INTRODUCTION OF A NOVEL ANOMALOUS SOUND DETECTION METHODOLOGY

Xiao Tan and S M Yiu

Dept of Computer Science, University of Hong Kong, Hong Kong

ABSTRACT

This paper is to introduce a novel semi-supervised methodology, the enhanced incremental principal component analysis ("IPCA") based deep convolutional neural network autoencoder ("DCNN-AE) for Anomalous Sound Detection ("ASD") with high accuracy and computing efficiency. This hybrid methodology is to adopt Enhanced IPCA to reduce the dimensionality and then to use DCNN-AE to extract the features of the sample sound and detect the anomality. In this project, 228 sets of normal sounds and 100 sets of anomaly sounds of same machine are used for the experiments. And the sound files of machines (stepper motors) for the experiments are collected from a plant site. 50 random test cases are executed to evaluate the performance of the algorithm with AUC, PAUC, F measure and Accuracy Score. IPCA Based DCNN-AE shows high accuracy with the average AUC of 0.815793282, comparing with that of Kmeans++ of 0.499545351, of Incremental PCA based DBSCAN clustering of 0.636348073, of Incremental based PCA based One-class SVM of 0.506749433 and of DCGAN of 0.716528104. From the perspective of computing efficiency, because of the dimensions-reduction by the IPCA layer, the average execution time of the new methodology is 15 minutes in the CPU computing module of 2.3 GHz quad-core processors, comparing with that of DCGAN with 90 minutes in GPU computing module of 4 to 8 kernels.

KEYWORDS

Internet of things ("IoT"), Anomalous Sound Detection ("ASD"), Audio Frames, Deep learning, Deep Convolutional Neural Network, Autoencoder.

1. INTRODUCTION

1.1. Anomalous Sound Detection and Issues of Implementation to Industry

Internet of things ("IoT") and industry 4.0 are becoming more and more important in both research and industrial domains. Industry 4.0 is a paradigm within automation and data transformation in manufacturing technologies, which help to build up the intelligent networking of devices, machines, and processes in the enterprises. Industrial Internet of Things ("IoT"), as one of the key components of industry 4.0, is to adopt the devices, including sensors, actuators, computers or "smart" objects to collect the operation data in the manufacturing process, to monitor the operation status of the machines and to detect the anomalous behaviours of the machines. The machine learning/deep learning algorithms are adopted widely to deploy the artificial intelligent agents for IoT data analysis.

Anomalous Sound Detection ("ASD") is one prominent approach of IoT, which is to adopt unsupervised or semi-supervised ML/DL methodologies to extract the features of the sound data of the equipment, and then to observe and detect the anomalous sound emitted from the target machinery as the basis of the warning of the operation status of the targets and send it to the human operators. All machinery in factories is subject to breakdown, which causes potential risk of loss for the companies [1]. Experienced technicians are able to diagnose such failures and the machines' operational conditions by listening to the machinery[2].As a major type of Predictive Maintenance Technologies, acoustic monitoring sensor could detect the anomalous changes of the machine's condition before the breakdown really happens. Based on the statistics result published by US Department of Energy in 2020, the functional predictive maintenance program reduces the maintenance costs of 25%-30%, eliminates the breakdown of 70%-75%, reduces the downtime of 35%-45% and increases the production of 20%-25%[5]. However, based on the studies in United States in the winter of 2020, only 12% maintenance resources and activities of average facilities are predictive [3]. Therefore, the need of the predictive maintenance is huge and on a high priority. especially when the machines work in an environment which is difficult for humans to enter or work continuously in (eg: the Deep Ocean Drifters), or there is lack of the labours in the companies which are quite normal in the developed countries or regions, the quality of the maintenance service will be impacted significantly due to the shortage of human-friendly environment or skilled workers. To resolve this issue, the technology of "smart" anomalous sound detection with high precise and sensitivity on operating acoustic data is developed.

Except for the lower cost of predictive maintenance, the reduced risk of deaths and severe injuries due to the equipment incidents is another important reason to implement the Anomalous Sound Detection technology in the maintenance practices. For example, incidents involving lifts, elevators and escalators each year kill 31 people and seriously injure about 17,000 people in the United States, according to the Centre to Protect Workers' Rights ("CPWR") analysis of data provided by the BLS (1992-2009) and the Consumer Product Safety Commission ("CPSC") (1997-2010)[4]. In Hong Kong, the annually Reported Lift Incident Records are 366 on average (2011-2021)[5]. With Anomalous Sound Detection technology, the working conditions of the lifts or elevators are capable to be monitored and predicted in time before the incidents really happen and therefore reduce the loss of the peoples' lives and weaken the harms to the safety of equipment located environments.

However, there are quite a few challenges of the ASD, although the importance of its role is increasing with the development of industry 4.0 and IoT. And those challenges are mainly related to the limit of the data collections, which includes:

Imbalanced Training Dataset. For most IoT applications, the anomaly events with a long time-series data are rare. Generally, the IoT generates a large volume of normal data with 24*7, and the actual anomalous sounds rarely occur and are highly diverse[6] because of the changing and complex operating conditions[7][8]. The ratio of anomalous data files to normal ones depends on the different types of machines, maintenance conditions and external environments eg: changes of seasons etc. In practice, the range of such ratio varies from 5% or less to more than 20% when the machines' conditions are getting damaged. Therefore, to identify the unknown anomalous sounds from the given exhaustive continuous normal data is a challenge for traditional ASD technology.

Noise. In most occasions, the sound data is collected in the real manufacturing environments where multiple machines are placed together to emit different types of sounds, therefore, the environmental noise is a potential issue to impact the data quality significantly[9]. Especially, the sudden changes of the noise characteristics increase the difficulty of the detection of the anomalous sound in the noise contaminated environment, and lead to cognitive performance decline[10]. Therefore, a machine learning methodology with noise-tolerance is crucial for the application for the ASD. However, the traditional methodologies, including both classification and clustering, are not capable to handle this issue from the nature.

Hardcoded Architecture. Because most different local environments have their own unique acoustic signatures, therefore the localized models are required for each different environment or domain. Lack of automatic self-adaptability and hardcoded framing are the defects of most algorithms because of the requirement to set different parameters for the different characteristics of the original acoustic data and the external environments. And such settings impact the efficiency and accuracy of ASD.

Computation Cost. To achieve the satisfactory performance of features extractions of audio files, the enormous size of the training data and the architecture of the machine learning or deep learning algorithms require high computation capability, and thus the computation cost keeps high. For example, for some generative adversarial network or autoencoder, the continuous computation capability of one or multiple GPUs are required when keeping successful batch-running. The computing cost can reach thousands of dollars per machine per month.

To resolve these four issues, different with the methodologies introduced in this chapter, as the semi-supervised algorithm, the Deep Convolutional Neural Network Autoencoder has the capability to train the labelled normal dataset and classify the unlabelled anomalous sound when lacking the known abnormal scenarios. To aware the noise robustness, a noise is collected in the environment for experiments purpose. Hence, the encoder inputs are the clean sample values plus the noise for training and prediction.

And the audio segments in the pre-processing phrase are referred as one input parameters of the DCNN Auto Encoder, therefore the adaptability of the deep learning algorithm is relatively higher than other unsupervised or semi-supervised algorithms with high accuracy and computing efficiency.

In the next chapter, several well-known unsupervised and supervised algorithms are introduced. The pros and cons of each algorithm are discussed.

