

Figure 5. Proposed attention-based ConvLSTM autoencoder model.

4.6. Model's Hyperparameter Optimization

In order to enhance the performance of the model, several hyperparameters need to be tuned and optimized [25]. Model hyperparameters are used in processes to help estimate the model parameters, which are learned and estimated during the training process of minimizing an objective loss function. Several hyperparameter optimization methods exist, such as grid search, random search, and gradient-based optimization. Grid search is an exhaustive search through a manually specified set of hyperparameter values, which is time-consuming and impacted by the problem of dimensionality [47]. Gradient-based optimization utilizes the gradient descent algorithm to compute the gradient with respect to the hyperparameters, but they only support continuous hyperparameters and can only detect a local optimum for non-convex hyperparameter optimization problems rather than a global optimum [48]. Finally, random search randomly searches the grid space and supports all types of hyperparameters, allowing a larger and more diverse grid space to be explored. Hence, random search is used as the optimization method in this work as it is more computationally efficient than grid search and gradient-based optimization.

Five hyperparameters are optimized in this work, as presented in Table 1 alongside all the values under consideration and in which layer they lie, if applicable. The selected hyperparameters are considered to be the most influential five hyperparameters on the model's performance, based on initial experiments. All hyperparameter value options are based on initial experiments as well. Activation function values considered include Rectified Linear Units (ReLU) [49], Leaky ReLU [50], Exponential Linear Units (ELU) [51], and Scaled ELU (SELU) [52]. Optimization algorithms considered include Adam [53], RMSProp [54], AdaDelta [55], and Stochastic Gradient Descent (SGD) [56]. Finally, the loss functions considered include Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root MSE (RMSE).

Table 1. Model hyperparameters.

Layer	Hyperparameter	Values
ConvLSTM	Activation Function	ReLU, Leaky RELU, ELU, SELU
N/A	Learning Rate	1×10^{-2}, 1×10^{-3}, 1×10^{-4}, 1×10^{-5}, 1×10^{-6}
N/A	Batch Size	16, 32, 64, 128, 256
N/A	Optimizer	Adam, RMSProp, ADADelta, SGD
N/A	Loss Function	MAE, MSE, RMSE

4.7. Compute Reconstruction Errors

The model's loss objective in this work is to minimize the reconstruction errors of the normal feature images during training, in order to accurately reconstruct the normal feature images and inaccurately reconstruct the anomalous feature images during the testing phase. The loss function used is dependent on the random search hyperparameter optimization method output for the loss function hyperparameter.

4.8. Dynamic Thresholding Mechanism

At this stage, an efficient anomaly thresholding mechanism is needed in order to detect anomalous reconstructed feature images. Often, the anomaly threshold is learned with supervised methods. However, as the nature of the data in the manufacturing domain is continuously changing and there exist insufficient labeled data for each class, supervised methods would not be optimal for use here [57]. Hence, a nonparametric and dynamic anomaly thresholding mechanism is proposed in this work, which calculates a different threshold for each time series pair based on statistical derivations, achieving high anomaly detection performance with low overhead. More specifically, a single threshold is set against every single time series pair in the feature image, based on the mean and standard deviation of that specific normal time series pair's reconstruction errors. Any time series pair value that surpasses the threshold at any time step during testing will be flagged as

anomalous and will flag the entire process at that time step as anomalous. Mathematically, the method is described as:

$$\epsilon_{ij} = \left(\mu(e_{ij}) + z\sigma(e_{ij})\right)^T \in \epsilon \tag{19}$$

where ϵ_{ij} indicates the threshold value for the i and j time series features pair across the entire time series, ϵ is the $(n+p) \times (n+p)$ threshold matrix, e_{ij} is the set of reconstruction errors for the normal i and j time series features pair, μ is the mean, σ is the standard deviation, and z is an ordered set of positive values representing the number of standard deviations. Values for z depend on context, with a range of two to five found to produce the most accurate experimental results in this work. The presented dynamic thresholding mechanism detects outliers, as well as localizing the anomaly root cause, by pinpointing the sensors that are causing the detected outlier.

5. Experiments

The data set used in this paper is the Computer Numerical Control (CNC) Mill Tool Wear data set provided by the University of Michigan [58] and found on Kaggle [59]. The data set consisted of a series of machining experiments run on $2 \times 2 \times 1.5$ inch wax blocks in a CNC milling machine in the System-level Manufacturing and Automation Research Testbed (SMART). The utilized SMART machinery is technologically advanced, resulting in more real-life SM data. Moreover, the data set provides a sufficient amount of time series features to construct rich feature images for training and inference, as well as providing many different experimental settings and tool wear conditions to empirically test ACLAE-DT. The machining data were collected from a CNC machine for variations in feed rate, tool condition, and clamping pressure, where each experiment produced a finished wax part with an "S" shape. A total of 44 time series readings from the 4 motors in the CNC machine, the X-axis, Y-axis, Z-axis, and spindle (S-axis), were collected at a sampling rate of 10 Hz. The time series readings include the motor's actual position, actual velocity, actual acceleration, command position, command velocity, command acceleration, current feedback, DC bus voltage, output current, output voltage, and output power. All available attributes were used in this paper. The data set contained a total of 25,286 time series measurements from the 18 experiments conducted, where 4 of these experiments failed the visual inspection check. Data from each experiment were in a separate .csv file.

