IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 19, 2021, accepted August 30, 2021, date of publication September 14, 2021, date of current version September 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3112545

SSS-AE: Anomaly Detection Using Self-Attention Based Sequence-to-Sequence Auto-Encoder in SMD Assembly Machine Sound

KI HYUN NAM, YOUNG JONG SONG, AND IL DONG YUN, (Member, IEEE)
Department of Computer Engineering, Hankuk University of Foreign Studies, Yongin 17035, South Korea

Corresponding author: Il Dong Yun (yun@hufs.ac.kr)

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) by the Ministry of Education, Science, Technology under Grant 2019R1A2C1085113, and in part by the Hankuk University of Foreign Studies Research Fund.

ABSTRACT A Surface-Mounted Device (SMD) assembly machine continuously assembles various products in real field. Unwanted situations such as assembly failure and device breakdown can occur at any time during the assembly process and result in costly losses. Anomaly detection techniques using deep learning are effective in detecting such abnormal situations. Two training scenarios, single-product learning and multi-product learning, can be considered for SMD anomaly detection workflows. Since there are not many products in previous studies, single-product learning is sufficient. However, multi-product learning is required when the number of products increases gradually. Successful multi-product learning on various assembly sound data in an industrial environment with limited resources requires efficient and light learning methods. In this paper, we propose robust model and effective data preprocessing method, Self-Attention based Sequence-to-Sequence Auto-Encoder (SSS-AE) and Temporal Adaptive Average Pooling (TAAP). For more accurate evaluation compared with the previous SMD anomaly detection studies, a new large-scale SMD dataset containing observed real abnormal products were collected and evaluated. As a result, we show that SSS-AE and TAAP are powerful and practical approaches for both single-product learning and multi-product learning.

INDEX TERMS Anomaly detection, auto-encoder, self-attention, sequence-to-sequence.

I. INTRODUCTION

Anomaly detection refers to the problem of detecting data that does not match the normal situation [1], [2] and extends to detailed domain such as damage detection [3]. At the SMD assembly site, various products are subsequently assembled in a continuous flow. In this process, it is very important to detect abnormal situations such as assembly failure and device breakdown immediately, because missing the abnormal situations can result in a huge cost loss. When SMD machine assembles a particular product, a sound occurs, which is an important clue in determining which product is assembled because it is dependent on the assembly operation. Considering the difficulty in collecting sufficient abnormal data in real-field, unsupervised learning that learns only normal data is a reasonable option.

The associate editor coordinating the review of this manuscript and approving it for publication was Mauro Fadda.

In unsupervised learning, Auto-encoder [4], [5] models are first trained for reconstructing the normal data, abnormal data are then detected based on their reconstruction error [6]. Existing SMD anomaly detection studies using auto-encoder include the works of Oh *et al.* [7] and Park *et al.* [8]. Oh *et al.* achieved good performance using auto-encoder based on CNN. But, it has a disadvantage that takes a long time to learn because a large number of parameters are required. On the other hand, Park *et al.* proposed Fast Adaptive RNN Encoder–Decoder (FARED), an encoder-decoder model based on stacked RNN. Unlike CNN based auto-encoder [7], the FARED has few parameters. Therefore, it show a fast training time that can be applied in real-time at the real site.

Two training scenarios can be considered to apply SMD anomaly detection models to workflow of real-field. Single-product learning scenario is that one anomaly detection model learns only one normal product. Single-product

learning is mainly chosen in situations that need to learn new normal product in real time. For single-product learning, the number of models is linearly proportional to the number of target products. Multi-product learning scenario means that one anomaly detection model learns multiple normal products simultaneously. The multi-product learning model learns and remembers the normal products handled by the single-product learning models in the past. In previous SMD anomaly detection studies, single-product learning is sufficient because the number of products is small. However the number of products increases every day in real-field, which enhances the necessity of multi-product learning instead of single-product learning. Furthermore, in order to attempt multi-product learning, the number of products collected must vary sufficiently, which could not be attempted in previous studies. In this work, we attempt multi-product learning for the first time with newly collected large dataset.

To successfully perform two training scenarios under the industrial situations where resources are not enough, we propose two effective methods: Self-Attention based Sequence-to-Sequence Auto-Encoder (SSS-AE) and Temporal Adaptive Average Pooling (TAAP). As mentioned earlier, we base auto-encoder because it is proper to learn representation without label. When training dataset grows and becomes complex, such as multi-product learning, increasing the size of neural network for higher capacity is a convincing choice. However, in our case, we introduce Self-Attention [9] and Sequence-to-Sequence (Seq2Seq) [10] because we need to minimize the increase in model size. Self-attention allows the model to focus on meaningful information by calculating internal relationships to the input sequential vector itself. This is effective for auto-encoder to learn representation without any external information. Seq2Seq can pass the last hidden state of the encoder to the initial state of the decoder. In other words, the encoder encapsulates the pattern of sequential data and passes it to the decoder. Therefore, Seq2Seq allows auto-encoder to make abundant use of temporal information when learning representation for the sequential data domain. These two mechanisms can easily scale with little change, even in traditional RNN models, and require only a small amount of parameters increase. We demonstrate that the SSS-AE outperforms the previous model in single-product learning and multi-product learning through performance comparison. SSS-AE is one of main contributions. Furthermore, we introduce Temporal Adaptive Average Pooling (TAAP), an effective data preprocessing method that can significantly improve anomaly detection performance. This also works effectively on previous model as well. TAAP is another of main contributions. Experimental results explain how much each mechanism contributes to performance improvement and, as a result, verify that the SSS-AE and TAAP have a remarkable ability to reliably manage large normal products. To conduct various and accurate experiments compared to previous studies, we directly collected various products and consisted a new large-scale SMD dataset. We plan to release the large-scale SMD dataset with additional information in follow-up study. We release our SMD dataset examples and source code publicly available.[1]

This paper is organized as follows: We describe background in Section 2. We introduce new large-scale SMD dataset for experiments in Section 3. We explain data preprocessing in Section 4. We propose the robust model SSAE in Section 5. We show experimental results in Section 6. We conclude our conclusions in Section 7.

II. BACKGROUND

Recurrent Neural Network (RNN) is used to learn the temporal dependencies of sequential data by applying memory or state to feedforward neural networks. RNNs have made many advances in various sequential data domains, including speech processing such as speech recognition [11]–[15] and speech synthesis [16], [17], and language processing such as machine translation [10], [18]. The ability of RNNs to model sequential data is sufficiently validated, but when learning long sequential data, the vanishing gradient problem occurs.

Long Short-Term Memory (LSTM) [19] is a modified RNN using three gates (i.e. input, forget, and output), and two states (i.e. hidden state and cell state) to solve this problem. LSTM is calculated at time step t as follows:

$$i_t = \sigma(\mathbf{W}_{ii}x_t + \mathbf{W}_{hi}h_{t-1} + b_i) \quad (1)$$
$$f_t = \sigma(\mathbf{W}_{if}x_t + \mathbf{W}_{hf}h_{t-1} + b_f) \quad (2)$$
$$o_t = \sigma(\mathbf{W}_{io}x_t + \mathbf{W}_{ho}h_{t-1} + b_o) \quad (3)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh(\mathbf{W}_{ic}x_t + \mathbf{W}_{hc}h_{t-1} + b_c) \quad (4)$$
$$h_t = o_t \circ \tanh(c_t) \quad (5)$$

where x_t is the input at time step t, and i_t, f_t, o_t are the input, forget, and output gates, and h_t, c_t are the hidden and cell states, respectively. The $\mathbf{W}$, b, σ and $\circ$ are denote weight matrices, bias vectors, the sigmoid function and the Hadamard product (i.e. element-wise product).

