POLITECNICO DI TORINO

Master's Degree in MECHATRONIC ENGINEERING

Master's Degree Thesis

Development and implementation of an ensemble of intelligent techniques for the detection of abnormal conditions in safety-critical applications

Supervisors

Prof. NICOLA PEDRONI

Candidate

Jiaxing ZHANG

July 2024

Summary

An ensemble of statistical techniques and artificial intelligence methods is proposed for the detection of abnormal conditions (anomalies) in safety-critical applications. Stacked Autoencoders (SAEs), Autoencoders(AEs), isolation forest (IF), the Local Outlier Factor (LOF) method and Gaussian Mixture Models (GMMs) are considered in the ensemble. The outputs of these algorithms (i.e., properly defined anomaly scores) are normalized within an original probabilistic framework and combined for the robust and conservative identification of anomalies in datasets: different combination functions are employed, including "averaging", "maximization" and "AKPV", which averages the scores of the top three anomaly detectors. SAEs are also used as efficient tools for dimensionality reduction in the overall framework. The developed approach is tested on two functional datasets made of time series: 1) the ECG5000 dataset; and 2) a set of transients representing the time evolution of several physical parameters (e.g., fluid temperatures) taken from the simulator of a Generation-IV nuclear reactor, i.e., a Molten Salt Fast Reactor (MSFR). The global accuracy and the Area Under The Curve (AUC)-Receiver Operating Characteristics (ROC) scores are used for assessing the performance of the individual methods and of the ensemble. The results show that the use of an ensemble improves the performance in the detection of anomalies with respect to the individual detectors.

Acknowledgements

At the end of this long and difficult journey at Politecnico, I would like to express my warmest thanks to my supervisor, prof. Nicola Pedroni, for the valuable suggestion on my thesis work.

I would like to express my deepest gratitude to my parents for their unwavering support and encouragement throughout my journey.

I am also immensely grateful to my friends, whose support and camaraderie have been invaluable.

Table of Contents

List of Tables

List of Figures

Acronyms

DL	Deep Learning
AE	Autoencoder
SAE	Stacked Autoencoder
LSTM	Long Short-Term Memory
LSTM-AE	Long Short-Term memory based Autoencoder
FC-AE	Fully-Connected Autoencoder
LOF	Local Outlier Factor
IF	Isolation Forest
GMM	Gaussian Mixture Model
T-SNE	T-distributed Stochastic Neighbor Embedding

TPR	True Positive Rate
FPR	False Positive Rate
TNR	True Negative Rate
TNR	False Negative Rate

Chapter 1

Introduction

1.1 Background

Since the early 20th century, some fields such as Data Science, Machine Learning and Deep Learning has drawn significant attention across diverse industries. Thanks to the advancement of computing power and the growth of database, it is possible to implement such techniques in different industrial fields and applications, especially anomaly detection, one of the vital tasks [1].

Anomaly detection, also known as outlier detection, plays a crucial role in a wide range of fields, including healthcare, finance, cybersecurity, and industrial systems. In the operation and control of safety-critical systems and plants, operators are typically required to promptly identify their abnormal conditions and the corresponding causes, considering the time evolution of continuously. Many anomaly detection algorithms for time series signals have been proposed in recent years. The main categories of anomaly detection algorithms are based on statistics, density, distance, clustering, isolation, ensemble, and subspace[1]. In many machine learning tasks, ensemble-based methods are well known for their better performance compared to the individual methods[1]. Among almost all the anomaly detectors had been proposed, it is difficult to find a single winning algorithm or method that performs well across different datasets, despite the significant advancement in time series anomaly detection area[2].

The choice of applying the ensemble method to specific datasets, ECG5000 and MSFR datasets, is motivated by the real-world relevance of anomaly detection in healthcare and nuclear systems. Exploring and enhancing the performance of anomaly detection methods in these domains aims to contribute methodologies applicable to critical sectors.

This thesis holds significance in advancing the field of (times series) anomaly/outlier detection by means of the following contributions:

1. Most methods rely on their own “arbitrary” definitions of outliers and of how they differ from normal datapoints. This results from the lack of a rigorous definition of what constitutes “normal (resp., abnormal) behavior” and of a corresponding quantitative measure that can be universally accepted as such. To tackle this issue, the outlier detection algorithms here employed are originally tailored to produce normalized anomaly scores: actually, in many contexts, a quantifiable “score of outlyingness” is much more useful than a simple binary label of “inlier” or “outlier” (that many available techniques only provide), since there will always be data whose outlyingness may be considered too small to classify them as outliers, but whose score is still sufficient to be relevant when analyzing their taxonomy. In particular, in this thesis the outputs of the detection algorithms (i.e., properly defined anomaly scores) are normalized within an original probabilistic framework to motivate a statistical interpretability of the unified scores. Outlier scores usually are not equally distributed in their value range but follow a much more complex distribution, which is hard to grasp analytically, and many assumptions must be made to get a reliable result. Instead, it becomes much easier to analyze the actual score distribution on the data set without assuming anything on the distribution of the data. In [3] several normalization functions are proposed that are based on classical Gaussian, Gamma, F and Cauchy Probability Distribution Functions (PDFs). It should be again pointed out that we do not draw any assumptions on the distribution of the data, but on the distribution of the derived outlier scores. The intuition of this approach is the following: we do not have a direct way to interpret the score, we instead evaluate how “unusual” the score of a point is, using the algorithms output score as a one-dimensional feature projection of the data. Let us note that any distribution function can be used for this purpose analogously, depending on how well it fits to the distribution of the derived outlier scores. In this work, we employ an Empirical Cumulative Distribution Function (CDF) approach to fit the available scores and avoid making any assumption on their distribution.

2. Due to the very broad and diverse nature and taxonomy of the time series outliers, the performance of the available detection methods largely depends on the “physical origin” of the anomaly, on the nature and spatial distribution of the data points (i.e., depending on the type, sparsity and homogeneity of the dataset). In practice, no single detection algorithm can universally apply to all datasets. To address this problem, several diverse detection algorithms are considered within an ensemble framework and all the normalized anomaly scores produced are aggregated to get a unique, robust and reliable score. The underlying idea is that some algorithms will perform well on a particular subset of points (i.e., the outliers that are the most “obvious” from their

respective standpoints), whereas other algorithms will do better on other subsets of points. By so doing, the ensemble combination is often able to obtain more accurate and robust results because of its ability to combine the outputs of multiple algorithms and to “correct” errors committed by the single, diverse ensemble members. In this thesis, the (probabilistic) normalized anomaly scores are combined for the robust and conservative identification of anomalies in datasets: different combination functions are employed, including "averaging", "maximization", "damped average" and "AKPV", which averages the scores of the top three anomaly detectors.

3. The diversity of the existing methods hinders the comparability of the results. In an unsupervised framework the only valid way of testing the detection capabilities of a detection algorithm is to compare its provided results through a chosen set of toy examples. Most examples simply rely on the comparison of their detection rates in rather simple models, and they are almost never tested on sets of functional data coming from real systems or generated by numerical simulations. In this respect, in the present thesis the effectiveness of the proposed approaches is systematically assessed by means of several diverse statistical indicators. In particular, the global accuracy, the True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), False Positive Rate (FPR) and the Area Under The Curve (AUC)-Receiver Operating Characteristics (ROC) scores are used for assessing the performance of the individual methods and of the ensemble.

4. Two relevant and realistic functional datasets made of time series and taken from “safety-critical” sectors are considered: 1) the ECG5000 dataset (a collection of heartbeat signals); and 2) a set of transients representing the time evolution of several physical parameters (e.g., fluid temperatures) taken from the simulator of a Generation-IV nuclear reactor, i.e., a Molten Salt Fast Reactor (MSFR).

The outcomes of this thesis have the potential to improve the accuracy and efficiency of anomaly detection in practical scenarios, thereby enhancing the reliability and security of systems in various domains.

1.2 Overview of this thesis structure

The thesis is organized as follows. In chapter 2, we discuss related the frontier research in anomaly detection which contains the technologies with Machine Learning (ML) and Deep learning (DL). The overview of the method and algorithms that are used in anomaly detection is also presented in this section. In chapter 3, it contains the detailed introduction of algorithms that are AE, SAE, LSTM, T-SNE, LOF, GMM, IF. In chapter 4, the general introductio of ensemble, ecdf and the methods of combination of scores are presented. In chapter 5, it is the description of dataset, experimental setting and discussion of results. And it also contains the method related to ensemble of outputs from these methods and algorithms. In section 6, it is the conclusion.

Chapter 2

Related works

2.1 Frontier Research in Anomaly Detection

There are lots of survey works that introduce anomaly detection and compare performance of different algorithms and methods. Take [1] as example, the authors introduce the definition of anomaly and categories of anomaly. And the authors compare different algorithms among statistics-based, density-based, distance-based, clustering-based, isolation-based, ensemble-based, and subspace-based. In order to extract global feature from time series data, the authors of [4] proposed a model combined Transformer and 1-D CNN. This model consists of encoder with multiple Transformer layers and a decoder with one 1-D CNN layer. This method utilizes the self-attention mechanism from Transformer to extract global features of time series data. And some ensemble methods based machine learning models have developed. Such as [5], one ensemble method is proposed, named LODA, which ensemble different 'weak' detectors as a 'strong' detector to achieve high performance in particular tasks. LODA is a collection of k one-dimensional histograms, each approximating the probability density of input data projected onto a single projection vector. The projection vectors diversify individual histograms, improving the performance of a single detector. LODA emphasizes the speed of anomaly detection procedure so that it can operate in real-time system. And [6], which introduces the LSCP framework, which addresses the challenge of combining base outlier detectors in unsupervised outlier ensembles by emphasizing data locality and dynamic selection.

2.2 Common Algorithms for Anomaly detection

2.2.1 AE: Autoencoder

Autoencoder are a type of neural network used for unsupervised learning. For the sake of unsupervised learning, no need for labeled anomaly data, which can be scarce or difficult to obtain. The autoencoders can be used as data reconstruction and dimension reduction. But only one autoencoder is not always extract feature efficient, so the stacked autoencoders are introduced.

2.2.2 SAE: Stacked Autoencoder

Stacked Autoencoder is a variant of Autoencoder, which is wieldy used in anomaly detection tasks. By stacking more than one autoencders, SAE can reduce dimension lower than single AE. In time series signal anomaly detection, LSTM based block and CNN based block are always stacked together to extract local feature from original data. Furthermore, to extract global feature from complex time series signal, Self-attention mechanism based Autoencoder structure become more and more popular.

2.2.3 LOF: Local Outlier Factor

LOF is degree of being an outlier for each object proposed in [7]. It is more meaningful to assign to each object a degree of being an outlier, leading to the introduction of the concept of LOF, which quantifies how outlying an object is. This approach addresses the limitations of existing outlier detection methods and provides a more comprehensive understanding of outliers in datasets. The LOFs of inliers always have values that more or less equal to 1. And the ones of outliers always have values more than 1. The higher it is, the more it indicates an outlier for each object.

2.2.4 IF: Isolation Forest

Isolation Forest is a machine learning algorithm proposed in [8]. IF is a model-based anomaly detection method that isolates anomalies instead of profiling normal instances, achieving high detection performance and fast execution given the assumption that anomalies are data patterns which differ from normal instances and the number is significant less that the number of normal data. The IF algorithm is a two-stage process for anomaly detection. In the training stage, isolation trees are constructed using sub-samples of the training set. The tree height limit is set based on the sub-sampling size, focusing on data points with shorter-than-average path lengths, as they are more likely to be anomalies. The number of trees and the sub-sampling size are input parameters that can be adjusted based on the desired detection performance [8]. What's more, ReMass-iForest [9] is variant of iForest, which use relative score instead of global score as used in iForest. Due to relative score, ReMass-iForest is able to detect local anomalies, which is shortcoming of iForest [1].

2.2.5 GMM: Gaussian Mixture Model

Gaussian Mixture Model is also called Finite Mixture Model (FMM). GMM is an ancient model that more than 100 years old. Thanks to Expectation Maximization (EM) algorithm, FMM refreshed. The EM algorithm is ideally suited to problems that estimation of the parameters [10]. GMM fits multiple Gaussian models from dataset, which the number of models could be consider as hyperparameter and the parameters of models (for gaussian distribution, the parameters are mean and variance) could be found by EM algorithm. Given the assumption that normal data concentrate with high probability density and abnormal data lie on the region with low probability density, probability density for each object could be set as a criteria to judge abnormality, which lower values represent anomalies [11].

2.2.6 OCSVM: One Class Support Vector Machine

SVMs use reprocessing strategy in learning by mapping the input space to feature space for classification. And SVMs try to find a hyperplane that maximizes the margin between itself and the nearest training points [12]. OCSVM is a variant of SVM for anomaly detection. OCSVM tries to find a hyperplane to separate the training data form the origin, considering the abnormal data lying on the origin. This stimulates the thought that this consideration is the weak point of OCSVM as this requires the origin to be a member of the outlier class [13].

2.2.7 k-NN: k-nearest neighbors

K-NN assums that the abnormal data is far away fram its-nearest neighbour, anormaly score computed as distance between data instace and its k-NN [1]. K-NN is expensive on staring and computation complexity by storing all training data and calculating the distance of testing data and all training data. Base k-NN, some more improved variants are proposed, like kth-NN [14], Sp [15].

2.2.8 k-means

K-means is an unsupervised learning algorithm primarily used for clustering data into 'k' clusters based on feature similarity. For anomaly detection, k-means can help identify data points that do not fit well into any cluster, which are considered anomalies or outliers. Furthermore, Outlier Finding Process (OFP) [16] is a anomaly detetction method based k-means. It uses modified k-means clustering and then minimum spanning tree is constructed. The tree with less nodes are considered as anomalies [1].

Chapter 3

Methodologies considered in this thesis

3.1 Autoencoder

The autoencoder (AE) stands as an importable neural network architecture within machine learning. Recognized for its capacity in unsupervised learning, autoencoders operate without the necessity of explicit labels during the training phase. It's worth noting that autoencoders can be more accurately described as self-supervised, as they derive their own labels from the training data [7]. AE also function as a generative model, and AEs adept at producing reconstructed data. This characteristic has led to widespread utilization of autoencoders in anomaly detection tasks. Typically, an autoencoder is trained to accurately reconstruct normal data while encountering difficulty in reconstructing abnormal data, thus yielding a substantial reconstruction error. Additionally, the output of the 'bottleneck' layer, termed the 'code', may exhibit distinctions between normal and anomalous data. The measurement of 'reconstruction error' commonly serves as the criterion for distinguishing between anomalies and normal data.

An autoencoder consists of encoder and decoder. The encoder is responsible for mapping input data to a latent space (usually in the lower dimension in comparison with original dimension). Typically composed of multiple hidden layers, each containing multiple neurons. The decoder receives the encoded data and attempts to reconstruct the original input. Structurally symmetrical to the encoder, using multiple hidden layers to progressively reflect the structure of the input data. The output of the encoder is the encoding in the latent space, which is a compressed representation of the input data. The dimensionality of this latent space is usually lower than that of the input data, achieving dimensionality reduction. The structure of AE is shown in the figure 3.1. Image source is [17].

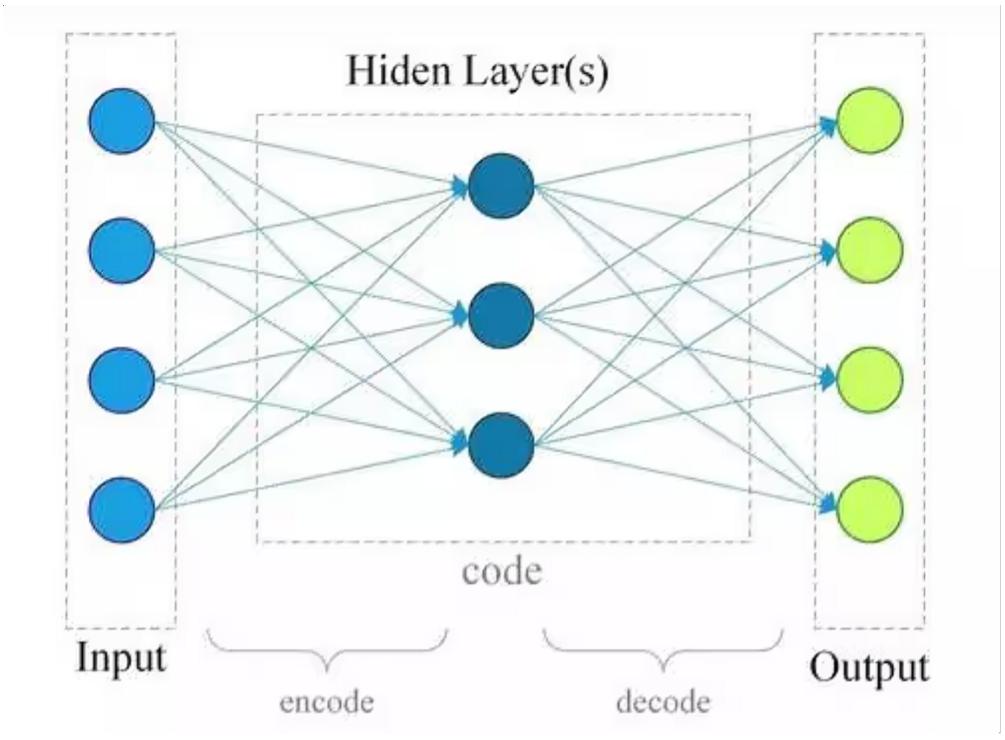


Figure 3.1: sample structure of autoencoder

Let a dataset $X = [x_1, x_2, x_3, \dots, x_N]$ be *i.i.d.* sample of data, where $x_i \in R^d, X \in R^{N\times d}$. feed X to autoencoder, the encoder of AE performs dimensional reduction on X to the defined dimension of latent codes. And the latent representation is $a^{(1)} = [a_1^{(1)}, a_2^{(1)}, a_3^{(1)}, \dots, a_N^{(1)}]$

$$a^{(1)} = f(W_1 X + B_1)$$

Where f, W_1, B_1 are activation function, weights matrix and bias matrix respectively. As a sequence, decoder of AE starts to reconstrue the original data from latent codes a to reconstructed output is

$$X_reconstructed = g(M_2 a + C2)$$

where g, M_2, C_2 are activation function, weights matrix and bias matrix. Through adding more layers to encoder and decoder, AE could be built as deep autoencoder, shown in figure 3.2. Image source is [18].

