Unsupervised Online Anomaly Detection on Multivariate Sensing Time Series Data for Smart Manufacturing

Ruei-Jie Hsieh
Dept. of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
eunahsieh@lsalab.cs.nthu.edu.tw

Jerry Chou
Dept. of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
jchou@lsalab.cs.nthu.edu.tw

Chih-Hsiang Ho
Dept. of Smart System Institute
Institute for Information Industry
Taipei, Taiwan
andrew.ho@iii.org.tw

***Abstract*—The emergence of IoT and AI has brought revolutionary change in various application domains. One of them is Industry 4.0, also called Smart Manufacturing, which aims to achieve highly flexible and automated production processes. In this paper, we study a use case of anomaly detection in smart manufacturing using the real data collected from the sensing devices of a factory production line. Our goal is to improve the anomaly detection accuracy at an earlier stage of production line, so that cost and time wasted by possible production failures can be reduced. To overcome the limited and irregular anomaly patterns found from our multivariate sensor dataset, we proposed an unsupervised real-time anomaly detection algorithm based on LSTM-based Auto-Encoder. Our evaluations show that our approach achieved almost 90% accuracy for both precision and recall while other classification or regression based methods only reached 70% $\sim$ 85%.**

***Index Terms*—Deep Learning, Machine Learning, Anomaly Detection, Multivariate Time-Series Data, Long Short-Term Memory, Autoencoder**

I. Introduction

A fourth industrial revolution is occurring in global manufacturing. The concept of industry 4.0 is firstly published by Kagermann in 2011 [1], and now it has become a strategic initiative in many countries to address the need of advanced manufacturing with higher efficiency, lower costs, and mass-personalization of products and services. The goal of the initiative is to transform industrial manufacturing through digitalization and exploitation of potentials of new technologies, including cyber-physical systems (CPS), the internet of things (IoT), cloud computing and artificial intelligence(AI), etc. One of the examples is Smart manufacturing [2], which aims to build a fully integrated, collaborative manufacturing system that responds in real time to meet changing demands and conditions in the factory, in the supply network and in customer needs.

One of the important problems in Smart manufacturing is anomaly detection [3], [4]. It enables predictive maintenance [5] which reduces maintenance frequency to the lowest possible state leading to a huge cost saving in keeping resources in normal working condition. Especially, as the scale and complexity of manufacturing continue to grow in a rapid speed, the probability of equipment or product failures also increases. To minimize the cost of failures, we must detect anomalous behavior of mechanical devices in the production line in order to predict potential problems as early as possible, so that the production plans can be adjusted accordingly to avoid failures, and the failures already happened can be contained in their early stage to avoid cascading effects.

In this paper, we study the anomaly detection problem using a multi-variate time series dataset collected from the sensors installed on the manufacturing equipment of a factory processing line. The problem of anomaly detection is particularly challenged because the anomaly data records are limited, the anomaly patterns are highly irregular, and the detection must be accurate in a timing fashion. Several approaches have been proposed previously as discussed in Section III. Traditionally, rule-based policies are applied for detection. The rules are specified according to the experiences, domain knowledge or ad hoc data analysis. Hence it is vulnerable to unseen anomaly, and it cannot be easily generalized to other fields or environments. Machine learning techniques became more commonly seen with the emergence paradigm of big data. Some of these approaches are based on time series analysis models, such as Autoregressive Integrated Moving Average. Others use classification algorithm, such as Dynamic Time Warping or KNN. But most classifiers perform poorly on imbalanced datasets since they are intended to maximize the accuracy.

To address aforementioned challenges, we developed an unsupervised ANN (Artificial neural network) detection model based on the LSTM-based Auto-Encoder. The LSTM cell has the ability to capture temporal dependencies in a multivariate sensor data. The Auto-Encoder network architecture is used to learn the normal behavior without labeled dataset. Once the model is well trained, the online evaluation data is fed into our model as a set of sequences by a sliding window for real time anomaly detection. Finally, an alerts is triggered when a time interval is voted as an anomalous data point by majority. To the best of knowledge, such approach has not been proposed for anomaly detection on multi-variate time series data, nor has it been evaluated by real dataset from Smart manufacturing. Our evaluations show that our approach achieved almost 90% accuracy for both precision and recall while other classifiers (i.i., CNN and DTW k-nn) or regression based methods (i.e., Vector Auto Regressive) only reached only 70% $\sim$ 85%.

The remainder of this paper is organized as follows: Section II briefly reviews Long Short-Term Memory(LSTM) and Autoencoder. Section III introduces different approaches of anomaly detection tasks. Section IV specifies the objective for this paper and subsequently discussed the challenges of Multivariate time-series anomaly detection tasks. Later describes the use case scenario of our automatic production line anomaly detection task. Section V provides the details of the data preprocessing and the architecture of the proposed three phase LSTM-based Autoencoder. The experimental results will be discussed in Section VI. Finally, Section VII concludes this paper.