Sequence-to-Sequence (Seq2Seq) [10], [18] model is one of the RNN-based Encoder-Decoder models. In conventional encoder-decoder model, encoder learns to encode source data into latent representation, and decoder learns to decode a given latent representation into target data. Seq2Seq is an enhanced RNN encoder-decoder to model sequential data. In Seq2Seq, the RNN of encoder calculates the source sequence $\mathbf{x} = [x_0, x_1, \ldots, x_{L-1}]$ sequentially and outputs it to the hidden states $\mathbf{z} = [z_0, z_2, \ldots, z_{L-1}]$. If the RNN layer is LSTM, hidden states $\mathbf{z}$ are calculated according to (5). An important feature of Seq2Seq is that it passes the last hidden state z_{L-1} of the encoder RNN to the initial state of the decoder RNN. This means that encoder passes summary information of the source sequence to decoder. Summary information can be a temporal feature of sequential data. To predict y_t of the target sequence $\mathbf{y} = [y_0, y_1, \ldots, y_{T-1}]$ as timestep t, the input of the decoder has two options as follows:

[1] https://github.com/HUFS-VLab/tf-SSS-AE

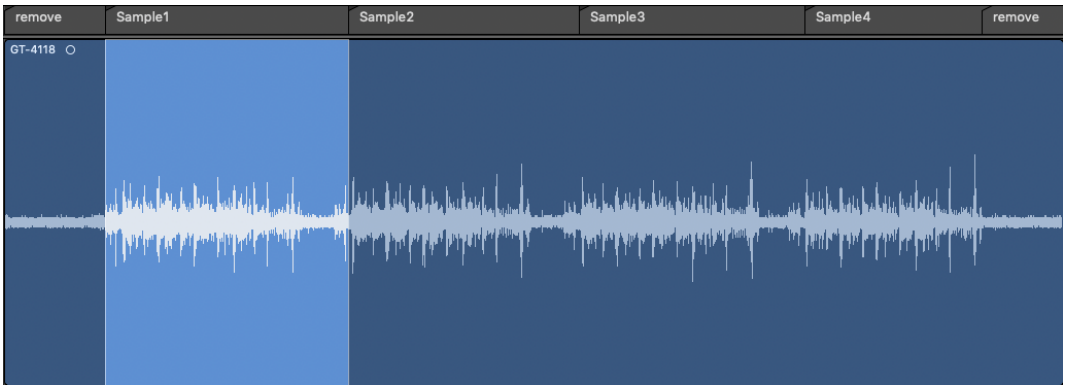


FIGURE 1. Example of sound data collection process.

TABLE 1. Large SMD dataset consisting of 30 normal products and 6 abnormal products. Bold text means abnormal products. Abnormal products are excluded from training.

Name	Time($\mu\pm\sigma$)	Amount	Name	Time($\mu\pm\sigma$)	Amount	Name	Time($\mu\pm\sigma$)	Amount
AT2-AB	40.3s±2.1s	100(30/70)	GT-4118	35.9s±4.1s	43	ST-3214	26.8s±1.5s	108
AT2-IN88	8.3s±0s	63	**GT-4118-1**	**33.9s±3.3s**	**14**	**ST-3214-1**	**27s±1.2s**	**100**
C-SERIES	12.6±0.8s	41	**GT-4118-2**	**28.4s±2.7s**	**100**	**ST-3214-2**	**27.5s±0.9s**	**100**
CLR-085	34s±3.5s	51	HV-M1230C	49.8s±2.6s	39	ST-3428	30.3s±2.7s	89
CT-C112B	45.4s±3.6s	22	MG-A121H	30.1s±2s	68	ST-3624	28.5s±3.6s	77
CT-C112T	40.1s±0.9s	24	NA-9289	10.5s±0.1s	55	ST-3704	27.9s±1.5s	54
CT-C121B	33.9s±7.7s	18	NA-9473	13.1s±0s	53	ST-3708	40s±2.9s	64
CT-C134B	44.4s±14.4s	33	RT-12DF	36.6s±0.9s	29	**ST-3708-1**	**31s±2.1s**	**35**
CT-C134T	34.5s±2.9s	44	ST-2524	20.3s±0.7s	46	**ST-3708-2**	**36s±7.4s**	**82**
GT-3214	28.3s±2.6s	33	ST-2624	19.5s±0.8s	43	ST-3804	25.2s±1.3s	83
GT-5112	25.6s±1.7s	51	ST-2744	17.4s±0.6s	76	ST-7111	8.7s±0.9s	30
GW-9XXX	20.8s±2.1s	93	ST-3424	25.9s±3.5s	81	TSIO-2002	19.8s±0.7s	65

1) y_{t-1} *of Target Sequence* **y***:* This option is primarily chosen for speech recognition [14] or machine translation [10], [18].
2) *Output of the Encoder:* This is usually considered in unsupervised learning task [20], [21].

Existing studies using Seq2Seq models mainly use the first option, but they have poor performance on long sequence data.

Attention mechanism is a powerful method to overcome the vulnerability of Seq2Seq to the long sequence data [14], [16], [17]. Attention calculates and selects important hidden states (scoring and focusing) for input y_{t-1} in the entire hidden states of the encoder when the decoder generates output y_t from input y_{t-1}. Generally, scoring is similar to similarity measure, and various score functions [22], [23] have proposed, including cosine similarity [24]. Self-attention or intra-attention is a mechanism for calculating the relationship between different positions of the same input. For example, the relationship between different words in the same sentence. By this mechanism, self-attention can enrich the input representation without external information. Self-attention has not only achieved great results in natural language processing [9], [25]–[28], but has also succeeded in many other tasks in various areas [29].

Auto-encoder [4], [5] is an unsupervised learning model composed of encoder and decoder, and it learns the representation of input data by compressing the input data and reconstructing it back. Auto-encoder successfully performs learning representation in various areas, including text generation [30] and anomaly detection [2], [31], [32]. The task of auto-encoder targeting sequence data is to learn representation that reflect temporal information. Seq2Seq helps auto-encoder learn the representation of sequence data [20], [33]–[35]. This is because Seq2Seq is specialized in extracting temporal features of input sequence. Recently, auto-encoder with attention have also been studied [30], [36]–[38]. For Seq2Seq based auto-encoder with self-attention [30], [36], [37], self-attention calculates the relationship between different positions or states inside the input sequence. This relationship highlights which states of the input representation the encoder or decoder should focus on.

