Deep anomaly detection for industrial systems: a case study

Feng Xue[1], Weizhong Yan[1], Tianyi Wang[1], Hao Huang[2], Bojun Feng[1]

[1] *GE Research, Niskayuna, NY, 12065, USA*
{*xue,yan,wangt,bojun.feng*}*@ge.com*
[2] *GE Research, San Ramon, CA, 94583, USA*
hao.huang1@ge.com

ABSTRACT

We explore the use of deep neural networks for anomaly detection of industrial systems where the data are multivariate time series measurements. We formulate the problem as a self-supervised learning where data under normal operation are used to train a deep neural network autoregressive model, i.e., use a window of time series data to predict future data values. The aim of such a model is to learn to represent the system dynamic behavior under normal conditions, while expect higher model vs. measurement discrepancies under faulty conditions. In real world applications, many control settings are discrete in nature. In this paper, vector embedding and joint losses are employed to deal with such situations. Both LSTM and CNN based deep neural network backbones are studied on the Secure Water Treatment (SWaT) testbed datasets. Also, Support Vector Data Description (SVDD) method is adapted to such anomaly detection settings with deep neural networks. Evaluation methods and results are discussed based on the SWaT dataset along with potential pitfalls.

1. INTRODUCTION

Deep neural networks have made tremendous progress recent years in a number of areas, particularly in image and natural language processing (He, Zhang, Ren, & Sun, 2016; Liu et al., 2017; He, Gkioxari, Dollar, & Girshick, 2017; Devlin, Chang, Lee, & Toutanova, 2019). In a recent survey (Khan & Yairi, 2018), deep learning has been reported to perform competitively in a number of asset health management applications, namely anomaly detection and diagnosis.

Industrial systems, such as a power plant, are critical infrastructures where accurate anomaly detection is import to plant operation. The increasing deployment of sensors and systematic data collection systems present an opportunity to bring deep neural networks to bear on this problem. On the other hand, industrial systems are very complex by design. A typical industrial system has hundreds of tags including measurements, control signals, and operation settings. There are still challenges on a number of fronts when employing deep learning neural networks for industrial system anomaly detection:

- what is a proper problem formulation for neural network training?
- what data preprocess one should carry to present data to a neural network?
- what backbone neural network architecture is a preferred choice?

In the pursuit of these questions, this paper present a case study on the Secure Water Treatment (SWaT) testbed dataset (Goh, Adepu, Junejo, & Mathur, 2016). We formulate the problem as a self-supervised learning setting where data under normal operation are used to train a deep neural network autoregressive model, i.e., use a window of time series data to predict future data values. Self-supervised learning is a form of learning that trains a neural network on an artificially formulated learning task to learn a useful representation of the underlying problem. In our case, an autoregressive model is formulated as the self-supervised task to learn the representation of the underlying system's dynamic behavior under normal conditions, while expect higher model vs. measurement discrepancies under faulty conditions. In this regard, we can leverage the abundance of normal operation data usually available in real-world industrial applications. With this setup, we do not need to go through a laboriously manual labeling effort to select normal operation data. Instead, a heuristic procedure based on maintenance records can be easily used to define normal operation data. The fact that the majority of the operation data is normal also helps to ensure reasonable training data quality. In this paper, we also introduce vector embedding and joint losses as a way to deal with discrete control settings in real world applications.

