Cyber Resilience Model Based on a Self Supervised Anomaly Detection Approach

Eko Budi Cahyono[1], Suriani Binti Mohd Sam[2], Noor Hafizah Binti Hassan[3], Amrul Faruq[4]
Faculty of Artificial Intelligence, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia[1, 2, 3]
Faculty of Engineering, Universitas Muhammadiyah Malang, Malang, Indonesia[1, 4]

***Abstract*—Cyber resilience plays an important role in dealing with cybersecurity and business continuity uncertainty in the post-COVID-19 era. The fundamental problem of cyber resilience is the complexity of real-world problems. Therefore, it is necessary to reduce the complexity of real-world problems to be simple and easy to analyze through cyber resilience model. The first part is the representational model by utilizes world models. It utilizes the stochastic nature of latent data to generate log-likelihood values by data-generating process. The second part is the inference model. This concludes the observation of log-likelihoods using a self-supervised anomaly detection approach. This is related to optimizing decision boundary in anomaly detection, which is achieved by supervising two competing hypotheses based on bias-variance alignment and likelihood ratios. The optimization operates a dynamic threshold supervised by a supervisory signal from the underlying structure of log-likelihoods. The paper contributes by conducting research on the cyber resilience model from the perspective of statistical machine learning. It enhances the representational modeling of world models with the Gaussian mixture model for multimodal regression (GMMR). Additionally, it examines the issue of misleading log-likelihood for out-of-distribution inputs caused by the generalization error and optimizes decision boundary in minimizing the generalization error with a new metric named the harmonic likelihood ratio (HLR). Finally, it aims to boost the performance of anomaly detection using self-supervised learning.**

Keywords*—Cybersecurity; anomaly detection; cyber resilience model; statistical machine learning; data generating process; bias variance alignment; likelihood ratios; self-supervised learning

I. Introduction

In the post-COVID-19 era, cyber resilience plays an important role in dealing with cybersecurity and business continuity uncertainty. The problems and methodology for cyber resilience, especially in the scope of small and medium enterprises (SMEs), have been described systematically in the cyber resilience progression model [5], [6]. The model helps SMEs prioritize cyber resilience proposing the natural evolution of implementation. It designed to be more flexible based on available resources. It focuses on insights into operationalizing cyber resilience strategies by describing ten domains. One of the ten domains is detection processes and continuous monitoring has been chosen as the central domain [4]. In description, it is explained that there are two strategies for this domain: actively monitor the company's assets and define a detection process that specifies when to escalate anomaly into incidents [5], [6]. This description allows self-supervised anomaly detection technology that works fully automatically to detect and monitor processes.

The cyber resilience model that will be studied refers to technical architectures in [4]. The technical architecture was built on five layers. The five layers were services, data, generative models, data analysis, and resilience scale [4]. Each layer had different structures and functions but was part of a system. The technical architectures are necessarily simplified without changing the fundamental architecture so that the complexity of the problem is reduced and more focused on anomaly detection. The fundamental architecture that continues to go is Gaussian mixture models (GMMs), one of the first-generation generative models with high flexibility in handling data distribution, statistical modeling, and inference. GMMs are the fundamental architecture of cyber resilience with the underlying structure of a probabilistic model approach [4]. However, GMMs fail if applied directly as the basis of cyber resilience model. GMMs fail in high-dimensional data. GMMs can be used for density estimation but the perfect density model cannot guarantee anomaly detection [23]. Also, GMMs work in an unsupervised setting that often produces high false positive rates in a dynamic system [28]. The paper studies GMMs as a reliable generative model for anomaly detection.

The cyber resilience model designed by a two-part model: the representational model in layers 1 to 3, which play a role in data collection, and the inference model in layers 4 and 5, which play a role in data analysis. The complexity of real-world problems of cyber resilience encapsulated by data collection and analysis as the key of a data-driven approach. The paper studies critically the problems of the representational and inference model to set up the appropriate cyber resilience model, such as misleading log-likelihood, the bias-variance alignment, and two competing hypotheses.