III. DATASET

In previous study [8], there were only 6 products. However, since the number of training products increases rapidly in the real-field, this is a motivation to supervise many products simultaneously. The data collection process is the same as in the previous studies [7], [8], but 30 new normal products were collected for various experiments. Figure 1 shows an example of the data collection process. Moreover, abnormal data of three products *GT-4118*, *ST-3214*, and *ST-3708* were collected from actual errors and were classified with two

different error levels for each of them. The level of error was categorized according to the level of noise a person hears. These two error levels are expressed as *Error 1* and *Error 2* respectively and are called *XX-OOO-1* and *XX-OOO-2* by product name. *Error 1* means the degree to which it is slightly recognizable when a person hears it because there is not much noise. *Error 2* means the degree to which it is easily recognizable because it contains distinctly loud noise. As a result, the dataset is composed of 36 different products, and we name this dataset a Large SMD dataset, **LSMD**. Error 2 products were recorded at 96kHz, and all other products were recorded at 192kHz. More specific information is summarized in Table 1. To show that training is possible even in a short length of time, 30 samples were randomly extracted for training, which takes approximately 15 minutes. If the total number of samples for the product was less than 30, the training and test samples were randomly separated at a ratio of 7:3. Since abnormal products are not subject to training, all of them are used as test samples. With these guidelines, for single-product learning and multi-product learning, the training and test set are configured and we summarized this procedure in Algorithm 1 and 2.

Algorithm 1 Split Train/Test Set for Single-Product Learning

Input: A whole dataset $\mathcal{D} = \{p^{(i)}\}_{i=1}^{N}$, where p is a set of samples for a product and N is the number of the products.
Output: Train and test set for i-th product training
Training set $Tr^{(i)} = p^{(i)}_{1:M}$, where M is 30 if $M > 30$ else 70% of K and K is the number of $p^{(i)}$
Test set $Ts^{(i)} = \{Positive\ samples, Negative\ samples\}$
Positive samples $= p^{(i)}_{M:K}$
Negative samples = a set of samples of other products other than $p^{(i)}$

Algorithm 2 Split Train/Test Set for Multi-Product Learning

Input: A whole dataset $\mathcal{D} = \{p^{(i)}\}_{i=1}^{N}$, where p is a set of samples for a product and N is the number of the products.
Output: Train and test set for C products training
Training set $Tr^{C} = \{Tr^{(i)}\}_{i=1}^{C}$
Test set $Ts^{C} = \{Positive\ samples, Negative\ samples\}$
Positive samples $= \{Ts^{(i)}_{Pos}\}_{i=1}^{C}$, where *Pos* is *Positive*
Negative samples = a set of samples of other products other than C products

IV. DATA PREPROCESSING

We use the sound data by converting it to the spectrogram s in the time-frequency domain. $s \in \mathcal{R}^{T \times D}$, where T is the length of the spectrogram or the number of the spectrums and D is the number of frequency bins. So, $s_{i,j}$ denotes the j-th frequency bin in the i-th spectrum. Previous studies input data into the models by truncating sound sample [8] or spectrogram [7] to several fixed size data (i.e. fragmented data) after or during data preprocessing. Because, long sequence data is a problem for RNN based models. Park *et al.* average the spectrogram of the fragmented data, compresses it into one spectrum, and creates one input sequence consisting of a series of spectra. In this paper, this method called Local Average Pooling (LAP). Previous methods only provide fragmented acoustic patterns of sound to model. Therefore, we introduce Temporal Adaptive Average Pooling (TAAP), a new data preprocessing method that allows the length of the source spectrogram to be compressed to the user's target length. TAAP can create relatively short sequence data whereas preserving the overall acoustic pattern of the source sound as much as possible.

Since the length T of the spectrogram is variable, TAAP compress using an adaptive kernel to reduce the variable length T to the desired length L (shorter than T). TAAP only reduce the length of the spectrogram by averaging. In TAAP, the size of kernel has $k \times 1$. As k must be an integer, T must be an integer multiple of L. Otherwise, the source spectrogram s will have to be truncated or padded by zero. In this paper, we use zero-padding. TAAP is simply calculated as follows:

$$k = \frac{T}{L} \tag{6}$$

$$\bar{s}_j = \frac{1}{k} \sum_{i=j\times k}^{(j+1)\times k-1} s_i \tag{7}$$

where $j = 0, 1, \cdots, L-1$. According to (7), a sequence $\bar{\mathbf{s}} = [\bar{s}_0, \ldots, \bar{s}_{L-1}]$ could be derived from the original sequence s. This means that each of the k spectrums can be averaged into one spectrum, eliminating unnecessary temporal information, while reducing the time resolution. Using TAAP can provide the following benefits such as:

1) Regardless of the length of the input sequence, it can be reduced to a fixed length, reducing sequential operations and reliably training the model.
2) Data can be easily organized in batch unit without truncating or padding, thus minimize data corruption and increase the computation speed through mini-batch.

After TAAP, we normalize the sequence $\bar{\mathbf{s}}$ by min-max normalization.

V. MODEL

In this section, we describe the SSS-AE in detail. Figure 2 illustrates the SSS-AE model. The SSS-AE consists of attentional encoder and attentional decoder. For attention, SSS-AE uses the Multi-Head Self-Attention of Transformer [9]. All RNN layers of the SSS-AE are Long Short Term Memory (LSTM) [19] layer. We denote $\mathbf{x} = [x_0, x_1, \ldots, x_{L-1}]$ as the source sequence data, and $\mathbf{y} = [y_0, y_1, \ldots, y_{L-1}]$ as output sequence data. The source sequence $\mathbf{x}$ same as acoustic feature $\bar{\mathbf{s}}$. SSS-AE models the output $\mathbf{y}$ to be reconstructed same to the input $\mathbf{x}$ as follows:

$$\mathbf{y} = \mathbf{x} \tag{8}$$

$$P(\mathbf{y} \mid \mathbf{z}) = P(\mathbf{z} \mid \mathbf{x})P(\mathbf{x}) \tag{9}$$

where $\mathbf{z} = [z_0, z_2, \ldots, z_{L-1}]$ is the representation.