In this paper, every measurement is taken as an independent observation within a sliding window to identify normal or anomalous behavior and to pinpoint the sensors that flagged windows as anomalous. Any measurement that is part of the 4 experiments with a failed visual inspection check contains at least a single anomalous time series reading.

In order to evaluate ACLAE-DT's anomaly detection performance, the attention-based ConvLSTM autoencoder model is compared with seven baseline methods. The baseline methods comprise an ML classification method, a classical forecasting method, three state-of-the-art DL methods, and two variants of ACLAE-DT. The classical and ML methods are evaluated to demonstrate the effectiveness of using a DL model, and the baseline DL methods are evaluated to demonstrate the effectiveness of a ConvLSTM autoencoder. Moreover, the two variants of ACLAE-DT are evaluated to demonstrate the effectiveness of each component within the model. The same numbers of layers, hyperparameters, and components are used for each DL method, if applicable. The baseline methods are as follows.

1. SVM: An ML method that classifies whether a test data point is an anomaly or not based on the learned decision function from the training data.
2. Auto-Regressive Integrated Moving Average (ARIMA): A classical prediction model that captures the temporal dependencies in the training data to forecast the predicted values of the testing data.
3. LSTM Autoencoder: A DL method that utilizes LSTM networks in both the encoder and decoder.

4. ConvLSTM-LSTM Autoencoder: A DL method that utilizes ConvLSTM networks in the encoder and LSTM networks in the decoder.
5. CNN-LSTM Autoencoder: A DL method that utilizes CNN-LSTM networks in both the encoder and decoder.
6. ACLAE-DT Shallow: An ACLAE-DT variant that utilizes ACLAE-DT's model without the last MaxPool3D and ConvLSTM encoder components and without the first UpSample3D and ConvLSTM decoder components.
7. ACLAE-DT No-Attention: An ACLAE-DT variant that utilizes ACLAE-DT's model without attention.

To empirically examine the models, three different experiments are conducted. The models are tested using a window size of 10 with a step size of 2 in Experiment 1, a window size of 30 with a step size of 5 in Experiment 2, and a window size of 60 with a step size of 10 in Experiment 3. All DL-based models are trained for 250 epochs. Moreover, comparison metrics are employed to evaluate the models used and compare their anomaly detection performance. In order to fully capture the values of the true and false positives and negatives for each model, the precision, recall, and F1 score metrics are utilized, as well as the time taken to train each model. An anomalous window is defined as a poorly reconstructed feature image with a value that surpasses the corresponding threshold. True positives in this work indicate anomalous windows correctly classified as anomalous, and true negatives indicate non-anomalous windows correctly classified as non-anomalous. All experiments are repeated five times and the average results are computed for performance comparison.

Afterwards, ACLAE-DT's results are analyzed to pinpoint the readings that flagged windows as anomalous. It is important to localize the anomaly root cause during a production failure to plan adequate countermeasures and fix the system.

All networks are built and implemented in Python 3.7.9, using the Tensorflow [60] and Keras [61] libraries. All work is run on a machine comprising an NVIDIA GeForce GTX 1650 4 GB, a 16 GB DDR4 2666 MHz RAM, and a 9th Generation Intel Core i7-9750H Processor.

6. Performance Evaluation

6.1. Anomaly Detection Results

The anomaly detection performance for each model under the three different experimental settings is illustrated in Tables 2–4, respectively. Note that the results for ARIMA and SVM are the same across all three experiments, as they do not consider window sizes and step sizes in their algorithmic calculations. Table 2 demonstrates the performance evaluation of all eight models in Experiment 1. It can be observed that ARIMA detected anomalies better than SVM, indicating that the data set had a temporal dependency feature that could not be captured by the classification method. However, all the DL-based methods performed better than ARIMA, indicating DL's strength in capturing more complex structures and modeling a finer multivariate temporal dependency and correlation from the data set. Furthermore, it can be observed that all variants of ACLAE-DT performed better than the three baseline DL models based on every single evaluation metric used, while taking less time to train. The full ACLAE-DT model performed at least 4.8% better in every single evaluation metric, while taking at least 22.9% less time to train than the three baseline DL models. Moreover, the full ACLAE-DT model performed either similarly to or better than the two variant baseline models, while taking 6.7% less time to train than the shallow model but 3.1% more time than the no-attention model.

Table 2. Anomaly detection results using a window size of 10 with a step size of 2.

Method	Precision	Recall	F1 Score	Train Time (s)
SVM (Linear Kernel)	0.15	0.17	0.16	14
ARIMA (2,1,2)	0.52	0.59	0.56	98
LSTM Autoencoder	0.83	0.80	0.82	13,468
ConvLSTM-LSTM Autoencoder	0.80	0.84	0.82	11,914
CNN-LSTM Autoencoder	0.84	0.84	0.84	10,136
ACLAE-DT Shallow	0.94	0.87	0.90	8372
ACLAE-DT No Attention	0.95	0.87	0.91	7574
ACLAE-DT Full	0.95	0.88	0.92	7812

Similar observations can be drawn from Tables 3 and 4, which demonstrate the performance evaluation of all eight models in Experiment 2 and Experiment 3, respectively. As the window size and step size increased, the training time for all ACLAE-DT variants decreased, whilst average performance improved. This was not the case with the DL baseline models, however, as training time and performance did not follow a general trend as window sizes and step sizes changed. In Experiment 2, the full ACLAE-DT model performed at least 5.8% better in every single evaluation metric, while taking at least 65.8% less time to train than the three baseline DL models. In Experiment 3, the full ACLAE-DT model performed at least 16.5% better in every single evaluation metric, while taking at least 196.7% less time to train than the three baseline DL models. In both experiments, the full ACLAE-DT model performed better than the two variant baseline models, while taking less time to train than the shallow model but more time than the no-attention model.