1.2. Limitation of Existing Solutions

As one of the hottest topics in the research field of industry 4.0, Anomalous Sound Detection develops quite fast during the past decades. As defined by V. Chandola, A. Banerjee, and V. Kumar in 2009[11], anomalies refer to the behaviour patterns that do not conform to a well-defined notion of normal behaviour. Based on the definition, the characteristics of the anomalies to differ them from the normal ones includes (1) the scarcity as the anomaly event occurs quite less frequently than normal ones; (2) the features which are able to be identified or extracted from the normal ones;(3) the meaning to be represented by the anomality[12]. Because of the characteristics of the anomalies, many algorithms, mainly unsupervised and semi-supervised, are created for the fault diagnose and anomaly prediction. Several of them are discussed here.

Classification is one type of the earliest unsupervised machine learning (“ML”) methodology, in which Principal Component Analysis (“PCA”), and Linear Discriminant Analysis (“LDA”) are widely used to reduce the dimensions by looking for the linear combinations of variables[13]. The main challenge of such classification is that the characteristics of the data should be known in advance, eg: the count of the variables, the class should be Gaussian etc. Another disadvantage is the accuracy when deploying such algorithms in practice, especially when predicting the changes of the temporal shapes after the long-time training of the normal dataset.

Clustering is another type of the unsupervised ML methodology in ASD. In clustering, K-means is the most popular unsupervised algorithm [14]. The advantage of K-means is the high efficiency because of its linear complexity. The disadvantages include that the count of clusters should be pre-set as an input and that the inconsistent clustering results due to its random starting point selection. Agglomerative Hierarchical Clustering is another widely-used clustering algorithm[15]. The advantage of the hierarchical algorithm is that it doesn't require the count of clusters as an input. The disadvantages include the low efficiency and the low accuracy when processing a large volume of data because of its dendrogram [16].

Both classification and clustering algorithms are less noise-tolerant because of their architecture. That means, these two types of algorithms perform weak when deploying to noise contaminated environments.

Support Vector Machine ("SVM") and Convolutional Neural Network ("CNN") are two well-known supervised ML algorithms [17][18]. The accuracy of supervised ML algorithms can be much high. But such methodology cannot be deployed to imbalanced training datasets because of the lack of known anomalous samples.

Except for the traditional algorithms discussed above, the latest developments in the ASD algorithms include the Variational Autoencoder ("VAE")[19] and the Deep Convolutional Generative Adversarial Network ("DCGAN")[20], and other generative adversarial networks for the unsupervised deep learning. With appropriate architecture design, such unsupervised DL algorithms can achieve quite high accuracy for with imbalanced training datasets. The disadvantage is that those algorithms require relatively higher computation capability and costs.
In this paper, the incremental principal component analysis based deep convolutional neural network autoencoder is introduced. The details of the new DL algorithm will be described in the section B, C, D, E F and G.

1.3. Methodology Overview

The proposed autoencoder was designed in Tensorflow and Pytorch which are Google and Facebook open-source software for machine learning respectively. With the dimensionality reduction by Enhanced IPCA, the autoencoder has the capability to learn efficient feature representations of the input data with sole label, i.e. the training requires only one label for the normal sound datasets, and then reconstruct the input to generate the output based on the useful features learned by the autoencoder. It consists of two parts. One is an encoder $g = enc(x)$, which encodes the input x to a learned feature representation g. The encoder extracts meaningful features at the bottleneck layer, known as the code or latent space, which typically has a much lower dimensionality than the input data. This ensures that, instead of copying the input data, only the useful features are adopted in the calculation scope to reconstruct the output. The other is the decoder $\hat{x} = dec(g)$ which attempts to recapture the original input by decoding the useful features stored in the bottleneck layer. For a deep convolutional AE model, the encoder $enc_\theta(x)$ and decoder $dec_\gamma(g)$ are modelled as deep convolutional networks with parameters θ and γ, respectively. Therefore, the training of the deep convolutional neural network autoencoder is described as the formula below:

$$\min_{\theta,\gamma} \sum_{x} Error_{recn}(x, \hat{x}),$$

$where\ \hat{x} = dec_\gamma(enc_\theta(x))$

For $Error_{recn}(x, \hat{x})$,, we can use mean-squared error ("MSE") as the evaluation of the reconstruction errors between the input x and the output $\hat{x}$.

This methodology is capable to achieve high performance without the requirement on high computing capability especially for the large volume and high dimensional machines' sound files. The details of the encoder and decoder are described in the next section C.

1.4. Imbalanced Dataset

IPCA based DCNN-AE, as a semi-supervised machine learning algorithm, is capable to train the large-sized normal dataset and detect the anomality when a small-sized abnormous data enters.

DCNN-AE utilizes the encoder-decoder architecture to train the input data to estimate the output data to be similar to the input. Therefore, when the input data is anormal, with the trained architecture, the output generated will be anormal, theoretically.

Because of its ignorance of the normality of input dataset, unbalance is not an issue to impact the performance of the algorithm in nature.

1.5. Noise Awareness and Tolerance

Noise identification, noise removal and noise accommodation are the major topics to deal with the unwanted background noise when analyzing anomalous sound. The architecture of the new methodology adopts the noise accommodation which to immune the background noise from the anomalous sound observations [11][12]. Considering the complex environments to collect the sounds files, i.e. factories or construction sites, in an attempt to improve the robustness against noise of the algorithm, the noise awareness is included in the data pre-processing phase with a Gaussian white noise to be added in the original audio files and the signal to noise ratio("SNR") is 6.9897 (SNR=10*Log(1/0.2) while 0.2 is the ratio of noise to real sound, or the noise significancy factor). Hence, the encoder input actually is the clean sample values plus the noise:

$$\tilde{x} = x + \epsilon,$$

$where\ \epsilon \sim \mathcal{N}(-1,1) * noise\ significancy\ factor.$ And the decoded output is redefined as $dec_\gamma(enc_\theta(\tilde{x}))$.

This is to increase the noise awareness in the algorithm and was shown to results in better generalizing representations, which als0o appears to capture semantic information of the input. And based on the experiments, the maximum noise tolerance (measured by the noise significancy factor) of the IPCA based DCNN-AE is 0.5.

1.6. Parameterized Framing Technology

DCNN-AE in this paper adopts the parameterized architecture instead of hardcoded when defining the parameters of the Incremental PCA layer and the input data shapes of autoencoder layer. Different with the hardcoded input shapes, eg: 256*256 or 512*512, the temporal of input data is decided by the size of the audio data files. Each audio file is framed to the shape of Maximum Squared Root of the audio file data size (which is calculated to be 210) * Maximum Squared Root of the audio file data size (same as 210). The enhanced IPCA is capable to reduce the dimensionality to the calculated framing for DCNN-AE to proceed. When the environment changes and the raw sound data is changed, the algorithm will change the temporal of the input sound data automatically.

Such design, on some level, helps to build the auto-localized models to improve the self-adaptability of the different environments.

1.7. Less Requirement of Computing Capability and Computing Cost

Enhanced IPCA based DCNN-AE includes the two layers of IPCA and DCNN-AE, which allows the dimensions of the audio data to be reduced to adapt to the optimized architecture of the Autoencoder, and thus requires less computing capability of GPU.

The high-dimensional audio data files can be trained and predicted in normal CPU instead of GPU or TPU, after the technique of dimensions-reduction by enhanced IPCA, with minimum impacts on the performance. The computing cost because of the continuous GPU(s) consumptions can be reduced significantly as well.