2. RELATED WORK

Anomaly detection (AD) has been a extensively studied research topic. In literature, there are numerous AD methods available (Chandola, Banerjee, & Kumar, 2009; Zimek, Schubert, & Kriegel, 2012; Chalapathy & Chawla, 2019). Those anomaly detection methods can be broadly categorized into either supervised or semi-supervised. While supervised methods require both normal and abnormal samples, semi-supervised methods work on normal data only. Since anomalies are a rare event in most of real-world applications, semi-supervised methods tend to be better fitted and thus have been more popularly used. Given abundant samples of normal operation, semi-supervised anomaly detection learns the normal behavior (or to learn the boundary of the normal samples) and then detect any deviations from the normal behavior as anomalies. Traditional methods in both supervised and semi-supervised groups consists of various statistics and machine learning techniques. The survey papers (Chandola et al., 2009; Zimek et al., 2012) provided a good summary of these traditional methods. Our focus in this paper is on using deep learning for anomaly detection in the context of industrial systems applications where data used for detecting anomaly are primarily time-series sensor measurements. In this scope, deep learning has been dominantly used as for learning normal behavior, thus as semi-supervised anomaly detection. Generally speaking, those deep learning-based anomaly detection methods have two broad settings, *indirect (2-step)* and *direct(1-step)* (Yan, 2019; Chalapathy & Chawla, 2019).

Indirect setting. In indirect setting, deep learning is used as feature learning (or representation learning) and such learned features are then used as inputs to conventional detection models. In this category, deep generative networks, e.g., autoencoder (AE) (Yan, 2019; Zhou & Paffenroth, 2017), variational autoencoder (VAE) (Chen, Shi, Zhao, & Liang, 2019; An & Cho, 2015) and GANs (Li et al., 2019; Choi, Lim, Choi, & Kim, 2020), have been the popular choice in literature. While the network architectures used in the deep generative networks can be feedforward (FF), convolutional (CNN), and recurrent (RNN), for time series data, CNN and RNN are more effective as they are able to capture the temporal dependence of time series more effectively. To capture temporal dependence of time series, CNN (Wen & Keyes, 2019; Zhang et al., 2018) and LSTM (Malhotra, Vig, Shroff, & Agarwal, 2015; Chen et al., 2019; Guo et al., 2018) have been used as the network architecture of the autoencoder. Several works also introduced attention mechanism into time-series modeling, for example, (Yuan et al., 2018; Zhang et al., 2018). Furthermore, prediction-based deep learning for learning normal behavior has also been explored (Ahmad, Lavin, Purdy, & Agha, 2017). For example, in (Munir, Siddiqui, Dengel, & Ahmed, 2019), a time series prediction model that takes a window of time series as the input and predict next time stamp was used to model the time series normality. A deep CNN was used as the prediction model and the Euclidean distance of the prediction errors was used as the anomaly score for anomaly detection.

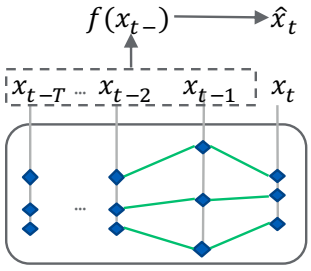

Figure 1. Autoregressive formulation: $\hat{x}_t = f(x_{t-})$

Direct setting. Unlike in indirect setting where the network training objective is not customized for anomaly detection, in Direct (1-step) setting, both feature learning and the anomaly detection model are learned jointly. Doing so ensures the network is optimal in terms of the objective criteria for anomaly detection. Literature in this category is relatively sparse. In (Ruff et al., 2018), a LSTM-based anomaly detection framework was introduced, where the parameters of both the LSTM and the anomaly detectors (OC-SVM and SVDD) are jointly optimized. Other works include (Chalapathy, Menon, & Chawla, 2018) and (Ergen & Kozat, 2019).

3. PROBLEM FORMULATION

In a typical industrial setting, we usually have abundance of normal operation data, while there are only very small number of faulty cases. Our formulation follows a self-supervised format, in which a model is trained to learn a representation of normal operation. Here, we formulate the self-supervised task as an autoregressive task. We want the model to learn an autoregressive representation of the underlying system, as shown in Figure 1. Let x_t be a data sample at time t, the autoregressive learning is try to estimate x_t given all observations up to time $t-1$. The model tries to learn an function, $\hat{x}_t = f(x_{t-})$, such that $\hat{x}_t$ is as close to the observation x_t as possible. In practice, we use a window length of T as the input instead of all the observations prior to time t. This window length is a parameter to be adjusted for a particular application.