As the core of the representational model, the world model [14] that lies in layer 2 takes the principal role in data-generating process to strengthen a data-driven approach. Hence, it needs to be enhanced to become part of the representational model that meet the requirements for data collection in a new setting. That is different from it utilized previously. For developing the data-generating process, the world model needs to avoid generating misleading log-likelihood [3]. Also, it needs to be integrated in a supervised setting [11], [12] to align the self-labeling problem of an unsupervised learning algorithm [1], [28].

The inference model in layers 4 and 5 focuses more on inferring the observed samples with data analysis directed by anomaly detection. The concept of anomaly detection explains that samples can be distinguished into normal and anomaly

samples [28], [35] with a two-class decision boundary [1], [8] determined by two competing hypotheses [33], [41]. The factual problem is aligning a two-class decision boundary with two competing hypotheses so that the model can perform anomaly detection properly, which will be studied in this paper.

II. Review of Existing Techniques

One fundamental study as the basis of cyber resilience model is anomaly detection [4]. Anomaly detection, in this case, is similar tasks to novelty detection (ND) [1], one-class classification (OCC) [29], and out-of-distribution (OOD) [8], [27], [33], [41]. An anomaly is an observation that deviates significantly from some concepts of normality [35]. This definition is also suitable for ND, OCC, and OOD [16], [35]. If the distribution of normal instances is P, an anomaly is a sample drawn from any distribution other than P. Furthermore, ND has a new region of P; OCC has a one-class decision boundary of P; OOD has fine-grained P during training [28], [35]. Thus, there is a shared concept between anomaly detection and ND, OCC, OOD.

The cyber resilience model represents a cyber resilience system as a technical framework to measure the resilience scale. The resilience scale denotes a scale of the service resilience against cyber threats. In the domain of cyber resilience, numerous services like Domain Name System (DNS), firewall, web services, resource planning, and supply chain are required to explore anomaly detection for monitoring the continuity of services. It is necessary for one specific service to bring about the model as suggested in [4] which points to DNS. This choice is reasonable because DNS used by most services and applications of the Internet, even critical infrastructures of the Internet. Also, DNS provides an authentic data distribution of anomalies by the system so that anomaly detection works in factuality. DNS is the hierarchical name system that uses the globally distributed database and stores the information about every Internet domain [39]. DNS information is stored on DNS servers and can be accessed anytime based on user queries. DNS is like a phone book. It contains addresses that make it easier for users to access the Internet in cyberspace. Each address has a unique identity and is different from one another. This identity is called an Internet Protocol (IP) address. DNS translates these numbers into names because IP addresses are complicated for users to remember. DNS is made simple to solve IP address resolution issues, however, it does not come with a security proficiency by nature [4]. Therefore, a DNS measurement method is needed to enable security proficiency by design.

DNS measurement methods are categorized into two types: passive and active DNS. Passive DNS [39] allows observation of DNS ecosystem limited to clients, resolver servers, authoritative servers and the networks between the three. Active DNS allows observing a global hierarchical DNS ecosystem. It involves very large DNS networks. It does not pay attention to the complete contents of the query and response from a particular client, so it has difficult to apply for data collection and analysis. The mechanism for DNS queries only supports one type of query. One query of the domain name and one response of an IP address. Other lookup keys must be converted to domain names before being used in DNS queries. Thus, passive DNS provides advantages over active DNS in reconstructing specific queries/responses useful for anomaly detection.

Passive DNS, as a local representative of the active DNS ecosystem, contributes to efficient traffic data collection. Many DNS traffic analysis uses passive DNS including anomaly detection. The earlier DNS anomaly detection is based on supervised model. The model produce high precision, low false positive rates, and efficient anomaly detection in specific patterns [28]. However, the model depend on training data. The ability to recognize anomaly patterns depends on the anomaly patterns that have been trained during training time [16], [28], [35]. The trained anomaly patterns are limited to the capacity of datasets. While DNS ecosystem is always evolving, threats to DNS security never stop, and DNS resilience needs to be monitored continuously. Therefore, supervised models are less suitable for DNS traffic anomaly detection in dynamic real world environments.