Table 3. Anomaly detection results using a window size of 30 with a step size of 5.

Method	Precision	Recall	F1 Score	Train Time (s)
SVM (Linear Kernel)	0.15	0.17	0.16	14
ARIMA (2,1,2)	0.52	0.59	0.56	98
LSTM Autoencoder	0.82	0.83	0.83	15,932
ConvLSTM-LSTM Autoencoder	0.79	0.84	0.81	8274
CNN-LSTM Autoencoder	0.83	0.85	0.84	5362
ACLAE-DT Shallow	0.91	0.89	0.90	3388
ACLAE-DT No Attention	0.95	0.88	0.90	3122
ACLAE-DT Full	0.96	0.90	0.93	3234

Table 4. Anomaly detection results using a window size of 60 with a step size of 10.

Method	Precision	Recall	F1 Score	Train Time (s)
SVM (Linear Kernel)	0.15	0.17	0.16	14
ARIMA (2,1,2)	0.52	0.59	0.56	98
LSTM Autoencoder	0.79	0.83	0.82	7462
ConvLSTM-LSTM Autoencoder	0.77	0.82	0.79	13,496
CNN-LSTM Autoencoder	0.84	0.85	0.85	5152
ACLAE-DT Shallow	0.96	0.99	0.97	2814
ACLAE-DT No Attention	0.97	0.99	0.98	1638
ACLAE-DT Full	0.99	1.00	1.00	1736

All the previous results demonstrate the strength of utilizing a DL-based anomaly detection model in multivariate time series, as SVM and ARIMA failed to capture complex relationships. Moreover, the results demonstrate the effectiveness of deploying ConvLSTM networks compared to LSTM networks or CNNs in both the encoder and decoder, as ACLAE-DT was capable of capturing the inter-sensor correlations and temporal patterns of multivariate time series effectively. The results further demonstrate the effectiveness of constructing a deeper model and adding attention to it, particularly for when the window size is 30 and above, as performance constantly improved. The full ACLAE-DT model and

its variants performed the best in Experiment 3, whilst taking the least amount of training time. The full ACLAE-DT model had the best model performance out of all the compared models in the aforementioned experimental setting, attaining a perfect recall and F1 score and an almost perfect precision score, with an average training time of 1736 s.

Previous works in the manufacturing domain for tool wear prediction [62–66] utilized different data sets, which either include one or a few time series features, making it difficult for direct comparison with ACLAE-DT as it is tested against a richer data set with 44 time series features. In the literature, no other work has utilized the entire data set in [59] to simultaneously detect anomalies and identify the anomalous root causes while employing ML or DL methods. Moreover, the accuracy metric was utilized in [62–66], potentially being misleading since the metric can be heavily skewed towards finding non-anomalous points, which usually dominate real-life data sets, and the accuracy metric was not used in this paper for direct score comparisons.

6.2. Anomaly Root Cause Identification Results

If the reconstruction error of an inter-correlation between two time series crossed the set threshold for a particular window, then the corresponding pair of sensors were signified as contributors towards the anomalous window. The three sensor readings that contributed the most towards the flagged anomalous windows in Experiment 3 are visualized in Figure 6. The x-axis indicates the sensor reading features, and the y-axis indicates the feature's anomalous window appearance percentage. It can be observed from the figure that the readings from the x-axis motor had the greatest influence on the success of the visual inspection check, as they contributed the most towards flagging a window as anomalous. The X1-OutputCurrent feature had the greatest influence as it passed its threshold in 95.7% of the windows, followed by the X1-DCBusVoltage feature as it passed its threshold in 88.2% of the windows, followed by the X1-ActualAcceleration feature as it passed its threshold in 78.7% of the windows.

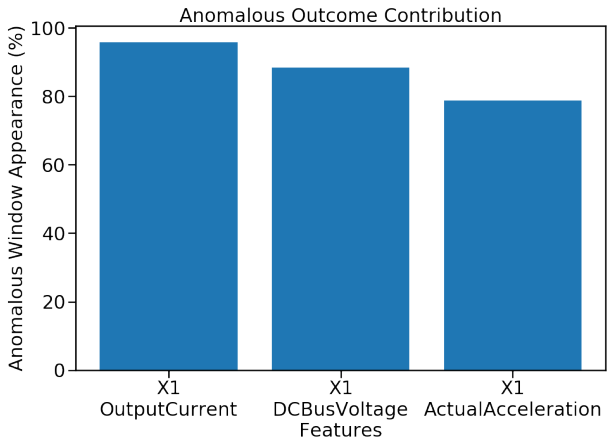

Figure 6. Anomaly root cause feature analysis.