For a given problem, we use the normal operation data to train a model to approximate $\hat{x}_t = f(x_{t-})$. The deviation between estimated and measured observation is obtained by a form of distance function: $d_t = d(\hat{x}_t, x_t)$. Such deviation is a measure of deviations to normal operation. Hence, a data sample with a deviation that is above a defined threshold is regarded as anomalous.

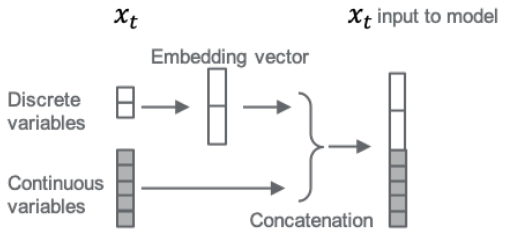

Figure 2. Discrete data embedding as input data to backbone models

4. DEEP LEARNING ARCHITECTURES

A number of neural network architecture can be used to learn the mentioned functional approximation. We explored Long Short-Term Memory (LSTM) recurrent neural network (Hochreiter & Schmidhuber, 1997), Convolution Neural Network (CNN), and traditionally fully connected Neural Network (NN) for our purpose. In the following subsections, we will describe the details of the models and setups for our experiments.

4.1. Discrete Input Embedding

In industrial applications, the time series data usually comprise of both discrete and continuous values. Sensor measurements are mostly collected as continuous signals, while control settings can be either continuous or discrete. For discrete data, especially non-ordinal data, a direct normalization that map these discrete values to a continuous space is rather arbitrary. To deal with this issue, we propose to jointly learn an embedding vector for each discrete variable along with other model parameters. This is inspired by neural language modeling approaches (Bengio, Ducharme, Vincent, & Jauvin, 2003), in which each word is mapped to a vector space of fixed size (the vector is called the embedding of the word). In our case, the embedding is a vector representation of the underlying discrete variable. These embedding vectors are included in the model parameters that behave as regular parameters. They are randomly initialized and then modified by the training algorithm like the other parameters in the model. This embedding transformation is illustrated in Figure 2. For each data sample x_t in the time series data, this embedding transformation is performed first before it is presented as input to the backbone model.

4.2. Mix Type of Target Variables

In the autoregressive setting, we have the choice to select a subset of the time series variables to serve as the target. A common setup is to estimate the measurement variables, i.e. usually continuous variables. However, it maybe beneficial to estimate the discrete variables as well. These discrete variables usually represent control settings. It might be advantageous to have the model to learn a control setting for the next time step given the current the state. Therefore, we have investigated two types of target variables in our study: 1) continuous target variables only; 2) mix type of continuous and discrete target variables.

The two settings are different mainly in their loss configurations. For continuous variables only setting, we use the mean square error as the loss function. Let x_t^c as the continuous target variables, the loss is simply as in Eq. 1, in which C is the set of continuous target variables.

$$L_{mse} = \frac{1}{C}\sum_{c\in C}\frac{1}{|C|}(\hat{x_t^c} - x_t^c)^2 \tag{1}$$

For the mix type setting, the loss is a joint of two parts: mean square error for continuous values and cross entropy for discrete values. In this case, the number of output for each discrete target variable is determined by its cardinality. For each discrete target variable d, the model output $\hat{x_t^d}$ is a probability vector with a length $|d|$, while the corresponding target variable is encoded as a one-hot vector. In this way, the cross entropy loss can be applied to these outputs. D is the set of discrete target variables.

$$L_{ce} = -\sum_{d\in D}\sum_{i\in|d|} x_t^{d_i}\log \hat{x^{d_i}}_t \tag{2}$$

For continuous variables only setting, loss $L = L_{mse}$. In the mixed type setting, the total loss is $L = L_{mse} + w_{ce}L_{ce}$, in which w_{ce} is the weight on cross entropy loss. We refer this as the *joint* approach in the experiment discussion later.