Since DNS traffic is dynamic, unsupervised model is inherently suitable for anomaly detection. Labeling normal and anomaly traffic are not necessary during training time. The model accepts all types of unlabeled traffic and classify it with certain rules into two classes that can distinguish normal or anomaly traffic. The model does not depend on training data [16], [28], [35]. The ability to recognize anomaly patterns does not depend on the trained anomaly patterns during training time. The model can recognize new anomaly patterns using a classifier algorithm. However, the model often produces high false positive rates and fails to recognize patterns from the genuine labeled data [28]. The model's ability to recognize normal and anomaly patterns is not as precise as the supervised model because there is no strong connection between the model and the genuine labeled data [28]. In this model, the genuine labeled data is not defined. Normal and anomaly labels are not explicitly defined.

A machine learning algorithm that makes use of the structure within data for self-labeling is self-supervised model [16], [32]. Self-supervised model has been designed as a hybrid approach of supervised and unsupervised models. The model produces pseudo-labels, but the model directs pseudo-labels to follow the underlying structure of the genuine labeled data. Anomaly detection can utilizes self-supervised model based on likelihood ratios [32], [33], [41]. Likelihood ratios can manage the relationship between supervised and unsupervised models. Likelihood ratios have signal to measure the performance of regression and classification functions in a supervised setting [19], [22], [25], [37]. Likelihood ratios also have signal to control self-labeling in an unsupervised setting [1], [2], [19], [21], [33], [41].

III. Methodology

Cyber resilience plays an important role in facing business continuity uncertainty. Outside of business matters, business continuity uncertainty can be affected by data uncertainty in data collection and analysis. Various sources, such as noisy data, incomplete data, sampling errors, measurement errors generalization errors, and anomalies, which will cause data uncertainty. It is necessary to manage data uncertainty to

improve the reliability of data collection and analysis. One of the approaches chosen to address the problem of data uncertainty is to study anomaly detection in more depth. It has several indicators that will be used to obtain reliable data collection and analysis. The indicators are bias, variance, and likelihood ratios.

This section explains the research methodology of cyber resilience model, consist of the representational model to realize reliable data collection using data-generating process and the inference model to realize reliable data analysis based on likelihood ratios. Process flow diagram of the methodology as seen in Fig, 1.

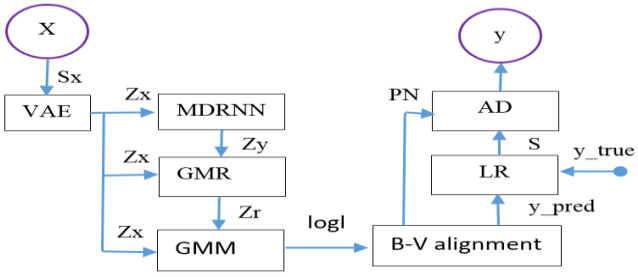


Fig. 1. Process flow diagram of the methodology.

A. Characteristics of Raw Datasets

The raw dataset is a collection of data from a network capture-based DNS logger. The raw dataset was taken from November 2018 to February 2020 on DNS servers (dnsanalyzer.info) amount 1,000 samples (Sx). The tool used to collect the raw dataset is GoPassiveDNS [26].

The raw dataset contains three feature vectors (X): TTL (x_1), latency (x_2), and throughput (x_3). The K-Means algorithm was used to collect data for classifying positive and negative classes from the raw dataset. The negative class have fine grain structure with two type subdistributions: the sample variance of subdistributions in one standard deviation (inliers) and the sample variance of subdistributions more than two standard deviations (outliers). Inliers of the negative class represent DNS normal. Inliers of the negative class have the lowest anomaly level in the resilience scale. The positive class have fine grain structure with three type subdistributions: the sample variance of subdistributions in one standard deviation (inliers), the sample variance of subdistributions more than two standard deviations (outliers), and anomaly. Anomaly refers to the response of RCODE more than 0 (zero) by GoPassiveDNS. RCODE more than 0 indicates DNS failure [26]. Otherwise, RCODE equal to 0 indicates DNS normal. The anomaly subdistribution have the highest anomaly level in the resilience scale.