The details of the algorithm will be introduced in Chapter II.

2. ENHANCED INCREMENTAL PRINCIPAL COMPONENT ANALYSIS BASED DEEP CONVOLUTIONAL NEURAL NETWORK- AUTOENCODER

2.1. Dimensionality Reduction by Enhanced Incremental Principal Component Analysis

Principal Component Analysis ("PCA") is a classical statistical method to reduce the dimensions in a dataset with many interrelated variables. Since being raised by Pearson as early as in 1901 and Hotelling in 1930's, PCA has been widely used in science and business domains as an unsupervised method[21]. The principal of PCA is to seek the subspace of largest variance in dataset. For data given as a set of n vectors in Rd, $x_1, x_2, \dots x_n$, denote by X which is the normalized matrix: $X = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$ [22]. The PCA is to find the k-dimensional subspace so that the projected data on the subspace has largest variance. Assume $W \in R_d * k$, PCA could be formulated as the following optimization problem[23]:

$$\max_{W \in R^{d*k}, W^{TW}=I} ||XW||_F^2$$

Where $\|\cdot\|_F$ is the Frobenius norm. The above optimization is an inherently non-convex optimization problem even for k=1. However, in the classical PCA algorithm, when finding the leading principal component, there are two methods: one is the singular value decomposition (SVD), which requires super-linear time and potential $O(d^2)$ space so that it is prohibitive for large scale machine learning problems. The other is Power method which is to estimate the matrix X. The approximation method requires the data to be well-conditioned, and the spectral gap, i.e. the distance between the largest and second largest eigenvalues of X, to be bounded away from zero. This method can improve the computing efficiency but might loss the useful data as the pass-over from the leading components.

In 2002 a new PCA algorithm for incremental computation of PCA is introduced by Arta$\tilde{c}$ et al., 2002[23]. This method is to make the simultaneous learning and recognition possible. This algorithm is to update the original eigenspace and mean continuously with the learning rate, and store only the data projected onto it. It releases the restriction of the classical PCA that all the sample data are required to be available as a priority, and therefore more adaptive for the demand online learning.

In this paper, an enhanced incremental PCA is adopted for dimension reduction. The summary of the new algorithm is:

1. Locate the target object in the first frame, and initialize the eigen basis U to be empty, and the mean μ to be the appearance of the target in the first frame. The effective number of observations so far is n=1.
2. Then advance to the next frame. Draw particles from the particle filter, according to the dynamical model. And the dynamics between states in this space is modeled by Brownian motion. Each parameter in the affine transformation of X is modeled independently by a Gaussian distribution around its counterpart in X_{t-1}, and therefore the motion between frames is itself and an affine transformation. Specifically, $p(X_t \mid X_{t-1}) = \mathcal{N}(X_t ;\, X_{t-1}, \psi)$, where ψ is a diagonal covariance matrix whose elements are the corresponding variances of affirm parameters.
3. For each particle, extract the corresponding window from the current frame and calculate its weight, which is its likelihood under the observation model.
4. Perform an incremental update (with a forgetting factor) of the eigen basis, mean, and effective number of observations.
5. Go to step 3.

The enhanced incremental algorithm is based on the Sequential Karhunen-Loeve ("SKL") algorithm of Levy and Lindenbaum (2000)[24], which takes QR decomposition and computes the SVD of basis vectors. The computational advantage of SKL algorithm is that it has space and time complexity constant in n. The space complexity is reduced to $O(d(k+m))$, and the computational requirements are reduced to $O(dm^2)$ for recomputing the entire SVD[25]. However, this algorithm does not account for the sample mean of the training data, which changes over time as a new data arrive. The enhanced incremental PCA is to augment the new training data with an additional vector carefully chosen to correct for the time-varying mean. The principal computation process is:

Given U and $\sum$ from the SVD of (A – $\overline{I_A}$), as well as $\overline{I_A}$, n, and B, compute $\overline{I_C}$ as well as U' and $\sum$' from the SVD of (C-$\overline{I_C}$):

1. Compute the mean vector $\frac{1}{m}\Sigma_{i=n+1}^{n+m} I_i$, and $\overline{I_C} = \frac{n}{n+m}\overline{I_A} + \frac{m}{n+m}\overline{I_B}$.
2. Form the matrix $\hat{B}$ = $[(I_{m+1} - \overline{I_B}) \ldots (I_{n+m} - \overline{I_B})\sqrt{\frac{nm}{n+m}}(\overline{I_B} - \overline{I_A})]$.
3. Compute $\tilde{B} = orth\left(\hat{B} - UU^{\intercal}\hat{B}\right)$ and $R = \begin{bmatrix} \Sigma & U^{\intercal}\hat{B} \\ 0 & \tilde{B}(\hat{B} - UU^{\intercal}\hat{B}) \end{bmatrix}$, and $\hat{B}$ will be one column larger than in the SKL algorithm.
4. Compute the SVD of R: $R \overset{\text{SVD}}{=} \tilde{U}\tilde{\sum}\tilde{V}^{\intercal}$
5. Finally $U' = [U, \tilde{B}]\tilde{U}$ and $\sum' = \tilde{\sum}$.

In this algorithm, an important consideration is the effect of the forgetting factor on the mean of eigen basis when reducing the contribution of each block of data to the overall covariance modelled by an additional factor of f2 at each SVD update. And the forgetting factor are multiplied with the incremental update of the eigen basis, mean and effective number of observations to generate the new frame.

Another enhancement is in the Sequential Inference Model. When calculating the affine motion parameters (and thereby the location) of the target at time t, the Condensation algorithm (proposed by Isard and Blake in 1996)[26], based on factor sampling, is adopted to approximate an arbitrary distribution of observations with a stochastically generated set of weighted samples, as it evolves over time.

The new PCA algorithm above is adopted to improve the accuracy with computing efficiency. By Enhanced IPCA, the components for DCNN-AE to extract features can be reduced to fit for the optimized architecture designs of DCNN-AE based on the parameterized framing shapes.

2.2. Deep Convolutional Neural Network Autoencoder

An autoencoder is a feed-forward neural network and is an unsupervised or semi-supervised algorithm that applies backpropagation, setting the target value to be equal to the input [27]. Generally speaking, it is trained to approximate the identification so that the output $\hat{x}$ is similar to the input x.

The autoencoder is consisted of encoder which obtains a compressed encoder from the input and of decoder which reconstructs the data from encoding. The encoder maps the input information to the compressed nodes and creates the bottleneck which forces the network to extract the compressed low-dimensional representations from high-dimensional data [28]. And based on the extracted information in the bottleneck, or, hidden layer, the decoder reconstructs the input data, with the backpropagation to minimize the reconstruction error which is defined as Mean Squared Error (“MSE”). The structure can be described as the two formulas below [29][30].

$$\text{Encoder: } h = \sigma(W_{xh}x + b_{xh}).$$

Here h is the hidden layer, or bottleneck, σ is the non-linear mapping relationship between the compressed nodes in the hidden layer and the original inputs. W_{xh} are the weights when mapping the original inputs to the nodes of the hidden layers. b_{xh} are the biases from the original inputs to the nodes of the hidden layers.

$$\text{Decoder: } \hat{x} = \hat{\sigma}(W_{hx}h + b_{hx}).$$

Here $\hat{x}$ is the output of the decoder, and $\hat{\sigma}$ is the non-linear mapping relationship between the compressed nodes in hidden layers and the final output. $W_{hx}$are the weights to map the compressed nodes in the hidden layers to the outputs. And $b_{hx}$are the biases from the compressed data in the hidden layers to the output[31][32][33].