II. Background

In this section we briefly describe the architecture of a Long Short-Term Memory and the structure of AutoEncoder model.

A. Long Short-Term Memory(LSTM)

A recurrent neural network(RNN) is a class of artificial neural networks which has the ability of exhibiting temporal dynamic behavior. It processes input sequences of arbitrary length by recursively activating transition function on a hidden state vector h_t. At each timestep t, there are two inputs to the hidden layer: the output of the previous layer h_{t-1}, and the input at that timestep x_t. The former input is multiplied by a weight matrix W_x and W_h and later feed into the non-linear transition function(e.g. $tanh()$) to produce hidden state vector h_t.

$$h_t = tanh(W_x X_t + W_h h_{t+1} + b) \tag{1}$$

During training RNN, the same weights W_x and W_h are applied repeatedly at each timestep. This can grow or decay exponentially over long sequences [6] [7], which makes it difficult to learn long-distance correlations in a sequence. To solve the exploding or vanishing gradients problem, the LSTM architecture introduces a memory cell which changes in response to inputs to preserve cell state over long periods of time [8]. LSTM was proposed by [9] in 1997, and has already proven to be a powerful technique for addressing the problem of time series prediction.

Define d memory dimensions of LSTM units at each timestep t to be a collection of vectors in $\mathbb{R}^d$. Each unit contains an input gate i_t, a forget gate f_t, an output gate o_t as shown in Fig. 1, a memory cell state c_t and a hidden state h_t making it easier to capture long-term dependencies. Depending on the strength of the information each node receives, it will decide to block it or pass it on. When it is transferred through these cells, the information is also filtered with the set of weights associated with the cells.

LSTM has the power to incorporate a behavior into a network by training it with time series data. A prediction is made by two part: the value of the input and its position in the time series. This means that two same input value at different times probably results in two different outputs and it makes LSTM to be a powerful technique for addressing the problem of time series prediction.

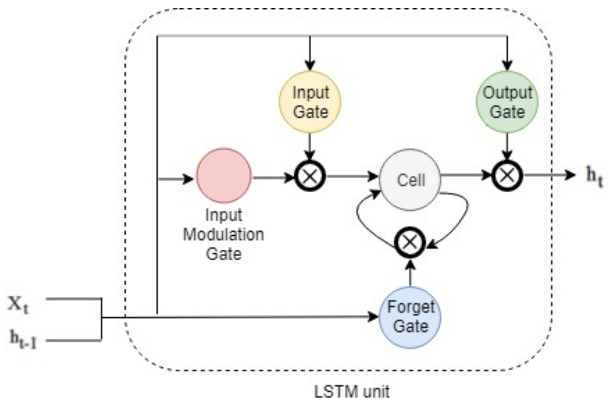


Fig. 1. LSTM unit

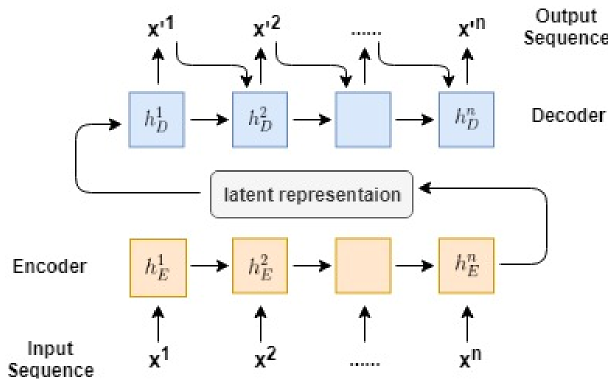


Fig. 2. sequence-to-sequence model architecture

B. AutoEncoder

An autoencoder is a neural network that is trained by unsupervised learning, which is trained to learn reconstructions that are close to its original input. In general, autoencoder consists of an encoder and a decoder network as shown in Fig. 2. The encoder takes the original input and extracts a fixed-sized latent representation, which dimensionality is usually smaller than the input. The decoder further maps the latent representation back to the original input space as a reconstruction. An autoencoder learns to minimize the reconstruction error between original input and reconstructed input. In the other words, the autoencoder learns to extract meaningful information that sufficiently explains the characteristics of the data.

In general, autoencoders for sequences with temporal dependencies are implemented as sequence-to-sequence(Seq2Seq) models [10]. Sequence-to-sequence model consists of an encoder and a decoder network. This is widely used for machine translation task. Fig. 2 shows the architecture of sequence-to-sequence model. The encoder takes the sequence x as input and calculates hidden states hE for every time step. The resulting hidden representation is fed into the decoder. Each decoder step receives the previous decoder state h_D^{i-1} and the previous reconstructed output $x'^{(i-1)}$ to calculate h_D^i and $x'^{(i)}$.