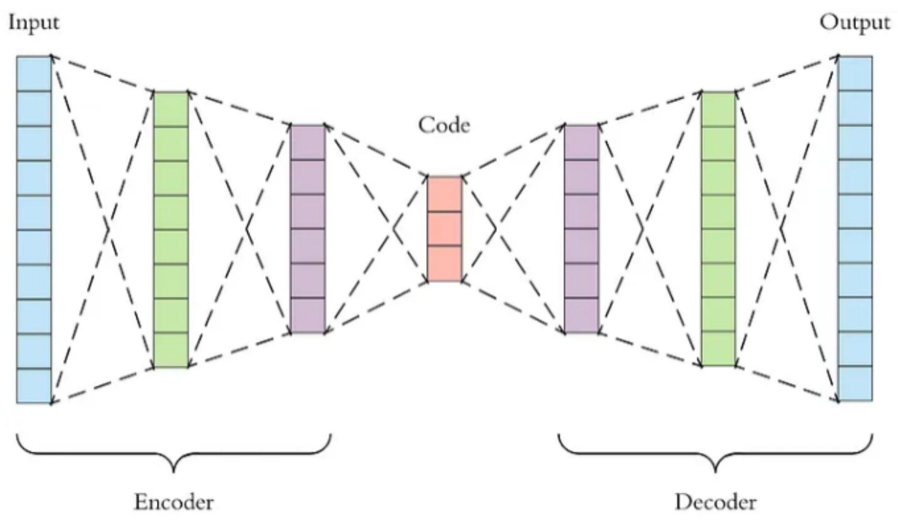


Figure 3.2: sample structure of deep autoencoder

In terms of deep autoencoder, the laten representation

$$a^{(1)} = f_n(W_n \dots f_2(W_2 f_1(W_1 X + B_1) + B_2) + \dots + B_n)$$

where n is the number of layers of encoder. The reconsturucted data is

$$X_{reconstructed} = g_m(M_m \dots g_2(M_2 g_1(M_1 a + C_1) + C_2) + \dots + C_m)$$

where m is the number of layers of decoders.

3.2 Stacked Autoencoder

Stacked Autoencoder (SAE) emerges as a variant of conventional AE, which always boasts better performance not only on reconstruct data but also feature extraction. SAE consists of multiple AEs (usually with multiple layers). In SAE, each AE has an encoder and a decoder. In usage phase, the more meaningful output is not reconstructed data rather than the 'code' that the outputs of 'bottleneck' layer. However, in training phase, reconstructed error is used for updating wights with back propagation.

In SAE, the encoding from the previous AE is connected to the input of the subsequent autoencoder, forming a cascading structure. This connection continues until the final AE, whereupon the encoding is linked to the decoder input of the subsequent autoencoder in reverse order, ultimately reaching the initial AE. This results in a stacked structure where each autoencoder relies on the representation learned by the previous autoencoder, the 'code' of each autoencoder serves as the input for the next one. This allows the network to learn hierarchical representations of the data, with each layer capturing more complex features. In general, the outer AE engage to reconstruct data and inner AE engage to feature extraction.

In practice, the outermost autoencoders primarily focus on data reconstruction, while the inner autoencoders emphasize feature extraction. Therefore, during the training phase, careful attention must be directed towards the reconstruction ability of the outer autoencoders, while the inner autoencoders prioritize feature extraction. For a visual depiction of the SAE structure, refer the figure 3.3. Image source is [19]

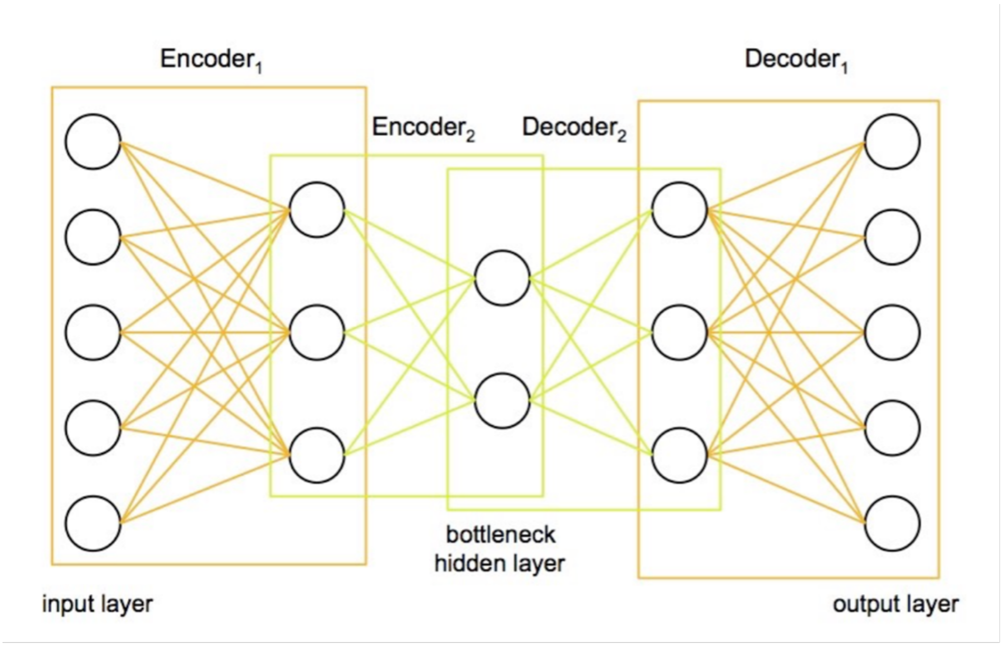


Figure 3.3: sample structure of SAE

In terms of deep autoencoder, the latent representation

$$a^{(1)} = f_{encoder\ n}(f_{encodern-1} \cdots f_{encoder2}(f_{encoder1}(X)))$$

$$f_{encoder1} = f_{1m_1}(W_{1m_1} \dots f_{12}(W_{12} f_{11}(W_{11}X + B_{11}) + B_{12}) + \dots B_{1m_1})$$

$$f_{encodern} = f_{nm_n}(W_{nm_n} \dots f_{n2}(W_{n2} f_{n1}(W_{n1}X^{(n-1)} + B_{n1}) + B_{n2}) + \dots B_{nm_1})$$

Where W_{nm_n} is the $m_n th$ wights matrix function for the nth encoder, B_{nm_n} is the $m_n th$ bias matrix for the nth encoder, and f_{nm_n} is the $m_n th$ active function for the nth encoder. X^{n-1} is the output of 'bottleneck' layer of the $(n-1)th$ encoder. The reconstructed data is

$$X_{reconstructed} = g_{decoder\ n}(g_{decodern-1} \cdots g_{decoder2}(g_{decoder1}(X)))$$

$$g_{decoder1} = g_{1m_1}(M_{1m_1} \ldots g_{12}(M_{12}g_{11}(M_{11}a + C_{11}) + C_{12}) + \ldots C_{1m_1})$$

$$g_{decodern} = g_{nm_n}(M_{nm_n} \cdots g_{n2}(M_{n2}g_{n1}(M_{n1}a^{(n-1)} + C_{n1}) + C_{n2}) + \ldots C_{nm_1})$$

Where M_{nm_n} is the m_nth wights matrix function for the nth decoder, C_{nm_n} is the m_nth bias matrix for the nth decoder, and g_{nm_n} is the m_nth active function for the nth decoder. a^{n-1} is the output of 'bottleneck' layer of the $(n-1)th$ decoder. And a is the 'code'.

Thank to this chain structure, encoder of SAE could increasing do dimensional reduction and extract high-level features from original data [19]. The bottleneck dimension of SAE could be lower than one of single AE.

3.3 LTSM: Long Short-Term Memory

Long Short-term Memory (LSTM) block is widely used in Recurrent Neural Networks (RNNs), particularly acclaimed for its effectiveness in extracting temporal features from time series data. In traditional RNNs, recurrence means repetitive multiplication, which introduces two issues, gradient vanish and exploring. These challenges impede the training of RNNs and hinder their ability to capture information over long sequences.

To address these obstacles, the LSTM architecture was introduced as a specialized variant of RNNs. LSTM is specifically designed to solve the long-term dependency problem in traditional RNNs. Gradient exploring is achieved through the incorporation of 'memory unit' and 'gate' mechanisms. This memory unit can selectively remember or forget information and does so through a gating mechanism.

LSTM comprises three gates: input gate, forget gate and output gate. These gates regulate the flow of information, which how input is processed in the memory unit, when previous information is forgotten with forget gate, and new information is propagated forward with output gate.

Overall, the architectural design of LSTM empowers it to effectively process and retain information over extended sequences, making it particularly adept at handling temporal data. The structure of LSTM block is shown in the figure 3.4. Image source is [19].

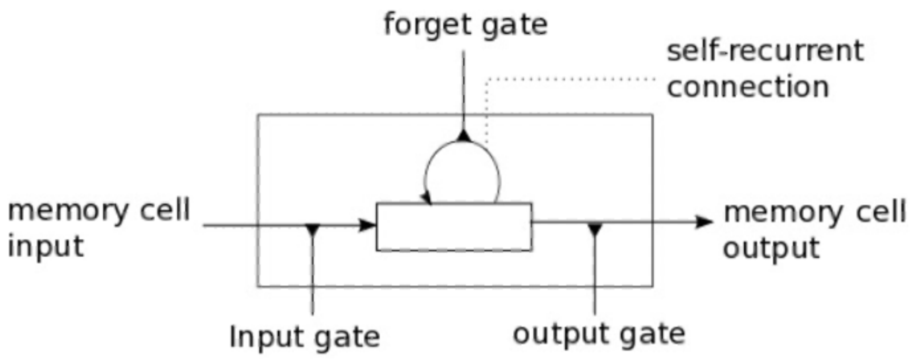


Figure 3.4: sample structure of LSTM block

In real usage case, there are five activation functions added for input gate signal, memory cell input gate, forget gate, memory cell output gate and output gate. And there are four inputs, and one outputs. The output of the block is recurrently connected back to the block input and all of the gates [20]. The real usage structure is shown in the figure 3.5.

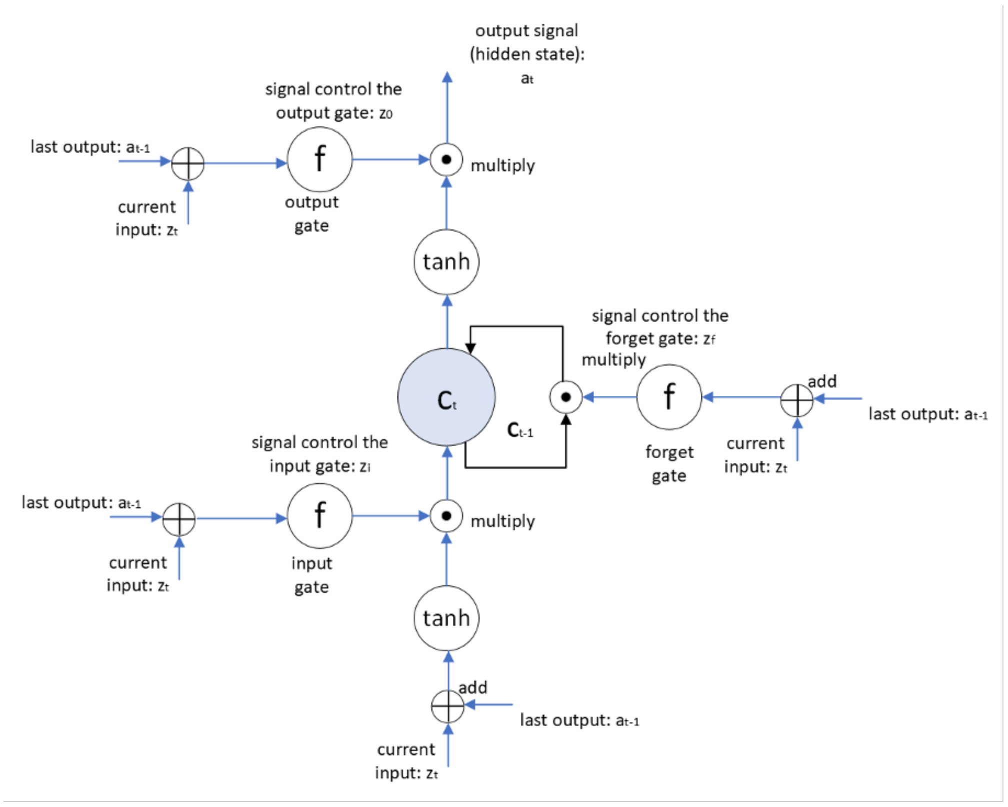


Figure 3.5: sample structure of LSTM

Where f is gate activation function, always is sigmoid function to mimic open and close.

According to the structure displayed by above image.

$$z_i = f(W_{ii}z_t + b_{ii} + W_{ai}a_{t-1} + b_{ai})$$

$$z_o = f(W_{io}z_t + b_{io} + W_{ao}a_{t-1} + b_{ao})$$

$$z_f = f(W_{if}z_t + b_{if} + W_{af}a_{t-1} + b_{af})$$

$$i_t = tanh(W_{ii}z_t + b_{it} + W_{at}a_{t-1} + b_{at})$$

$$C_t = z_f \otimes C_{t-1} + z_i \otimes tanh(i_t)$$

$$a_t = z_o \otimes tanh(C_t)$$

Where z_i, z_o, z_f are signals that control input gate, output gate and forget gate respectively. And i_t is input signal for ‘memory unit’ at the current time, where W_{it} and W_{at} are the weights associated with current input z_t and last time hidden state $a_(t-1)$, respectively, while b_{it} and b_{at} are bias vectors. For input gate, W_{ii}

and W_{ai} are the weights associated with current input z_t and last time hidden state a_{t-1}, respectively, while b_{ii} and b_{ai} are bias vectors. In terms of output gate, W_{io} and W_{ao} are the weights associated with current input z_t and last time hidden state a_{t-1}, respectively, while b_{io} and b_{ao} are bias vectors. For forget gate, W_{if} and W_{af} are the weights associated with current input z_t and last time hidden state a_{t-1}, respectively, while b_{if} and b_{af} are bias vectors. C_t is current cell state, and C_{t-1} is cell state at the previous time stamp. In the real usage, during training, all the weights metrics and bias vectors are calculated by backpropagation through time (BPTT) [21] algorithm like other neural networks (NNs).

3.4 T-SNE: T-Distributed Stochastic Neighbor Embedding

T-Distributed Stochastic Neighbor Embedding (T-SNE) is a dimensionality reduction technique used to visualize high-dimensional data. In comparison with Principal Component Analysis (PCA), a very famous linear dimensionality reduction technique, T-SNE mainly does non-linear dimensional reduction. For the sake of visualization, T-SNE maps the high-dimensional (hundreds even thousands dimension) datapoints to two or three dimensional datapoint [22]. T-SNE is a variant of Stochastic Neighbor Embedding (SNE) with improvements. T-SNE tries to find a similar distribution on low-dimensional space from high-dimensional space, in the other words, T-SNE tries to adjust the points in low-dimensional space in order to let the points that are similar in high-dimensional space become in low-dimensional space.

Given the data $X = \{x_1, x_2, \ldots, x_n\}$, and an initial hyperparameter perplexity: *prep*. Perplexity can be interpreted as a smooth measure the effective number of neighbors. The performance of SNE is fairly robust to changes in the perplexity, and typical values are between 5 and 50 [22]. The parameter is, in a sense, a guess about the number of close neighbors each point has. The perplexity value has a complex effect on the resulting pictures [23]. And optimization parameters: number of iterations T, learning rate: η and momentum: $\alpha(t)$ are also user defined as initial hyperparameters. T-SNE calculates pairwise $p_{(j|i)}$ in original data space with pre-defined *perp*.

$$p_{(j|i)} = \frac{exp(\frac{-\|x_j - x_i\|)^2}{2\sigma_i^2})}{\sum_{k \neq i} exp(\frac{-\|x_j - x_i\|)^2}{2\sigma_i^2})}$$

Where $p_{(i|i)} = 0$, and σ_i is variance of Gaussian distribution that centered on datapoint x_i, and it is selected according to *perp* with the equation:

$$prep(P_i) = 2^{H(P_i)}$$

where P_i is is probability induced by σ_i, and $H(P_i)$ is Shannon entropy of P_i, which is:

$$H(p_i) = -\sum_j p_{j|i} log_2 p_{j|i}$$

SNE performs a binary search for the value of σ_i that produces a P_i with a fixed perplexity[22].

And set:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2}$$

And then sample initila solution $Y^{(0)} = \{y_1, y_2, \ldots, y_n\}$, which has a difference on the final result because the lost function is non-convex. Subsequently, T-SNE performs T iterations for finding the optimized solution with equation:

$$q_{ij} = \frac{(1 + \|y_j - y_i\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$

And the gradient is:

$$\frac{\partial L}{\partial y_i} = 4\sum_j (p_{ij} - qij)(y_i - y_j)((1 + \|y_j - y_i\|^2)^{-1})$$

Where L is loss function powered by KL divergence:

$$L = KL(P\|Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$$

$$Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial L}{\partial Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$$

3.5 LOF: Local Outlier Factor

LOF is an algorithm used to detect outliers in a data set. LOF is able to identify data that has anomalous behavior relative to its neighboring data points, not just globally. This makes it useful in various fields such as anomaly detection, data cleaning, and pattern recognition. The core idea of LOF is to determine the degree of anomaly by calculating the density ratio of each data point to its neighboring points. Points with lower density may be outliers. This local approach makes LOF perform well when processing data in regions of different densities. LOF is a degree of being an outlier to each object.

Given the data $X = x_1, x_2, \ldots, x_n$, to calculated the LOF with equation:

$$LOF_k(x_i) = \frac{1}{|N_k(x_i)|} \sum_{o \in N_k(x_i)} \frac{lrd_r(o)}{lrd_k(x_i)}$$

Where $lrd_k(.)$ is the local reachability density of each data, $N_k(x_i)$ is the $k-nearest$ neighbors of observation x_i. $|N_k(x_i|$ is the number of observations in $N_k(x_i)$. Local reachability density $lrd_k(.)$ could be given by the equation:

$$lrd_k(x_i) = 1/\frac{\sum_{o \in N_k(x_i)} reach\_dist_k(x_i,0)}{|N_k(x_i)|}$$

Where $reach\_dist_k(x_i, o)$ is reachability distance of observation x_i with respect to observation o, which is defined as

$$reach\_dist_k(x_i, o) = max(d_k(o), d(x_i, o))$$

Where $d_k(o)$ is the kth smallest distance among the distances from observation o to its neighbors. And $d(x_i, o)$ is the distance between observation x_i and observation o. $|N_k(x_i)|$ is always greater than or equal k. In LOF, k and the method to calculate the distance could be considered as hyperparameters [7].

3.6 IF: Isolation Forest

IF is an anomaly detection method based tree structure [8]. The main idea of Isolation Forest is to isolate outliers by constructing randomly divided binary trees. Unlike other algorithms, Isolation Forest adopts a bottom-up tree building strategy, that is, starting from the entire data and gradually creating subtrees. In the process of building the tree, the algorithm divides the data into different subsets through random selection of features and random cuts. In Isolation Forest, normal samples usually require a long path to be isolated, while outliers are relatively short. Therefore, by measuring the path length required for the sample to be isolated, it can be used to determine whether the sample is an outlier. The advantage of this approach is that outliers have relatively short isolated paths in the tree and therefore can be detected faster, while normal samples require longer paths, making the algorithm more efficient. In this case, IF consider the path length after normalization as anomaly score, which higher score, the more it indicates an outlier for each object.

IF is distinguished from existing model-based, distance-based and density-based methods with such characters [8]:

- 1. IF allows to use small data samples to build batter isolation trees (IF) with the help of isolation characteristic.

- 2. IF eliminates major computational through removing the distance or density measures.

- 3. IF has a linear time complexity with a low constant and a low memory requirement, so that it occupies low memory usage.
- 4. IF is high scalable

Through building iTrees, IF builds isolation forest. Given the dataset $X = x_1, x_2, \dots, x_n$ of n instances from a d-variat distribution, to build an iTree. And X is divided by selecting an attribution randomly until the meet the following requirements [8]:

- 1. The number of datapoint in remanding dataset $|x| = 1$.
- 2. The tree reaches a heigh limits l, usually $l = celling(log_2 \varphi)$. Where φ is the size of dataset.
- 3. All data in remanding dataset are same.