X1-OutputCurrent was further analyzed in order to have a thorough understanding of ACLAE-DT's mechanism and results. Figure 7 visualizes three charts within a specific time series cross-section: (a) X1-OutputCurrent original vs. reconstructed normal data, (b) X1-OutputCurrent original vs. reconstructed anomalous data, and (c) X1-OutputCurrent reconstructed normal vs. anomalous data errors with the calculated dynamic threshold boundary in red. In chart (a), it can be observed that ACLAE-DT was able to reconstruct the original normal data well for most data points, with a small margin of error. In chart (b), it can be observed that ACLAE-DT was not able to reconstruct the original anomalous data well, particularly for the reading peaks, as it had never observed similar system statuses before. Finally, in chart (c), it can be realized that the reconstructed anomalous data errors crossed the threshold frequently, whereas the reconstructed normal data errors never

crossed the threshold. It is important to note that many of the reconstructed anomalous errors were just shy of crossing the threshold, indicating that when the inter-correlation between X1-OutputCurrent and another time series was computed, the results were bound to cross the set threshold if the time series contained novel system behavior, contributing to X1-OutputCurrent's high anomalous correlation.

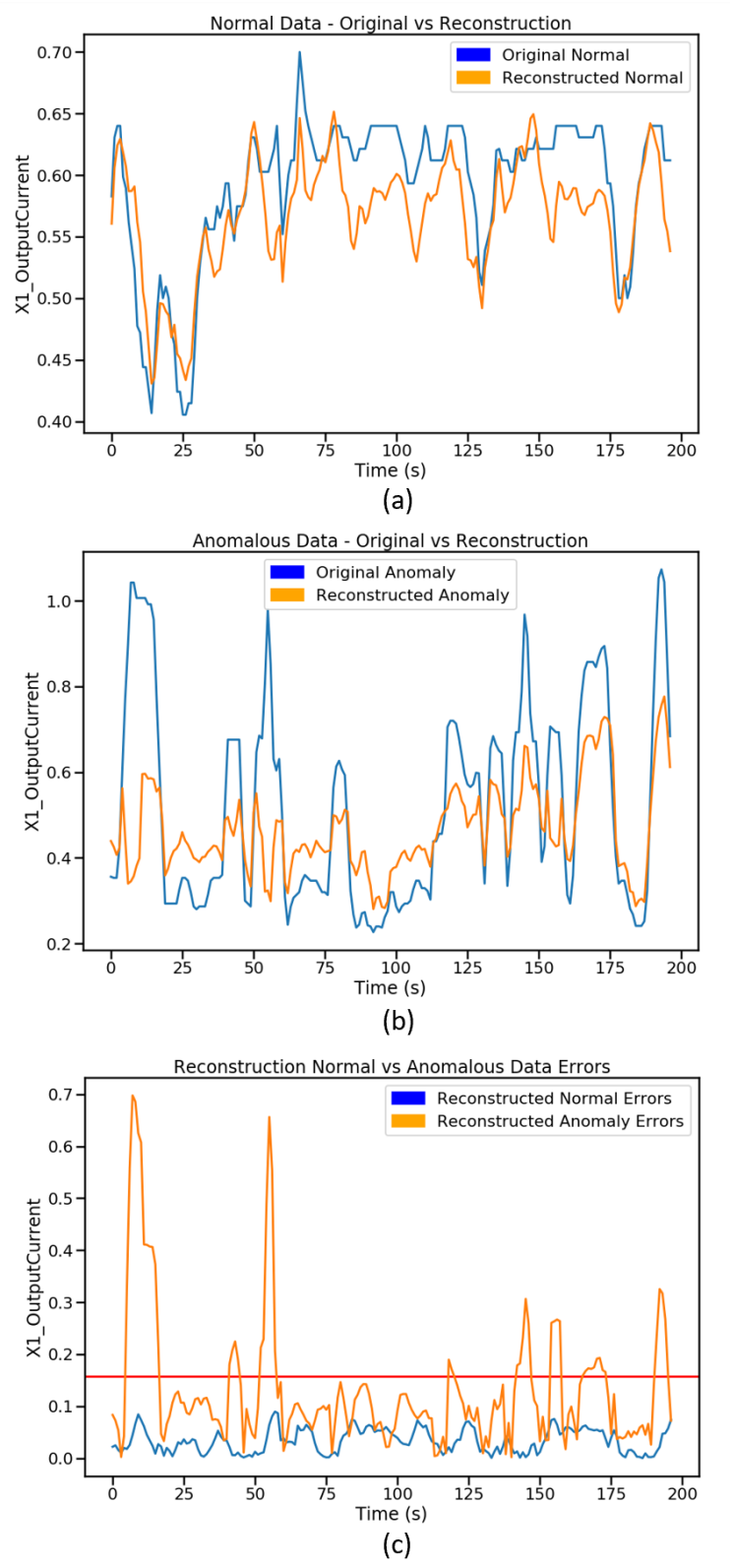

Figure 7. X1-OutputCurrent time series measurement analysis.

6.3. Execution Time and Memory Requirements

To further evaluate ACLAE-DT against the baseline methods, the execution time and memory requirements of each method for a single window in Experiment 3 were calculated, as visualized in Figures 8 and 9, respectively. Ten experimental executions were conducted and the average results were used. SVM and ARIMA were not included in the comparison due to their poor anomaly detection performance, deeming both methods unsuitable for use in real-life SM processes.