The raw dataset exhibit different data characteristics named the generator of in-distribution, which is the source of normal datasets, and the generator of out-of-distribution, which is the source of anomaly datasets. The raw dataset needs to be further processed to generate reliable datasets.

TABLE I. CHARACTERISTICS OF RAW DATASETS

Datasets	In distribution		Out-of-distribution		
Classes	Negative		Positive		
Types	Inlier	Outlier	Inlier	Outlier	Anomaly
Variances	≤68%	>95%	≤68%	>95%	-
RCODE	0	0	0	0	>0
Subdistributions	1	2	3	4	5
Samples	500	25	75	200	200
Anomaly levels	1	2	3	4	5

Table I presents two sources of raw datasets: In-distribution (ID) and out-of-distribution (OOD). The ID dataset comes from the generator of in-distribution. The OOD dataset comes from the generator of out-of-distribution. Conceptually, the two generators should produce different two sources of datasets and naturally not overlap each other. However, in practice, the two generators cannot avoid producing two overlapping datasets.

Table I explains that the overlap occurs due to the two approaches taken in data collection. The first approach, data collection uses DNS sensors. The sensors are limited in representing data from the real world with only three feature vectors. The sensors are also limited in responding accurately to RCODE. Furthermore, the sensors can only classify DNS events by default based on the value of RCODE.

TABLE II. DNS RESPONSE CODE

RCODE	**Message**	**Description**
0	NOERROR	DNS query completed successfully
1	FORMERR	DNS query format error
2	SERVFAIL	Server failed to complete the DNS request
3	NXDOMAIN	Domain name does not exist
4	NOTIMP	Function not implemented
5	REFUSED	The server refused to answer for the query
6	YXDOMAIN	Name that should not exist, does exist
7	XRRSET	RRset that should not exist, does exist
8	NOTAUTH	Server not authoritative for the zone
9	NOTZONE	Name not in zone

Table II defines the DNS response code which have been implemented in GoPassiveDNS [26]. It describes RCODE values of the various types of DNS events that occur most

frequently. Refers to Table I that RCODE produces high false negative rates. Subdistribution 1, 2, 3, and 4 are classified as DNS normal even though some are DNS failure according to K-Means. In contrast, the second approach, data collection uses K-Means to classify two different datasets. Table I shows that K-Means produce high false positive rates. Subdistribution 3, 4, and 5 are classified as the positive class even though some are the negative class according to RCODE. The problem of the first approach is that normal and anomaly labeling in the raw dataset is based on RCODE even though many failure functions cannot be recognized by RCODE. The second approach uses the algorithm of machine learning. The K-Means is the simplest unsupervised learning algorithm for basic self-labeling. It can be utilized to classify the raw dataset into two classes on a specific rule.

It is a similarity between the two approaches. None of anomaly samples in subdistribution 5 that has been classified by the first approach are part of normal samples that has been classified by the second approach. In other words, anomaly samples from the first approach is the same as anomaly samples from the second approach. So, it is concluded that the classification of one sample as anomaly and another sample as not anomaly is true. However, there are anomaly samples from the second approach that are not anomaly samples from the first approach such as samples in subdistribution 3 and 4. Therefore, the first approach is not able to recognize new patterns of anomaly other than those already recognized by RCODE. This urges the use of machine learning algorithms to better recognize new patterns of anomalies. Meanwhile, the second approach is able to recognize new patterns of anomaly other than those already recognized by the first approach. However, not all of the new anomaly patterns of the second approach are true due to the limitations of K-Means based on a specific shape of the decision boundary and no training to do for anomaly samples.

B. Data Collection

The representational model are useful in improving the reliability of data collection. The reliability of data collection is improved by modeling the raw dataset into latent samples (Zx), a sequence of latent samples (Zy), and a density of latent samples (logl).