The number of iTrees in IF could be considered as hyperparameter t. As usual, $t = 100$ [8]. After training phase, one IF has been built with t iTrees. Each iTree contributes a path length $h(x_i)$ for each data point x_i. Anomaly score for each observation $s(x_i)$:

$$s(x_i) = 2^{\frac{E(h(x_i))}{c(n)}}$$

Where, $E(h(x_i))$ is the averae of $h(x_i)$ from all iTress. And c(n) is the average of $h(x_i)$ given n iTrees, which could be considered as normalization of $h(x_i)$, could be calculated by

$$c(n) = 2H(n-1) - (\frac{2(n-1)}{n})$$

where, $H(i)$ is harmonic number and it can be calculated by $H(i) = ln(i) + 0.5772$.

3.7 GMM: Gaussian Mixture Model

GMM is a collection of Gaussian distribution models that represent overall population, which each Gaussian distribution model represent distribution of subpopulation. In anomaly detection field, each Gaussian distribution model represents normal subpopulation. In this case, anomalous data points lie very 'edging' region standing for very small probability density, with assumption that the number of anomalies is significant less that the number of normal data. Under the help of EM algorithm, GMM is reborn. But it is possible for GMM to reach a local optimization or saddle point, since EM algorithm maximizes the likelihood literately.

Given a dataset $X = \{x_1, x_2, \ldots, x_N\}, x_i \in R^n$. And a GMM has parameters that means and variances for each single model. And the formular is following:

$$p(x_i) = \sum_i^K \phi_i N(x_i|\mu_i, \Sigma_i)$$

$$N(x_i|\mu_i, |\Sigma_i) = \frac{1}{\sqrt{2\pi^K|\Sigma_i|}} exp(-\frac{1}{2}(x_i - \mu_i)^T \Sigma_i^{-1}(x_i - \mu_i))$$

$$\sum_i^K \phi_i = 1$$

Where the probability density for each observation $p(x_i)$ is the weighted sum of all probability densities from each single. And ϕ_i is weight for each model. What's more, if ϕ_i is learned, it could be seen as a-posteritori information. Instead, if ϕ_i is not learned, it could be seen as a-priori information. K is the number of components, which could be considered as hyperparameter.

For training phase, it's time for EM algorithm to play.

First step is that randomly defined initial parameters $\hat{\mu}$ and $\hat{\Sigma}$ for K componnets

$$\hat{\mu} = \{\hat{\mu_1}, \hat{\mu_2}, \ldots, \hat{\mu_K}\}$$

$$\hat{\Sigma} = \{\hat{\Sigma_1}, \hat{\Sigma_2}, \ldots, \hat{\Sigma_K}\}$$

$$\hat{\phi} = \{\hat{\phi_1}, \hat{\phi_2}, \ldots, \hat{\phi_K}\} = \{\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K}\}$$

Where $\hat{\mu}$ is means that are usually randomly selection from datapoints. And $\hat{\Sigma}$ is initial variance that are initial with the equation.

$$\hat{\Sigma} = \frac{1}{N}\sum_{i=1}^N (x_i - \bar{x})^2$$

$$\bar{x} = \frac{1}{N}\sum_{i=1}^N (x_i)$$

where $\hat{\phi}$ is set as average.

And the second step is the first step for EM algorithm, which is Expectation (E) step. In E step, the probability $\hat{\gamma_{lk}}$ of x_i from each mode is calculated with the initial parameters.

$$\hat{\gamma_{lk}} = \frac{\hat{\phi_k} N(x_i|\hat{\mu_k}, \hat{\Sigma_k})}{\sum_{j=1}^K \hat{\phi_j}(x_i|\hat{\mu_j}, \hat{\Sigma_j})}$$

Subsequently, with $\hat{\gamma_{lk}}$, all parameters are calculated $\hat{mu}, \hat{\Sigma}$ and $\hat{\phi}$ again with the following equation:

$$\hat{\phi_k} = \sum_{i=1}^{N} \frac{\hat{\gamma_{lk}}}{N}$$

$$\hat{\mu_k} = \frac{\sum_{i=1}^{N} \hat{\gamma_{lk}} x_i}{\sum_{i=1}^{N} \hat{\gamma_{lk}}}$$

$$\hat{\Sigma_k} = \sqrt{\frac{\sum_{i=1}^{N} \hat{\gamma_{lk}} (x_i - \hat{\mu_k})^T (x_i - \hat{\mu_k})}{\sum_{i=1}^{N} \hat{\gamma_{lk}}}}$$

EM algorithm will repeat expectation (E) and maximization (M) steps until the parameter's estimates converge.

Chapter 4

Ensemble based outlier detection

4.1 General introduction of ensemble

Ensemble methods are techniques that create multiple models (in outlier detection field, it could be called as detectors), and combine them to produce improved results. The central idea is that a group of weak detectors can come together to form a strong detectors, reducing errors and improving prediction accuracy.

The command types of ensemble methods:

1. Bagging: Involves training multiple models on different random subsets of the training data and combining their outputs

2. Boosting: Sequentially trains models, with each new model focusing on correcting errors made by previous models.

3. Stacking: Uses predictions of several models (base learners) as input features for a higher-level model (meta-learner), which makes the final prediction.

4. Voting: Combines predictions of multiple models through majority voting (for classification) or averaging (for regression).

5. Bagged Ensembles of Neural Networks: Trains several neural networks on different bootstrap samples of the training data and averages their predictions.

6. Snapshot Ensembles: Saves multiple snapshots of a single neural network during training and combines their predictions.

The advantages of ensemble:

1. Improved Accuracy: Ensembles often achieve higher accuracy than individual models by combining their strengths and reducing weaknesses.

2. Robustness: They are generally more robust to overfitting, as the ensemble averages out the noise and reduces the variance of predictions.

3. Error Reduction: Aggregating predictions helps in reducing both bias and variance, leading to better generalization on unseen data.

4. Versatility: Ensembles can combine different types of models to leverage the unique strengths of each.

The elements that give the ensemble advantages also induce the disadvantages. The disadvantages of ensemble:

1. Increased Complexity: Training and managing multiple models induces more computation complexity, in comparison with single model.

2. Interpretability: Ensembles can be harder to interpret compared to single models, especially when combining different model types, even though the interpretability problem commonly exists in deep learning field.

3. Inference Time: Predictions can be slower, as they involve running multiple models, which can be problematic for real-time applications. For the sake of real-time applications, ensemble could be more expensive.

4. Resource consuming: Maintaining and updating multiple models can be challenging and resource-intensive. Deploying multiple detectors needs more hardware resourse. For some resource limited hardware, ensemble could be impossible.

5. Hyperparameter Tuning: Ensemble methods often require careful tuning of hyperparameters for each base model and the ensemble itself, adding to the complexity of the process. In this thesis, we use autoencoders to do dimensional reduction. The dimension of the 'botllneck' layer is crucial. Too small dimension could cause feature lossing and too large dimension could make the dimensional reduction meaningless.

Since ensemble combines the different anomaly scores from the different algorithms and each algorithm of the ensemble provides scores of different magnitude and nature, in order to aggregate them, they need to be normalized and made "homogeneous" by transformation. In this work, we employ an Empirical Cumulative Distribution Function (CDF) approach to fit the available scores and avoid making any assumption on their distribution.

4.2 Scores Transformation – CDF

The Empirical Cumulative Distribution Function (CDF) is a statistical tool used to visualize the distribution of a dataset. It's constructed by sorting the data in ascending order and assigning each data point a cumulative probability based on its rank. In this thesis, we use ecdf to make the scores from different models in the same range (from 0 to 1).

The first step to creat the ecdf is sorting the data to arrange the data points in ascending order. And then, the second step, calculate cumulative probability. For each data point, assign a cumulative probability that represents the proportion of data points less than and equal to that value. This can be calculated using the formula:

$$P(x) = \frac{number\ of\ data\ points <= x}{total\ number\ of\ data\ points}$$

For visualisation, it would draw a diagram that rises in the shape of a staircase. The least largest value has the one hundred percent. Ortherwise, the least value has the smallest pertages.

4.3 The method of combiantion scores

The methods of combination scores are used in this thesis:

1. Maximization: The maximization defines the final scores to each datapoint as the maximum of the scores among all detectors.
2. Average: The average algorithm attributes an outlier score as final scores to each datapoint based on the average of the scores yielded by each individual detector.
3. Damped Average: The damped average defines the final scores to each datapoint as the average of the square root of anomaly scores from all detectors.
4. AKPV: The AKPV attributes the final anomaly score to each datapoint based on on the average of the scores of top three detectors.

Chapter 5

Experimental Design and Results Analysis

5.1 Experimental Design and Results Analysis on ECG5000

In this thesis, we utilize Stacked Autoencoder (SAE) (on ECG5000 dataste) and AEs (on MSFR dataset) not only as an anomaly detector but also for dimensional reduction purposes. The reconstructed error serves as the criterion for determining whether each observation is normal or anomaly. Additionally, the “code” extracted from the “bottleneck” layers of SAE and AEs are employed for subsequent anomaly detection algorithms.

5.1.1 Description of ECG5000 dataset

The first dataset is ECG5000 [24] which is a collection of heartbeat signals. There are five thousand of samples in this collection, whose number is large enough for training a deep neural network. The original dataset of "ECG5000" is a 20-hour long ECG downloaded from Physionet[25] which is “dataset bank”. The original dataset is “BIDMC Congestive Heart Failure Database” in the original publication [26], also called “chfdb”. ECG5000 is the record of “chf07”. This database of “chfdb” contains long-term ECG recordings from 15 subjects (11 men, aged 22 to 71, and 4 women, aged 54 to 63) with severe congestive heart failure (NYHA class 3–4) [27]. Signal preprocessing is crucial not only in biomedical field but also in other industrial fields due to presence of noise, artifacts and any trends that can influence the results [28]. According to the discerption of ECG5000 [10], the signals were pre-processed in two steps:

1. Extracting each heartbeat

2. Making each heartbeat equal length using interpolation

This dataset was originally used in "A general framework for never-ending learning from time series stream" [29].

In ECG5000 dataset, there are five categories which are normal, ‘PVC’, ‘R on T’, ‘SP’ and ‘UB’, In this thesis, the four categories expect for ‘normal’, are considered as severe congestive heart failure and fused together as anomaly. The sample waveform of heartbeats is shown in the figure 5.1.

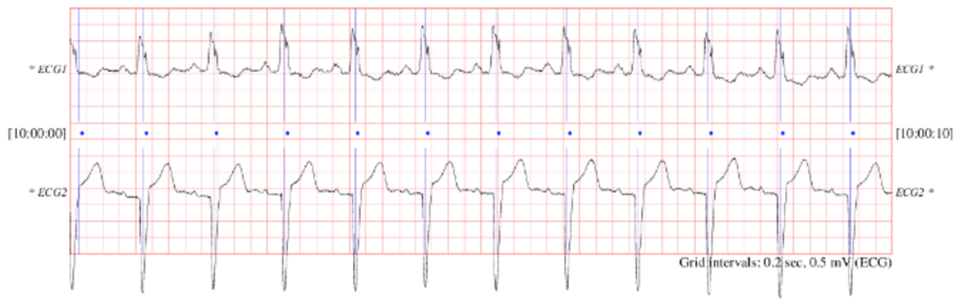


Figure 5.1: Waveform of heartbeats signal

5.1.2 ECG5000 Data waveform

ECG5000 dataset is ready dataset, in the other words, ECG5000 dataset can be used directly. ECG5000 dataset contains 5000 one-lead sequences and every sequence has 140 points(the sequence length is 140), which means that each time series data is 140-point sequence. To say it directively, the dimension of the signal is [5000, 1, 140] (number of observation, the number of lead of each observation, the number of points of each sequence) The signals in ECG5000 dataset is heartbeats signals, the ready heartbeats signal is shown in the figure 5.2. This waveform looks different from the figure 5.1 due to pre-processing.

5.1.3 Model Structure Definition

There are two autoencoders being employed on ECG5000 dataset, which are SAE and FC-AE.

Considering the drwaback of LSTM, which LSTM models have strong memory capabilities and may overfit on the training set, especially when the amount of data is small or there is a lot of noise, FC-AE is used as a supplement of SAE.

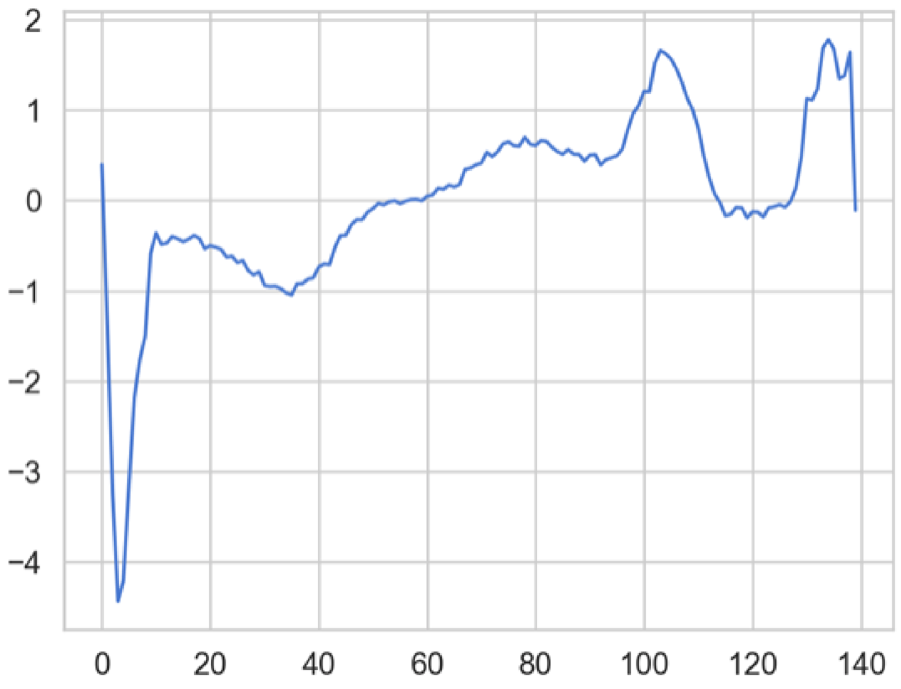


Figure 5.2: waveform of heartbeats signal

The objective for SAE is dimensional reduction and getting reconstructed error. In terms of FC-AE, the reconstructed error is the only output, as the criteria for anomaly detection.

Stacked Autoencoder (SAE)

Taking the consideration of the necessary of extracting temporal information, LSTM-based AE is widely used in anomaly detection field [30]. The thesis employed a configuration comprising four layers of LSTM, two in encoder and two in decoder, and one fully connected layer as the output layer for the “external” Autoencoder (AE) in the Stacked Autoencoder (SAE) framework. Additionally, for the sake of more efficient dimensional reduction, a fully connected AE (FC-AE) was utilized as the “internal” AE. During the training phase, the LSTM-based AE and FC-AE were trained independently. The “code” extracted from the “bottleneck” layer of the LSTM-based AE served as the training data for the FC-AE. The overall structure of SAE in the training phase is shown in the figure 5.3. The "external" AE make the main contribute to reconstruct data, and the "internal" AE mainly focus on the dimensional reduction. Thus, the reconstructed error is sensitive to ’external’ AE and the ’code’ is sensitive to the ’internal’ AE.

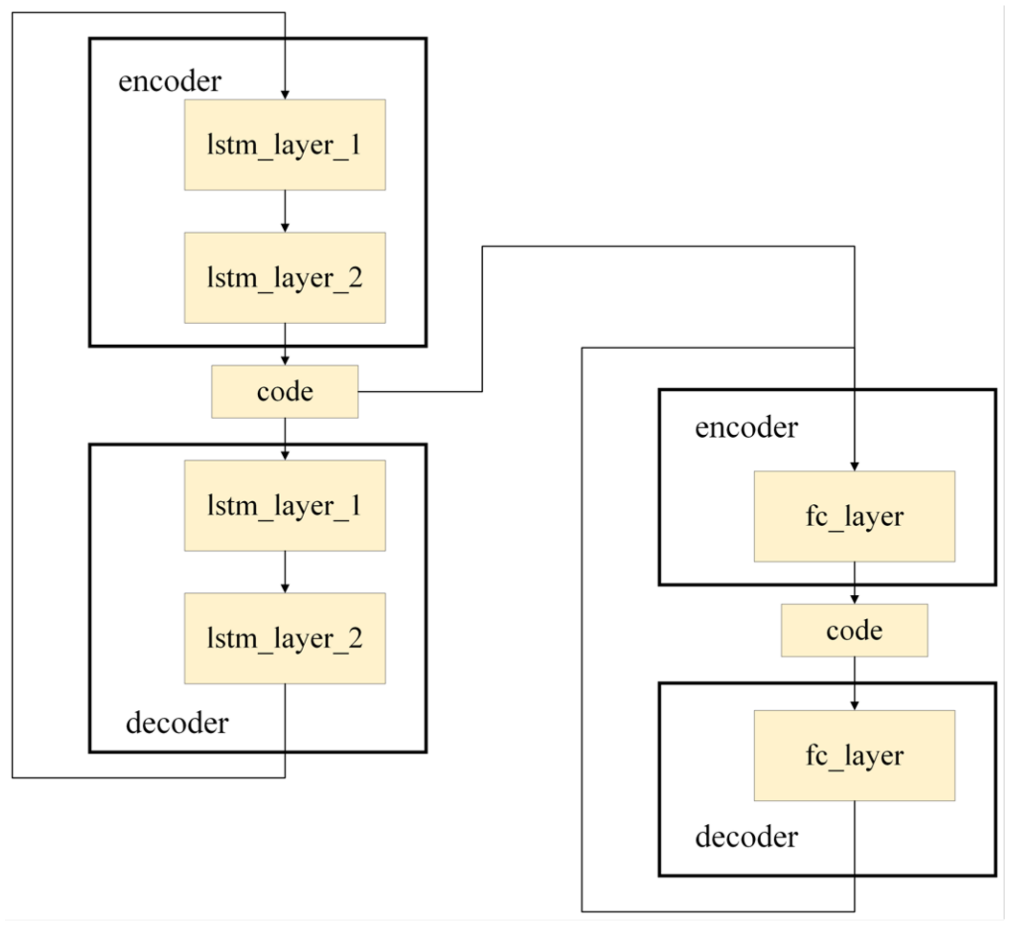


Figure 5.3: overall structure of SAE in training phase

In the testing or usage phase, the two Autoencoders (AEs) are stacked together to form the Stacked Autoencoder (SAE). The output from the 'bottleneck' layer of the LSTM-based AE is directly connected to the FC-AE as input. And the output of decoder of FC-AE works as the input of decoder of LSTM-based AE. The overall structure of SAE in the usage phase is shown in the figure 5.4.

5.1.4 Experiment on ECG5000 dataset

SAE Model Design

In LSTM based SAE, the "external" AE is LSTM-AE and "internal" AE is FC-AE.

The details of LSTM-AE is shown in the figure 5.5, and the details of FC-AE is shown in the figure 5.6. After stacked, the details of SAE is shown in the figure 5.7.