III. Related work

For the anomaly detection tasks, we can classify previous work methods in to the following three categories:

Rule-based approach: A rule-based system is made up of a set of rules, which typically takes the form of if-then rules. In general, such rules can be manually constructed according to the experiences with regard to different tasks [11]. For the purpose of defending sovereignty, protecting infrastructures, countering terrorism, detecting illegal activities, [12] propose a rule-based expert system by variety of sensor data streams and knowledge of the maritime domain in support of maritime anomaly detection. [13] is the anomaly detection task to the medical area presents an algorithm for performing early detection of disease outbreaks by rule-based anomaly detection algorithm that characterizes each anomalous pattern with a rule.

Classification approach: Classification algorithms is a straightforward approach for pattern recognition in time series data. Distance-based methods along with k-nearest neighbors have proven to be successful in classifying multivariate time series [14]. To measure a distance between two sequences, plenty of research [15] [16] indicates Dynamic Time Warping (DTW) as the best distance-based measure to use along k-NN . Deep learning has also yielded promising results for multivariate time series classification. A recent study [17] propose an effective Multi-Channel Deep Convolutional Neural Network (MC-DCNN). MC-DCNN takes input from each channel and learns features individually, the latent features are later fed into an MLP to perform classification.

Forecasting(Prediction-based) approach: Forecasting Models are trained to predict the next values for a given input sequence. Such a forecasting model can also be used for anomaly detection. The model is trained solely on normal data, once sufficiently trained, it can be utilized to detect anomalies by comparing the predicted value with the actual value recorded by sensors. If the error exceeds a predefined threshold value, the sample can be labeled as anomalous. There are several prediction-based algorithms for anomaly detection, such as 1-class SVMs [18] or Deep Neural Networks (DNN) with Long Short-Term Memory (LSTM) cells as their predictive model [19] [20].

IV. Problem Description

A. Objective and Challenges

Nowadays manufacturing industry utilizes numerous sensors to monitor machines behavior. The reliable detection of anomalous patterns in continuous sensor data is an important challenge in todays manufacturing industry, as the abnormal behavior can be a sign of some problem in the production line which might eventually lead to failure. Currently, this process is handled by engineers, who have sufficient knowledge of the domain. These engineers manually detect anomalous samples based on their experience. However, when it comes to multivariate time-series sensor data, this task can be really hard and prone to human error. Besides, it's difficult to accomplish it without extensive domain knowledge.

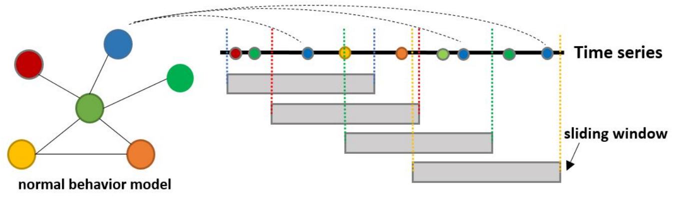


Fig. 3. Online detection procedure

Training a classifier requires a sufficient amount of labeled training examples with a balanced class distribution. The tasks of anomaly detection is characterized by extremely imbalanced data distribution. That is, the normal samples are available but the number of anomalous samples is inadequate. Moreover, in general the abnormal behavior is irregular. It's not easy to detect the anomalous behaviors by pattern recognition and classification. We cannot detect the unseen anomalous behavior if we don't have this kind of samples in our training data.

The challenges of anomaly detection in multivariate sensor data can be summarized as follows:

- Without sufficient knowledge of the domain
- Multivariate data consisting of several different sensors
- Temporal dependencies
- Extremely imbalanced data distribution
- Irregular anomalous behavior

The objective of this work is to apply deep learning methods for anomalous patterns detection within sensor data of automatic production line. Deep learning allows modeling complicated behavior based on unknown underlying rules from a specific domain. Anomalies in sensor data can be defined as previously unseen patterns, since the recorded data is expected to consist solely of normal sequences. The algorithm for anomaly detection should be able to detect known anomalies as well as generalize to new and unknown anomalies. To achieve online detection, we utilize a $sliding window$ approach [21]. The basic idea of online detection procedure is illustrated in Fig. 3. Since the model is trained on normal data which is a fixed-length sequence of preceding steps. Once the model is well trained, it can achieve online anomaly detection by comparing the predicted value at each timestep with the actual sequence.

B. Use case scenario

In this paper, we use a real case scenario and data provided by a manufacturing company to evaluate and validate our deep

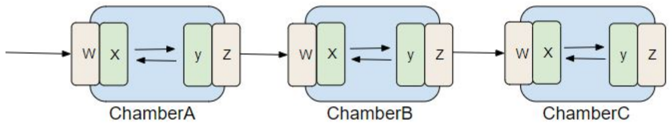


Fig. 4. Production line

learning based anomaly detection approach. The detailed of its production process and sensing data are given as follows.