The architecture of standard autoencoder is shown in Fig 1.

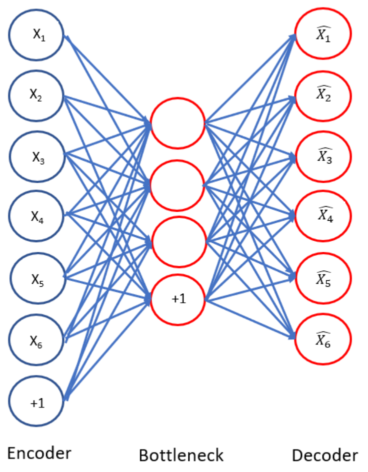


Fig. 1. Standard Autoencoder

Deep Convolutional Neural Network Autoencoder (“DCNN-AE”) is the autoencoder composed of multiple convolutional layers in encoder and inverse convolutional layers in decoder [19]. In this paper, there are four convolutional layers in encoder part, each with batch-normalization, Leaky ReLU and Max-pooling to construct a one typical compressed pattern. In decoder, there are four convolutional transpose layers respectively, each with batch-normalization, Leaky ReLU and Up-sampling to set up the reconstruction pattern. The architecture of the DCNN-AE is shown as Fig 2.

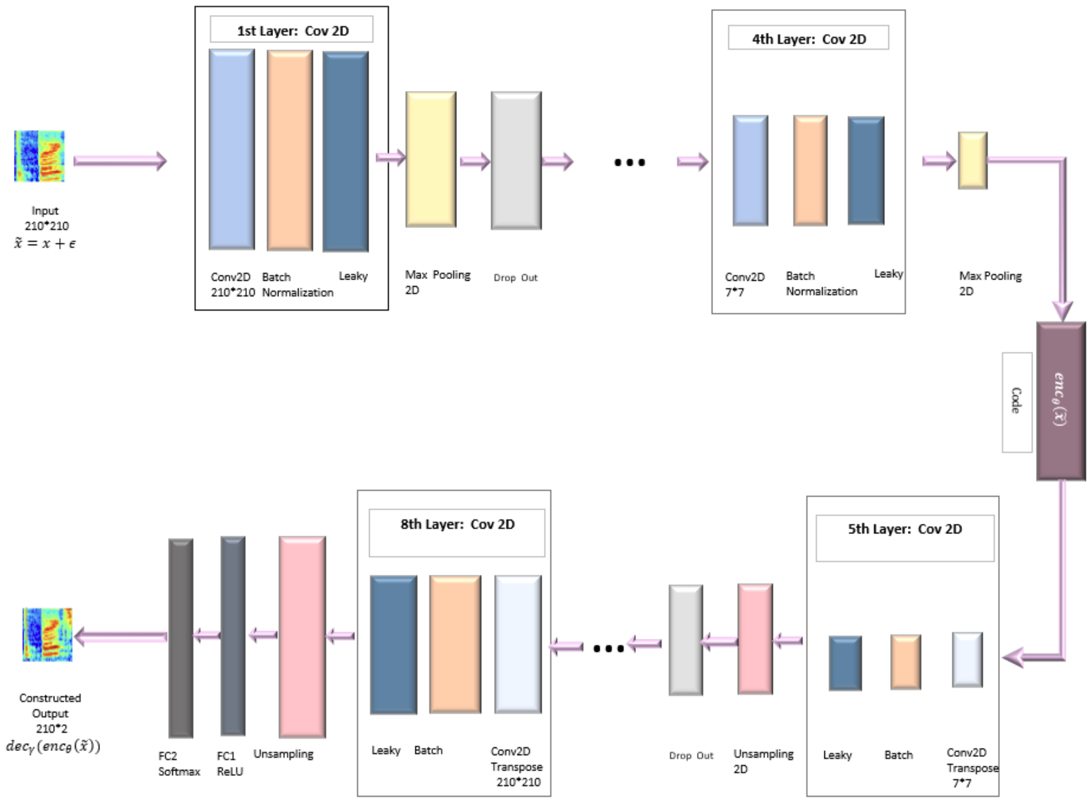


Fig. 2. Architecture of Deep Convolutional Neural Network Autoencoder

The details of the DCNN-AE algorithm include[34]:

Convolutional and Inverse Convolutional Layers

Convolutional Layers are applied to extract the features based on the patches randomly sampled from the input, being learned with the weighted matrices known as the filtering masks and kernels, and then being processed by non-linear operations to generate the feature vector as the output[35].The principle of the conventional layer of CNN is:

$$h_{i,j,k}^{l} = {w_k^l}^T x_{i,j}^{l} + b_k^l$$

Where $h_{i,j,k}^{l}$ is the feature value at location (i,j) in the kth feature map of lth layer, ${w_k^l}^T$and b_k^l are the weight vector and bias of the kth filter of the lth layer respectively, and $x_{i,j}^{l}$ is the input patch centered at location (i,j) of the lth layer.

In this paper, 2-dimensional convolutional layers are deployed in the encoder part, and 2-dimensional transpose convolutional layers are deployed in decoder part to reconstruct the information transformed and compressed from the original inputs [36].

Batch Normalization Layers are added after the convolutional and inverse convolutional layers to normalize the activations of the previous layers each batch [37]. Assume in the $n+1^{th}$ hiden layer and $h_i^{(n)}$, i=1,2,…,m be the values of the inputs from the previous layer for the a batch $\mathcal{B}$, the principal of the batch normalization is described as the formulas shown as below:

$$h_i^{(n+1)} = BN_{\gamma,\varepsilon}\left(h_i^{(n)}\right) = \gamma \dot{h}_i^{(n)} + \varepsilon$$

Where:

1. $\dot{h}^{(n)} = \frac{h^{(n)} - \mu\mathcal{B}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$
2. $\mu\mathcal{B} = \frac{1}{m}\sum_{i=1}^{m} h_i^{(n)}$
3. $\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m}(h_i^{(n)} - \mu\mathcal{B})^2$

Here $BN_{\gamma,\varphi}$ refers to the Batch Normalization function to the two learning parameters of $\gamma\,,\varepsilon.\, h_i^{(n)}$ is the ith compressed node in the nth hidden layer. 1 describes the calculation of $\dot{h}^{(n)}$ which is the covariance of $h^{(n)}$.

From the principal formulas above, we can understand that Batch Normalization applies a transformation that maintains the mean activation of every unit close to zero, then applies scaling and shifting, parameterized by two learnable parameters γ, ε. By this, batch normalization whitens the data which is well known for speeding up learning in neural networks and reduces internal covariance shift, which is used for the change of a layer’s activations distribution over time.

Pooling

Pooling layer is adopted to reduce the features extracted by convolutional layers. Four 2-dimesional Max Pooling layers are applied among four convolutional layers to use maximum values of each cluster of neurons of the prior layer. This is to reduce the dimensionality and the overfitting by introducing the abstracted form of the low-level features generated by the first convolutional layer. This pooling method is typically effective in this research because the original input data are pre-processed and transformed to the categories with data type as of integer.

Tab 1 shows the details of all the layers in this DCNN-AE algorithm.

Full Connected Layer and Classifier

The fully connected layer is utilized to associate neurons within outputs in the last inverse convolutional layer, which are connected to the Softmax classifier.