From the figure 5.5, the "Estimated Total Size" is the model size, which the size

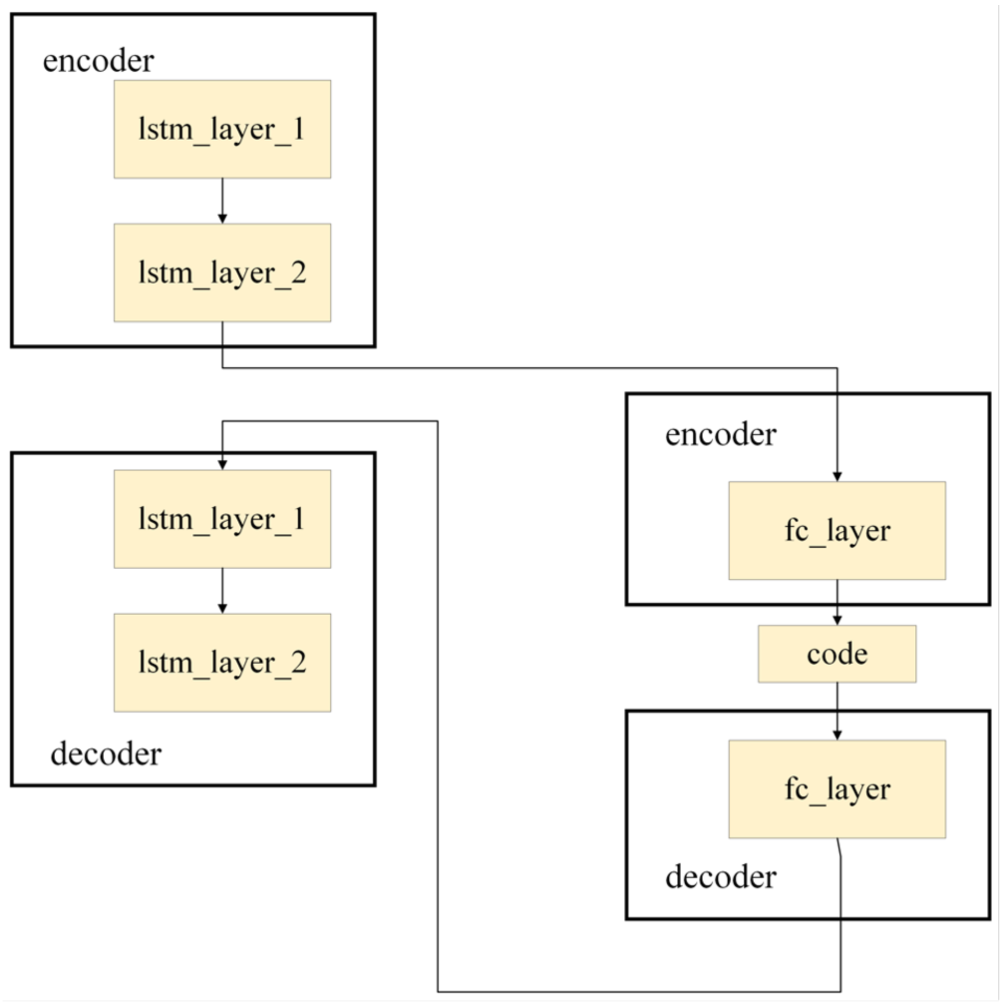
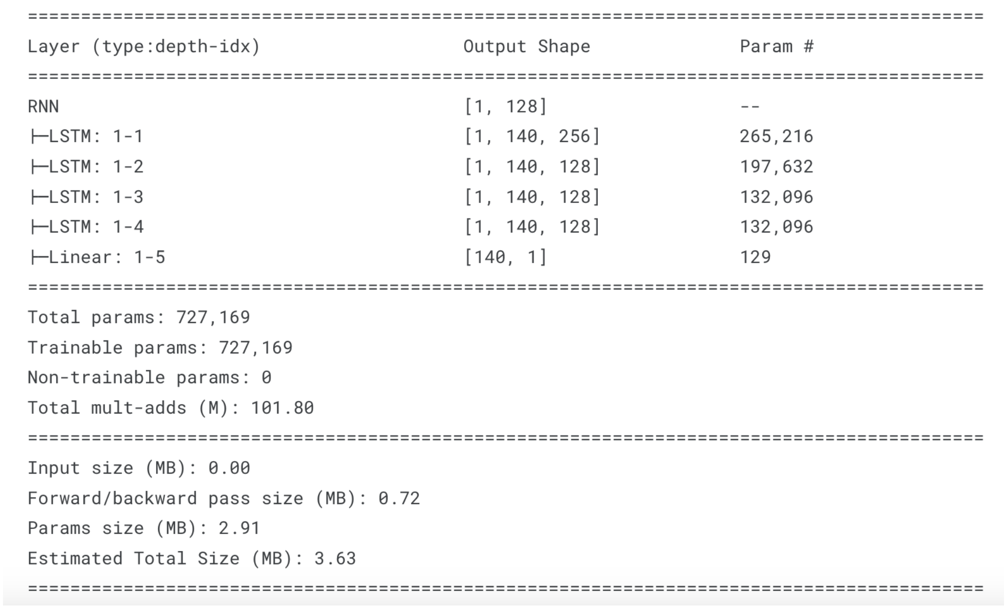


Figure 5.4: overall structure of SAE in usage phase

of memory occupied by the model when the model is running, and it is 3.63MB. The "Total params" and "Trainable params" are the total number of parameters and the number of parameters within training in the model. In this model "Total params" is equal to "Trainable params". They are 727,169.

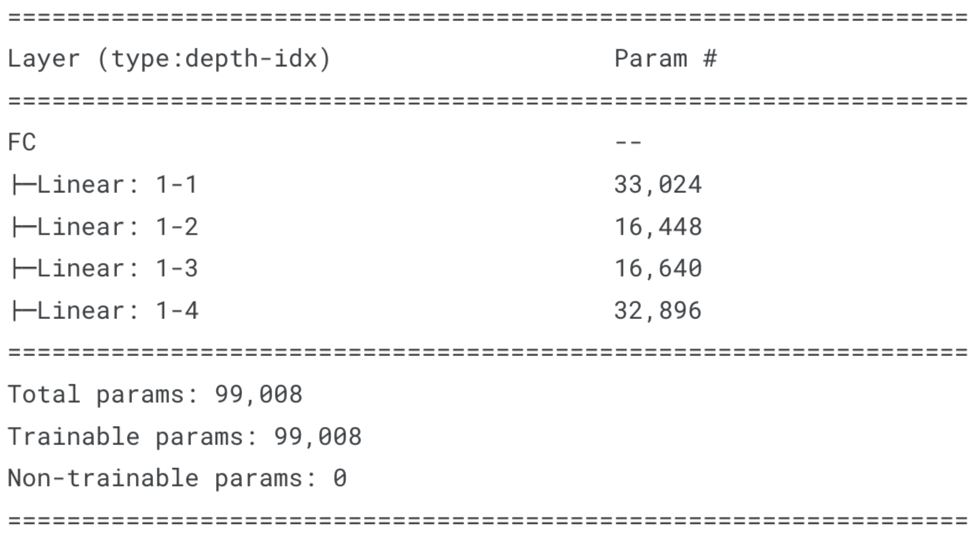
From the figure 5.6 and the figure 5.7, the "Estimated Total Size" could be figured out as 0.4 MB (Estimated Total Size of SAE - Estimated Total Size of LSTM-AE). And the "Total params" is 99,008. In terms of SAE, the "Estimated Total Size" is 4.03 MB and the "Total params" is 826,177 ("Total params" of LSTM-AE + "Total params" of "FC-AE").

By comparison between LSTM-AE and SAE, stacking makes the model size increased slightly for the sake of dimensional reduction.



```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
RNN                                      [1, 128]                  --
├─LSTM: 1-1                              [1, 140, 256]             265,216
├─LSTM: 1-2                              [1, 140, 128]             197,632
├─LSTM: 1-3                              [1, 140, 128]             132,096
├─LSTM: 1-4                              [1, 140, 128]             132,096
├─Linear: 1-5                            [140, 1]                  129
==========================================================================================
Total params: 727,169
Trainable params: 727,169
Non-trainable params: 0
Total mult-adds (M): 101.80
==========================================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.72
Params size (MB): 2.91
Estimated Total Size (MB): 3.63
==========================================================================================
```

Figure 5.5: The details of LSTM-AE



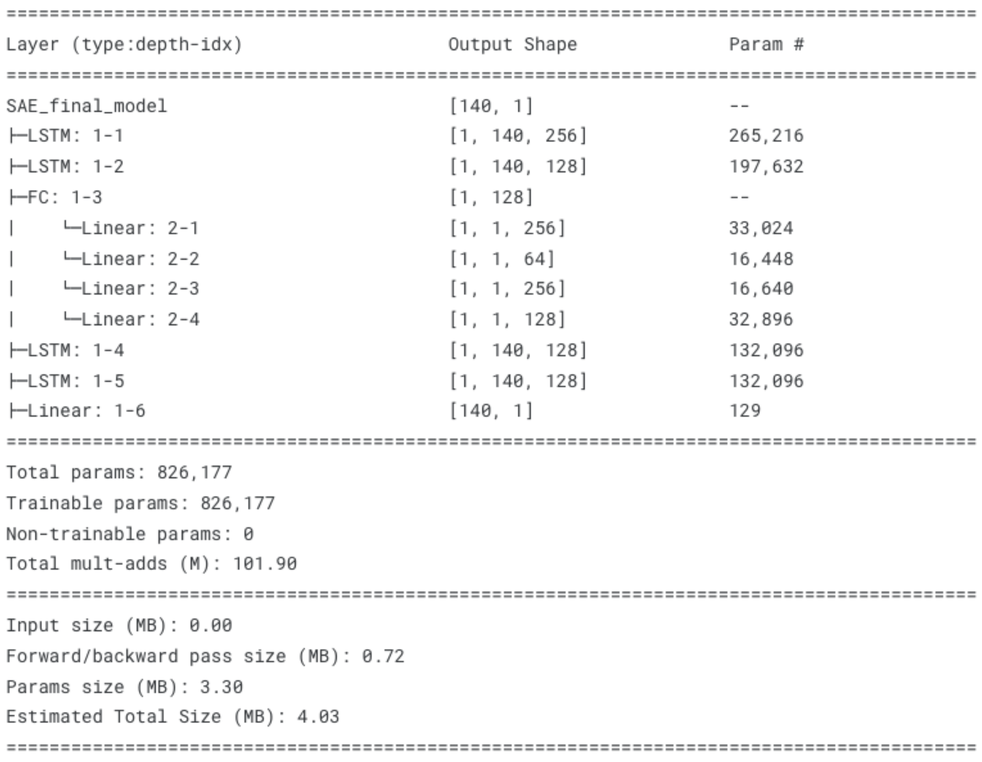
```
=================================================================
Layer (type:depth-idx)                   Param #
=================================================================
FC                                       --
├─Linear: 1-1                            33,024
├─Linear: 1-2                            16,448
├─Linear: 1-3                            16,640
├─Linear: 1-4                            32,896
=================================================================
Total params: 99,008
Trainable params: 99,008
Non-trainable params: 0
=================================================================
```

Figure 5.6: The details of FC-AE

Individual FC-AE Model Design

The details of FC-AE is shown in the figure 5.8. From the figure 5.8, the "Estimated Total Size" is 0.09, and the "Total params" is 21,008.

The advantage of FC-AE is light weight. In this thesis, introducing the individual



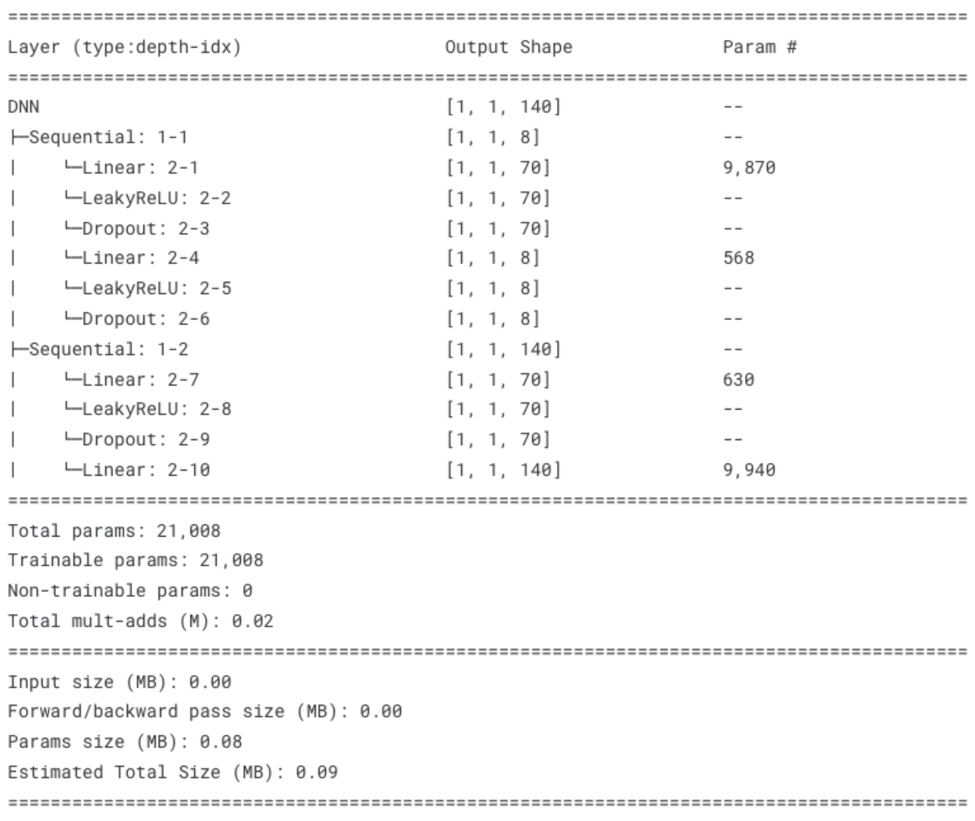
```
=======================================================================================
Layer (type:depth-idx)                    Output Shape              Param #
=======================================================================================
SAE_final_model                           [140, 1]                  --
├─LSTM: 1-1                               [1, 140, 256]             265,216
├─LSTM: 1-2                               [1, 140, 128]             197,632
├─FC: 1-3                                 [1, 128]                  --
|    └─Linear: 2-1                        [1, 1, 256]               33,024
|    └─Linear: 2-2                        [1, 1, 64]                16,448
|    └─Linear: 2-3                        [1, 1, 256]               16,640
|    └─Linear: 2-4                        [1, 1, 128]               32,896
├─LSTM: 1-4                               [1, 140, 128]             132,096
├─LSTM: 1-5                               [1, 140, 128]             132,096
├─Linear: 1-6                             [140, 1]                  129
=======================================================================================
Total params: 826,177
Trainable params: 826,177
Non-trainable params: 0
Total mult-adds (M): 101.90
=======================================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.72
Params size (MB): 3.30
Estimated Total Size (MB): 4.03
=======================================================================================
```

Figure 5.7: The details of SAE

FC-AE in usage phase does not introduce more significant running time, which makes that individual FC-AE could be used as a complementary tool for SAE. Even though, SAE has ability to learn the latent representation and reconstruct input well, SAE still hard to learning everything percisely.

Experimental Design

The training settings is in the table 5.1, where Absolute Difference error is also known as L1 loss (L1_loss): $l(x, \hat{x})=|x - \hat{x}|$. Mean squared error loss (MSE_loss): $l(x, \hat{x})=(x - \hat{x})^2$. Optimization functions are Adam [31] that considers the ‘inertia’ of gradient. With the help of gradient ‘inertia’, Adam could push neural networks (NNs) pass local optimization and saddle points, which makes the NN less sensitive to hyperparameters. And scheduler make the learning rate dynamic. For the scheduler of "ConsineAnnealingWarmRestarters", it make the leaning rate change as consine signal and it also help NNs pass local optimization. Thanks to ’Adam’ and ’learning rate scheduler’ make NNs less sensitive to hyperparameters. In terms of the "Batch Size", "1" means that the data could be fed to model one by one, which sacrifices efficiency but make it more accurate. And the "random seed" is for



```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
DNN                                      [1, 1, 140]               --
├─Sequential: 1-1                        [1, 1, 8]                 --
|    └─Linear: 2-1                       [1, 1, 70]                9,870
|    └─LeakyReLU: 2-2                    [1, 1, 70]                --
|    └─Dropout: 2-3                      [1, 1, 70]                --
|    └─Linear: 2-4                       [1, 1, 8]                 568
|    └─LeakyReLU: 2-5                    [1, 1, 8]                 --
|    └─Dropout: 2-6                      [1, 1, 8]                 --
├─Sequential: 1-2                        [1, 1, 140]               --
|    └─Linear: 2-7                       [1, 1, 70]                630
|    └─LeakyReLU: 2-8                    [1, 1, 70]                --
|    └─Dropout: 2-9                      [1, 1, 70]                --
|    └─Linear: 2-10                      [1, 1, 140]               9,940
==========================================================================================
Total params: 21,008
Trainable params: 21,008
Non-trainable params: 0
Total mult-adds (M): 0.02
==========================================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.08
Estimated Total Size (MB): 0.09
==========================================================================================
```

Figure 5.8: The details of individual FC-AE

reproduction. This is because, during trainning phase, the model could initialize the initial hyperparameters randomly. And during spliting the dataset, the total dataset could be splited into trining dataset, validation dataset and test dataset randomly. These two procedures could be controlled by define the same "random seed".

	LSTM-AE/FC-AE	FC-AE
Batch Size	1	1
Optimal function	Adam	Adam
Loss Function	Absolute difference(L1)/Mean Square Error (MSE)	Absolute difference(L1)
scheduler	ConsineAnnealingWarmRestarters/None	ConsineAnnealingWarmRestarters
Training epochs	200/50	200
Early stop	50/None	50
Learning Rate	0.001	0.001
Random Seed	19980425	19980425
Platform	Nvidia T4 GPU on Kaggle	Nvidia T4 GPU on Kaggle

Table 5.1: Training settings for ECG5000 for SAE and FC-AE.

Results Analysis

Firstly, the comparison between SAE and individual FC-AE is shown in the table 5.2.

	SAE	FC-AE	average	maximum	Damped average
TNR	0.8942	0.6173	0.8000	0.9317	0.8000
FNR	0.0040	0.0146	0.0076	0.0026	0.0076
TPR	0.9960	0.9854	0.9924	0.9974	0.9924
FPR	0.1058	0.3827	0.2000	0.0683	0.2000
Accuracy	0.9923	0.9720	0.9853	0.9950	0.9853
AUC	0.9970	0.9835	0.9943	0.9934	0.9943

Table 5.2: Comparison among SAE, individual FC-AE and ensemble.

The accuracy of SAE is 0.9923 that is better than 0.9720 of individual FC-AE. And in terms of TNR, SAE is even significant better than FC-AE. However, after ensemble, the performance still gets improvemts. For accuracy, it grows from 0.9923 to 0.9950, that is a slight improvement. And the TNR grows from 0.8942 to 0.9317, which could be meaningful improvement. Considering all performance matrix, after ensemble of maximum, the performance gets general improvements.

The redults reveal that even thought the overall performance of FC-AE is worse than the performance of SAE, individual FC-AE is still helpful for better performace after ensemble.