Fig. 4 shows there are three chamber(A, B, C) in the production line, the product will move from A→B→C in order. Each chamber has 4 sensors(W, X, Y, Z) inside. When the sensor detects product it will output signal '1', otherwise it outputs '0'. When $sensor_w$ outputs signal '1', it indicates the product is already in the chamber. The product moves back and forth between $sensor_x$ and $sensor_y$ until the process completed. Then, the product will keep moving on and trigger $sensor_z$ outputs a signal which means the process in this chamber is finished. The behavior of all three chambers (i.e., $chamber_A$,$chamber_B$,$chamber_C$) are similar. But in dataset available to us, there are a lot of more anomalies in $chamber_B$ than $chamber_A$ and $chamber_C$. Hence, our evaluations also try to train the model for normal pattern using the data from $chamber_A$, and then evaluate the model on the data from $chamber_B$ instead. Since our approach can detect anomaly on any chambers before products go through the whole production line, early failure detection is achieved.

V. Methodology

A. Model Selection

For most of the anomaly detection tasks, we don't have sufficient labeled data, and the anomalous samples are very limited. Additionally, it may be very hard to recognize the anomalous behavior since the anomalous pattern is irregular. Because of above reasons a binary classification model cannot be properly trained. Therefore, we take an alternative approach based on reconstruction method—Autoencoder. The Autoencoder model consists of an encoder and a decoder network, where both are implemented as recurrent neural networks. Since anomaly events are rare, we assume the training dataset only contains normal pattern, and doesn't need to be labeled. This reconstruction method allows us only train on normal sequences, so that the model of normal pattern can be used to detect previously unseen anomaly. To detect anomalous sequences in the automatic process machine, we must consider the temporal dependencies in a time series dataset. Hence, we apply a LSTM-based Autoencoder as shown in Fig. 5.

B. Data Preprocessing

A strength of neural network model is its ability to perform feature selection automatically from training process. So our data preprocessing steps mostly focus on converting the raw data shown in Fig. 6 into a form that can be fed into our neural network model as follows.

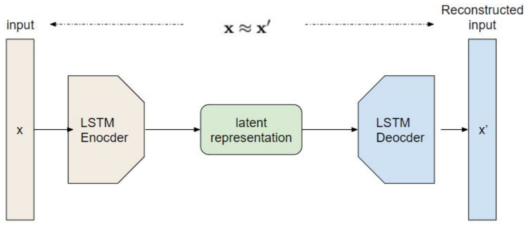
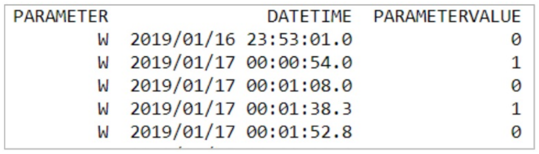


Fig. 5. LSTM AutoEncoder structure



PARAMETER	DATETIME	PARAMETERVALUE
W	2019/01/16 23:53:01.0	0
W	2019/01/17 00:00:54.0	1
W	2019/01/17 00:01:08.0	0
W	2019/01/17 00:01:38.3	1
W	2019/01/17 00:01:52.8	0

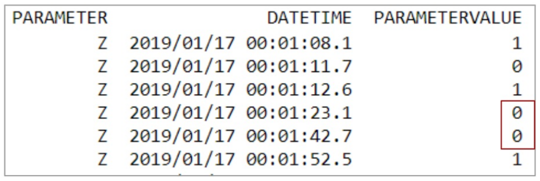
Fig. 6. Raw data

- Cleaning data: The output signals from a sensor should only switches between '0' and '1'. So, as shown in Figure 7, any parameter value remains the same across consecutive time intervals is treated as duplicate signal, and removed from the dataset.
- Convert DATETIME into UNIX timestamp: It is for the ease of computations in our computing programs.
- Convert timestamp into same interval: The raw data contains event-based records which only have records when sensor output is triggered. But our detection model is time interval-based. So we fill in records for each fixed time interval between the parameter value is changed. The time interval in our study is 1s.
- Integrate the dataset from all sensors into a multi-variate time series dataset: Figure 8 shows the formatted data structure for our LSTM auto-encoder network.
- Organize the input sequence: Since our data is recorded continuously(time series data), so we need to use a sliding window of fixed length to segment the data. Given that each timestamp S consists four sensors signal $\{W_i, X_i, Y_i, Z_i\}$. Assume the window size is WS, our data would later presents as $\{S_1, S_2, ..., S_{WS}\}$, $\{S_2, S_3, ..., S_{WS+1}\}, \cdots$, $\{S_n, S_{n+1}, \cdots, S_{WS+n}\}$.

C. Model Training & Evaluation

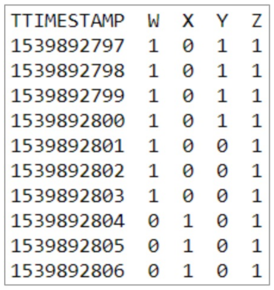
Our approach based on LSTM-based autoencoder can be divided into three phases as shown in Figure 9.