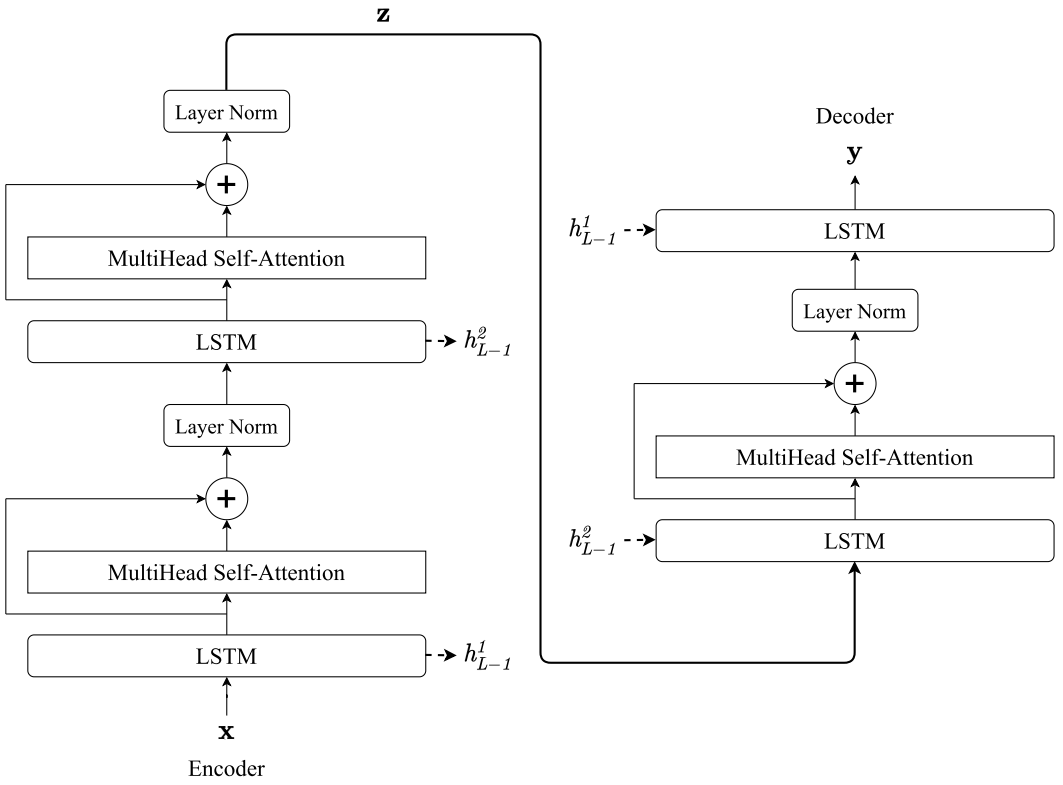


FIGURE 2. Illustration of the self-attention sequence-to-sequence based auto-encoder.

TABLE 2. Description of the neural network models for experiments. Encoder and decoder attributes denote the ratio of dimension reduction or increase of LSTMs in encoder and decoder. For example, (r_1, r_2) means that the first LSTM and the second LSTM scale dimension of input representation by r_1 and r_2.

Models	Seq2Seq	Encoder	Decoder	Attention	Parameters
Enc-Dec-AE	-	$(1,\frac{1}{2})$	(1,2)	-	122,560
Seq2Seq-AE	✓	$(1,\frac{1}{2})$	(1,2)	-	122,560
Seq2Seq-AE-large	✓	$(1,\frac{1}{2},1,\frac{1}{2})$	(1,2,1,2)	-	245,120
SSS-AE	✓	$(1,\frac{1}{2})$	(1,2)	✓	151,840

We train the model to minimize Mean Squared Error (MSE) function as the loss function $\mathcal{L}$, which computes the difference between a set $\mathbf{X}$ of source sequence $\mathbf{x}$ and a set $\mathbf{Y}$ of output sequence $\mathbf{y}$.

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{X}_i - \mathbf{Y}_i\|^2 \tag{10}$$

where n is the number of training data.

A. ENCODER

The encoder and decoder both consist of N blocks. The block is a sequential connection of LSTM, Multi-Head Self-Attention, and Layer Normalization [39]. All LSTM's, including the decoder, perform to calculate temporal relationship or information for the input representation. Multi Head Self-Attention calculates the relationship between different hidden states of the input representation. This process enriches the input representation. We use a Residual Connection [40] to add the outputs of each sub-layer, and Layer Normalization. The encoder's LSTM maps the input representation to a low-dimensional space through dimension reduction. The last hidden state h^i_{L-1} of the i-th encoder's LSTM learns the temporal summary of the input representation and passes it to the initial state of the decoder's LSTM. The last hidden state h^i_{L-1} is also part of the latent representation. As a result, the encoder maps the source $\mathbf{x}$ to the latent representation $\mathbf{z}$.

B. DECODER

The decoder is similar to the encoder, but the last top block uses only one LSTM layer for reconstruction. The initial state of the j-th decoder's LSTM is entered with the last hidden state h^i_{L-1} of the i-th encoder's LSTM. This can provide temporal summary in advance when the decoder's LSTM maps the input representation to a high-dimensional space. We construct the encoder and decoder in a symmetric

structure to transfer temporal summary information of the same feature-level. Therefore, i and j are $j = N + 1 - i$. As a result, the decoder reconstructs the source data $\mathbf{x}$ from the latent representation $\mathbf{z}$. In this work, we construct two blocks each in the encoder and decoder. The second block of the encoder and decoder scale dimension of the input representation by $1/2$ and 2.

VI. EXPERIMENTAL RESULTS

A. EXPERIMENTAL SETUP

All sound samples were transformed to mel-spectrograms, which are calculated with Hann window, 2048 window size, 512 stride size and 80 mel-filter banks using `Librosa`. We used the magnitude of mel-spectrogram.

In all experiments, we focus on which factors have a significant impact on anomaly detection performance. We pay attention to 5 factors as the following:

1) *Training Scenarios:* We conduct experiments around two training scenarios. In single-product learning, the amount of training data is quite small (in this work, around 15 minutes per product). Therefore, we note whether the single-product learning model is robust for small training data. For multi-product learning, we experiment by increasing the number of training products gradually. The added products are selected in the order of close reconstruction error for the trained product of the single-product learning model. In multi-product learning, we assess whether the model can distinguish abnormal products well, even if the size and distribution of the training dataset increases and becomes more complex.
2) *Data Preprocessing:* We conjecture that providing intact acoustic pattern of sound to sound-based anomaly detection models is the most important factor in performance. We demonstrate the necessity of suitable data preprocessing through the results of applying LAP and TAAP.
3) *Reconstruction Target:* A typical auto-encoder reconstructs the output to the same as the input sequence. However, some studies [41], [42], including Park *et al.* [8], reconstruct next sequence data that appears after the input sequence. This is working well, but it is not the dominant method, so we check the actual effect.
4) *Mechanisms of Neural Network:* Anomaly detection model using auto-encoder rely on the latent space learning capability of auto-encoder. We check what mechanisms are effective in learning latent space for sequence data.
5) *Parameter Size:* We verify that the size of the model is related to performance. This supports that the performance of the proposed model is not just due to increasing in the number of parameters.

Table 2 summarizes the neural network models used in all experiments. All experiments were optimized using Adam optimizer [43], mini-batches of size 64. Models were early-stopped based on the loss value, but generally trained for up to 5000 epochs. The learning rate was initialized by 0.005. Attention is Multi-Head Self Attention and the number of heads is 8. All training were performed on a machine consisting of a single NVIDIA GTX 1080 TI GPU with 11 GB of memory, Intel Xeon, and 16 GB of DDR4 RAM.

In the case of anomaly detection using reconstruction error like MSE, the detection performance continues to change depending on how the threshold is set. Therefore, most of the anomaly detection studies use the receiver operating characteristic curve (ROC curve) or area under the ROC curve (AUC) as metric. In this work, the evaluation is based on the AUC score.