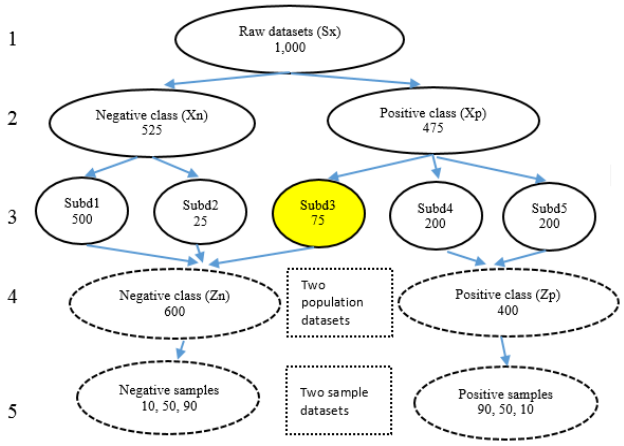


Fig. 2. Process flow diagram of data collection.

Fig. 2 describes process flow diagram of data collection which consists of five steps. The first step is initial data collection which produces raw datasets as 1,000 samples (Sx). The samples are a data distribution of 200 data points so raw datasets are 200,000 data points in size. The samples are designed as a data distribution to understand the underlying generator of datasets through latent variable models. The samples have parameters from a data distribution that show the principal component of datasets.

The second step classifies positive (Xp) and negative (Xn) classes from raw datasets using the K-Means algorithm. The positive class are samples that assert the presence of anomaly. The negative class are samples that assert the absence of anomaly. The K-Means algorithm succeeded in classifying raw datasets into 525 negative samples and 475 positive samples from selected samples.

In the third step, each class is divided into two subdistributions, namely inliers and outliers, so that in total there are four subdistributions plus one specific subdistribution produced by DNS sensors as the ground truth of anomaly samples. So the total becomes five subdistributions, each of 500, 25, 75, 200, and 200 selected samples respectively. Inliers and outliers are formed using an outlier detection technique [42].

The fourth step as the key of reliable data collection is processes that generate data specifically event, memory, and density processes. The three processes set up the data-generating process properly. A detailed explanation of each process is in the following subsection. Furthermore, one sample subdistribution differs from another. It estimated by the Expectation-Maximization (EM) algorithm [9].

TABLE III. EM AS A SAMPLE CLASSIFIER

	subd2	**subd3**	**subd4**
subd1	1.419	2.949	5.026
subd5	5.423	3.893	1.816
DI	4.004	0.944	3.210

abbreviations: subd = subdistribution, DI = difference index

Table III explains the process of classifying normal and anomaly samples using EM. Two polars are defined as the centroids of normal and anomaly classes. Subdistribution 1 with the lowest anomaly level selected as the polar of normal class. Subdistribution 5 with the highest anomaly level selected as the polar of anomaly class. The smaller difference between a subdistribution and two polars defines the label of subdistribution. As subdistribution 2 and 3 are closer to subdistribution 1 than 5, subdistribution 2 and 3 are classified as normal class. As well subdistribution 4 is closer to subdistribution 5 than 1, subdistribution 4 is classified as anomaly class. It can be seen that K-Means and EM are the same in classifying subdistribution 1, 2, 4, and 5 but different in classifying subdistribution 3. K-Means classify subdistribution 3 as anomaly class while EM classify subdistribution 3 as normal class. K-Means classify patterns with a specific shape for all subdistributions, while EM classifies patterns dynamically following the underlying

structure of each subdistribution. It seems EM is more flexible than K-Means. EM is not limited by shape but depends on unobserved latent variables performed properly by data-generating process.

The output of data-generating process is primary datasets (logl) that consist of 1,000 samples treated as a population of observation. The samples come from five subdistributions of raw datasets, each of 500, 25, 75, 200, and 200 respectively. In the last step, the observation sample is generated as a random variable of normal and anomaly samples. One hundred random variables configure the observed samples that are taken randomly from the population. Be appointed, the sample size is 10% of the population provided for any observation. The data-generating process will consider data collection more appropriate as required.