Softmax classifier layer, which is used for classification according to the features, is after multiple convolutional autoencoder layers, max pooling layers, Leaky ReLU layers and full connected layers. In this paper, the anomalous sound detection results are divided into two classes, therefore the classifier result can be written by the formula as below:

$$\widehat{p_i} = \frac{e^{(\hat{x}_i)}}{\sum_{k=1}^{2} e^{(\hat{x}_k)}}, \text{ i = 0, 1}$$

Here $\widehat{p_i}$ is the classifier represents the probability of nodules and no nodules. $\hat{x}_i = \hat{\sigma}\sum_{t=1}^{T}(W_{hx}h + b_{hx})$ represents the T output features generated through the full connected layer where W_{hx} and b_{hx} represent the weight and error in decoder part respectively, $\hat{\sigma}$ represents the nonlinear function in decoder, in this paper it refers to ReLU and Leaky ReLU.

ReLU and Leaky ReLU

"Rectified Linear Unit" ("ReLU") is one of the most notable non-saturated non-linear activation functions. The principle of ReLU is shown as below:

a. With default values, it returns element-wise max(x, 0). Otherwise, it follows:
b. f(x) = max_value for x >= max_value,
f(x) = x for threshold <= x < max_value,
f(x) = alpha * (x – threshold) for otherwise.

The pro of ReLU is to prune the negative input to reduce the sparsity in the neural network, and the con of ReLU is that such pruning might impact the accuracy of the computation because the possible useful features contained in the negative inputs to build the discriminative high-level features.

Leaky ReLU is a variant of ReLU by assigning none zero output for negative input. Instead of mapping negative input to zero, the Leaky ReLU uses a predefined linear function to compress negative input and therefore enables the features of negative parts contained[38]. Therefore Leaky ReLU achieves the better trade-off between the network sparsity and its input information. Both ReLU and Leaky ReLU are unbounded functions and solves the gradient vanishing.

Table 1. Layers of the DCNN-AE Algorithm

Seq	Layer Type	Kernel Size	Feature Maps	Output Size	Padding
1	Input	-	-	None,210,210,1	-
2	Cov 1	2*2	32	None,210,210,32	Same
3	Batch Normalization				
4	Leaky ReLU				
5	Max Pool 1	(2,2)		None,105,105,32	Same
6	Drop Out	0.1			
7	Cov 2	3*3	64	None,105,105,64	Same
8	Batch Normalization				
9	Leaky ReLU				
10	Max Pool 2	(3,3)		None,35,35,64	Same
11	Drop Out	0.1			
12	Cov 2	5*5	128	None,35,35,128	Same
13	Batch Normalization				
14	Leaky ReLU				
15	Max Pool 2	(5,5)		None,7,7,128	Same
16	Drop Out	0.1			
17	Cov 2	7*7	256	None,7,7,256	Same
18	Batch Normalization				
19	Leaky ReLU				
20	Max Pool 2	(7,7)		None,1,1,256	Same
21	Drop Out	0.1			
22	Sequential				
23	Cov Transpose 1	1*1	128	None, 1,1,128	Same
24	Batch Normalization				
25	Leaky ReLU				
26	Up-sample 1	7		None,7,7,128	Same
27	Drop Out	0.1			
28	Cov Transpose 2	7*7	64	None,7,7,64	Same
29	Batch Normalization				
30	Leaky ReLU				
31	Up-sample 2	5		None,35,35,64	Same
32	Drop Out	0.1			
33	Cov Transpose 2	35*35	32	None,35,35,32	Same
34	Batch Normalization				
35	Leaky ReLU				
36	Up-sample 2	3		None,105,105,32	Same
37	Drop Out	0.1			
38	Cov Transpose 2	105*105	1	None,105,105,1	Same
39	Batch Normalization				
40	Leaky ReLU				
41	Up-sample 2	2		None,210,210,1	Same
42	Drop Out	0.1			
43	FC1 (Linear)	210*210		None,210,20	

44	FC1 (ReLU)			None,20	
45	FC1 (Drop Out)	0.2			
46	FC2 (Softmax)		2	None,2	
47	Output			None,2	Same

Regulatory and Other Hyperparameters

Gaussian Dropout. Gaussian noise is injected to hidden layers of neural networks to act as a powerful regularity, and it is only active during training of the network.

Optimizer[39][40]. AdaMax is a method for stochastic optimization. It is to compute individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. Different with Adam which updates exponential moving average of the gradients and the squared gradient based on the unlimited variants, AdaMax updates the gradients with a simpler bound of maximum value of the calculated second momentum.

In this paper, the learning rate of AdaMax is set as 1e-4, beta1 is set as 0.5 and decay rate is set as 1e-9. “Mean Squared Error” and “Mean Absolute Error” are adopted as the parameters for the cost evaluations.

The details of the hyperparameters are listed in Tab 2.

Table 2. Hyperparameters of the DCNN-AE Algorithm

Hyperparameter	Setting
optimizer	Admax (lr=1e-4, beta1=0.5)
dropout rate	0.1
batch normalization	Momentum=0.8
Leaky ReLU	Alpha=0.2
batch size	16
noise initializer	Random Uniform (-1,1)
noise significancy factor	0.2
loss	mse
monitor	mae
epoch	400

3. EXPERIMENT SETUP

3.1. Dataset and Pre-processing

The data is collected from the stepper motors in a plant site in Changzhou City of Jiangsu Province in China. The data consists of the normal/anomalous operating sounds of the real machines. Each recording is a single-channel 2-sec length audio that includes both a target machine's operating sound and added noise. The sample rate is standard as of 44,100.

In the experiment, the train dataset only includes the normal data from one machine type, and we randomly pick up 100 normal wave files in the total 228 sample normal for consecutive 50 times. The test dataset includes 10 anormal audio files, randomly selected from the total 120 sample anormal data files. The prediction focuses on the turning point from consecutive 100 normal audio files to 10 anomalous audio files.

The steps of data pre-processing include:

Random selection. 100 normal audio files and 10 anormal audio files are selected randomly from the 228 respective sample normal and 120 anormal audio files.

Conversion. Convert the audio files to the arrays of readable digits.

Framing. Calculate the frame rates and reframe the files with adjusted frame rates.

Adding label. Add the label as of “0” as a column in train dataset only to indicate that the status of all the data in the data frame are “normal”.

Scalarization and Normalization. Scale and normalize both the train and the test datasets.
For the experiments, the train data size is 88,200,000, and the prediction data size is 8,820,000. The experiments run for consecutive 50 times for each algorithm with the datasets picked up randomly in the total dataset. The null values of any attributes are filtered out during the pre-processing.

3.2. Benchmark System and Result

A simple Deep Convolutional Neural Network (“DCGAN”) is adopted as the benchmark performance of unsupervised anomaly detection for the experimental dataset. Generative Adversarial Network is a deep neural net architecture comprised of a pair of “adversarial” models called the generator and the discriminator respectively[41][42]. The generator is to capture a noise vector to map to the data distribution as the fake input of the discriminator. The discriminator is to discriminate or identify the two inputs, one from the real data distribution and the other from the fake data distribution generated by the generator, with some policies. Fig 1 is the architecture of the general adversarial network.