B. COMPARISON RESULTS

We denote baseline models for FARED and SSS-AE as follows: 1) SSS-AE: it is the SSS-AE model adopting TAAP, and reconstructs the input sequence $\mathbf{x}$, 2) FARED: it means the Enc-Dec-4 model using LAP and reconstructs the next sequence $\mathbf{x}_{next}$ that appears after the input sequence $\mathbf{x}$. Table 3 shows the results of 4 models under single-product learning with 5 factors. The left and right sides of the $\rightarrow$ mean the input and reconstructed data. Table 4 depicts the results of 2 baseline models for SSS-AE and FARED, under multi-product learning. Multi-product learning experiments gradually increase training products based on the reconstruction error distributions of single-product learning on three products *GT-4118, ST-3214*, and *ST-3708*. Figure 3 is the average result for Table 4. In Figure 3, performance sometimes improves as the number of training products increases. We conjecture that the newly added products did not interfere with training because they are quite similar to the previous learned products. Table 5 shows the results of 4 models under multi-product learning. In all scenarios, FARED has poor performance. On the other hand, SSS-AE demonstrates pretty performance improvement. In particular, this improvement is higher in multi-product learning. We give additional information in the Appendix.

1) DATA PREPROCESSING

In all models and scenarios, TAAP performs much better than LAP. This demonstrates that TAAP is powerful in handling information from the sound data. For TAAP, the best performance is obtained by setting the target length L to 16. However, setting the length L too short or too long caused performance degradation. As a result, we can find that among the 5 factors, data preprocessing contributes the most to performance improvement. These results support the necessity of effective data preprocessing.

2) RECONSTRUCTION TARGET

Reconstructing the next sequence $\mathbf{x}_{next}$ is not bad, but reconstructing the input sequence $\mathbf{x}$ still gives higher performance. The difference in performance depending on the reconstruction target is not small. This implies that methods

TABLE 3. AUC score of each auto-encoder model on 30 normal products under single-product learning with 5 factors. The left and right sides of the right arrow mean the input and reconstructed data. x means the input sequence. x_{next} means the next sequence that appears the input sequence x. In TAAP, we set the target length L to 16.

Models	Enc-Dec-AE	Seq2Seq-AE	Seq2Seq-AE-large	SSS-AE
Local-Average-Pooling				
$\mathbf{x} \rightarrow \mathbf{x}_{next}$	0.903	0.918	0.910	0.921
$\mathbf{x} \rightarrow \mathbf{x}$	0.934	0.946	0.930	**0.948**
Temporal-Adaptive-Average-Pooling				
$\mathbf{x} \rightarrow \mathbf{x}_{next}$	0.964	0.979	0.978	0.980
$\mathbf{x} \rightarrow \mathbf{x}$	0.991	**0.995**	0.990	**0.995**

TABLE 4. Description of AUC score for the number of training products under multi-product learning. We use 2 baseline models, SSS-AE and FARED. N product means the number of training products. The right dash arrow indicates that the number of training products is increasing gradually, starting from the single normal product.

N products	3	6	9	12	15	18	21	24	27	Average
FARED										
GT-4118 --→	0.896	0.910	0.873	0.858	0.821	0.767	0.773	0.787	0.814	0.865
ST-3214 --→	0.893	0.899	0.845	0.818	0.874	0.820	0.799	0.791	0.814	0.875
ST-3708 --→	0.893	0.902	0.845	0.858	0.813	0.816	0.835	0.791	0.814	0.876
SSS-AE										
GT-4118 --→	0.970	0.956	0.973	0.998	0.939	0.934	0.936	0.927	0.937	0.952
ST-3214 --→	0.986	0.968	0.973	0.974	0.995	0.935	0.936	0.934	0.938	0.960
ST-3708 --→	0.984	0.953	0.989	0.995	0.949	0.942	0.941	0.925	0.936	0.957

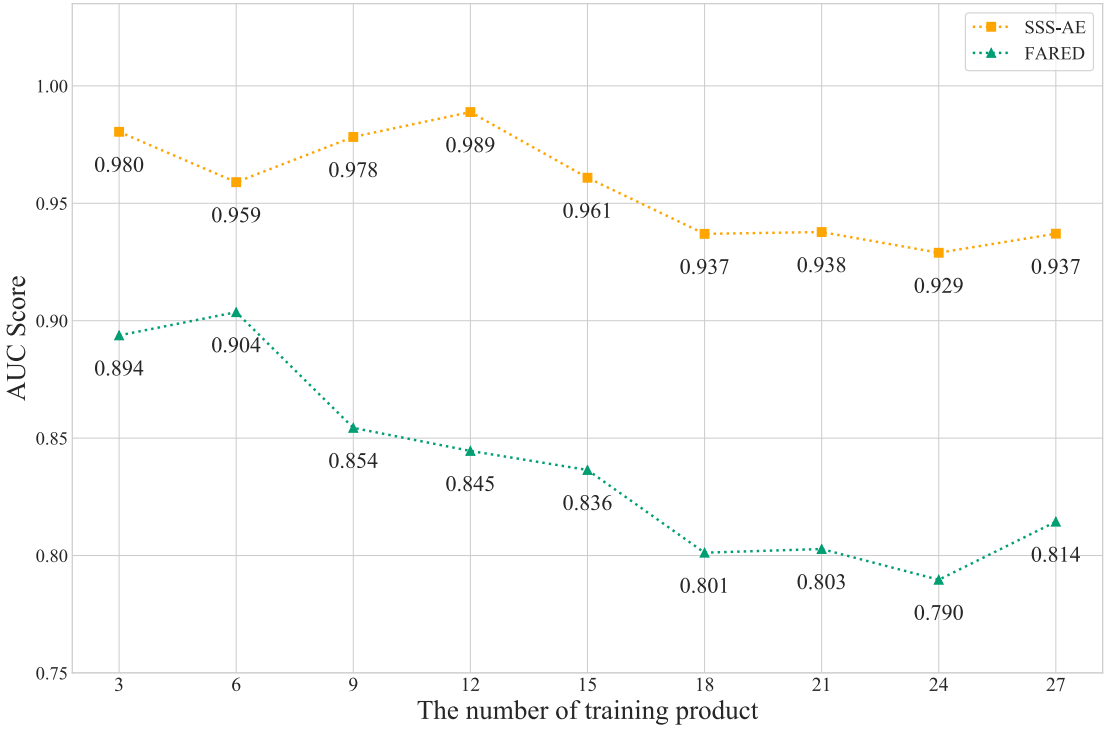


FIGURE 3. Average AUC score of 2 baseline models, SSS-AE and FARED for Table 4.

TABLE 5. AUC score of each auto-encoder model under multi-product learning.

Models	Enc-Dec-AE + LAP + $\mathbf{x} \rightarrow \mathbf{x}_{next}$	Enc-Dec-AE + TAAP + $\mathbf{x} \rightarrow \mathbf{x}$	Seq2Seq-AE + TAAP + $\mathbf{x} \rightarrow \mathbf{x}$	Seq2Seq-AE-large + TAAP + $\mathbf{x} \rightarrow \mathbf{x}$	SSS-AE + TAAP + $\mathbf{x} \rightarrow \mathbf{x}$
AUC	0.838	0.926	0.936	0.944	**0.956**

such as reconstructing the next sequence that appears after the input sequence can make the auto-encoder difficult to train.