4.3. Recurrent Neural Network

The RNN network we use involves layers of LSTM and a linear layer. As show in Figure 3, the input vector is fed to the LSTM cell one at a time, and there is a linear layer that maps the LSTM hidden state at each time step to the target variables.

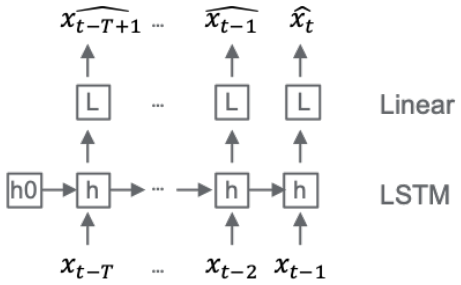

Figure 3. RNN architecture for autoregressive representation

In the training phase of the LSTM, we have the network to output all the estimation at each time step. There, we could

use all the outputs in the loss function or selective using only the last time step output. In testing, only the last time step is used as the estimation.

4.4. Convolution and Fully Connected Neural Network

For Convolution Neural Network (CNN) and fully connected Neural Network (NN), a time series $[x_{t-T}, ..., x_{t-1}]$ with length of T is the input to the network. In the case the CNN, the input is passed through layers of 1-d convolution. This is followed by a fully connected layer to map all the features from convolution operation to output $\hat{x}_t$. In the case of NN, the input is passed through multiple fully connected layers, and output $\hat{x}_t$.

4.5. Deep Support Vector Data Description Model

Traditional SVDD, a SVM-based one-class classifier, finds the smallest hypersphere that encloses the normal data in feature space. To be effective, the traditional SVDD, or shallow SVDD, requires labor intensive feature engineering. To address this issue, recently Ruff et al (Ruff et al., 2018) proposed the Deep SVDD, a neural network with a specially defined objective function such that it can learn feature representation and the smallest hypersphere together. They defined the optimization objective as in Eq. 3. It has two terms. The first one represents the average distance between the samples mapped to feature representation space via the network,$\psi(\boldsymbol{x}_i; \boldsymbol{W})$, and the center of the hypersphere, $\boldsymbol{c}$. And the second one is the standard regularization term.

$$L_{svdd} = \frac{1}{n}\sum_{i=1}^{n} \left\|\psi(\boldsymbol{x}_i; \boldsymbol{W}) - \boldsymbol{c}\right\|^2 + \frac{\lambda}{2}\sum_{l=1}^{L} \left\|\boldsymbol{W}^l\right\|_F^2 \qquad (3)$$

The network parameters are optimized using back propagation with the stochastic gradient descent (SGD). After the network is trained using normal samples, the anomaly score for a test sample $\boldsymbol{x}_t$ is given as: $s(\boldsymbol{x}_t) = \left\|\psi(\boldsymbol{x}_t; \boldsymbol{W}) - \boldsymbol{c}\right\|^2$.

5. CASE STUDY

We use the SWaT testbed data (Goh et al., 2016) from University of Technology and Design in Singapore for our case study. The testbed was built to facilitate cyber-security research. SWaT is a scaled down water treatment plant, capable of producing five gallons per minute of safe drinking water. It replicates a typical modern water treatment plant in cities. Raw water is treated in a six stage process, consisting of physical processes such as ultra filtration, de-chlorination, and reverse osmosis. SWaT consists of a layered communication network, Programmable Logic Controllers (PLCs), Human Machine Interfaces (HMIs), a Supervisory Control and Data Acquisition (SCADA) workstation, and a Historian. The plant process is shown in Figure 4. Details of the data collection and cyber attacks are described in (Goh et al., 2016).