The results of SAE and individual FC-AE is shown in table 5.3, where LSTM-AE stands for the 'external' AE for SAE. From the "Mean of Loss", LSTM-AE creats the best result among LSTM-AE, SAE and FC-AE. SAE still have a "FC-AE" as "internal" AE, which tend to reconstruct the 'code's from the "bottleneck" layer of "external" AE, LSTM-AE, and it also induces the reconstructed error on the reconstructed "code". Reconstructed "code" with error make the decoder of "LSTM-AE" "understand" the "code" worse than pure LSTM-AE.

	LSTM-AE	SAE	FC-AE
Mean of Loss	9.36	10.62	20.89
Dimension of code	[1,128]	[1,64]	[1,8]

Table 5.3: Results for ECG5000 for SAE and FC-AE.

Reconstructed data from pure LSTM-AE and input data are show in the figure 5.9, where the blue curve is original input data, and the orange curve is reconstructed data. In the figure 5.10, where the blue curve is original input data, and the orange curve is reconstructed data from SAE. And in the figure 5.11, where the blue curve is original input data, and the orange curve is reconstructed data from LSTM-AE and the green curve is reconstructed data from SAE.

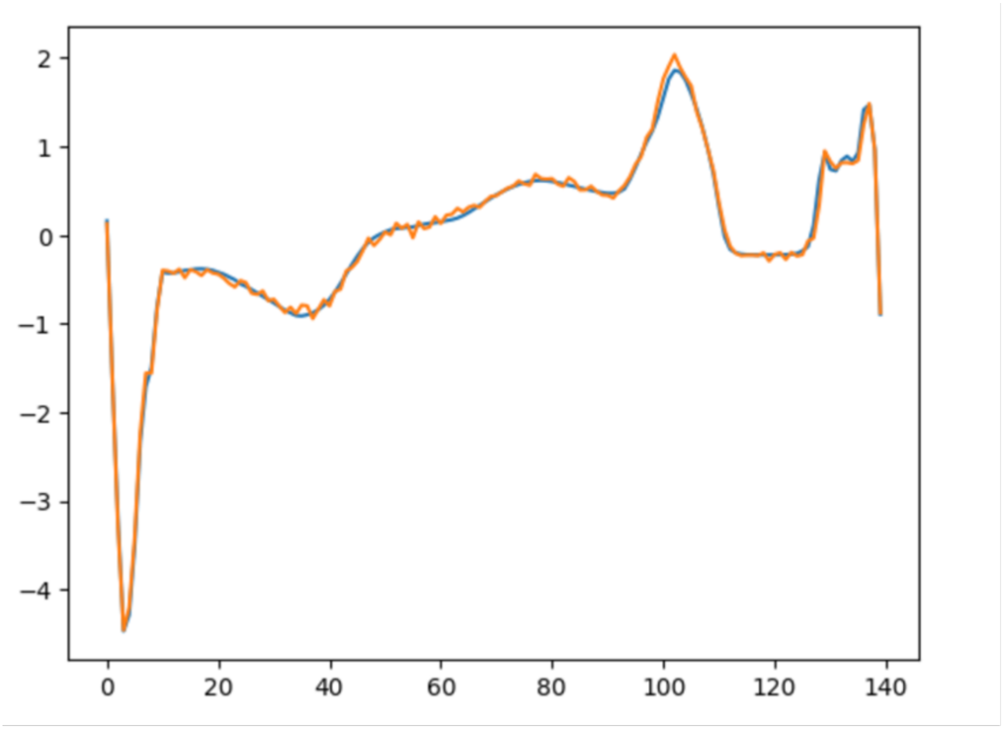


Figure 5.9: comparison between input data and reconstructed data from LSTM-AE

In order to check models performance on dimensional reduction, T-SNE is employed on the 'codes' from the 'bottleneck' layers of LSTM-SAE and FC-AE. The results are shown in the figures 5.12 and 5.13. In these two figures, the blue points are for anomaly and the red points stand for the normal data. According to the figures 5.12 and 5.13, the abnormal datapoints and normal datapoints separates generally, even though some special points lie in the wrong region.

Stacking another AE introduced slight larger reconstructed error than without stacking. And in terms of the results of T-SNE employed on the 'codes', SAE return the similar result as AE. For dimensional reduction, SAE do dimensional reduction from [1,140] to [1,64], in comparison with LSTM-AE [1,128], SAE is more efficient even though slight performance sacrifice.

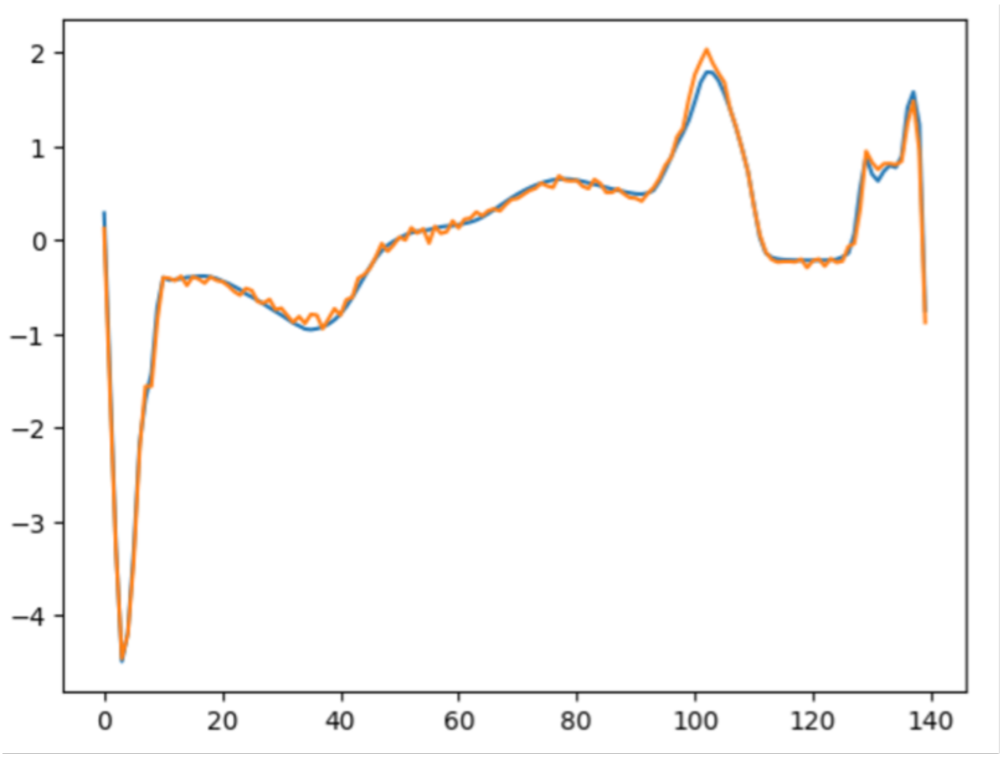


Figure 5.10: comparison between input data and reconstructed data from LSTM-SAE

Ensemble

The performance is evaluated with 2607 data points, in which there are 526 normal data points and 2081 abnormal data points. These abnormal data points are separated to 104 batch with 20 data points in one batch. And the 526 normal data points and every 20 abnormal data points are combined to feed to different models. The global performance matrix in the table 5.4. In general, all ensemble methods got improvement in comparison with any single method. And ensemble method of ‘maximum’ performs best in terms of all evaluation. In the table 5.4, where ’LSTM-AE’ stands for stacked autoencoder based LSTM. ’FC-AE’ stands for autoencoder based fully connected structure. ’Average’ stands for the combination method of average sum. ’Maximum’ stands for selecting the maximum value among four methods. ’Damped average’ stands for average sum on the square root of anomaly scores. ’AKPV’ stands for average sum of the top 3 anomaly detectors.

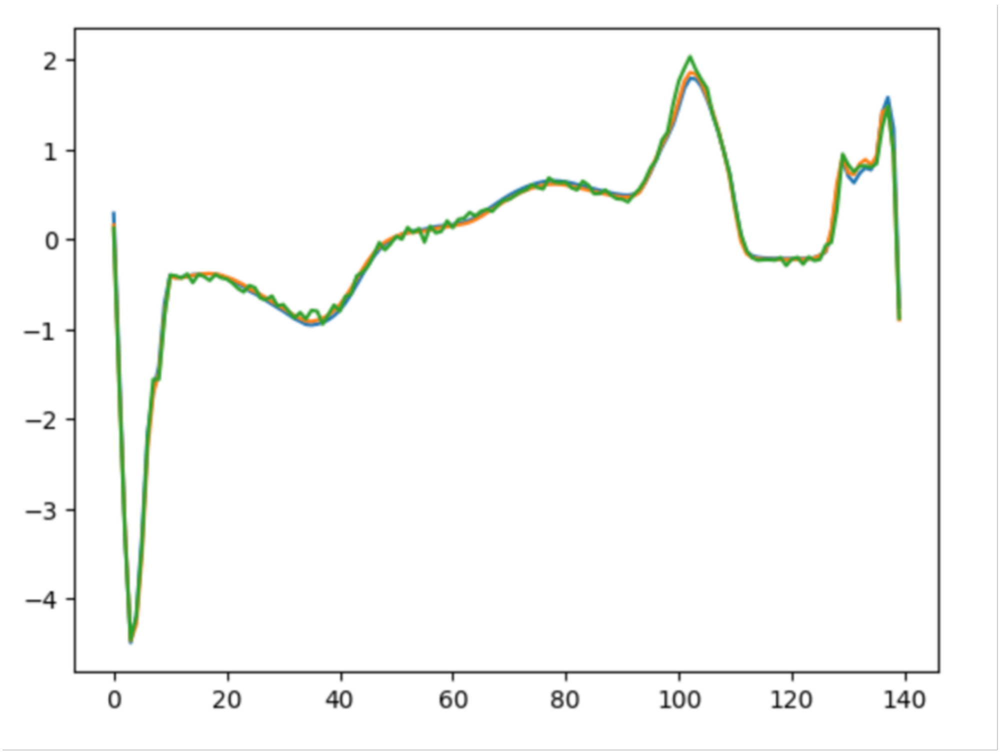


Figure 5.11: comparison between input data and reconstructed data from LSTM-SAE and reconstructed data from LSTM-AE

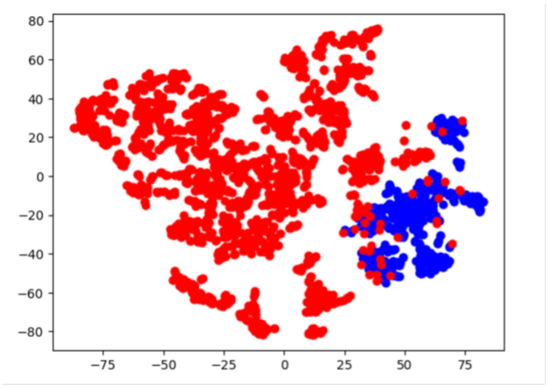


Figure 5.12: TSNE-SAE

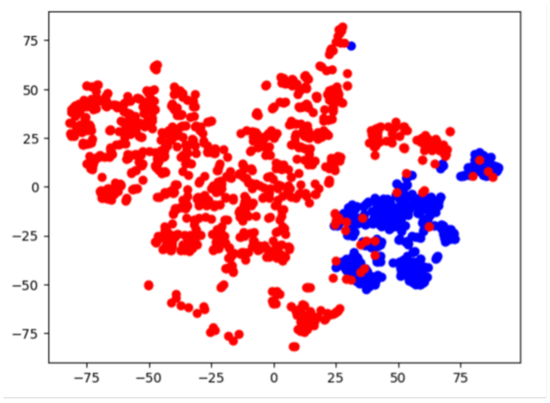


Figure 5.13: TSNE-AE

	SAE	AE	LOF	GMM	average	maximum	Damped average	AKPV
TNR	0.8942	0.6173	0.9505	0.9058	0.9457	0.9894	0.9457	0.9534
FNR	0.0040	0.0146	0.0019	0.0942	0.0021	0.0040	0.0021	0.0018
TPR	0.9960	0.9854	0.9981	0.9964	0.9979	0.9996	0.9979	0.9982
FPR	0.1058	0.3827	0.0495	0.0036	0.0543	0.0106	0.0543	0.9982
Accuracy	0.9923	0.9720	0.9964	0.9931	0.9960	0.9992	0.9960	0.9966
AUC	0.9970	0.9835	0.9989	0.9981	0.9985	0.9923	0.9985	0.9981

Table 5.4: Ensemble performance matrix on ECG5000

5.2 Experimental Design and Results Analysis on MSFR dataset

In terms of MSFR dataset, one data point contains four sequences. For this case study, we did three different experiments, which a bidirectional LSTM-based AE is fed by four sequences, and the code from bidirectional LSTM-based AE as input to LOF, GMM, IF. The second experiment is that there are four fully connected AEs for the four sequences and connected the 'codes' as one 'big' code fed to LOF, GMM, IF. The third case is that there are four fully connected AEs and four LOFs and four GMMs and four IFs for the four sequences. The reconstructed error serves as the criterion for determining the normalcy or anomaly of each observation. Additionally, the code extracted from the bottleneck layer is employed for subsequent algorithms.

5.2.1 Description of MSFR dataset

The second application is obtained from a MSFR simulator, developed by [32]. It has been developed as a useful tool for analyzing the reactor parameters and testing the control strategies for the EU SAMOFAR project. The simulator was designed with the Modelica language, an object-oriented, declarative, multi-domain modeling language for component-oriented modeling of complex engineering systems. Since its acasual modeling characteristics, the language is suitable for building a flexible and reusable scheme for a complex system. The thermal-hydraulic aspect has been carried out using the ThermoPower library. Plant simulators have already been developed for the ALFRED (Advanced Lead Fast Reactor European Demonstrator) project and for the IRIS reactor, adopting the Modelica language. Dymola (Dynamic Modelling Laboratory) is the simulating software in which the current simulator has been built. It is equipped with state-of-the-art implicit numerical integration algorithms, as DASSL [33].

Since a MSFR has the salt mixture that acts as fuel and coolant at the same time, a strong coupling between the thermo-dynamic variables and the neutronic parameters is present. So, it is necessary to take into account the motion of the delayed neutron precursors: a one-dimensional approximation has been applied, that gives good results for the neutronic modeling and it is computationally efficient. A point kinetic modelling that is able to deal with the motion of precursors has been used, based on 0D-1D approach. A complementary 2D MATLAB model has been built to compare the results of the simulator for what concern the neutronic part.

The scheme of the MSFR model is shown in the following figure 5.14, from [32].

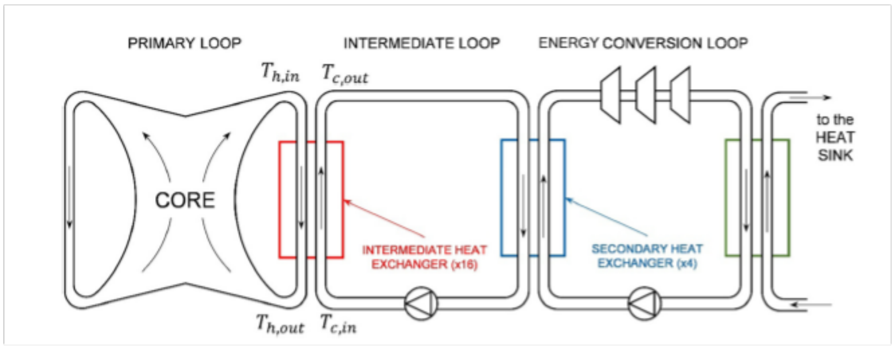


Figure 5.14: MSFR scheme

The plant presents three different loops: in the primary and intermediate loops, the fluoride based molten salt extracts thermal power from the core. The third one is a classic Brayton -Joule cycle. The reactor generates 3000 MW_{th}, the fuel salt has an average temperature of 700 ^oC, entering in the core at 650 ^oC and exiting at 750 ^oC. The total salt volume is 18 m^3.

In the present work, the plant behavior has been simulated, with the "injection" of faults, randomly sampled. It has been simulated a pump failure, varying the mass flowrate from 0% to 90% with respect to the nominal value. The FDD detection has been carried out 50 times, with 501 normal transients and one anomaly for each iteration. Four different temperatures have been studied: the maximum and minimum temperature in the intermediate loop and the maximum and minimum one in the fuel circuit.

The molten salt presents two limits. The upper one is related to the maximum temperature that can be tolerated by the structural material and it is set to 1373 K (or 1100 ^oC). The minimum allowed temperature is set at 858 K, when the fluid freezes.

The simulator has performed 1000 runs, with 499 faults and 501 normal successful transients. The most common failure criterion is the physical one, when a temperature exceeds the limits, with 280 failures. Numerical failures happen 219 times during the simulations.

n figure 5.15, from [34], the various failure mode types are presented, including the numerical ones: the most common one regards the freezing of the fluid in the intermediate circuit.

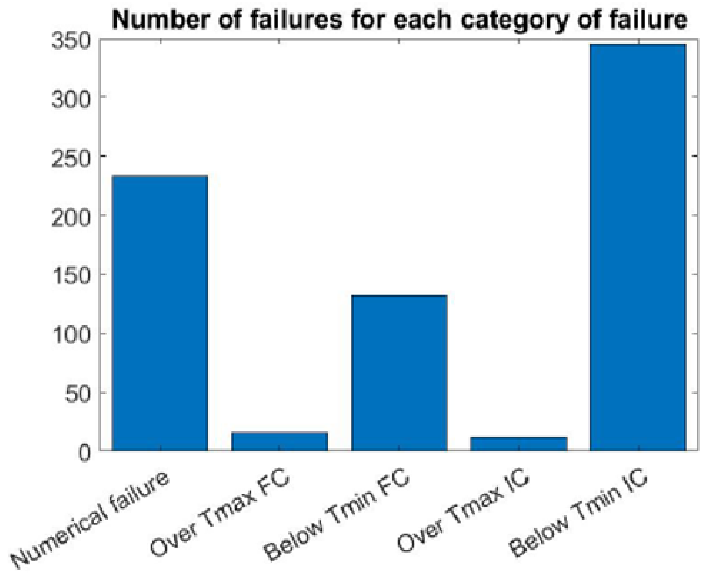


Figure 5.15: Most common failure modes

In the present work, the implementation regards the minimum temperature in the intermediate circuit, which is the one with the highest number of failures. In figure 5.16, a transient with the representation of a numerical failure is presented. In figure 5.17, a normal behavior and a physical failure is presented.