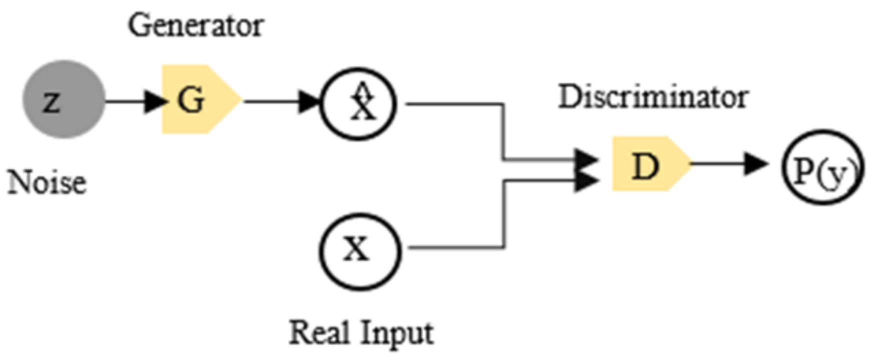


Fig. 3. Architecture of Standard GAN

DCGAN is one type of GAN to apply several convolutional layers in the architecture of GAN. The details are listed in the table 3.

Table 3. Parameters of the DCGAN

Hyperparameter	Setting
optimizer	Admax of Discriminator (lr=1e-5, beta1=0.5), Admax of Generator (lr=1e-5, beta1=0.5), Admax of Compiled (lr=2e-5, beta1=0.5).
dropout rate	0.1
batch normalization	Momentum=0.8
Leaky ReLU	Alpha=0.2
batch size	16
noise initializer	Random Uniform (-1,1)
loss	Mean Squared Error
monitor	Mean Absolute Error
epoch	2000

With the same audio data, the experimental result is shown as table 4. From the experiments' result, it can be seen that DCGAN is capable to achieve the accuracy of 0.716528104 which is relatively satisfying. Besides, the average execution time of DCGAN is 90 minutes in the GPU computing module. The computation cost of DCGAN is relatively high.

Table 4. Experimental Result of the DCGAN

AUC	PAUC	F Accuracy Score	MSE	Average Execution Time
0.716528104	0.706213763	0.51664777	0.483352227	90 mins

3.3. Noise Tolerance Test

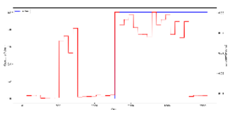
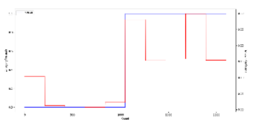
Another series of experiments are conducted to test the maximum noise tolerance of the enhanced IPCA based DCNN-AE. Based on the experiments results, the performance of the algorithm is impacted when the SNR is 3.0103 (SNR=10*Log(1/0.5)), in which 0.5 is the noise significancy factor.

3.4. Comparison of Hardcoded Architecture and Parameterized Architecture

The third series of experiments are to compare the prediction performance between the hardcoded architecture and parameterized architecture. The experiments' results show that the parameterized architecture achieves higher accuracy within less training time. For example, when the shape of framing is set as 256*256, the average AUC of the hardcoded architecture is around 0.7 with average execution time of 35 minutes, and the average AUC of the parameterized architecture is more than 0.8 with the average execution time of 15 minutes.

From the experiment result of one sample shown in Table 5, it can be observed that although the AUC of hardcoded architecture is 0.8875, the MSE is much higher than that of parameterized architecture. Therefore, the simulation result is not so satisfying comparing with parameterized architecture.

Table 5. Single Example: Comparison Between Hardcoded Architecture and Parameterized Architecture

Evaluation Item	Hardcoded Architecture	Parameterized Architecture
AUC	0.8875	1.0
pAUC	0.8026315789473684	1.0
MSE	0.46381313	0.31623548
Graph of the comparison between actual and predicted values		

3.5. Computer Module and OS Parameters

All computations except basement system ("DCGAN") were executed on 2.3 GHz quad-core processors (Turbo Boost up to achieve 3.8 GHz) PC computer with 8 Gb RAM iOS.
DCGAN was executed in Google Cloud Platform with 1 NVIDIA P100 GPU.

4. RESULTS AND ANALYSIS

The experiment is to pick up 110 datasets with 100 normal audio files from total 228 normal files as training dataset and 10 anormal audio files from total 100 anomalous files as test dataset to test the IPCA Based DCNN-AE algorithm. Tab 6 shows the summary of the test results.

Table 6. Summary of Performance Evaluation

Number of Test Cases	AUC	PAUC	MSE	Average Execution Time
50	0.815793282	0.713118455	0.27791114	15 mins

From Table 6, it can be seen that IPCA based DCNN-AE shows high accuracy with the average AUC and PAUC between 0.815793282 and 0.713118455. And the average execution time is about 15 mins for total data size of 97,020,000 (normal audio dataset for training:88,200,000, anomalous audio dataset for prediction: 8,820,000).

Fig 4 shows a sample of the prediction results when predicting the turning point changing from normal status to anomalous status. The results show that the IPCA based DCNN-AE predicts the turning point with high accuracy, and the AUC of the prediction reaches 1.0.

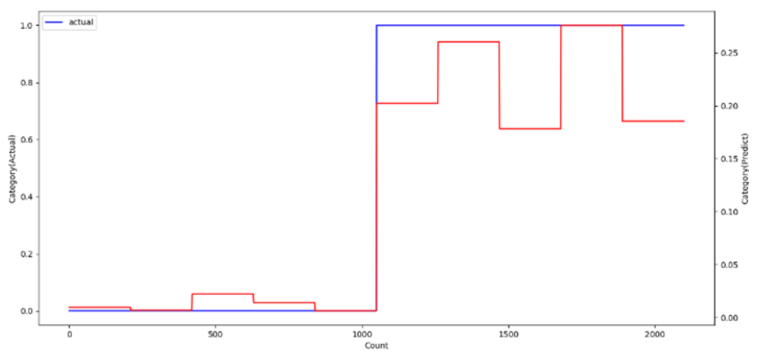


Fig. 4. ROC Curve of IPCA Based DCNN-AE Algorithms

Fig 5 shows the error rate and the loss rate of the sample training process of the IPCA based DCNN-AE. These two graphs show that the validation error and loss error decreases significantly in the first 50 batches and then keeps stable in the left batches.

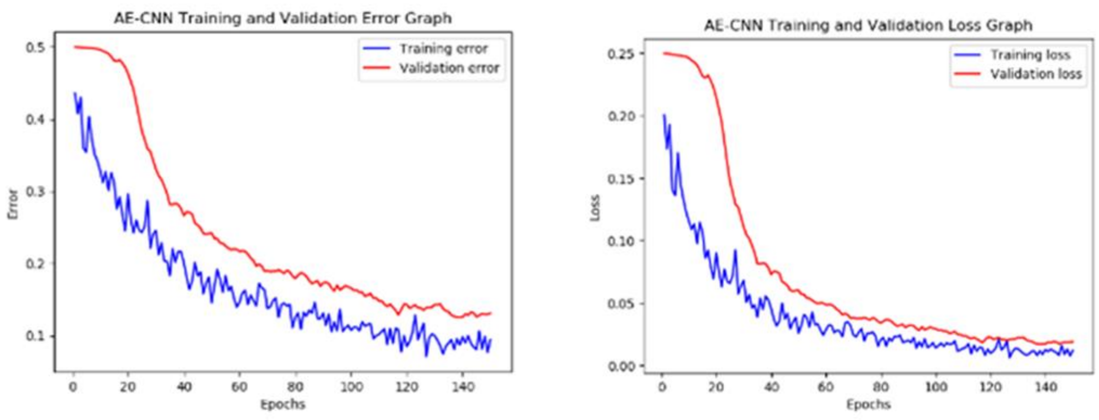


Fig. 5. Error Rate and Loss Rate of One Sample of DCNN-AE

Table 7 shows the AUC and PAUC comparisons among 4 semi-supervised and unsupervised machine learning algorithms: Kmeans++, one-class SVM[16], density-based spatial clustering of applications with noise ("DBSCAN")[43][44] and IPCA Based DCNN-AE.