3) MODELS

Despite same parameter size, Seq2Seq based model outperforms simple encoder-decoder in all scenarios. This proves

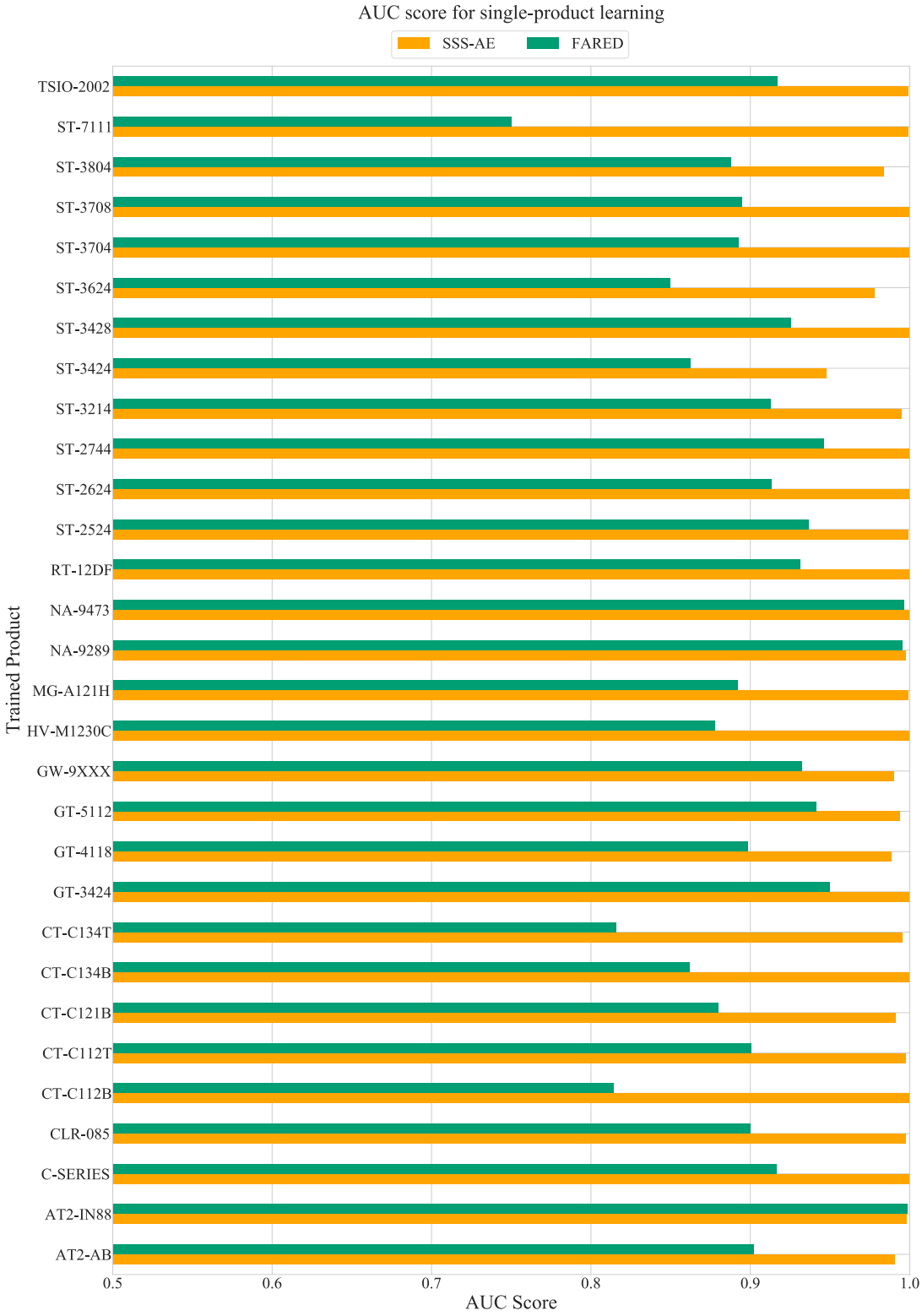
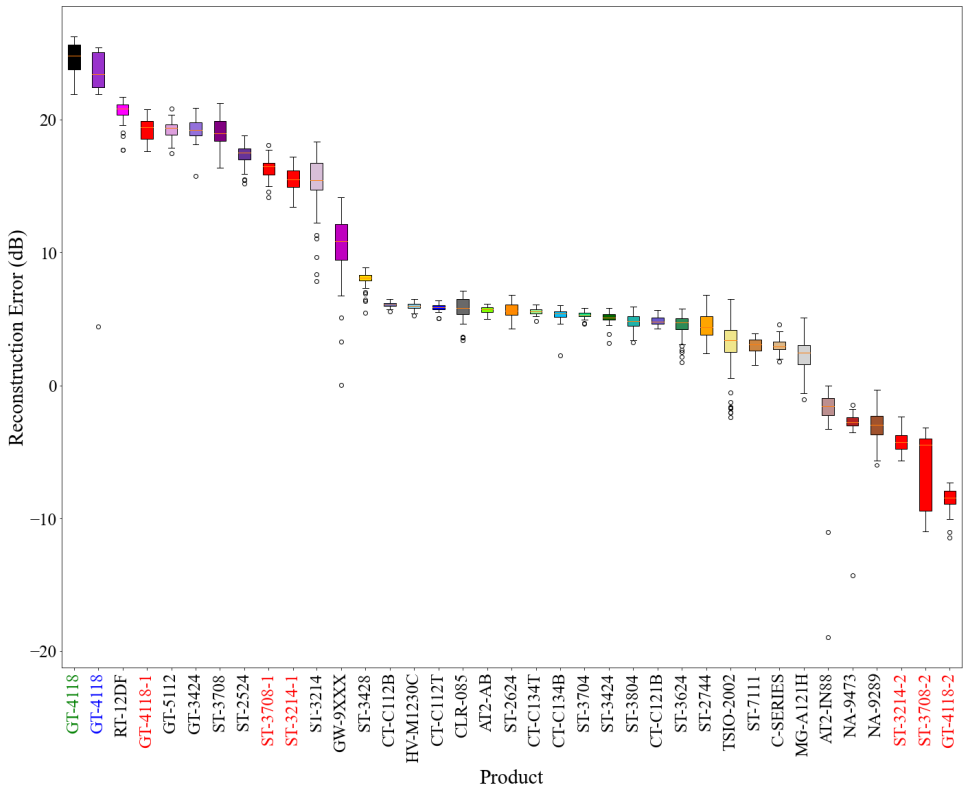
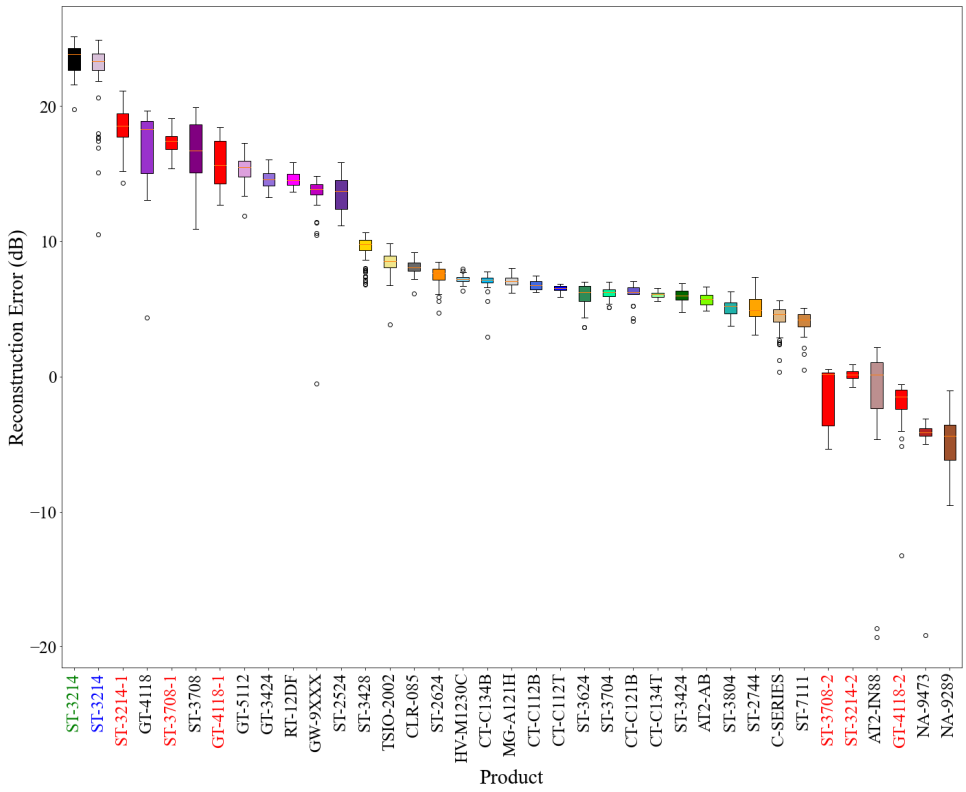