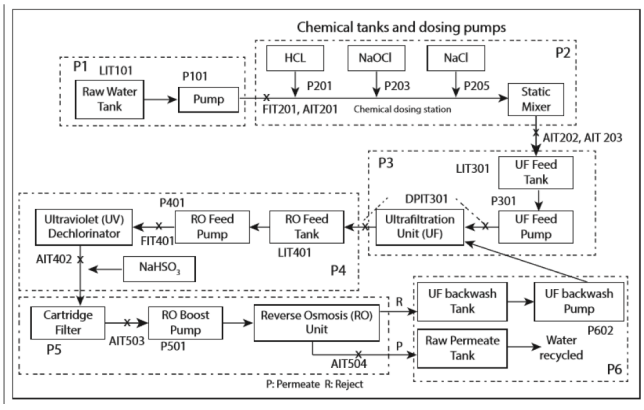

Figure 4. SWaT testbed process overview

5.1. Dataset

The data consists of 11 days continuous operation of the SWaT testbed. The first 7 days consist of a normal operation period. Cyber attacks were launched during the remaining four days. These attacks were of various intents and lasted between a few minutes to an hour. Over the whole period, data sample is collected at a frequency of 1/second. During the last 4 days, 41 episodes of cyber attack are simulated on the testbed. 36 of them are physical attacks. From a fault detection point view, these simulated physical cyber attacks are equivalent to malfunction of sensors or actuators. A sample of the 41 episodes are listed in Figure 5. For example, attacks 1, 2, and 4 are actual physical changes. This can be regarded as simulated malfunction of valves at different stages of water treatment process. Attack 3 can be regarded as a small drift of the sensor reading. Attack 5 is all about network, which we do not consider them in the study. We consider the 36 attacks that are physical in nature, which can be regarded as simulated faults in an industrial system.

Attack #	Start Time	End Time	Attack Point	Start State	Attack	Actual Change
1	28/12/2015 10:29:14	10:44:53	MV-101	MV-101 is closed	Open MV-101	Yes
2	28/12/2015 10:51:08	10:58:30	P-102	P-101 is on where as P-102 is off	Turn on P-102	Yes
3	28/12/2015 11:22:00	11:28:22	LIT-101	Water level between L and H	Increase by 1 mm every second	No
4	28/12/2015 11:47:39	11:54:08	MV-504	MV-504 is closed	Open MV-504	Yes
5	28/12/2015 11:58:20		No Physical Impact Attack			

Figure 5. Sample attacks from SWaT dataset

5.2. Performance Evaluation

In an industrial setting, operators typically concern about improving the detection rate of fault events while reducing the number of false positives over a certain period of normal operation. While fault event has only a small number of occurrences, normal operation spans over a long period of time. In real world, we thus would like to count the true positives at the event level, while false positives at the model's decision level.

In the SWaT dataset, there are no repeated events. We adopted a sample based evaluation method. In this case, there is an event start time and end time. Data samples from the time window $[E_s - T, E_e + T]$ are considered faulty operations. Since we do not have clear understanding about the underlying behavior of the system after a fault. It may take long time for the system to recover back to its normal operation condition. Such time to recovery would also depend on fault types and the underlying physical reactions and controls. In order to take such a phenomena into consideration, we define normal operation in two ways. One way is to take a separate test data (the last 1 day in our experiments) out of the first 7 days of normal operation period. This will ensure the quality of normal operation data. We refer this as normal hold off measure. The other way is to treat all the attack data samples that fall outside of all the attack event windows as normal operation. We refer this as attack data only measure. As we will discuss in the later section, such treatment may be prone to issues.

In our experiments, we use both ROC (Receiver Operating Characteristic) and PR (Precision-Recall) curve to evaluate the performance. AUC (Area Under Curve) and Average Precision are calculated as the overall performance measure.

5.3. Model Settings

After removing all the constant variable from the combined normal and faulty data, we have 25 continuous variables and 20 discrete variables. In the standard setup, we use all the variables as inputs to estimate all the continuous variables at the next time step. In the joint estimation setup, the model learns to estimate both continuous and discrete variables. Data samples are kept as the original data, which are sampled at 1/second.

For LSTM, we use 2 layers each with 50 hidden units, followed by a linear layer that map the hidden variable to output. For joint loss with LSTM as the backbone, each discrete variable has a vector length of its cardinality in the output. We set $w_{ce} = 1e - 2$ in our experiment. We use a time window $T = 120$ for both settings.