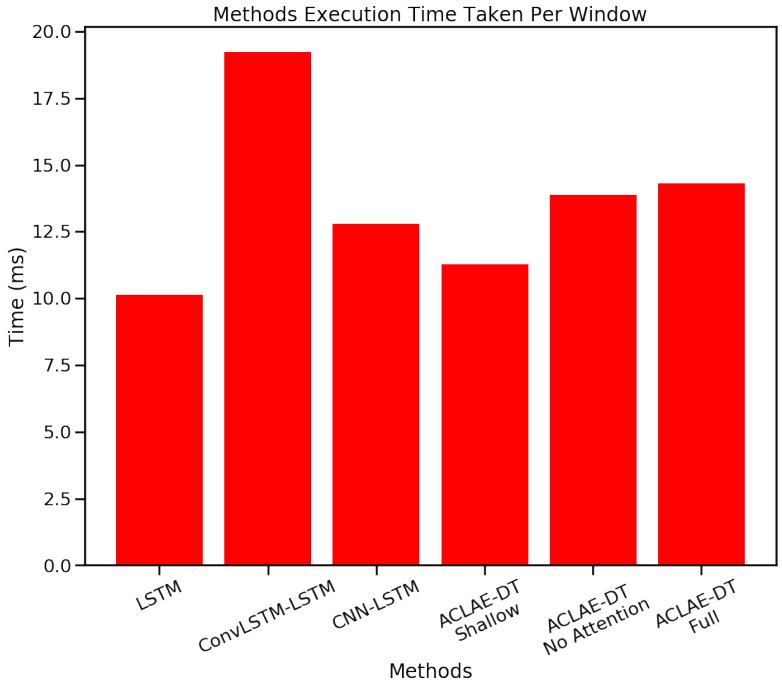

Figure 8. Execution time taken per window.

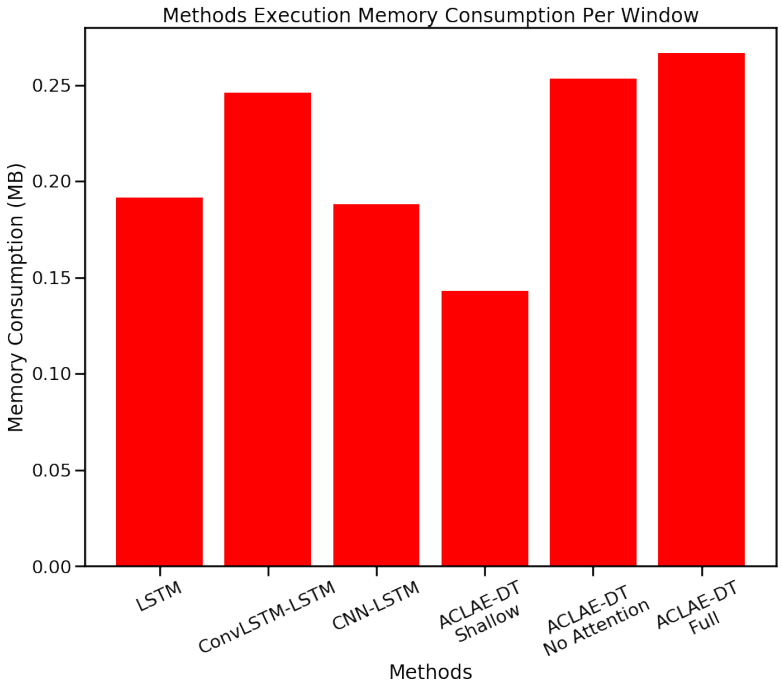

Figure 9. Execution memory consumption per window.

It can be observed from Figure 8 that the LSTM autoencoder method took the shortest execution time per window, with around 10 ms, followed by the ACLAE-DT shallow method with around 11.25 ms. The ACLAE-DT full method took around 14 ms, 40% more time than the LSTM autoencoder method and 24.4% more time than the ACLAE-DT shallow method. Moreover, it can be drawn from Figure 9 that the ACLAE-DT shallow method required the least amount of memory during execution, with around 0.14 MB. ACLAE-DT no-attention and ACLAE-DT full required the largest amount of memory out of all methods, with memory consumption of around 0.25 and 0.26 MB, respectively. From the drawn observations, it is evident that the tradeoff of using the full ACLAE-DT method with its superior anomaly detection performance, root cause detection identification, and short training time is its greater execution time and memory consumption. If a method's execution time and memory consumption are of higher importance than the method's anomaly detection performance and training time during real-life production, then the ACLAE-DT shallow method can be utilized as it performed very closely to the ACLAE-DT full method and required the second least amount of execution time and the least amount of execution memory out of all methods.

7. Conclusions

In this paper, a novel unsupervised attention-based deep ConvLSTM autoencoder with a dynamic thresholding mechanism framework, ACLAE-DT, was proposed to detect anomalies in a real-life manufacturing multivariate time series data set. The framework first normalized and enriched the raw time series with contextual information and sliding windows, before constructing feature images to capture system statuses across different time steps. The feature images were then input into an attention-based deep ConvLSTM autoencoder to be reconstructed, with an aim to minimize the reconstruction errors. The computed reconstruction errors were then subjected to a dynamic, nonparametric thresholding mechanism that utilized the mean and standard deviation of the normal reconstruction errors to compute a specific threshold for each time series pair, in order to detect and diagnose the anomalies.