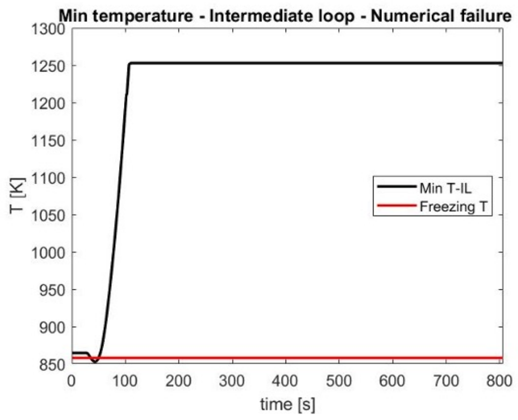


Figure 5.16: T IC min, numerical failure

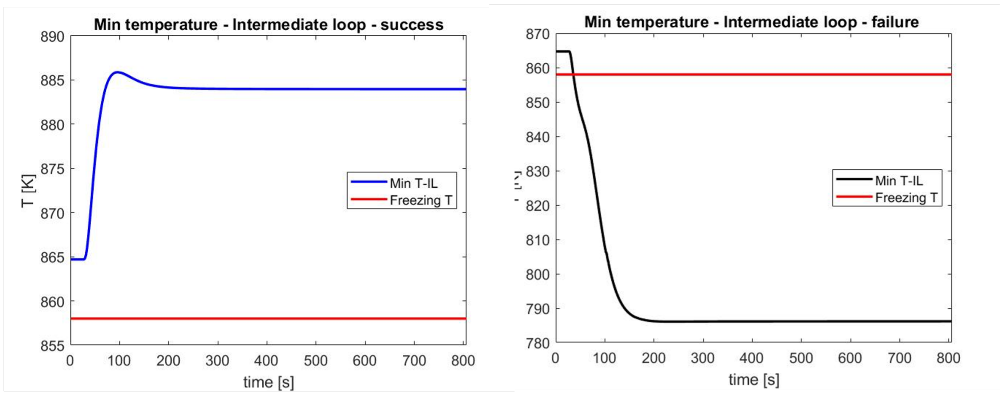


Figure 5.17: T IC min, success and physical failure

5.2.2 The first case (BiLSTM-AE)

The bidirectional LSTM augments standard LSTMs to improve the model's performance on feature extraction on long time-dependencies. It trains two LSTMs on the input data. The first LSTM on the original input data and the other on a reversed replica of the input data. The training of BiLSTM is on all past and future input information available within a particular time frame. Intuitively, BiLSTM process input data in two directions (thus, from left-to-right and right-to-left) using a forward hidden layer and a backward hidden layer [35]. By combining the outputs of these two layers, BiLSTMs can utilize information from both the past and the future of a given point in the sequence, providing a richer and more comprehensive understanding of the data.

The general architecture of BiLSTM is shown in figure 5.18. Image source is [36]. There are two LSTM blocks, one of blocks get signal from the beginning to the end of signals and another block get signal from the end to the beginning of the signals.

BiLSTM-based AE model design

There is a one-dimensional Convolutional layer for extracting feature across four sequences in encoder and one transposed one-dimensional Convolutional layer to convert the dimension as same as input data in decoder. For time dependency feature, two bidirectional Long Short Term Memory layers are employed in encoder and two bidirectional Long Short Term Memory layers are employed in decoder. The general structure of model is shown in the figure 5.19. The 'code' (output from bottleneck layer) is fed to LOF and GMM and IF as input.

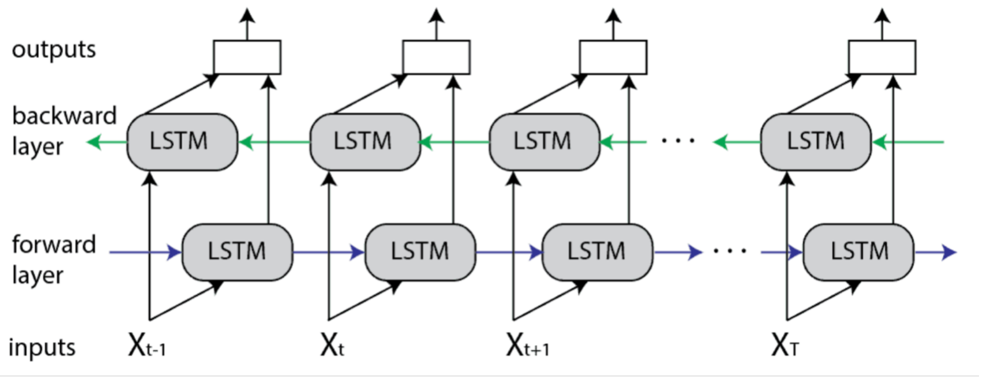


Figure 5.18: Bidirectional LSTM architecture

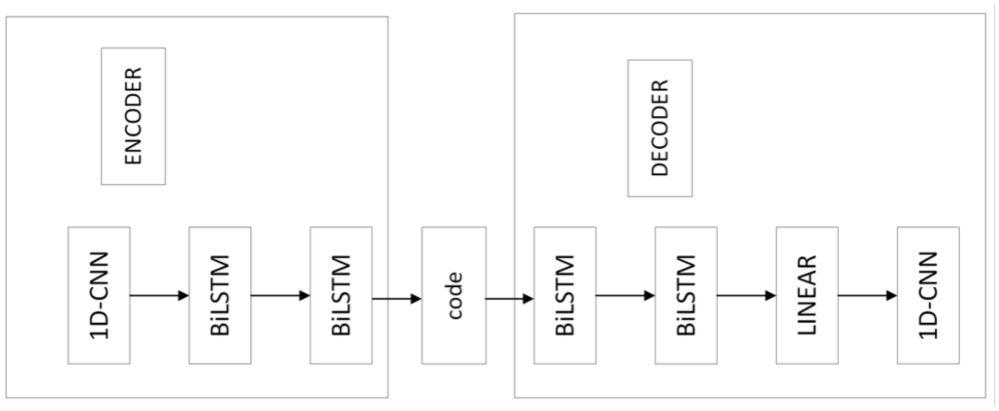


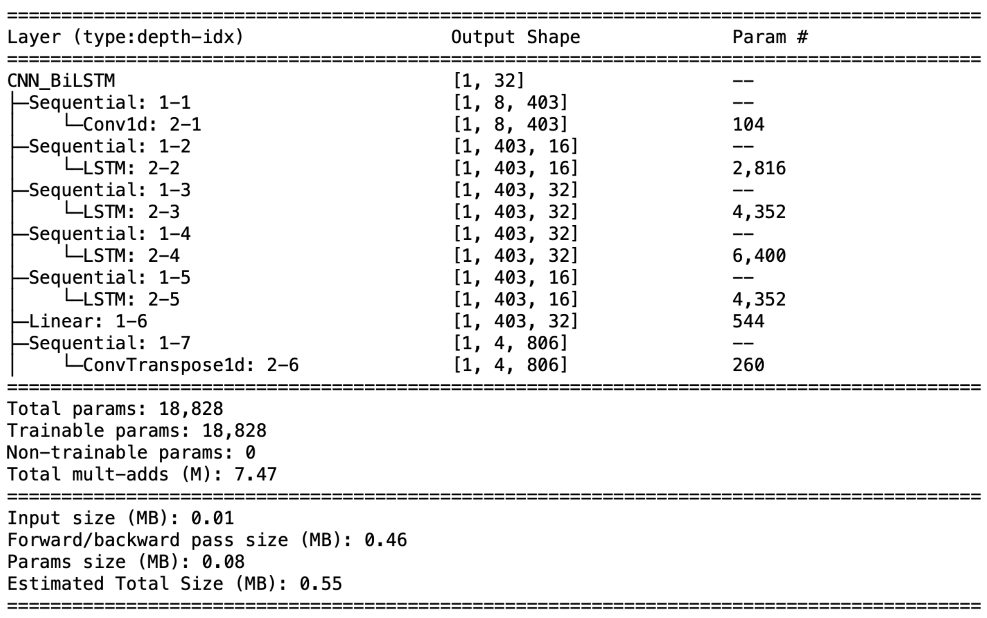
Figure 5.19: general structure of model

The details of BiLSTM-AE

The details of BiLSTM-AE is shown in the fugure 5.20. In the figure 5.20, where, the "Total params" is 18.828 and the "Estimated Total Size" is 0.55 MB, could be considered as small, and the "Total mult-adds" is 7,47 millions times, that means that one forward processing contains 7.47 millions operations. This is the reason that it needs long time to train and inference.

Experimental Design

Only one BiLSTM-AE is trained not only for anomaly detection but also for dimensional reduction, and the detail settings is shown in the table 5.5.



```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
CNN_BiLSTM                               [1, 32]                   --
├─Sequential: 1-1                        [1, 8, 403]               --
│    └─Conv1d: 2-1                       [1, 8, 403]               104
├─Sequential: 1-2                        [1, 403, 16]              --
│    └─LSTM: 2-2                         [1, 403, 16]              2,816
├─Sequential: 1-3                        [1, 403, 32]              --
│    └─LSTM: 2-3                         [1, 403, 32]              4,352
├─Sequential: 1-4                        [1, 403, 32]              --
│    └─LSTM: 2-4                         [1, 403, 32]              6,400
├─Sequential: 1-5                        [1, 403, 16]              --
│    └─LSTM: 2-5                         [1, 403, 16]              4,352
├─Linear: 1-6                            [1, 403, 32]              544
├─Sequential: 1-7                        [1, 4, 806]               --
│    └─ConvTranspose1d: 2-6              [1, 4, 806]               260
==========================================================================================
Total params: 18,828
Trainable params: 18,828
Non-trainable params: 0
Total mult-adds (M): 7.47
==========================================================================================
Input size (MB): 0.01
Forward/backward pass size (MB): 0.46
Params size (MB): 0.08
Estimated Total Size (MB): 0.55
==========================================================================================
```

Figure 5.20: The details of BiLSTM-AE

	BiLSTM_AE
Batch Size	32
Optimal Function	Adam
Loss Function	Mean Square Error (MSE)
Schedular	ConsineAnnealingWarmRestarters
Training opochs	1000
Early Stop	300
Learning Rate	0.01

Table 5.5: Experimental design for BiLSTM-AE on MSFR dataset

Results Analysis

With the help of BiLSTM-AE, the dimension of 'code' is [1,32].

Sample reconstructed data and input data are shows in the figures 5.21, 5.22, 5.23and 5.24, where the yellow curves are original signal and the blue curves are the reconstructed signals. And the reconstructed data for anomaly is shown in figure 5.25, where the yellow curve are original signal and the blue curves are the reconstructed signal.

According to the figures 5.21, 5.22, 5.23 and 5.24 the model can reconstruct

	BiLSTM_AE
Mean of Loss	1.90
Dimension of code	[1,32]

Table 5.6: Results Analysis

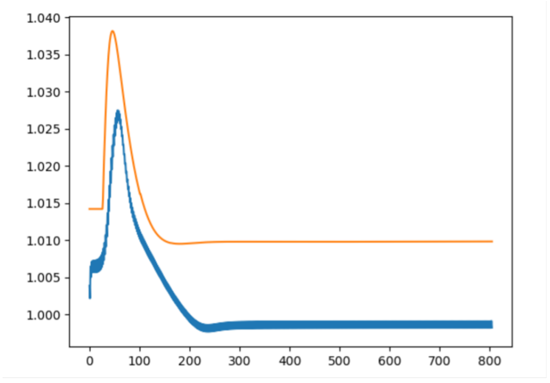


Figure 5.21: T-FC-MAX-success

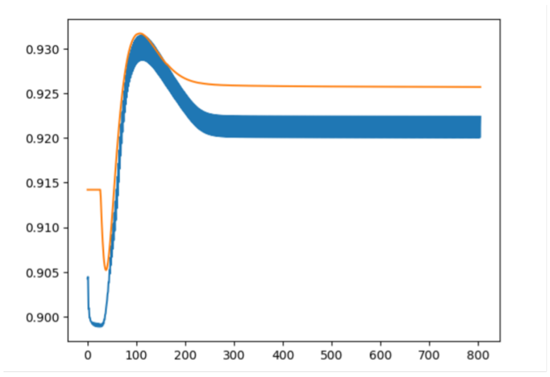


Figure 5.22: T-FC-MIN-success

normal data with small reconstructed errors and for the abnormal data, in the figure 5.25, the reconstructed error could be larger.

Ensemble

For evaluating, the same method as ‘ECG5000’ dataset is used. There are 51 normal data and 20 abnormal data in one test set and rangking them with ‘anomaly scores’ from large values to small. And then the number of anomaly data in the first 20 position and the number of normal data in the last 51 positions are used for calculating the TNR, FNR, TPR and FPR and accuracy. And then replace the 20 anomaly data with another 20 anomaly data until run out anomaly data. And then calculating the mean of such performance index.

The details of performance is shown in the table 5.7. According to the table 5.7, among all the individual methods, the dominating detector is GMM, with the help of low dimensinal ’code’ from BiLSTM-AE. After ensemble, the dominating one is ’Maximum’ with scarifying AUC. For the reason that ’Maximum’ gets the best performance, the four individual methods (BiLSTM-AE, GMM, LOF, IF) trend to give all data point the less anomaly scores than the ground truth (even though the groung truth is unknown) and ’maximum’ trend to give the larger value than the individual give. This could be found through that, among the four individual detectors, the TNRs are truly lower than the TPRs, which means in general the

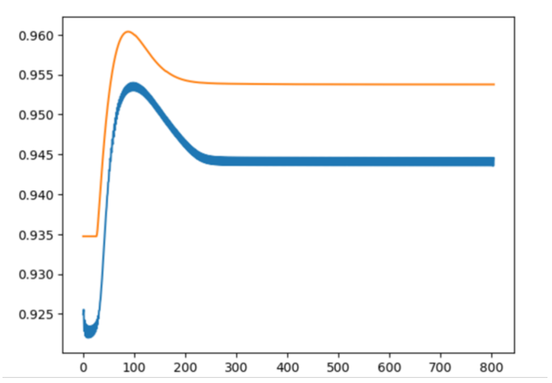


Figure 5.23: T-IC-MAX-success

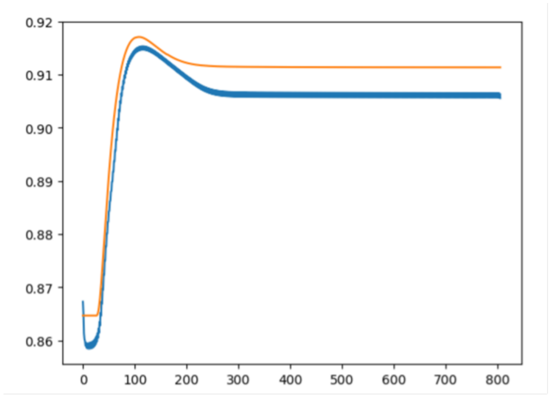


Figure 5.24: T-IC-MIN-success

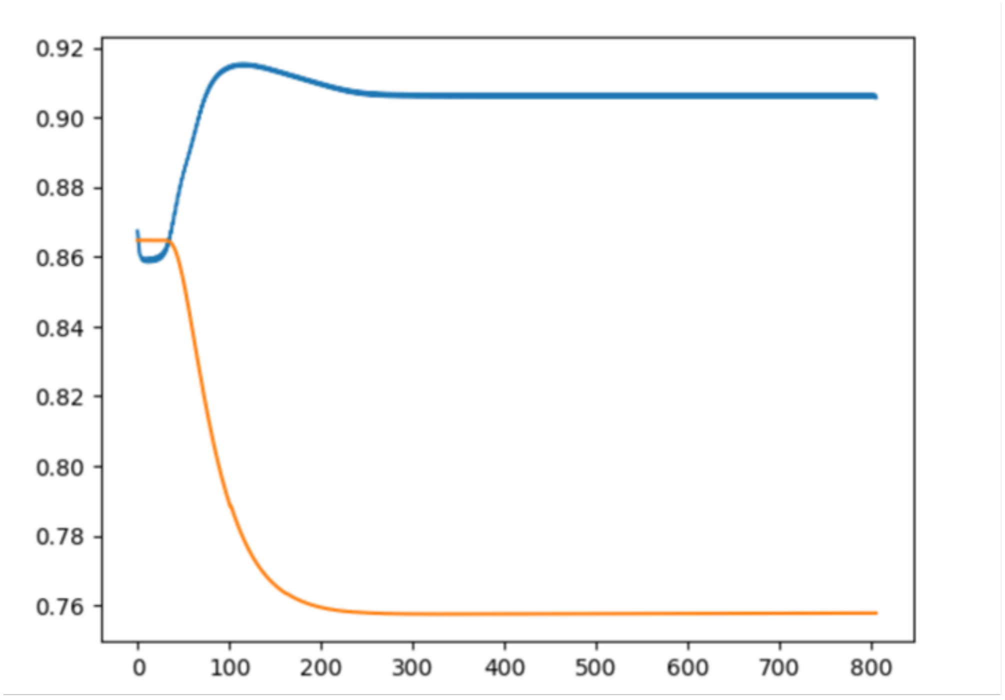


Figure 5.25: Failure Signal

anomal score for each observation is lower the ground truth. And for the scrifying of AUC, this is because selecting the maximum anomaly scores as the final anomaly scores could make all anomaly scores increase. In this case, the normal data are forced to have the large anomaly scores.

	BiLSTM_AE	GMM	LOF	IF	Maximum	Average	AKPV	Damped Average
TNR	0.8792	0.9208	0.8312	0.8250	0.9437	0.9208	0.9292	0.9208
FNR	0.0474	0.0310	0.0662	0.0686	0.0221	0.0310	0.0278	0.0310
TPR	0.9526	0.9690	0.9338	0.9314	0.9779	0.9690	0.9722	0.9690
FPR	0.1208	0.0792	0.1687	0.1750	0.0563	0.0792	0.0708	0.0792
Accuracy	0.9319	0.9554	0.9049	0.9014	0.9683	0.9554	0.9601	0.9554
AUC	0.9557	0.9791	0.9483	0.9589	0.9003	0.9862	0.9861	0.9856

Table 5.7: Ensemble performace of first case

5.2.3 The second case (four FC-AEs and one LOF, IF and GMM)

This case is for the case that 'four fully connected AEs, one GMM, one LOF, one IF'. In this case, the dominating method among individual method is 'AE'. After combination of (AEs, GMM, LOF, IF) the best situation happened on 'maximum' and 'AEs', which means ensemble did not get improvement.