C. The Event Process

The event process is the first part of data-generating process. It encodes DNS events into a latent variable model. It reproduces the idea from the world model [14] that pattern recognition was performed indirectly through a latent variable model. The model in latent space provides more advantages than in data space. The pattern encoded as a principal component is more concise. The principal component has information sufficiency that can be further encoded in another process smoothly, so it is easier to analyze in the next part of data-generating process. VAEs take the task of encoding for a latent variable model.

Variational autoencoders (VAEs) are an artificial neural network architecture in which the latent space has good properties to enable generative processes [21] utilizing stochastic backpropagation [34]. It plays a role in transforming data distribution in data space (X) into data distribution in latent space (Zx) by the latent variable model. It recognizes the characteristic of data distribution through EM by estimating the joint distribution between X and Zx [9]. EM will maximize the similarity between the two. If the joint distribution Zx is a close approximation of X, then Zx will be indistinguishable from X. Analysis of compressed data distribution produces a principal component. This is called dimensionality reduction. Principal components in latent space are more ready to use to recognize the behavior of data distribution than feature vectors in data space. It summarize the pattern of data distribution without eliminating the principal information substance.

VAEs use two neural networks: encoder (generative model) and decoder (variational approximation) [21]. The encoder part transforms samples of discrete data distribution from DNS events into continuous latent variables by taking the characteristic of probability distribution. In this case, probability distribution as the core of a latent variable model that has two parameters: mean (μ) and standard deviation (Σ). The mean represents the expected value of DNS events, and the standard deviation represents the variability of DNS events. The type of probability distribution to generate the latent sample is Gaussian distribution. So, a latent variable model may be called a Gaussian model. The use of VAE encoder as a generator of latent samples refers to the vision model of world models [14].

One important component of neural networks that quantifies the difference between the predicted and actual outputs is the loss function. There are two loss functions for VAEs: Mean Square Error (MSE) and Kullback–Leibler Divergence (KLD). These two loss functions are standard for measuring how fit a model explains the data. The decoder part reconstructs latent samples into the predicted outputs (S'x). In the experiment, the decoder part is only needed at training to measure how fit a Gaussian model has been encodes DNS events by comparing the difference between the predicted and actual outputs (S'x-Sx). Flow diagram of VAEs as seen in Fig. 3.

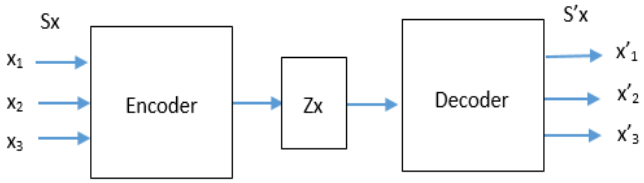


Fig. 3. Flow diagram of VAEs for the event process.

The performance of training can be evaluated from training and testing loss. Suppose the loss gets smaller for larger epochs, the loss shift convergence to a specific value, testing loss is smaller than training loss, that are indicating that VAE training has achieved appropriate performance tasks. Latent variables at training are set to $\mu + \Sigma \times \varepsilon$. The ε is random noise added to the latent variable so that the loss will always converge to a specific value [21], [34]. Some of the reasons testing loss is smaller than training loss are that at training: (1) the latent variable overloaded with random noise, whereas, at testing, it does not, (2) testing loss measured after training, and (3) training loss measured for all epochs while testing loss measured once after completing the training epoch. However, training and testing loss are limited to optimizing the loss function in estimating the difference between the actual (Sx) and predicted (S'x) DNS events and not optimizing model parameters. The model parameter is optimized by the architecture of encoder and decoder, hyperparameter tuning, and using more flexible prior distributions. This means that the model parameter of training is better than those of testing because it is used as a reference for testing. The performance of VAEs training as seen in Fig. 5(a).