For CNN, we use 2 layers each with 50 channels of kernel size of 3, stride of 1, followed by a linear layer that map the output of the last convolutional layer to the final output, with time window size of $T = 10$.

For fully connected neural network, we use 2 layers each with 50 hidden units, followed by a linear layer that map the output of the last layer to the final output, using the same time window size is $T = 10$.

For deepDVDD, the mapping network is a feed forward neural network with one hidden layer. The number of neurons in the hidden layer is 50. The number of outputs (mapped dimension) of the network is 20. We use window size $T = 120$ in this case.

In all the experiments, we use the Adam optimizer (Kingma & Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and a learning rate of $lr = 10^{-3}$. LSTM model uses the standard *sigmoid* and *tanh* activation function, while ReLU is used in CNN and NN model.

We adopted a longer time window for LSTM and found it beneficiary. On the hand, a longer time window did not benefit CNN and fully connected neural network models. Hence, we adopted a shorter window for those models.

5.4. Result and Discussion

The LSTM results are shown in Figure 6. We can see the a big different between the two measures described earlier. With normal data hold off measure, we almost achieved perfect performance. However, using attack data only, the performance is not ideal. A further investigation indicates that that the system may never restore back to normal operation after a number of attacks.

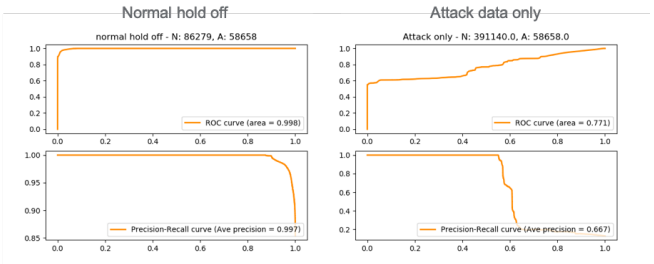

Figure 6. ROC and PR curve for LSTM

As shown in Figure 7, we can see that operation regime has shifted drastically after a number of attacks. P201 starts to oscillate between two settings and AIT201 starts to drift from normal operation regimes. In our experiments, almost all this period of operation is flagged as abnormal. We would argue that the model behavior is an appropriate one given such an out of normal operation.

We thus conducted an experiment of dropping variable A201 for both training and testing. We see a clear improvement on attack measurement as shown in Figure 8.

As far as training loss, we conducted experiments to compare two ways: 1) using all the outputs from the window; 2) using only the last output. From a estimation error point of view, it seems that the second way performs better as shown in Figure 9. Maintaining consistency between training and inference provides better performance.

In addition, we also demonstrated that joint LSTM achieved smaller representation error comparing with standard LSTM, as showed in Figure 10 in the same experiment setting.

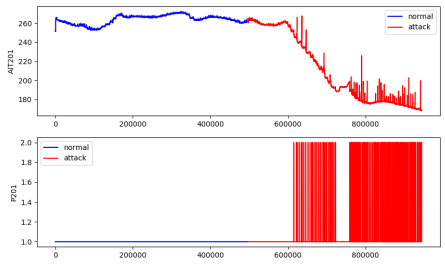

Figure 7. Signal and control operation regime change: red curve for attack period of time, blue for normal operation; P201 has maintained constant during normal operation

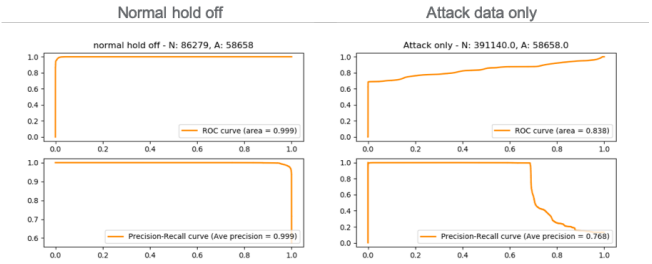

Figure 8. ROC and PR curve for LSTM after dropping A201

For overall performance with attack data only measure, the different models perform similarly as shown in Table 1, although LSTM with joint loss show some advantages.