- Phase 1: Time-series Modeling:
  In this phase, first we use the historical data collected from sensors to train our LSTM-based Autoencoder for recognizing the normal pattern of a chamber in the production line. The autoencoder model encodes a sequence of data from the input and extracts a latent representation. The latent representation is feature vectors that hold the information, the features, that represents the input. The



```
PARAMETER                DATETIME  PARAMETERVALUE
        Z  2019/01/17 00:01:08.1                1
        Z  2019/01/17 00:01:11.7                0
        Z  2019/01/17 00:01:12.6                1
        Z  2019/01/17 00:01:23.1                0
        Z  2019/01/17 00:01:42.7                0
        Z  2019/01/17 00:01:52.5                1
```

Fig. 7. duplicate data



TTIMESTAMP	W	X	Y	Z
1539892797	1	0	1	1
1539892798	1	0	1	1
1539892799	1	0	1	1
1539892800	1	0	1	1
1539892801	1	0	0	1
1539892802	1	0	0	1
1539892803	1	0	0	1
1539892804	0	1	0	1
1539892805	0	1	0	1
1539892806	0	1	0	1

Fig. 8. multi-variate time series data as input

decoder takes these feature vectors as input and gives the best closest match to the inputs. The output of the model represents the reconstruction error of the given inputs. Hence, the loss function is based on computing the reconstruction error. Reconstruction error is calculated by comparing the original data(S) and the prediction(S') as below.

$$E_t = MSE(S_t, S'_t) = \sum_{i=1}^{k} \left\| S_t^i - S_t^{'i} \right\|^2 \qquad (2)$$

where k is the number of sensors.
The optimizer will train both encoder and decoder together in order to lower the loss. The model is well trained when the reconstruction error is small enough for all the input sequences from the training dataset. Then the testing data is fed into the model for online anomaly detection. A normal pattern seen by the autoencoder should have a small reconstruction error, while an unseen abnormal pattern likely to have a larger output values.

- Phase 2: Anomaly Detection:
  In Anomaly detection phase, we attain reconstruction error of each timestamp in the input sequence by our model. If we capture the significant reconstruction error between the original input and the prediction, it might imply this sample is a anomalous timestamp. We use a threshold ϵ to separate normal and abnormal samples. If the reconstruction error of a sample is larger than ϵ, it would be labeled as an anomaly from the test. The setting of ϵ is observed by the distribution of reconstruction errors from the training data as explained from our experimental results in Section VI-B2.
- Phase 3: Alerting:
  Since our input is a sequence, every timestamp in this sequence results in a predicted-label(normal/anomaly). Assume window size is WS, for each timestamp it will be covered by WS times, and thus received WS predicted-labels. Therefore, we apply majority voting to make the final decision. If the sample of a timestamp received more than half of the votes as anomaly, an alert is triggered to indicate a possible failure has occurred at the given timestamp.

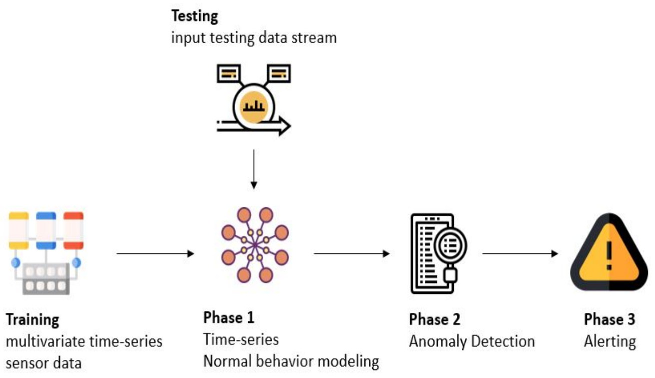


Fig. 9. Steps of using LSTM-based autoencoder for anomaly detection.

D. Transfer Learning

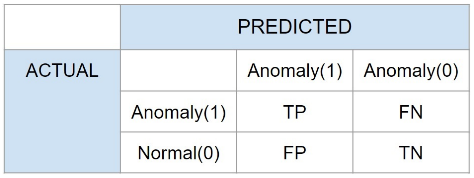
Considering we might not have sufficient amount of data for every chambers in our production line, and it could be time consuming to train a detection model for each chambers individually, we apply transfer leaning to reduce our model construction time.

Transfer learning [22] aims to extract the knowledge from one or more source tasks and applies the knowledge to a related target task. By retaining the knowledge (features, weights etc) from previously trained models, we could train a new model more efficiently with less training data.

Take our case study as an example. We observed that the behavior of all three chambers are similar. However, the number of normal data records from $chamber_B$ is much smaller than $chamber_A$. Therefore, we can apply transfer learning on the model previously trained by the data of $chamber_A$ for $chamber_B$. In our evaluation, we use the data of $chamber_B$ to retrain the last network layer of both encoder and decoder from the model of $chamber_A$. Our results show that transfer learning not only reduce the training time, but also improve detection accuracy.