The first step for this case is spliting the four-sequence-datapoint to four seperate sequences. And then feeding these four sequences to Four FC-AEs. This step is shown in the figure 5.26

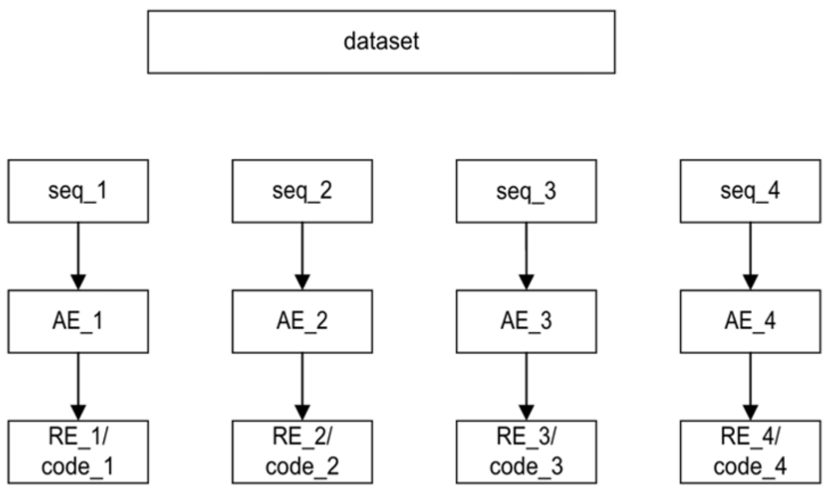


Figure 5.26: the first step of the second case

And the second step is that calculating the ecdf values for the four reconstruted errors and selected the maximum one among the four ecdf values. This step is shown in the figure 5.27

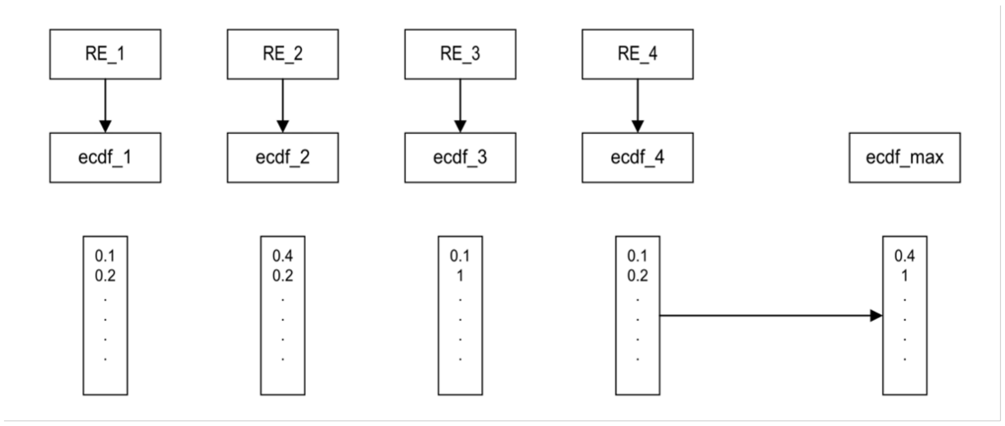


Figure 5.27: the second step of the second case

And the third step is that connecting the four 'codes' as a 'large' vector and feeding it to LOF, IF and GMM separately. And then calculating the ecdf values of the anomaly scores from these three models. This step is shown in the figure 5.28

And the forth step is that selecting the maximum one among these four ecdf values (for FC-AEs, LOF, IF and GMM). This step is shown in the figure 5.29

Model structure design

There are two similiar FC-AEs, one for the first sequence of T-FC-MAX shown in the figure 5.30 , another is for the other sequences of T-FC-MIN, T-IC-MAX and T-IC-MIN shown in the figure 5.31.

The details of the models

The details of the models are shown in the figure 5.32 and the figure 5.33. In comparison with the information of BiLSTM-AE, these two model could be considered as "large" model for the "Total params" is 441,169, and the "Estimated Total Size" is 1.78 MB. However, training these two models is faster than BiLSTM-AE. This is because, the "Totla mult-adds" is 0.44 M times, which is significant less that the BiLSTM-AE.

Experimental Design

The details of Experimental Design is shown in the table 5.8.

Results Analysis

The results from the four FC-AEs is shown in the table 5.9

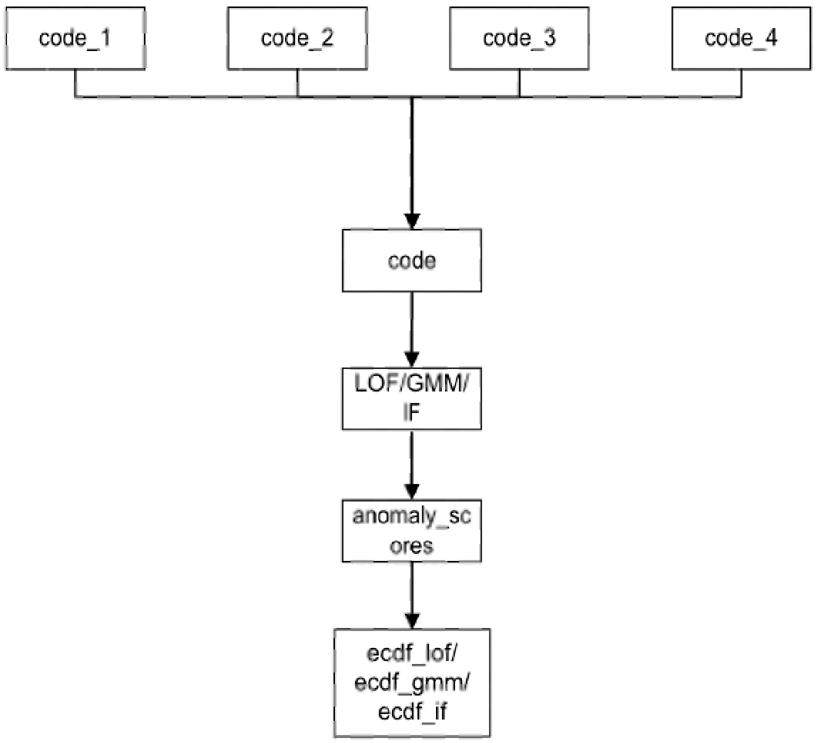


Figure 5.28: the third step of the second case

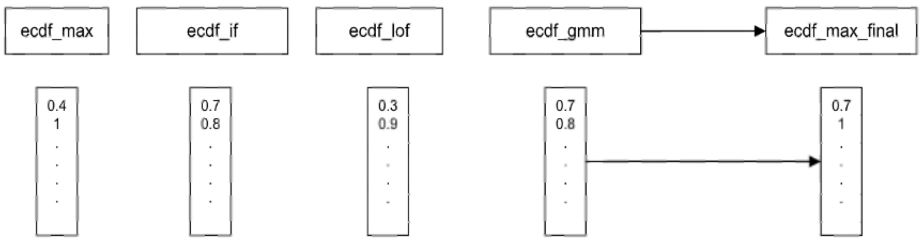


Figure 5.29: the forth step of the second case

Sample reconstructed data and input data are shows in the figures 5.34, 5.35, 5.36and 5.37, where the yellow curves are original signal and the blue curves are the reconstructed signals.

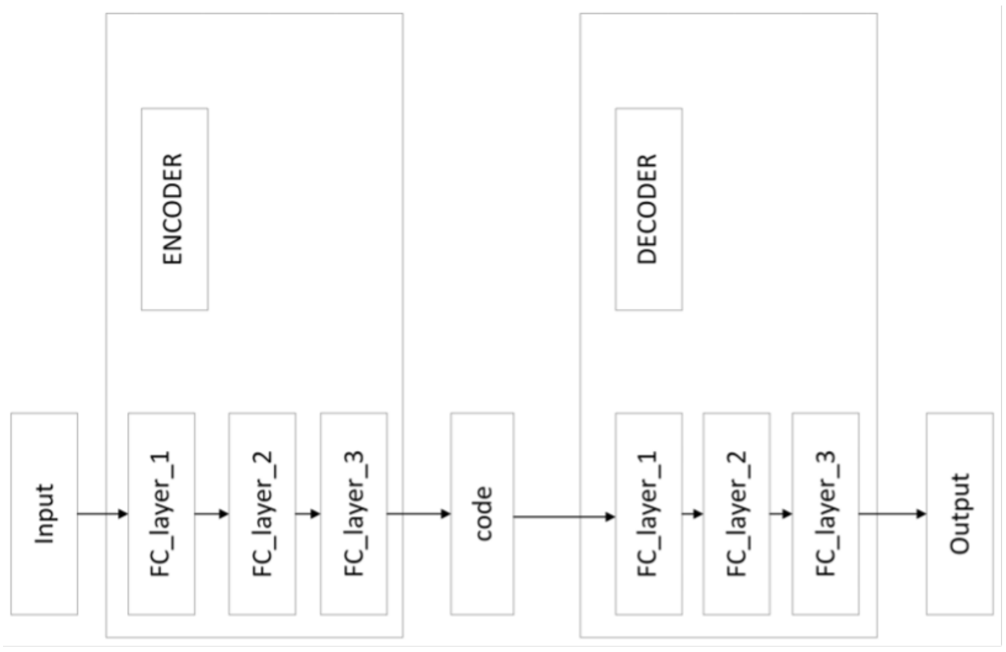


Figure 5.30: model structure for sequence of T-FC-MAX

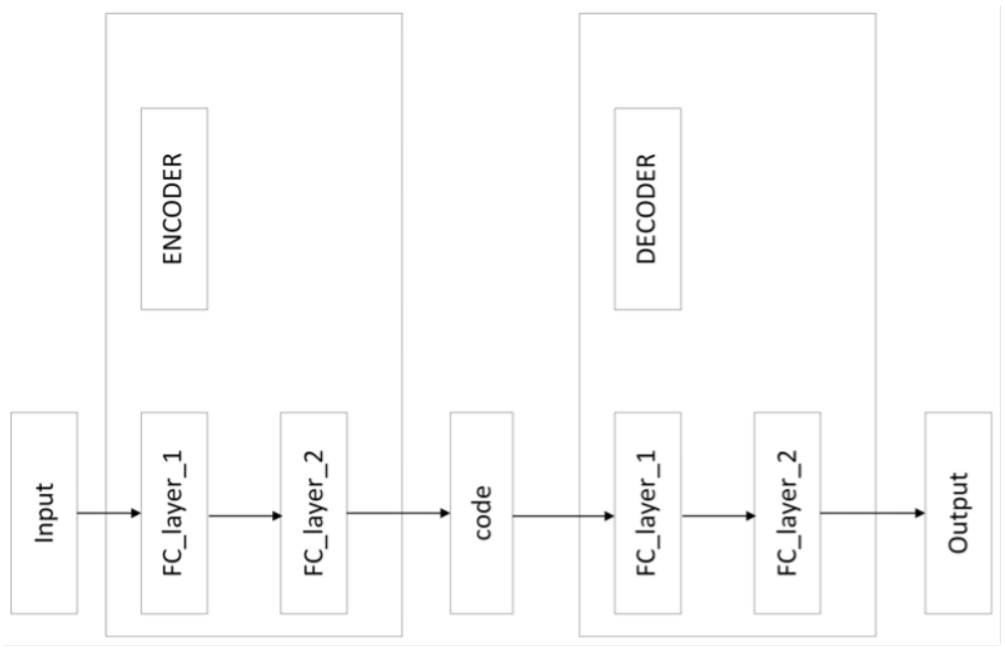
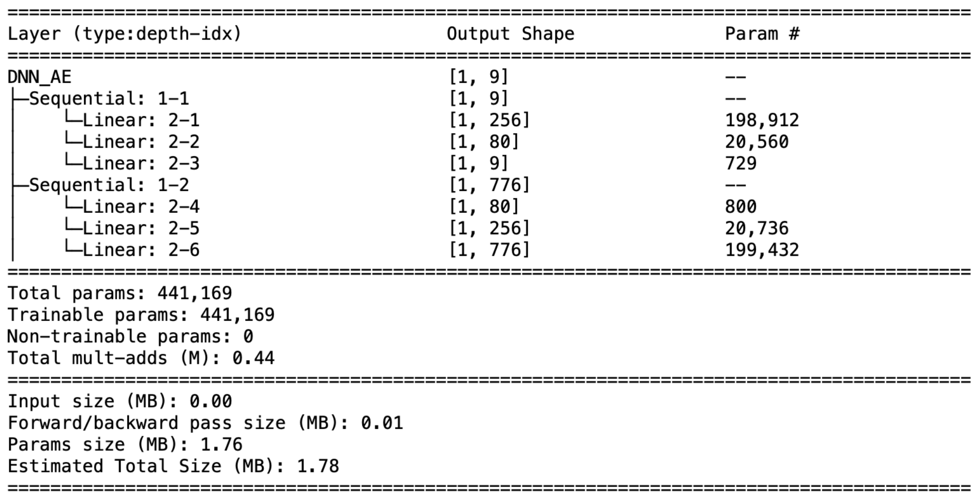
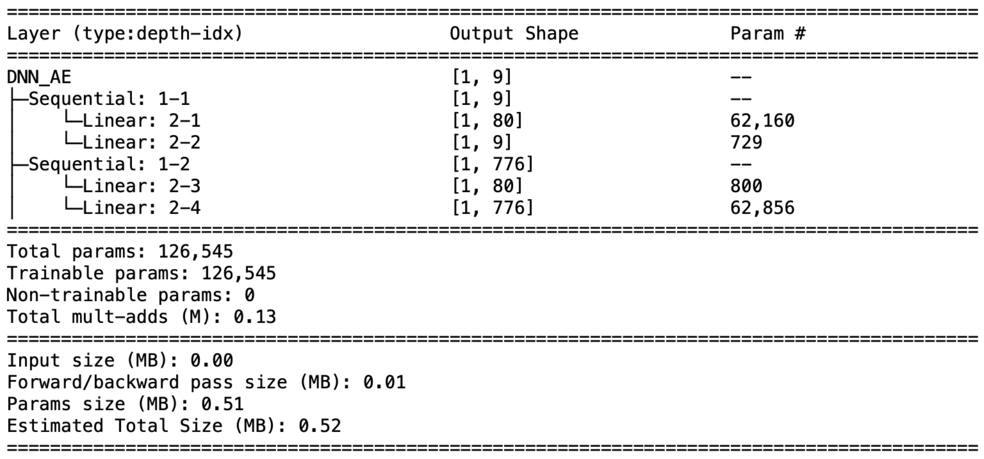


Figure 5.31: model structure for sequence of T-FC-MIN, T-IC-MAX and T-IC-MIN



```
===========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
===========================================================================================
DNN_AE                                   [1, 9]                    --
├─Sequential: 1-1                        [1, 9]                    --
│    └─Linear: 2-1                       [1, 256]                  198,912
│    └─Linear: 2-2                       [1, 80]                   20,560
│    └─Linear: 2-3                       [1, 9]                    729
├─Sequential: 1-2                        [1, 776]                  --
│    └─Linear: 2-4                       [1, 80]                   800
│    └─Linear: 2-5                       [1, 256]                  20,736
│    └─Linear: 2-6                       [1, 776]                  199,432
===========================================================================================
Total params: 441,169
Trainable params: 441,169
Non-trainable params: 0
Total mult-adds (M): 0.44
===========================================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.01
Params size (MB): 1.76
Estimated Total Size (MB): 1.78
===========================================================================================
```

Figure 5.32: The details of the model for the sequence of T-FC-MAX



```
===========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
===========================================================================================
DNN_AE                                   [1, 9]                    --
├─Sequential: 1-1                        [1, 9]                    --
│    └─Linear: 2-1                       [1, 80]                   62,160
│    └─Linear: 2-2                       [1, 9]                    729
├─Sequential: 1-2                        [1, 776]                  --
│    └─Linear: 2-3                       [1, 80]                   800
│    └─Linear: 2-4                       [1, 776]                  62,856
===========================================================================================
Total params: 126,545
Trainable params: 126,545
Non-trainable params: 0
Total mult-adds (M): 0.13
===========================================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.01
Params size (MB): 0.51
Estimated Total Size (MB): 0.52
===========================================================================================
```

Figure 5.33: The details of the model for the sequences of T-FC-MIN, T-IC-MAX and T-IC-MIN

Ensemble

For evaluating, the same method is employed. There are 51 normal data and 20 abnormal data in one test set and rangking them with 'anomaly scores' from large values to small. And then the number of anomaly data in the first 20 position and the number of normal data in the last 51 positions are used for calculating the TNR, FNR, TPR and FPR and accuracy. And then replace the 20 anomaly data with another 20 anomaly data until run out anomaly data. The detail is shown in the table 5.10. According to the table 5.10, the dominating method among all individual

	FC-AE (sequnence 1)	FC-AEs (sequences 2,3,4)
Batch Size	32	32
Optimal Function	Adam	Adam
Loss Function	Mean Square Error (MSE)	Mean Square Error (MSE)
Schedular	ConsineAnnealingWarmRestarters	None
Training epochs	1000	1000
Early Stop	None	300
Learning Rate	0.001	0.001
weight decay	0.1	0.1
platform	M1 CPU	M1 CPU

Table 5.8: Experimental design for FC-AEs on MSFR dataset

	FC-AE (seq 1)	FC-AE (seq 2)	FC-AE (seq 3)	FC-AE (seq 4)
Mean of Loss	38.42	16.64	10.11	66.60
Dimension of code	[1,9]	[1,9]	[1,9]	[1,9]

Table 5.9: Results on FC-AEs on MSFR dataset

detetors is AEs (four AE for four sequences). After ensemble, no improvements are got. This is because, the fully-connected AE did not extract features very well. Thus the 'code' after connecting makes other detetors confused because the features of four sequences are extracted separately with four independent AEs. This results in GMM, LOF, IF connot get good results.

5.2.4 The third case (four FC-AEs and four LOFs, IFs and GMMs)

The third case is for the case that 'four FC-AEs, four GMMs, four LOFs, four IFs'. That is that the four 'code's from four AEs go to the four GMM, LOFs, IFs. In this case, the dominating method among individual method is 'GMM'. After combination of (AEs, GMMs, LOFs, IFs) the best situation happened on 'maximum'.

The first step and the second step are as same as the first two steps of the second case. And the third step is that there are four 'codes' from four FC-AEs, and feeding them to four LOFs, IFs and GMMs for detecting the anomaly from four sequences separately. And then calculating the ecdf values and selecting the maximum one. This step is shown in the figure 5.38

And the forth step is as the same as of the forth step of the second case, which selecting the maximum one among these four ecdf values (for FC-AEs, LOFs, IFs and GMMs).

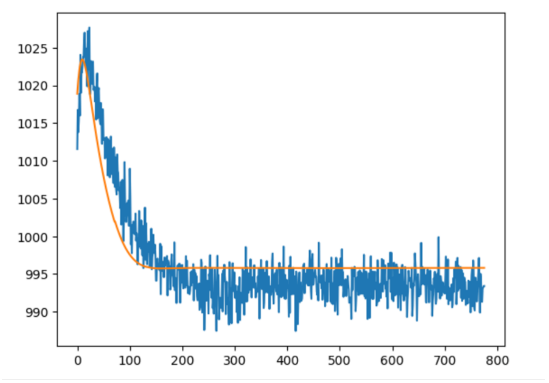


Figure 5.34: comparison between input data and reconstructed data from FC-AE (T_FC_MAX)

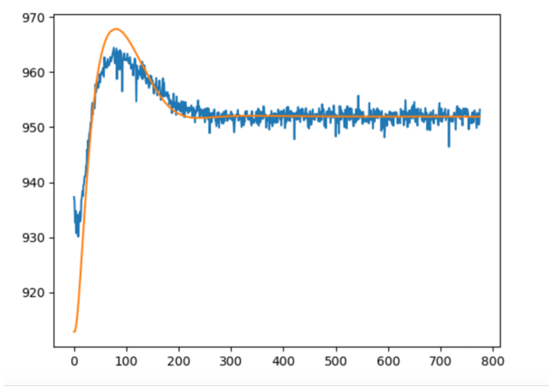


Figure 5.35: comparison between input data and reconstructed data from FC-AE (T-FC-MIN)

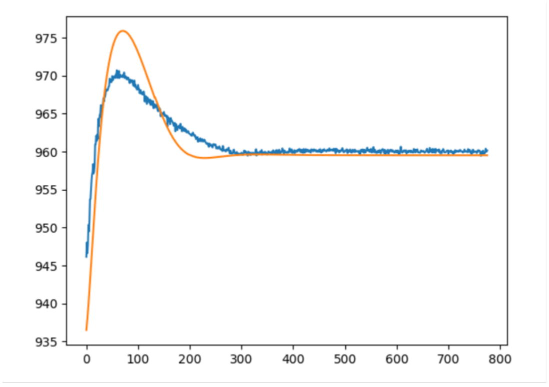


Figure 5.36: comparison between input data and reconstructed data from FC-AE (T-IC-MAX)

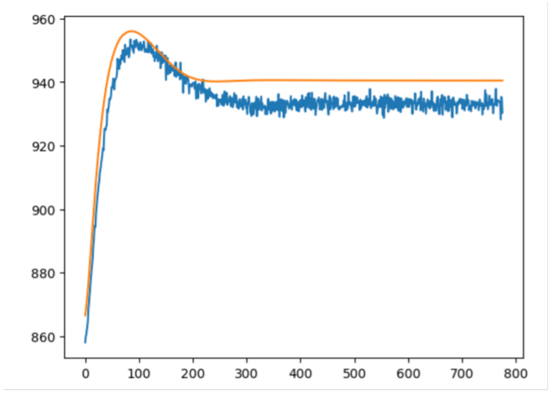


Figure 5.37: comparison between input data and reconstructed data from FC-AE (T-IC-MIN)

Ensembel

For evaluating, we use the same method as mentioned above. The detail is shown in the table 5.11. According to the table 5.11, all of these evaluation index get improvement, even though the models of Autoencoders are as the same as the models for the second case. In the third case, anomaly detection on every sequence is carried on desperately, which does not depend on the relation among the four sequences. This results in the better performance than the second case.