FIGURE 4. AUC score of training each single normal product under single-product learning with 2 baseline models.

that the temporal summary information calculated by the encoder can have a significant impact on the decoder. Self-attention also shows performance improvement in all scenarios, but there is one exception. For single product learning with TAAP in Table 3, Seq2Seq-AE and SSS-AE have no performance difference. However, SSS-AE outperforms

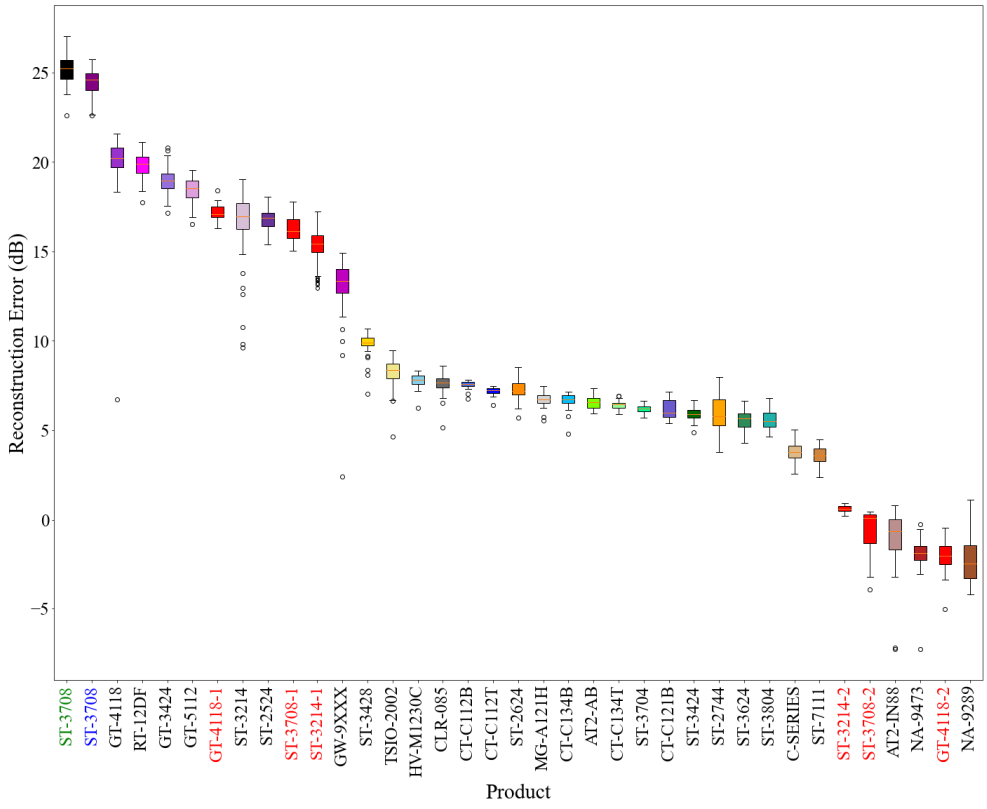


(a) GT-4118



(b) ST-3214

FIGURE 5. The distributions of reconstruction errors under single-product learning with SSS-AE. Reconstruction errors are expressed as SNR. Green and blue labels mean that the training and test sample of trained product. Red labels represent the error products.



(c) ST-3708

FIGURE 5. ***(Continued.)*** **The distributions of reconstruction errors under single-product learning with SSS-AE. Reconstruction errors are expressed as SNR. Green and blue labels mean that the training and test sample of trained product. Red labels represent the error products.**

TABLE 6. **Inference performance comparison of each auto-encoder model for under single-product learning. True Positive Rate (TPR) and False Positive Rate (FPR) were calculated based on thresholds showing the best performance for each model. Time means the inference average time that takes to process a single sample.**

Models	TPR	FPR	Time
Conv-AE	0.950	0.310	**0.008s**
FARED	0.892	0.064	0.015s
SSS-AE	**0.962**	**0.038**	0.016s

Seq2Seq-AE in multi-product learning with TAAP. We conjecture that for small training data such as single product learning, the performance of the models is already saturated by TAAP, but self-attention works well when large training data such as multi-product learning in Table 4. For LAP, self-attention shows better performance. As a result, SSS-AE with the Seq2Seq structure and the self-attention mechanism demonstrate 10% and 14% performance improvement over FARED for single product learning and multi-product learning. Specifically, when training large and complex dataset, such as multi-product learning, the components of SSS-AE show greater performance improvement. Furthermore, compared to Seq2Seq-AE-large, which has twice the number of parameters than Seq2Seq-AE, this result negates the assumption that the performance improvement of SSS-AE is simply due to the increased parameters.

4) INFERENCE PERFORMANCE

We summarized in Table 6 by comparing the inference time per sample and the detection performance for actual abnormal data for each anomaly detection model. {*GT-4118, ST-3214, ST-3708*} and *ERROR 1* and *ERROR 2* were used to check the ability to distinguish between normal data and actual abnormal data of the same product. True Positive Rate (TPR) and False Positive Rate (FPR) were calculated for each model based on the optimal thresholds obtained during the ROC calculation process. Conv-AE was built on the configuration of Oh *et al.* [7]. All models were preprocessed with the TAAP

method. Conv-AE showed high TPR and fast processing speed, but had the lowest FPR. FARED had decent FPR score, but had the lowest TPR of the three models. SSS-AE showed the best results in TPR and FPR. RNN-based models, SSS-AE and FARED, are about 7 ms slower than convolution based Conv-AE, but not significantly in the real field. These results show that SSS-AE is more practical than the conventional model.