VI. Experiment

This section presents the evaluation results of our approach using the real manufacturing dataset. First, we introduce the dataset and evaluation matrices in Section VI-A. Then, we compare the accuracy results of our method to several other previous approaches. Finally, we shows the impact of detection threshold, ϵ, the sliding window size, and the locality and size of training data.



	PREDICTED		
ACTUAL		Anomaly(1)	Anomaly(0)
	Anomaly(1)	TP	FN
	Normal(0)	FP	TN

Fig. 10. Confusion matrix

A. Experiment Setup

Our LSTM-based Autoencoder network model is implemented using the deep learning framework, Keras. The autoencoder consists of an encoder with one LSTM layer and a decoder with one LSTM layer and two stacked dense layers. Adam is used for optimizing the weights, as the popular algorithm in the field of deep learning because it achieves good results fast. The algorithms leverages the power of adaptive learning rates methods to find individual learning rates for each parameter. This makes it easier to tune the algorithms hyper-parameters based on similar experiments from recent literature.

The full training set consists 388791 samples which cover 168 hours time series sensor status. For this size of training set it make good balance between performance and training time in our experiments. The ratio of anomalies in our training set is 7% To verify the performance of our model, the data is split into training and test set. Since LSTM-based Autoencoder model is a unsupervised method that only trained on normal samples, the training set consists the majority of normal data which is assumed to show the normal behavior. The test set can be used to evaluate the accuracy of the model on previously unseen data.

We evaluate the capabilities of LSTM-based Autoencoder model by metrics such as precision, recall, and F1-score. As shown in Figure 10, precision is the probability of how many detected anomalies are accurate. Recall is the ratio of correctly predicted anomalies to the all actual anomalous samples in test set. F1-score gives a good estimation of anomaly detection tasks due to the unbalanced data composition which is defined as follows:

$$Recall = \frac{\#correct\ predictions}{\#true\ anomalies} = \frac{TP}{TP+FN} \tag{3}$$

$$Precision = \frac{\#correct\ predictions}{\#positive\ predictions} = \frac{TP}{TP+FP} \tag{4}$$

$$F_1 score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5}$$

B. Experiment Results

1) Evaluation metrics compare to other methods:

The time-series sensor data applying sliding window(30s) were fed as input to the model. We experiment our model on three different training/test set. The basic set trained on ChamberB(80%) of normal samples and tested on the rest of ChamberB(20%) shows a reasonable result compare to other methods. Training by ChamberA which consists of mostly normal data make a significant improvement.

We compared our method with traditional common statistical methods— Vector Auto Regression(VAR) and dynamic time wrapping k-NN classification, CNN classification. Vector Autoregression model(VAR) is a stochastic process model used to capture the linear interdependencies among multiple time series. A VAR model describes the evolution of a set of k variables over the same sample period ($t = 1, \cdots, T$) as a linear function of only their past values. Recently, there has been great success in time series analyses by applying dynamic time warping(DTW). Dynamic time wrapping(DTW) k-NN classification use DTW as the similarity score instead of Euclidean distance. The convolutional neural network (CNN) is a class of deep learning neural networks.A convolution can be seen as applying and sliding a filter over the time series.

In Fig. 11, we can see a great improvement on Precision and Recall comparing to the other classification models such as CNN and Dynamic time wrapping(DTW) k-NN. Since the classification model works better when the data is balanced, it can not reach a reasonable result in anomaly detection tasks. Machine learning classifiers are hard to cope with imbalanced training datasets as they are sensitive to the proportions of the different classes. As a consequence, these algorithms tend to favor the class with the largest proportion of observations (majority class), which may lead to misleading accuracy. In anomaly detection tasks, typically labeled data is imbalanced. This means that the anomalous records are rare. It's hard to train a binary classifier for this kind of imbalanced tasks. Besides, the results show that LSTM-based Autoencoder model trained by normal ChamberA is better than one of the most commonly used statistical methods for multivariate time-series forecasting Vector Auto Regressive (VAR). In a VAR model, each variable is a linear function of the past values of itself and the past values of all the other variables. It might be difficult to find a proper linear function to fit this time series data.

To make a further improvement of our method, we applied transfer learning. First, we trained a based normal model by $chamber_A$ and later retrained by partial of the target data set $chamber_B$. The model achieved an Precision of 90.68% , Recall of 89.82% and a F1-score of 90.24%. The main advantage of applying transfer learning is that once a based normal behavior model is well trained, we don't need a huge dataset to retrain this target dataset. We just need a partial of target dataset to fine tune the based normal behavior model.