Results demonstrated the effectiveness of ACLAE-DT, as it outperformed a classical approach, an ML approach, and three state-of-the-art DL approaches in detecting anomalous windows, while requiring less time to train than the latter approaches. Results further illustrated how ACLAE-DT was able to effectively diagnose the anomalies and locate the contributing features towards the anomalous windows. Moreover, the shallow variation of ACLAE-DT consumed the least amount of execution memory and the second least amount of execution time out the three state-of-the-art DL methods. All these results indicated the practicality and suitability of adopting ACLAE-DT for anomaly detection in produced robotic surface finishes and in real-life smart manufacturing processes. Furthermore, ACLAE-DT can be extended to detect anomalies in robotic processes within other IoT domains, such as data networks, wireless sensor networks, ubiquitous environments, and advanced healthcare. However, to fully ensure that ACLAE-DT is ready to be deployed in production for real-life smart manufacturing processes, further experiments with different data structures, behaviors, and features need to be conducted to assess the method's anomaly detection effectiveness in a variety of environments. Moreover, if the ACLAE-DT networks require retraining often in a production environment, a reduction in training time is required to maintain the anomaly detection effectiveness. As a future extension to this work, ACLAE-DT can be applied to another public data set to benchmark its performance with the conventional anomaly detection algorithms, and a reduction in the execution time and memory consumption of the full ACLAE-DT model while maintaining the superior anomaly detection and anomaly root cause identification performances can be explored.

Author Contributions: Conceptualization, T.T., S.A., R.M., A.S.; methodology, T.T.; software, T.T.; writing—original draft preparation, T.T.; writing—review and editing, T.T., S.A., A.S.; supervision, A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the Natural Research Council (NRC) of the Government of Canada under Project AM-105-1.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
ARIMA	Auto-Regressive Integrated Moving Average
CNC	Computer Numerical Control
CNN	Convolutional Neural Network
ConvLSTM	Convolutional Long Short-Term Memory
DL	Deep Learning
ELU	Exponential Linear Units
IIoT	Industrial Internet of Things
IoT	Internet of Things
K-NN	K-Nearest Neighbor
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
MSE	Mean Squared Error
ReLU	Rectified Linear Units
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SELU	Scaled Exponential Linear Units
SGD	Stochastic Gradient Descent
SMART	System-level Manufacturing and Automation Research Testbed
SVM	Support Vector Machine

References

1. Injadat, M.; Moubayed, A.; Nassif, A.B.; Shami, A. Multi-stage optimized machine learning framework for network intrusion detection. *IEEE Trans. Netw. Serv. Manag.* **2020**, *18*, 1803–1816. [CrossRef]
2. Moubayed, A.; Aqeeli, E.; Shami, A. Ensemble-based feature selection and classification model for dns typo-squatting detection. In Proceedings of the 2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), London, ON, Canada, 30 August–2 September 2020; pp. 1–6.
3. Salo, F.; Injadat, M.; Nassif, A.B.; Shami, A.; Essex, A. Data mining techniques in intrusion detection systems: A systematic literature review. *IEEE Access* **2018**, *6*, 56046–56058. [CrossRef]
4. Injadat, M.; Salo, F.; Nassif, A.B.; Essex, A.; Shami, A. Bayesian optimization with machine learning algorithms towards anomaly detection. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
5. Moubayed, A.; Injadat, M.; Shami, A.; Lutfiyya, H. Dns typo-squatting domain detection: A data analytics & machine learning based approach. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–7.
6. Yu, T.; Wang, X.; Shami, A. Recursive principal component analysis-based data outlier detection and sensor data aggregation in IoT systems. *IEEE Internet Things J.* **2017**, *4*, 2207–2216. [CrossRef]
7. Liu, J.; Guo, J.; Orlik, P.; Shibata, M.; Nakahara, D.; Mii, S.; Takáč, M. Anomaly detection in manufacturing systems using structured neural networks. In Proceedings of the 2018 13th World Congress on Intelligent Control and Automation (WCICA), Changsha, China, 4–8 July 2018; pp. 175–180.
8. Thakur, N.; Han, C.Y. An ambient intelligence-based human behavior monitoring framework for ubiquitous environments. *Information* **2021**, *12*, 81. [CrossRef]
9. Kusiak, A. Smart manufacturing. *Int. J. Prod. Res* **2018**, *56*, 508–517. [CrossRef]
10. Kang, H.S.; Lee, J.Y.; Choi, S.; Kim, H.; Park, J.H.; Son, J.Y.; Kim, B.H.; Do Noh, S. Smart manufacturing: Past research, present findings, and future directions. *Int. J. Precis. Eng. Manuf.-Green Technol.* **2016**, *3*, 111–128. [CrossRef]