Comperison the performace of BiLSTM-AE and AEs, AEs' performance are

	AEs	GMM	LOF	IF	Maximum	average	AKPV	Damped Average
TNR	0.9604	0.8750	0.9354	0.8208	0.9604	0.9000	0.9146	0.8979
FNR	0.0155	0.0490	0.0253	0.0703	0.0155	0.0392	0.0335	0.0400
TPR	0.9845	0.9510	0.9747	0.9297	0.9845	0.9608	0.9665	0.9600
FPR	0.0396	0.1250	0.0646	0.1792	0.0396	0.1000	0.0854	0.1021
Accuracy	0.9777	0.9296	0.9636	0.8991	0.9777	0.9437	0.9519	0.9425
AUC	0.9656	0.9366	0.9831	0.9625	0.8952	0.9713	0.9702	0.9696

Table 5.10: Ensemble performace of the second case

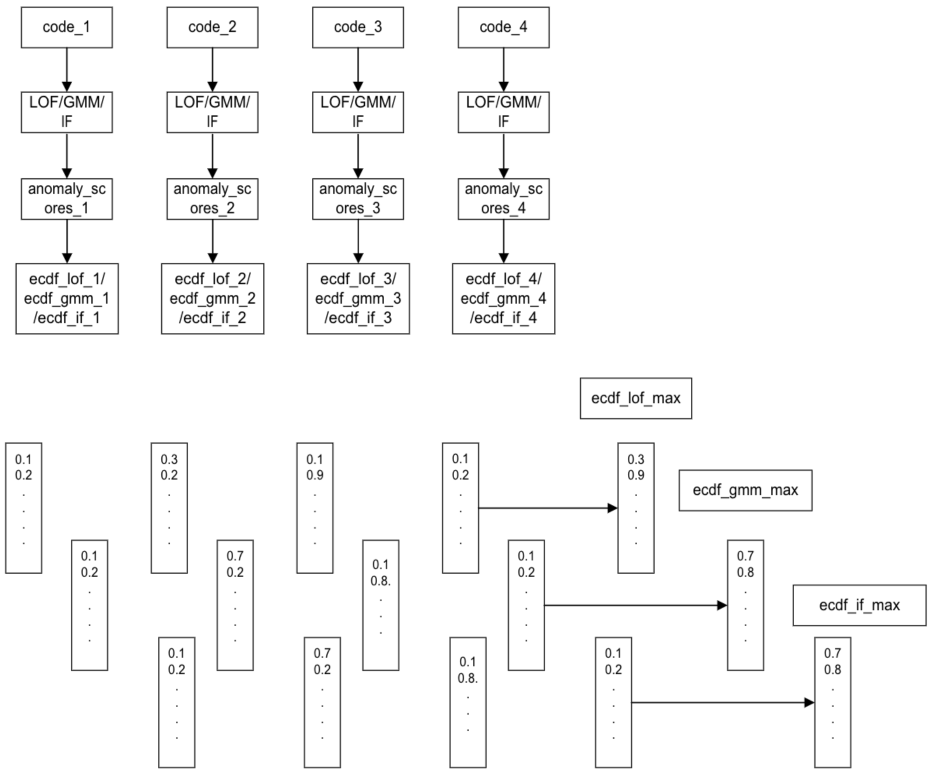


Figure 5.38: the third step of the third case

better than BiLSTM-AE's. This could be considered as motivation to ensemble more than one simple models. Deploying multiple simple models can save runtime without scarifying the performace.

	AEs	GMMs	LOFs	IFs	Maximum	average	AKPV	Damped Average
TNR	0.9604	0.8083	0.9583	0.8708	0.9688	0.8896	0.8854	0.8875
FNR	0.0155	0.0752	0.0163	0.0507	0.0123	0.0433	0.0449	0.0441
TPR	0.9845	0.9248	0.9837	0.9493	0.9877	0.9567	0.9551	0.9559
FPR	0.0396	0.1917	0.0417	0.1292	0.0312	0.1104	0.1146	0.1125
Accuracy	0.9777	0.8920	0.9765	0.9272	0.9824	0.9378	0.9354	0.9366
AUC	0.9656	0.9172	0.9904	0.9650	0.9632	0.9790	0.9735	0.9770

Table 5.11: Ensemble performace of the third case

Chapter 6

Conclusion

This thesis holds significance in advancing the field of (times series) anomaly/outlier detection by means of the following contributions:

1. Most methods rely on their own "arbitrary" definitions of outliers and of how they differ from normal datapoints. This results from the lack of a rigorous definition of what constitutes "normal (resp., abnormal) behavior" and of a corresponding quantitative measure that can be universally accepted as such. To tackle this issue, the outlier detection algorithms here employed are originally tailored to produce normalized anomaly scores: actually, in many contexts, a quantifiable "score of outlyingness" is much more useful than a simple binary label of "inlier" or "outlier" (that many available techniques only provide), since there will always be data whose outlyingness may be considered too small to classify them as outliers, but whose score is still sufficient to be relevant when analyzing their taxonomy. In particular, in this thesis the outputs of the detection algorithms (i.e., properly defined anomaly scores) are normalized within an original probabilistic framework to motivate a statistical interpretability of the unified scores. Outlier scores usually are not equally distributed in their value range but follow a much more complex distribution, which is hard to grasp analytically, and many assumptions must be made to get a reliable result. Instead, it becomes much easier to analyze the actual score distribution on the data set without assuming anything on the distribution of the data. In [3] several normalization functions are proposed that are based on classical Gaussian, Gamma, F and Cauchy Probability Distribution Functions (PDFs). It should be again pointed out that we do not draw any assumptions on the distribution of the data, but on the distribution of the derived outlier scores. The intuition of this approach is the following: we do not have a direct way to interpret the score, we instead evaluate how "unusual" the score of a point is, using the algorithms output score as a one-dimensional feature

projection of the data. Let us note that any distribution function can be used for this purpose analogously, depending on how well it fits to the distribution of the derived outlier scores. In this work, we employ an Empirical Cumulative Distribution Function (CDF) approach to fit the available scores and avoid making any assumption on their distribution.

2. Due to the very broad and diverse nature and taxonomy of the time series outliers, the performance of the available detection methods largely depends on the “physical origin” of the anomaly, on the nature and spatial distribution of the data points (i.e., depending on the type, sparsity and homogeneity of the dataset). In practice, no single detection algorithm can universally apply to all datasets. To address this problem, several diverse detection algorithms are considered within an ensemble framework and all the normalized anomaly scores produced are aggregated to get a unique, robust and reliable score. The underlying idea is that some algorithms will perform well on a particular subset of points (i.e., the outliers that are the most “obvious” from their respective standpoints), whereas other algorithms will do better on other subsets of points. By so doing, the ensemble combination is often able to obtain more accurate and robust results because of its ability to combine the outputs of multiple algorithms and to “correct” errors committed by the single, diverse ensemble members. In this thesis, the (probabilistic) normalized anomaly scores are combined for the robust and conservative identification of anomalies in datasets: different combination functions are employed, including "averaging", "maximization", "damped average" and "AKPV", which averages the scores of the top three anomaly detectors.

3. The diversity of the existing methods hinders the comparability of the results. In an unsupervised framework the only valid way of testing the detection capabilities of a detection algorithm is to compare its provided results through a chosen set of toy examples. Most examples simply rely on the comparison of their detection rates in rather simple models, and they are almost never tested on sets of functional data coming from real systems or generated by numerical simulations. In this respect, in the present thesis the effectiveness of the proposed approaches is systematically assessed by means of several diverse statistical indicators. In particular, the global accuracy, the True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR), False Positive Rate (FPR) and the Area Under The Curve (AUC)-Receiver Operating Characteristics (ROC) scores are used for assessing the performance of the individual methods and of the ensemble.

4. Two relevant and realistic functional datasets made of time series and taken from “safety-critical” sectors are considered: 1) the ECG5000 dataset (a

collection of heartbeat signals); and 2) a set of transients representing the time evolution of several physical parameters (e.g., fluid temperatures) taken from the simulator of a Generation-IV nuclear reactor, i.e., a Molten Salt Fast Reactor (MSFR).

From the camparison between the individual methods and the ensemble methods, the ensemble method of "maximum", that selecting the maximum one among the outputs from the different methods, always get improvements no matter on ECG5000 dataset and MSFR dataset. Even though in the second case on MSFR dataset, its performance is as the same as the dominating individual method. For ECG5000 dataset, the dominating method is related to "LOF" with accuracy of 0.9964, and then the "maximum" has the accuracy of 0.9992. For the MSFR dataset, for the first case, the dominating method is related to "GMM" with accuracy of 0.9554, and then the "maximum" has the accuracy of 0.9683. For the second case of MSFR dataset, the dominating method is related to "GMM" with accuracy of 0.9554, as the same as the ensemble method of "maximum". For the MSFR dataset, for the third case, the dominating method is related to "AEs" with accuracy of 0.9777, and then the "maximum" has the accuracy of 0.9824.

And in terms of comparison among individual methods on these two datasets, the dominating methods are different, which is also motivation to use different methods on the single dataset to find the more suitable method.

And the performance of the SAE and individual FC-AE on ECG5000 dataset indicates that even the simple model with unsatisfying performace could be complementary for the complex model for the accidental negligence of the complex model. By combination of SAE and FC-AE, the ensemble method of "maximum" also got improvement, in terms of accuracy, from 0.9923 to 0.9950

In terms of three cases on MSFR dataset, there are two different kinds of models, BiLSTM-AE and the combination of four FC-AEs. BiLSTM-AE usually is considered as more complex than FC-AEs and it also more difficult to train due to gradient exploring and gradient vanish and it gets better performance than single FC-AE as well. Whereas, for this case study, the performance of FC-AEs combination is better than the performance of BiLSTM, which motivates to use more than one simple models on complex data to get the satisified result. And combination of more than one simple models also could be faster than one complex model. In terms of computaional complexity, BiLSTM-AE has 7.47 M times total mult-adds and the four FC-AEs have 0.83 M total mult-adds totaly.

Bibliography

[1] Durgesh Samariya and Amit Thakkar. «A comprehensive survey of anomaly detection algorithms». In: *Annals of Data Science* 10.3 (2023), pp. 829–850 (cit. on pp. 1, 5, 7, 8).

[2] Kamran Shaukat, Talha Mahboob Alam, Suhuai Luo, Shakir Shabbir, Ibrahim A Hameed, Jiaming Li, Syed Konain Abbas, and Umair Javed. «A review of time-series anomaly detection techniques: A step to future perspectives». In: *Advances in Information and Communication: Proceedings of the 2021 Future of Information and Communication Conference (FICC), Volume 1.* Springer. 2021, pp. 865–877 (cit. on p. 1).

[3] Hans-Peter Kriegel, Peer Kroger, Erich Schubert, and Arthur Zimek. «Interpreting and unifying outlier scores». In: *Proceedings of the 2011 SIAM International Conference on Data Mining.* SIAM. 2011, pp. 13–24 (cit. on pp. 2, 54).

[4] Jina Kim, Hyeongwon Kang, and Pilsung Kang. «Time-series anomaly detection with stacked Transformer representations and 1D convolutional network». In: *Engineering Applications of Artificial Intelligence* 120 (2023), p. 105964 (cit. on p. 5).

[5] Tomáš Pevnỳ. «Loda: Lightweight on-line detector of anomalies». In: *Machine Learning* 102 (2016), pp. 275–304 (cit. on p. 5).

[6] Yue Zhao, Zain Nasrullah, Maciej K Hryniewicki, and Zheng Li. «LSCP: Locally selective combination in parallel outlier ensembles». In: *Proceedings of the 2019 SIAM International Conference on Data Mining.* SIAM. 2019, pp. 585–593 (cit. on p. 5).

[7] Markus Breunig, Peer Kröger, Raymond Ng, and Joerg Sander. «LOF: Identifying Density-Based Local Outliers.» In: *ACM Sigmod Record* 29 (June 2000), pp. 93–104. DOI: 10.1145/342009.335388 (cit. on pp. 6, 9, 18).

[8] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. «Isolation forest». In: *2008 eighth ieee international conference on data mining.* IEEE. 2008, pp. 413–422 (cit. on pp. 6, 7, 18, 19).

[9] Sunil Aryal, Kai Ming Ting, Jonathan R Wells, and Takashi Washio. «Improving iforest with relative mass». In: *Advances in Knowledge Discovery and Data Mining: 18th Pacific-Asia Conference, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. Proceedings, Part II 18.* Springer. 2014, pp. 510–521 (cit. on p. 7).

[10] T.K. Moon. «The expectation-maximization algorithm». In: *IEEE Signal Processing Magazine* 13.6 (1996), pp. 47–60. DOI: `10.1109/79.543975` (cit. on p. 7).

[11] Lishuai Li, R. John Hansman, Rafael Palacios, and Roy Welsch. «Anomaly detection via a Gaussian Mixture Model for flight operation and safety monitoring». In: *Transportation Research Part C: Emerging Technologies* 64 (2016), pp. 45–57. ISSN: 0968-090X. DOI: `https://doi.org/10.1016/j.trc.2016.01.007`. URL: `https://www.sciencedirect.com/science/article/pii/S0968090X16000188` (cit. on p. 7).

[12] Hyun Joon Shin, Dong-Hwan Eom, and Sung-Shick Kim. «One-class support vector machines—an application in machine fault detection and classification». In: *Computers & Industrial Engineering* 48.2 (2005), pp. 395–408 (cit. on p. 7).

[13] Abdenour Bounsiar and Michael G. Madden. «One-Class Support Vector Machines Revisited». In: *2014 International Conference on Information Science & Applications (ICISA).* 2014, pp. 1–4. DOI: `10.1109/ICISA.2014.6847442` (cit. on p. 7).

[14] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. «Efficient algorithms for mining outliers from large data sets». In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data.* 2000, pp. 427–438 (cit. on p. 7).

[15] Mahito Sugiyama and Karsten Borgwardt. «Rapid distance-based outlier detection via sampling». In: *Advances in neural information processing systems* 26 (2013) (cit. on p. 7).

[16] Mon-Fong Jiang, Shian-Shyong Tseng, and Chih-Ming Su. «Two-phase clustering process for outliers detection». In: *Pattern recognition letters* 22.6-7 (2001), pp. 691–700 (cit. on p. 8).

[17] Mohammad Irani Azad, Roozbeh Rajabi, and Abouzar Estebsari. «Non-intrusive load monitoring (nilm) using deep neural networks: A review». In: *2023 IEEE International Conference on Environment and Electrical Engineering and 2023 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe).* IEEE. 2023, pp. 1–6 (cit. on p. 9).

[18] Arden Dertat. *Applied Deep Learning - Part 3: Autoencoders.* `https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798` (cit. on p. 10).

[19] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. *Deep Learning Techniques for Music Generation – A Survey.* 2019. arXiv: `1709.01620 [cs.SD]` (cit. on pp. 12, 13).

[20] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles. «A review on the long short-term memory model». In: *Artificial Intelligence Review* 53.8 (2020), pp. 5929–5955 (cit. on p. 14).

[21] Paul J Werbos. «Backpropagation through time: what it does and how to do it». In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560 (cit. on p. 16).

[22] Laurens Van der Maaten and Geoffrey Hinton. «Visualizing data using t-SNE.» In: *Journal of machine learning research* 9.11 (2008) (cit. on pp. 16, 17).

[23] distill. *How to Use t-SNE Effectively.* `https://distill.pub/2016/misread-tsne/?_ga=2.135835192.888864733.1531353600-1779571267.1531353600` (cit. on p. 16).

[24] *Dataset: ECG5000.* `https://www.timeseriesclassification.com/description.php?Dataset=ECG5000` (cit. on p. 25).

[25] Ary L Goldberger et al. «PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals». In: *circulation* 101.23 (2000), e215–e220 (cit. on p. 25).

[26] Donald S Baim et al. «Survival of patients with severe congestive heart failure treated with oral milrinone». In: *Journal of the American College of Cardiology* 7.3 (1986), pp. 661–670 (cit. on p. 25).

[27] *BIDMC Congestive Heart Failure Database.* `https://physionet.org/content/chfdb/1.0.0/` (cit. on p. 25).

[28] Moumita Roy, Sukanta Majumder, Anindya Halder, and Utpal Biswas. «ECG-NET: A deep LSTM autoencoder for detecting anomalous ECG». In: *Engineering Applications of Artificial Intelligence* 124 (2023), p. 106484 (cit. on p. 25).

[29] Yanping Chen, Yuan Hao, Thanawin Rakthanmanon, Jesin Zakaria, Bing Hu, and Eamonn Keogh. «A general framework for never-ending learning from time series streams». In: *Data mining and knowledge discovery* 29 (2015), pp. 1622–1664 (cit. on p. 26).

[30] Moumita Roy, Sukanta Majumder, Anindya Halder, and Utpal Biswas. «ECG-NET: A deep LSTM autoencoder for detecting anomalous ECG». In: *Engineering Applications of Artificial Intelligence* 124 (2023), p. 106484 (cit. on p. 27).

[31] Diederik P Kingma and Jimmy Ba. «Adam: A method for stochastic optimization». In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 31).

[32] Claudio Tripodo, Andrea Di Ronco, Stefano Lorenzi, Antonio Cammi, et al. «Development of a control-oriented power plant simulator for the molten salt fast reactor». In: *EPJ Nuclear Sciences & Technologies* 5 (2019), pp. 1–24 (cit. on pp. 37, 38).

[33] S. Lorenzi. «SAMOSAFER School, Module 6 Operation and Control - MSFR Modelica Simulator». In: *Politecnico di Milano* (2021) (cit. on p. 37).

[34] Nicolo' Caruso. «Simulation-based exploration of the model of a Molten Salt Fast Reactor for the identification and classification of abnormal operating conditions». PhD thesis. Politecnico di Torino, 2023 (cit. on p. 39).

[35] Claudio Tripodo, Andrea Di Ronco, Stefano Lorenzi, Antonio Cammi, et al. «Development of a control-oriented power plant simulator for the molten salt fast reactor». In: *EPJ Nuclear Sciences & Technologies* 5 (2019), pp. 1–24 (cit. on p. 40).

[36] Enes Zvornicanin. *Differences Between Bidirectional and Unidirectional LSTM.* `https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm` (cit. on p. 40).