2) Varied threshold:

To distinguish anomalous samples from the sensor data, we need to set a threshold. The threshold ϵ is setting by the reconstruction error of training set. Although we assumed our training data consists solely of normal sequences, as we can see in Fig. 12, there's still some noise and few anomalous samples in our training data which made us need to assume the proportion of normal samples in training set. Once we decide the proportion of normal samples in our training set, we can find the value of threshold ϵ. That is to say, any sample in the

Model	Precision	Recall	F1-score
Trained by B	0.7253	0.8437	0.7800
Trained by A	0.8381	0.9014	0.8686
Transfer learning	0.9068	0.8982	0.9024
VAR	0.7652	0.8564	0.8082
CNN classification	0.7138	0.8251	0.7654
DTW k-NN classification	0.7331	0.8171	0.7728

Fig. 11. Experiment result of different models

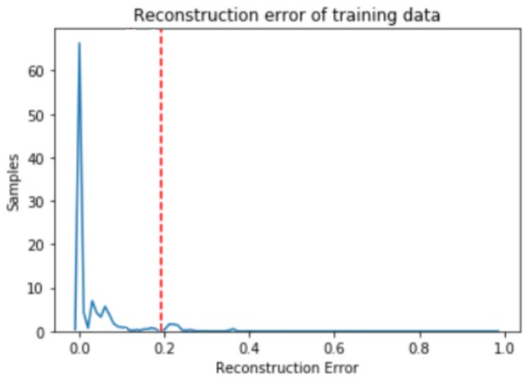


Fig. 12. Distribution of training set

test set which reconstruction error larger than ϵ it is considered as anomaly.

If we set a lower ϵ, we could reach a higher Recall value which means we captured most of the anomalous samples. However, it might lead a lower Precision since the false alarm rate. As shown in Fig. 13, the best setting found from our experiments is 0.93, which has the best F1-score and reaches the balance of Precision and Recall.

3) Data locality and training size effect:

In this experiment, Fig. 14 shows how changing the size of training data affects the anomaly detection performance and the effect of data locality. For a AutoEncoder model, it required a sufficient amount of training data to well construct the normal behavior. In Fig. 14, the learning curve represents the evolution of the performance as we increase the size of our training set. The more data we use to train our model, the better the predictive power. The F1-score raises as we increase the size of our dataset, because the model is able to generalize better from a higher amount of information.

Besides, if we select training set by temporal locality, it perform generally better than select randomly. Especially, when the training set size is small, the difference is obvious. When the training set is insufficient, the result is more related to the temporal locality. The reason is that the recent data can give more valuable information. As as we increase the size of our training set, the result shows that there's almost no difference since the training set is huge enough to construct a comprehensive normal behavior model.

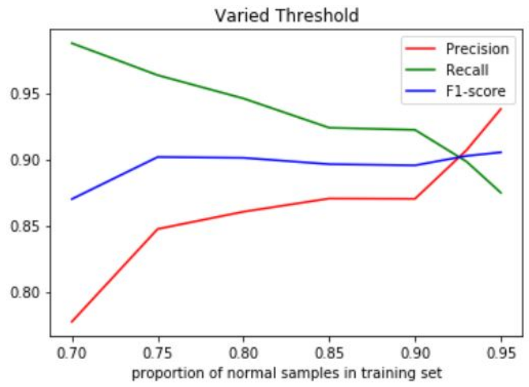


Fig. 13. Varied threshold

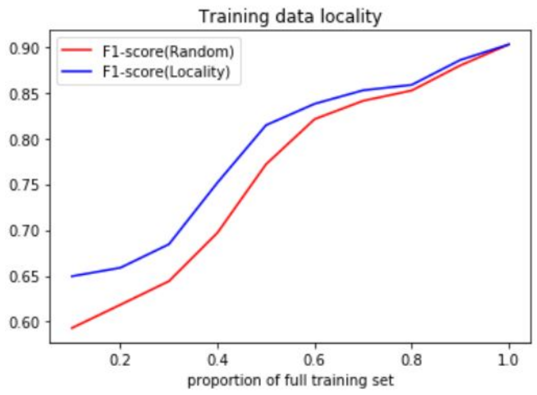


Fig. 14. Training set temporal locality

4) Effect of window length on average performance: Fig. 15 shows that the AutoEncoder architecture may work for different window lengths WS as we experimented the sliding window for a length between 10 to 50. In practice, a proper length of window relies on the nature of the data. In this case, the cycle of production line is 38 seconds. However, 38 seconds is not the best configuration in our results but 30 seconds. It has been shown that a small window length less than 30 is inadequate to induce a best personalized model on this production line. We have to choose WS large enough so that there is enough events stored for us to make a consistent inference. However, it doesn't mean that a large window size may lead better result since it does not provide any additional information but hide details we might interested in.

VII. Conclusion

In this work, we proposed a new neural network model based on LSTM-based Auto-Encoder for detecting anomaly on multi-variate time series dataset. By using the real use case scenario and datasets from a industry production line, we demonstrate our approach can be used for smart manufacturing to automated detect possible failures in a timing fashion without extensive domain knowledge.