From the experiments' results in Table 7, it can be observed that Incremental PCA based DCNN-AE shows the highest accuracy with the average AUC of 0.815793282, comparing with that of Kmeans++ of 0.499545351, of Incremental PCA based DBSCAN clustering of 0.636348073, of Incremental PCA based One-class SVM of 0.506749433 and of DCGAN of 0.716528104. From the four measures, Incremental PCA based DCNN-AE performs best among the unsupervised or semi-supervised algorithms.

Table 7. AUC Comparison Between Unsupervised and Semi-Supervised ML Algorithms

Machine Learning	**AUC**	**PAUC**	**F Accuracy Score**	**MSE**
Kmeans ++	0.499545351	0.499960161	0.499545351	0.500454647
IPCA+DBSCAN	0.636348073	0.636325593	0.636348073	0.363651927
IPCA+One-class SVM	0.506749433	0.500204267	0.497853267	0.493250568
DCGAN	0.716528104	0.706213763	0.51664777	0.483352227
IPCA+DCNN-AE	0.815793282	0.713118455	0.557364341	0.27791114

To compare the efficiency and computing cost of each unsupervised and semi-supervised algorithms, we adopt the average execution time and the GPU/CPU usage as the measurements. From Table 8, it can be observed that the IPCA based DBSCAN Clustering and the IPCA based One-class SVM consumes least execution time and computing cost. Considering the performance, the IPCA based DCNN-AE, although consumes more time as of 15 minutes, but reaches the highest accuracy comparing with other algorithms. And because the DCNN-AE doesn't require the computing resource of GPU(s), its computing cost is higher than clustering algorithms, and yet much lower than that of DCGAN.

Table 8. Efficiency and Computing Capability Requirement Comparison Between Unsupervised and Semi-Supervised ML Algorithms

ML/DL Algorithms	Average Execution Time Per Round (unit: minute)	CPU / GPU (unit: core)	Computing Cost Per Machine Per Month (unit: USD)
Kmeans ++	5	CPU	N. A
IPCA+DBSCANClustering	2	CPU	N. A
IPCA+One-class SVM	20	CPU	N. A
DCGAN	90	GPU (8 to 64)	1000 to 8000
IPCA+DCNN-AE	15	CPU	N. A

Fig 6 shows the ROC curves of those four semi-supervised and unsupervised machine learning algorithms. From the graph, it can be observed that DCNN-AE reaches the highest AUC value as of 0.815793282, one-class SVM and PCA+DBSCAN got the similar AUC of 0.65 and 0.54 respectively. The performances of Kmeans++ is the worst as of 0.5 respectively.

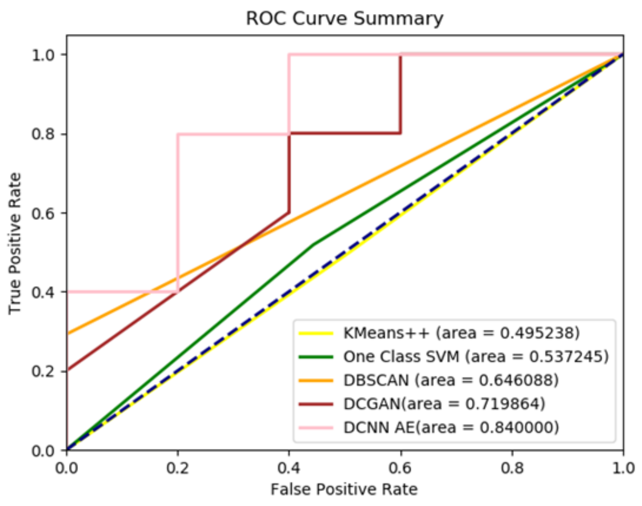


Fig. 6. ROC Curve of Unsupervised and Semi-supervised ML/DL Algorithms

5. Conclusion and Future Work

The enhanced IPCA Based DCNN-AE for Anomalous Sound Detection, when analyzing and detecting the machines' sounds of high-dimensional and significant density distributions between normal and anomaly, is proved to achieve high accuracy with lower computing cost because of the dimensions-reduction before DCNN-AE layers. However, the stability of the algorithm to keep the continuant high accuracy in each execution starting from a random point still needs large quantities of trainings to improve. Therefore, how to find the optimized parameters to keep balance among accuracy, efficiency and stability is important when considering deploying the algorithm in the industry wide.

In the experiment, 100 from total 228 normal files are adopted for training and 10 from total 120 anomalous files as test dataset. The train data size is 88,200,000, and the prediction data size is 8,820,000. With the increase of the data size to 1000 audio files for training and 100 files for test, the execution time is increased from 15 minutes to 120 minutes. However, the algorithm is still capable to be executed in CPU computing model with the unimpacted accuracy.

Furthermore, the application of the IPCA based DCNN-AE could be extended from Industry 4.0 to the intrusion detection of acoustic data. For example, the detection of voiceprint for smart appliances could adopt the new methodology with high computing efficiency and accuracy to reduce the risk of intrusion.

ACKNOWLEDGEMENTS

This project is partially supported by the funding from the HKU-SCF FinTechAcademy.

REFERENCES

[1] Harsh Purohit, Ryo Tanabe, Kenji Ichige, Takashi Endo, Yuki Nikaido, Kaori Suefusa, and Yohei Kawaguchi, “MIMII Dataset: Sound Dataset for Malfunctioning Industrial Machine Investigation and Inspection”, Proc. 4th Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE), 2019.

[2] Koizumi, Yuma, et al. "ToyADMOS: A dataset of miniature-machine operating sounds for anomalous sound detection." 2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA). IEEE, 2019.

[3] Sullivan, Greg, et al. Operations & maintenance best practices-a guide to achieving operational efficiency (release 3). No. PNNL-19634. Pacific Northwest National Lab. (PNNL), Richland, WA (United States), 2020.

[4] McCann, Michael, and N. Zaleski. "Deaths And Injuries Involving Elevators and Escalators." Retrieved on April 3 (2013): 2016.

[5] Electrical and Mechanical Services Department, “The Summaries of The Reported Lift Incidents.” Year from 2011 to 2021.

[6] Yuma Koizumi, Yohei Kawaguchi, Keisuke Imoto, Toshiki Nakamura, Yuki Nikaido, Ryo Tanabe, Harsh Purohit, Kaori Suefusa, Takashi Endo, Masahiro Yasuda, and Noboru Harada, "Description and Discussion on DCASE2020 Challenge Task2: Unsupervised Anomalous Sound Detection for Machine Condition Monitoring", arXiv e-prints: 2006.05822, 2020.

[7] Keith Grueneberg, Bongjun Ko, David Wood, Xiping Wang, Dean Steuer, Yeonsup Lim, “IoT Data Management System for Rapid Development of Machine Learning Models”, IEEE International Conference on Cognitive Computing (“ICCC”), 2019.

[8] Jae-joon Chung and Hyun-Jung Kim, “An Automobile Environment Detection System Based on Deep Neural Network and its Implementation Using IoT-Enabled In-Vehicle Air Quality Sensors”, Sustainability 2020, 12, 2475; doi:10.3390/su12062475, 2020.