11. Russo, I.; Confente, I.; Gligor, D.M.; Cobelli, N. The combined effect of product returns experience and switching costs on B2B customer re-purchase intent. *J. Bus. Ind. Mark.* **2017**, *32*, 664–676. [CrossRef]
12. Progress. Anomaly Detection & Prediction Decoded: 6 Industries, Copious Challenges, Extraordinary Impact. 2017. Available online: https://www.progress.com/docs/default-source/datarpm/progress_datarpm_cadp_ebook_anomaly_detection_in_6_industries.pdf?sfvrsn=82a183de_2 (accessed on 8 March 2022).
13. Yang, L.; Moubayed, A.; Hamieh, I.; Shami, A. Tree-based intelligent intrusion detection system in internet of vehicles. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
14. Kieu, T.; Yang, B.; Jensen, C.S. Outlier detection for multidimensional time series using deep neural networks. In Proceedings of the 2018 19th IEEE International Conference on Mobile Data Management (MDM), Aalborg, Denmark, 25–28 June 2018; pp. 125–134.
15. Aburakhia, S.; Tayeh, T.; Myers, R.; Shami, A. A Transfer Learning Framework for Anomaly Detection Using Model of Normality. In Proceedings of the 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 4–7 November 2020; pp. 55–61.
16. Guo, Y.; Liao, W.; Wang, Q.; Yu, L.; Ji, T.; Li, P. Multidimensional time series anomaly detection: A gru-based gaussian mixture variational autoencoder approach. In Proceedings of the Asian Conference on Machine Learning, Beijing, China, 14–16 November 2018; pp. 97–112.
17. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Volume 13.
18. Wang, J.; Ma, Y.; Zhang, L.; Gao, R.X.; Wu, D. Deep learning for smart manufacturing: Methods and applications. *J. Manuf. Syst.* **2018**, *48*, 144–156. [CrossRef]
19. Cook, A.A.; Mısırlı, G.; Fan, Z. Anomaly detection for IoT time-series data: A survey. *IEEE Internet Things J.* **2019**, *7*, 6481–6494. [CrossRef]
20. Chalapathy, R.; Chawla, S. Deep learning for anomaly detection: A survey. *arXiv* **2019**, arXiv:1901.03407.
21. Görnitz, N.; Kloft, M.; Rieck, K.; Brefeld, U. Toward supervised anomaly detection. *J. Artif. Intell. Res.* **2013**, *46*, 235–262. [CrossRef]
22. Kawachi, Y.; Koizumi, Y.; Harada, N. Complementary set variational autoencoder for supervised anomaly detection. In Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 15–20 April 2018; pp. 2366–2370.
23. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.Y.; Wong, W.K.; Woo, W.C. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In Proceedings of the Twenty-ninth Conference on Neural Information Processing Systems, Montréal, QC, Canada, 11–12 December 2015; pp. 1–9.
24. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
25. Yang, L.; Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* **2020**, *415*, 295–316. [CrossRef]
26. Saci, A.; Al-Dweik, A.; Shami, A. Autocorrelation Integrated Gaussian Based Anomaly Detection using Sensory Data in Industrial Manufacturing. *IEEE Sens. J.* **2021**, *21*, 9231–9241. [CrossRef]
27. Huo, W.; Wang, W.; Li, W. Anomalydetect: An online distance-based anomaly detection algorithm. In Proceedings of the International Conference on Web Services, Milan, Italy, 8–13 July 2019; pp. 63–79.
28. Li, J.; Izakian, H.; Pedrycz, W.; Jamal, I. Clustering-based anomaly detection in multivariate time series data. *Appl. Soft Comput.* **2021**, *100*, 106919. [CrossRef]
29. Gokcesu, K.; Kozat, S.S. Online anomaly detection with minimax optimal density estimation in nonstationary environments. *IEEE Trans. Signal Process* **2017**, *66*, 1213–1227. [CrossRef]
30. Xie, M.; Hu, J.; Han, S.; Chen, H.H. Scalable hypergrid k-NN-based online anomaly detection in wireless sensor networks. *Ieee Trans. Parallel Distrib. Syst.* **2012**, *24*, 1661–1670. [CrossRef]
31. Wang, Y.; Wong, J.; Miner, A. Anomaly intrusion detection using one class SVM. In Proceedings of the Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, West Point, NY, USA, 10–11 June 2004; pp. 358–364.
32. Ryan, J.; Lin, M.J.; Miikkulainen, R. Intrusion detection with neural networks. *Adv. Neural Inf. Process. Syst.* **1998**, *10*, 943–949.
33. Rasheed, W.; Tang, T.B. Anomaly detection of moderate traumatic brain injury using auto-regularized multi-instance one-class SVM. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2019**, *28*, 83–93. [CrossRef]
34. Kusiak, A. Smart manufacturing must embrace big data. *Nature* **2017**, *544*, 23. [CrossRef] [PubMed]
35. Tayeh, T.; Aburakhia, S.; Myers, R.; Shami, A. Distance-Based Anomaly Detection for Industrial Surfaces Using Triplet Networks. In Proceedings of the 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 4–7 November 2020; pp. 372–377.
36. Feng, C.; Li, T.; Chana, D. Multi-level anomaly detection in industrial control systems via package signatures and LSTM networks. In Proceedings of the 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Denver, CO, USA, 26–29 June 2017; pp. 261–272.
37. Bayram, B.; Duman, T.B.; Ince, G. Real time detection of acoustic anomalies in industrial processes using sequential autoencoders. *Expert Syst.* **2021**, *38*, e12564. [CrossRef]