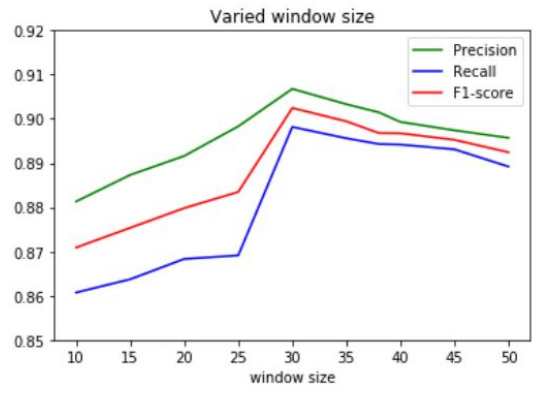


Fig. 15. Varied window size

We compared our method with traditional common statistical methods— Vector Auto Regression(VAR) and dynamic time wrapping k-NN classification, CNN classification.

Comparing to VAR which feeds past values of itself to a linear function, we got more than 10% in improvement. In contrast to these two classification models, our model improved F1-score by 13% on average. We also showed that our approach can be combined with transfer learning to save the training time and improve F1-score by 3.3% and Precision by 6.9%. Overall, our approach achieved almost 90% accuracy for both precision and recall while other classification or regression based methods only reached 70% $\sim$ 85%. In the future, we would like further improve our accuracy by investigating the abnormal patterns causing false negative or false positive in our evaluations.

References

[1] H. Kagermann, W. Lukas, and W. Wahlster. Industrie 4.0 - mit dem internet der dinge auf dem weg zur 4. industriellen revolution. *VDI Nachrichten*, 2011.

[2] Y. Lu, K.C. Morris, and S. Frechette. Current standards landscape for smart manufacturing systems. *NIST: Gaitehrsburg*, 8107, 2016.

[3] J. Liu, J. Guo, P. Orlik, M. Shibata, D. Nakahara, S. Mii, and M. Tak. Anomaly detection in manufacturing systems using structured neural networks. In *2018 13th World Congress on Intelligent Control and Automation (WCICA)*, pages 175–180, July 2018.

[4] F. Lopez, M. Saez, Y. Shao, E. C. Balta, J. Moyne, Z. M. Mao, K. Barton, and D. Tilbury. Categorization of anomalies in smart manufacturing systems to support the selection of detection mechanisms. *IEEE Robotics and Automation Letters*, 2(4):1885–1892, Oct 2017.

[5] Ke-Sheng Wang, Zhe Li, Jørgen Braaten, and Quan Yu. Interpretation and compensation of backlash error data in machine centers for intelligent predictive maintenance using anns. *Advances in Manufacturing*, 3(2):97–104, Jun 2015.

[6] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.

[7] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 04 1998.

[8] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July 2015. Association for Computational Linguistics.

[9] Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[10] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[11] Han Liu, Alexander Gegov, and Mihaela Cocea. Rule-based systems: a granular computing perspective. *Granular Computing*, 1(4):259–274, Dec 2016.

[12] Jean Roy. Rule-based expert system for maritime anomaly detection, 2010.

[13] Weng-Keen Wong, Andrew Moore, Gregory Cooper, and Michael Wagner. Rule-based anomaly pattern detection for detecting disease outbreaks. In *Eighteenth National Conference on Artificial Intelligence*, pages 217–223, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[14] C. Orsenigo and C. Vercellis. Combining discrete svm and fixed cardinality warping distances for multivariate time series classification. *Pattern Recogn.*, 43(11):3787–3794, November 2010.

[15] S. Seto, W. Zhang, and Y. Zhou. Multivariate time series classification using dynamic time warping template selection for human activity recognition. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 1399–1406, Dec 2015.

[16] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 262–270, New York, NY, USA, 2012. ACM.

[17] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J. Leon Zhao. Time series classification using multi-channels deep convolutional neural networks. In Feifei Li, Guoliang Li, Seung-won Hwang, Bin Yao, and Zhenjie Zhang, editors, *Web-Age Information Management*, pages 298–310, Cham, 2014. Springer International Publishing.

[18] Nauman Shahid, Ijaz Haider Naqvi, and Saad Bin Qaisar. One-class support vector machines: analysis of outlier detection for wireless sensor networks in harsh environments. *Artificial Intelligence Review*, 43(4):515–563, Apr 2015.

[19] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In *ESANN*, 2015.

[20] G. Loganathan, J. Samarabandu, and X. Wang. Sequence to sequence pattern learning algorithm for real-time anomaly detection in network traffic. In *2018 IEEE Canadian Conference on Electrical Computer Engineering (CCECE)*, pages 1–4, May 2018.

[21] Shuang Li, Yao Xie, Mehrdad Farajtabar, and Le Song. Detecting weak changes in dynamic events over networks. *IEEE Transactions on Signal and Information Processing over Networks*, PP, 03 2016.

[22] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.