[9] Hisashi Uematsu, Yuma Koizumi, Shoichiro Saito, Akira Nakagawa, and Noboru Harada, “Anomaly Detection Technique in Sound to Detect Faulty Equipment”, NTT Technical Review, Vol. 15 No. 8 Aug. 2017.

[10] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld, “Anomaly Detection for IoT Time-Series Data: A Survey”, DOI 10.1109/JIOT.2019.2958185, IEEE Internet of Things Journal.

[11] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A Survey." ACM computing surveys (CSUR) 41.3 (2009): 1-58.

[12] Mnasri, Zied, Stefano Rovetta, and Francesco Masulli. "Anomalous sound event detection: A survey of machine learning based methods and applications." Multimedia Tools and Applications 81.4 (2022): 5537-5586.

[13] Ricciardi, Carlo, et al. "Linear Discriminant Analysis and Principal Component Analysis to Predict Coronary Artery Disease." Health informatics journal 26.3 (2020): 2181-2192.

[14] Justin Salamon and Juan Pablo Bello, “Unsupervised Feature Learning for Urban Sound Classification”, 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[15] Peng Shi, Zhen Zhao, Huaqiang Zhong, Hangyu Shen and Lianhong Ding, “Agglomerative Hierarchical Clustering”, Concurrency Computat Pract Exper. 2021; 33:e6077.

[16] Shuangfei Zhai, Yu Cheng, Weining Lu, and Zhongfei (Mark) Zhang, “Deep Structured Energy Based Models for Anomaly Detection”, the 33rd International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48, 2016.
[17] Görnitz, Nico, Marius Kloft, Konrad Rieck, and Ulf Brefeld. "Toward Supervised Anomaly Detection." Journal of Artificial Intelligence Research 46 (2013): 235-262.
[18] Z Yin and J Hou, “Recent advances on SVM based fault diagnosis and process monitoring in complicated industrial processes”, Neurocomputing, 2016.
[19] Chao Dong, Chen Change Loy, and Xiaoou Tang, “Accelerating the Super-Resolution Convolutional Neural Network”, arXiv:1608.00367v1, Aug 2016.
[20] Yong Oh Lee, Jun Jo, Jongwoon Hwang, “Application of Deep Neural Network and Generative Adversarial Network to Industrial Maintenance: A Case Study of Induction Motor Fault Detection”, 2017 IEEE International Conference on Big Data (BIGDATA).
[21] Dan Garber, Elad Hazan, “Fast and Simple PCA via Convex Optimization”, arXiv:1509.05647v4, 25th Nov 2015.
[22] Akshay Balsubramani, Sanjoy Dasgupta, Yoav Freund, “The Fast Convergence of Incremental PCA”, arXiv:1501.03796v1 [cs.LG] 15 Jan 2015.
[23] Hugo Vieira Neto and Ulrich Nehmzow, “Incremental PCA: An Alternative Approach for Novelty Detection”, - Towards Autonomous Robotic Systems, 2005.
[24] Levy, Avraham, and Michael Lindenbaum. "Sequential Karhunen-Loeve basis extraction and its application to images." Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No. 98CB36269). Vol. 2. IEEE, 1998.
[25] David A. Ross, Jongwoo Lim, Ruei-Sung Lin, Ming-Hsuan Yang, “Incremental Learning for Robust Visual Tracking”. Int J Comput Vis (2008) 77: 125–141, DOI 10.1007/s11263-007-0075-7.
[26] Blake, Andrew, and Michael Isard. "The Condensation Algorithm-Conditional Density Propagation and Applications To Visual Tracking." Advances in Neural Information Processing Systems 9 (1996).
[27] Nauman Munir, Jinhyun Park, Hak-Joon Kim, Sung-Jin Song, Sung-Sik Kang, “Performance enhancement of convolutional neural network for ultrasonic flaw classification by adopting autoencoder”. NDT&E International 111 (2020) 102218.
[28] Burkni Palsson, Jakob Sigurdsson, Johnannes R. Sveinsson, and Magnus O. Ulfarsson, “Hyperspectral Unmixing Using a Neural Network Autoencoder”, Digital Object Identifier 10.1109/ACCESS.2018.2818280.
[29] Min Chen, Xiaobo Shi, Yin Zhang, Di Wu, Mohsen Guizani, “Deep Feature Learning for Medical Image Analysis with Convolutional Autoencoder Neural Network”, DOI 10.1109/TBDATA.2017.2717439, IEEE Transactions on Big Data.
[30] Chong Zhou, Randy C. Paffenroth, “Anomaly Detection with Robust Deep Autoencoders”, KDD 2017 Research Paper.
[31] Davide Chicco, Peter Sadowski, Pierre Baldi, “Deep Autoencoder Neural Networks for Gene Ontology Annotation Predictions”, ACM-BCB 2014.
[32] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, Dahua Lin, “Unsupervised Feature Learning via Non-Parametric Instance Discrimination”, CVPR 2018.
[33] Mayu Sakurada, Takehisa Yairi, “Anomaly Detection Using Autoencoders with Non-linear Dimensionality Reduction”, ACM 978-1-4503-3159-3, 2014.
[34] Tara N. Sainath, Abdel-Rahman Mohamed, Brian Kingsbury, Bhuvana Ramabhadran, “Deep Convolutional Neural Networks For LVCSR” 2013.
[35] Gu, Jiuxiang, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu et al. "Recent Advances in Convolutional Neural Networks." Pattern Recognition 77 (2018): 354-377.
[36] Steve Lawrence, C. Lee Giles, Ah Chung Tsoi, Andrew D. Back, “Face Recognition: A Convolutional Neural-Network Approach” Jan 1997.
[37] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet Classification with Deep Convolutional Neural Networks." Advances in neural information processing systems 25 (2012).
[38] Xiaohu Zhang, Yuexian Zou, Wei Shi, “Dilated Convolution Neural Network with Leaky ReLU for Environment Sound Classification”, 2017 22nd International Conference on Digital Signal Processing (“DSP”), 1-5, 2017.
[39] Diederik P. Kingma, Jimmy Lei Ba, “Adam: A Method for Stochastic Optimization”, ICLR 2015.

[40] Dokkyun Yi, Jaehyun Ahn and Sangmin Ji, “An Effective Optimization Method for Machine Learning Based on ADAM”, Applied Sciences, 2020.

[41] Xiao Tan, “Genetic Algorithm Based Bidirectional Generative Adversary Network for LIBOR Prediction”, Proceedings of the 8th International Conference on Emerging Internet, Data and Web Technologies (EIDWT 2020): Advances in Internet, Data and Web Technologies, Kitakyushu, Japan, 24-26 February 2020, p. 440-447.

[42] Jaroslav Kopčan, Ondrej Škvarek, Martin Klimo, “Anomaly detection using Autoencoders and Deep Convolution Generative Adversarial Networks”, 14th International Scientific Conference on Sustainable, Modern And Safe Transport.

[43] Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas, “C-DBSCAN: Density-Based Clustering with Constraints”, A. An et al. (Eds.): RSFDGrC 2007, LNAI 4482, pp. 216–223, 2007.

[44] Junhao Gan, and Yufei Tao, “On the Hardness and Approximation of Euclidean DBSCAN”, ACM Transactions on Database Systems, Vol. 1, No. 1, Article 1, Publication date: January 2016.