38. Benkabou, S.E.; Benabdeslem, K.; Kraus, V.; Bourhis, K.; Canitia, B. Local Anomaly Detection for Multivariate Time Series by Temporal Dependency Based on Poisson Model. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**.. [CrossRef] [PubMed]
39. Choi, T.; Lee, D.; Jung, Y.; Choi, H.J. Multivariate Time-series Anomaly Detection using SeqVAE-CNN Hybrid Model. In Proceedings of the 2022 International Conference on Information Networking (ICOIN), Jeju-si, Korea, 12–15 January 2022; pp. 250–253.
40. Zhang, C.; Song, D.; Chen, Y.; Feng, X.; Lumezanu, C.; Cheng, W.; Ni, J.; Zong, B.; Chen, H.; Chawla, N.V. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 1409–1416.
41. Valueva, M.V.; Nagornov, N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Math. Comput. Simul.* **2020**, *177*, 232–243. [CrossRef]
42. Jayalakshmi, T.; Santhakumaran, A. Statistical normalization and back propagation for classification. *Int. J. Comput. Sci. Appl.* **2011**, *3*, 1793–8201.
43. Yu, Y.; Zhu, Y.; Li, S.; Wan, D. Time series outlier detection based on sliding window prediction. *Math. Probl. Eng.* **2014**, *2014*, 879736. [CrossRef]
44. Gupta, M.; Sharma, A.B.; Chen, H.; Jiang, G. Context-Aware Time Series Anomaly Detection for Complex Systems. In Proceedings of the SDM Workshop on Data Mining for Service and Maintenance, Austin, TX, USA, 2–4 May 2013.
45. Guo, C.; Berkhahn, F. Entity embeddings of categorical variables. *arXiv* **2016**, arXiv:1604.06737.
46. Hallac, D.; Vare, S.; Boyd, S.; Leskovec, J. Toeplitz inverse covariance-based clustering of multivariate time series data. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 215–223.
47. Claesen, M.; Simm, J.; Popovic, D.; Moreau, Y.; De Moor, B. Easy hyperparameter search using optunity. *arXiv* **2014**, arXiv:1412.1114.
48. Zöller, M.A.; Huber, M.F. Benchmark and survey of automated machine learning frameworks. *arXiv* **2019**, arXiv:1904.12054.
49. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the ICML, Haifa, Israel, 21–24 June 2010.
50. Maas, A.L.; Hannun, A.Y.; Ng, A.Y. Rectifier nonlinearities improve neural network acoustic models. In Proceedings of the ICML, Atlanta, GA, USA, 16–21 June 2013; Volume 30, p. 3.
51. Clevert, D.A.; Unterthiner, T.; Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv* **2015**, arXiv:1511.07289.
52. Klambauer, G.; Unterthiner, T.; Mayr, A.; Hochreiter, S. Self-normalizing neural networks. *arXiv* **2017**, arXiv:1706.02515.
53. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
54. Tieleman, T.; Hinton, G.E. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *Coursera Neural Netw. Mach. Learn.* **2012**, *4*, 26–31.
55. Zeiler, M.D. Adadelta: An adaptive learning rate method. *arXiv* **2012**, arXiv:1212.5701.
56. Kiefer, J.; Wolfowitz, J. Stochastic estimation of the maximum of a regression function. *Ann. Math. Stat.* **1952**, *23*, 462–466. [CrossRef]
57. Chandola, V.; Banerjee, A.; Kumar, V. Survey of anomaly detection. *ACM Comput. Surv.* **2009**, *41*, 1–58. [CrossRef]
58. Kovalenko, I.; Saez, M.; Barton, K.; Tilbury, D. SMART: A system-level manufacturing and automation research testbed. *Smart Sustain. Manuf. Syst.* **2017**, *1*. [CrossRef]
59. Sun, S. CNC Mill Tool Wear. 2018. Available online: https://www.kaggle.com/shasun/tool-wear-detection-in-cnc-mill (accessed on 8 March 2022).
60. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
61. Chollet, F. Keras. 2015. Available online: https://github.com/fchollet/keras (accessed on 8 March 2022).
62. Kumar, S.; Kolekar, T.; Kotecha, K.; Patil, S.; Bongale, A. Performance evaluation for tool wear prediction based on Bi-directional, Encoder–Decoder and Hybrid Long Short-Term Memory models. *Int. J. Qual. Reliab. Manag.* **2022**.. [CrossRef]
63. Bazi, R.; Benkedjouh, T.; Habbouche, H.; Rechak, S.; Zerhouni, N. A hybrid CNN-BiLSTM approach-based variational mode decomposition for tool wear monitoring. *Int. J. Adv. Manuf. Technol.* **2022**, *119*, 3803–3817. [CrossRef]
64. Cai, W.; Zhang, W.; Hu, X.; Liu, Y. A hybrid information model based on long short-term memory network for tool condition monitoring. *J. Intell. Manuf.* **2020**, *31*, 1497–1510. [CrossRef]
65. Aghazadeh, F.; Tahan, A.; Thomas, M. Tool condition monitoring using spectral subtraction and convolutional neural networks in milling process. *Int. J. Adv. Manuf. Technol.* **2018**, *98*, 3217–3227. [CrossRef]
66. Martínez-Arellano, G.; Terrazas, G.; Ratchev, S. Tool wear classification using time series imaging and deep learning. *Int. J. Adv. Manuf. Technol.* **2019**, *104*, 3647–3662. [CrossRef]