Table 1. Fault detection comparison.

Methods	AUC	Ave_Precision
LSTM	0.838	0.768
CNN	0.869	0.788
NN	0.874	0.788
LSTM joint	**0.881**	**0.815**
DeepSVDD	0.834	0.748

For an easy comparison with a recent study (Inoue, Yamagata, Chen, Poskitt, & Sun, 2017) on the same dataset, we also select the best point solution based on F score. In Table 2, we compared results with two methods as reported in (Inoue et al., 2017): DNN and one-class SVM. The DNN method uses both LSTM and a staged partial estimation of actuator and sensor measurement for estimating outlier factor. LSTM with standard setup produce better performance in term of both precision and recall, while a number of other models in our setup produced better F scores.

6. CONCLUSION

In this paper, we described a setup on using deep neural networks for anomaly detection for industrial systems. The anomaly detection approach is formulated as a self-supervised task, i.e., learn the dynamic relationship of an

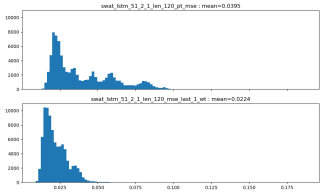

Figure 9. Testing error histogram from normal operation test data. Top: train with loss from the whole window; Bottom: train with loss from the last output only

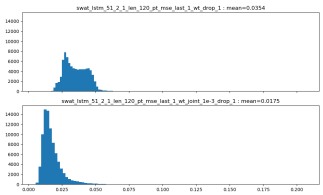

Figure 10. Testing error histogram from test normal operation data. Top: standard LSTM; Bottom: joint trained LSTM

industrial system as an autoregressive model. We introduced a number of techniques for dealing with industrial data when both discrete settings and continuous measurements are present. We also demonstrated that joint estimation of both continuous and discrete values can reduce estimation error and produce better overall performance comparing with its regular counterpart. We also compared a number of neural network architectures with a recent study on the same SWaT dataset, and showed that a number of models in our setup produce comparable or even better results. We also pointed out an issue with the SWaT dataset, and showed that the system operation largely drifted after some episodes of attacks. This makes the commonly adopted performance measure not reliable for the purpose of developing anomaly detection methods for industrial settings. As future work, we plan to conduct experiments using a different dataset to give us better views into the research questions we have posed.

ACKNOWLEDGMENT

This material is based upon work supported by the Department of Energy, National Energy Technology Laboratory under Award Number DE-FE0031763.

REFERENCES

Ahmad, S., Lavin, A., Purdy, S., & Agha, Z. (2017, November). Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, *262*, 134–147. doi: 10.1016/j.neucom.2017.04.070

An, J., & Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability..

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A Neural Probabilistic Language Model. *Journal of*

Table 2. Precision and recall select by best F score

	Methods	Precision	Recall	F score
Reference	DNN	0.98295	0.67847	0.80281
	One-class SVM	0.92500	0.69901	0.79628
Ours	LSTM	0.99402	0.68635	0.81202
	CNN	0.96897	0.67654	0.79677
	NN	0.96670	0.72458	**0.82831**
	LSTM joint	0.93730	0.71563	0.81161
	DeepSVDD	0.99060	0.63449	0.77353

Machine Learning Research, *3*(Feb), 1137–1155.

Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. *ArXiv*, *abs/1901.03407*.

Chalapathy, R., Menon, A. K., & Chawla, S. (2018). *Anomaly detection using one-class neural networks.*

Chandola, V., Banerjee, A., & Kumar, V. (2009, July). Anomaly detection: A survey. *ACM Computing Surveys*, *41*(3), 1–58. doi: 10.1145/1541880.1541882

Chen, R.-Q., Shi, G.-H., Zhao, W.-L., & Liang, C.-H. (2019). *Sequential vae-lstm for anomaly detection on time series.*

Choi, Y., Lim, H., Choi, H., & Kim, I. (2020). Gan-based anomaly detection and localization of multivariate time series data for power plant. In *2020 ieee international conference on big data and smart computing (bigcomp)* (p. 71-74).