VII. CONCLUSION

We experiment with two scenarios, single-product learning and multi-product learning, on a new large-scale SMD dataset. Multi-product learning is a challenge that previous studies have not tried. We verify that the proposed model and data preprocessing, SSS-AE and Temporal-Adaptive-Average-Pooling, perform single-product learning reliably and robust even on large SMD dataset and also effective in multi-product learning. Experimental results show that applying appropriate data preprocessing is a very important factor in improving the performance of anomaly detection. Moreover, for auto-encoders targeting the sequence data domain, these results support that the Seq2Seq structure and self-attention mechanism are still working well.

APPENDIX A

See Figure 4.

APPENDIX B

See Figure 5.

ACKNOWLEDGMENT

(Ki Hyun Nam and Young Jong Song contributed equally to this work.)

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[2] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019, *arXiv:1901.03407*. [Online]. Available: http://arxiv.org/abs/1901.03407

[3] Y.-J. Cha, W. Choi, and O. Büyüköztürk, "Deep learning-based crack damage detection using convolutional neural networks," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 32, no. 5, pp. 361–378, May 2017.

[4] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 1–38, 2010.

[5] L. Deng, M. L. Seltzer, D. Yu, A. Acero, A.-R. Mohamed, and G. Hinton, "Binary coding of speech spectrograms using a deep auto-encoder," in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc.*, 2010. [Online]. Available: https://scholar.google.com/scholar?start=0&hl=en&as_sdt=0,5&as_vis=1&cluster=11751390378439452975

[6] D. Wulsin, J. Blanco, R. Mani, and B. Litt, "Semi-supervised anomaly detection for EEG waveforms using deep belief nets," in *Proc. 9th Int. Conf. Mach. Learn. Appl.*, Dec. 2010, pp. 436–441.

[7] D. Y. Oh and I. D. Yun, "Residual error based anomaly detection using auto-encoder in SMD machine sound," *Sensors*, vol. 18, no. 5, p. 1308, Apr. 2018.

[8] Y. Park and I. Yun, "Fast adaptive RNN encoder–decoder for anomaly detection in SMD assembly machine," *Sensors*, vol. 18, no. 10, p. 3573, Oct. 2018.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[10] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[11] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, 2006, pp. 369–376.

[12] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[13] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, "Deep speech: Scaling up end-to-end speech recognition," 2014, *arXiv:1412.5567*. [Online]. Available: http://arxiv.org/abs/1412.5567

[14] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 4960–4964.

[15] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, and J.Chen, "Deep speech 2: End-to-end speech recognition in English and Mandarin," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 173–182.

[16] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards end-to-end speech synthesis," in *Proc. Interspeech*, Aug. 2017, pp. 4006–4010.

[17] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, R. A. Saurous, Y. Agiomvrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning wavenet on MEL spectrogram predictions," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 4779–4783.

[18] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, *arXiv:1406.1078*. [Online]. Available: http://arxiv.org/abs/1406.1078

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] A. Abid and J. Y. Zou, "Learning a warping distance from unlabeled time series using sequence autoencoders," in *Proc. NeurIPS*, 2018, pp. 1–9.

[21] A. Sagheer and M. Kotb, "Unsupervised pre-training of a deep LSTM-based stacked autoencoder for multivariate time series forecasting problems," *Sci. Rep.*, vol. 9, no. 1, pp. 1–16, Dec. 2019.

[22] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent., (ICLR)*, 2015, pp. 1–15.

[23] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1412–1421.

[24] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing machines," 2014, *arXiv:1410.5401*. [Online]. Available: http://arxiv.org/abs/1410.5401

[25] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 551–561.

[26] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 2249–2255.

[27] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," 2017, *arXiv:1703.03130*. [Online]. Available: http://arxiv.org/abs/1703.03130

[28] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," 2017, *arXiv:1705.04304*. [Online]. Available: http://arxiv.org/abs/1705.04304

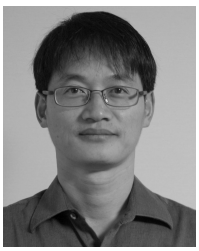
[29] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, "Stand-alone self-attention in vision models," 2019, *arXiv:1906.05909*. [Online]. Available: http://arxiv.org/abs/1906.05909

[30] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 7871–7880.

[31] H. Shao, M. Xia, J. Wan, and C. De Silva, "Modified stacked auto-encoder using adaptive Morlet wavelet for intelligent fault diagnosis of rotating machinery," *IEEE/ASME Trans. Mechatronics*, early access, Feb. 9, 2021, doi: 10.1109/TMECH.2021.3058061.
[32] Z. Wang and Y.-J. Cha, "Unsupervised deep learning approach using a deep auto-encoder with an one-class support vector machine to detect structural damage," *Struct. Health Monitor.*, vol. 20, no. 1, pp. 406–425, 2020.
[33] Y.-A. Chung, C.-C. Wu, C.-H. Shen, H.-Y. Lee, and L.-S. Lee, "Audio Word2 Vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder," in *Proc. Interspeech*, Sep. 2016, pp. 765–769.
[34] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," 2016, *arXiv:1607.00148*. [Online]. Available: http://arxiv.org/abs/1607.00148
[35] S. Amiriparian, M. Freitag, N. Cummins, and B. Schuller, *Sequence to Sequence Autoencoders for Unsupervised Representation Learning From Audio*. Augsburg, Germany: Universität Augsburg, 2017.
[36] J. Pereira and M. Silveira, "Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention," in *Proc. 17th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2018, pp. 1275–1282.
[37] M. Zhang, Y. Wu, W. Li, and W. Li, "Learning universal sentence representations with mean-max attention autoencoder," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 4514–4523.
[38] X. Liu, Z. Han, X. Wen, Y.-S. Liu, and M. Zwicker, "L2G auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 989–997.
[39] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*. [Online]. Available: http://arxiv.org/abs/1607.06450
[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[41] S. Chauhan and L. Vig, "Anomaly detection in ECG time signals via deep long short-term memory networks," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2015, pp. 1–7.
[42] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proc. ESANN*, vol. 89, 2015, pp. 89–94.
[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

KI HYUN NAM is currently pursuing the B.S. degree in computer engineering with Hankuk University of Foreign Studies, Yongin, South Korea. Since 2021, he has been a Research Assistant with Naver Clova, Seongnam, South Korea. His research interests include speech processing and representation learning.

YOUNG JONG SONG received the B.S. degree from Hankuk University of Foreign Studies, Yongin, South Korea, in 2020, where he is currently pursuing the M.S. degree in computer engineering. His research interests include speech processing and representation learning.



IL DONG YUN (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in control and instrumentation engineering from Seoul National University, Seoul, South Korea, in 1989, 1991, and 1997, respectively. From 1996 to 1997, he was with Daewoo Electronics, Seoul, as a Senior Engineer. Since 1997, he has been with Hankuk University of Foreign Studies, Yongin, South Korea. His research interests include medical image analysis and computer vision.