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, May). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*. (arXiv: 1810.04805)

Ergen, T., & Kozat, S. S. (2019). Unsupervised anomaly detection with lstm neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15. doi: 10.1109/tnnls.2019.2935975

Goh, J., Adepu, S., Junejo, K. N., & Mathur, A. (2016). A Dataset to Support Research in the Design of Secure Water Treatment Systems. In *CRITIS*. doi: 10.1007/978-3-319-71368-7-8

Guo, Y., Liao, W., Wang, Q., Yu, L., Ji, T., & Li, P. (2018). Multidimensional time series anomaly detection: A gru-based gaussian mixture variational autoencoder approach. In *Acml.*

He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2017, October). Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)* (pp. 2980–2988). Venice: IEEE. doi: 10.1109/ICCV.2017.322

He, K., Zhang, X., Ren, S., & Sun, J. (2016, June). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778). Las Vegas, NV, USA: IEEE. doi: 10.1109/CVPR.2016.90

Hochreiter, S., & Schmidhuber, J. (1997, November). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. doi: 10.1162/neco.1997.9.8.1735

Inoue, J., Yamagata, Y., Chen, Y., Poskitt, C. M., & Sun, J. (2017, November). Anomaly Detection for a Water Treatment System Using Unsupervised Machine Learning. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)* (pp. 1058–1065). New Orleans, LA: IEEE. doi: 10.1109/ICDMW.2017.149

Khan, S., & Yairi, T. (2018, July). A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, *107*, 241–265. doi: 10.1016/j.ymssp.2017.11.024

Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *ICLR*. (arXiv: 1412.6980)

Li, D., Chen, D., Jin, B., Shi, L., Goh, J., & Ng, S.-K. (2019). Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In I. V. Tetko, V. Kůrková, P. Karpov, & F. Theis (Eds.), *Artificial neural networks and machine learning – icann 2019: Text and time series* (pp. 703–716). Cham: Springer International Publishing.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., & Song, L. (2017, July). SphereFace: Deep Hypersphere Embedding for Face Recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 6738–6746). Honolulu, HI: IEEE. doi: 10.1109/CVPR.2017.713

Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long short term memory networks for anomaly detection in time series. In *Esann.*

Munir, M., Siddiqui, S. A., Dengel, A., & Ahmed, S. (2019). DeepAnT: A deep learning approach for unsupervised anomaly detection in time series. *IEEE Access*, *7*, 1991–2005. doi: 10.1109/ACCESS.2018.2886457

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., ... Kloft, M. (2018, 10–15 Jul). Deep one-class classification. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (Vol. 80, pp. 4393–4402). Stockholmsmässan, Stockholm Sweden: PMLR.

Wen, T., & Keyes, R. (2019). *Time series anomaly detection using convolutional neural networks and transfer learning.*

Yan, W. (2019, December). Detecting gas turbine combustor anomalies using semi-supervised anomaly detection with deep representation learning. *Cognitive Computation*. doi: 10.1007/s12559-019-09710-7

Yuan, Y., Xun, G., Ma, F., Wang, Y., Du, N., Jia, K., ... Zhang, A. (2018). Muvan: A multi-view attention network for multivariate temporal data. In *2018 ieee international conference on data mining (icdm)* (p. 717-726).

Zhang, C., Song, D., Chen, Y., Feng, X., Lumezanu, C., Cheng, W., ... Chawla, N. V. (2018). *A deep neural*

network for unsupervised anomaly detection and diagnosis in multivariate time series data.

Zhou, C., & Paffenroth, R. C. (2017). Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining* (p. 665–674). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3097983.3098052

Zimek, A., Schubert, E., & Kriegel, H.-P. (2012, October). A survey on unsupervised outlier detection in high-dimensional numerical data. *Stat. Anal. Data Min.*, *5*(5), 363–387. doi: 10.1002/sam.11161