VAEs are built with one input layer for three feature vectors, two hidden layers, each with sizes 64 and 16, and one output layer for training purpose only. One data distribution of the input layer come from two hundred data points. One latent sample is composed of two hundred latent vectors, in other words, the two hundred dimensions of Zx. Then, a μ vector of length two hundreds and a Σ vector of length also two hundreds, so the VAE parameters are four hundreds. In these configuration, two hundred latent vectors are a good sample size to reduce the log-likelihood of sampling errors. The observed samples need to be ensured to be independent (one discrete data of DNS events does not depend on other discrete data of DNS events in the same data distribution) and identically distributed (in one data distribution of DNS events, the probability distribution of data distribution is the same).

Consequently, a latent sample which generated by the model meets the criteria of independent and identically distributed (i.i.d) as a stochastic random sample, making it well-suited for addressing an observation sample of DNS events in latent space.

D. The Memory Process

The memory process is the second part of data-generating process that play a role in encoding a sequence of latent samples from the event process into a latent dynamics model. A latent sample which is the output of the event process does not have the sequence because it is an i.i.d sample. Meanwhile, one observation consists of many latent samples to form a sequence. The event process needs to be extended with another process called the memory process so that it can encode the sequence. Hence, the memory process is an extended event process.

Mixture density-recurrent neural networks (MDRNNs) are one specific neural network architecture that combines mixture density networks (MDNs) and recurrent neural networks (RNNs) to build a latent dynamics model. MDNs encode the output of a neural network parametrize a mixture of Gaussian distribution, which can model general conditional probability densities [2]. RNNs are an artificial neural network architecture of dynamic models that have been used to generate sequences [13], [15]. Dynamic models simplified representations of real-world problems by algorithms, such as dynamic models of signal processing, automatic speech recognition (ASR), and time-series forecasting. As a sequence generator, MDRNNs encode a sequence of latent samples Zx generated by a Gaussian model to latent samples Zy generated by a latent dynamics model. The use of MDRNNs as a sequence generator refers to the memory model of world models [14].

RNNs are used as the forefront of a sequence generator. RNNs have a recurrent layer (a cell) to remember its previous inputs by internal memory. A recurrent layer works with iterative sampling from the output, then feeding in the sample as input at the next step [13]. The problem with standard RNNs is that it is used only for basic sequential data tasks and cannot be used to store information about previous inputs for a very long time [13]. Hence, a type of RNNs called long short-term memory networks (LSTMs) [15] was designed to address the problem of standard RNNs. LSTMs can be used for advanced sequential data and artificial long-time lag tasks [15], such as tasks to predict a sequence of latent samples, and then store information in internal memory with memory cells and gating mechanisms for long-term. Memory cells store information from the previous step and use it to update the cell output for the current step. Gating mechanisms remove unimportant information, store new information, and encode information from the cell state to the output layer.

Practically, LSTMs need MDNs for modeling advanced sequential data. LSTMs and MDNs are compatible with each other. MDNs consist of two components: a feed-fordward neural network and a mixture model [2]. The output layer of LSTMs is the final layer of a feed-fordward neural network and a fully connected layer that connects all units in the layer directly to every unit in the previous layer, so it can be utilized as an interface between LSTMs and MDNs.

MDNs can smoothly encode the outputs of LSTMs to form the parameters of a mixture model, generally with Gaussian models for each mixture component [2], [10]. These parameters are mean (μ), standard deviation (Σ), and weight (π). The mean represents the expected value of DNS events, the standard deviation represents the variability of DNS events, and the weight represents mixing each Gaussian distribution of DNS events into a mixture distribution. These parameters involve shaping probability density functions (PDFs) for each mixture component and categorical distribution from the mixture weights [2], [10]. Additionally, there are two reasons why LSTMs require MDNs: different mixture components represent different stochastic events and different situations [10]. In other words, MDNs are able to model different stochastic events and situations for any DNS events in Gaussian models for each mixture component. Therefore, LSTMs and MDNs can seamlessly work together in MDRNNs. Flow diagram of MDRNNs as seen in Fig. 4.

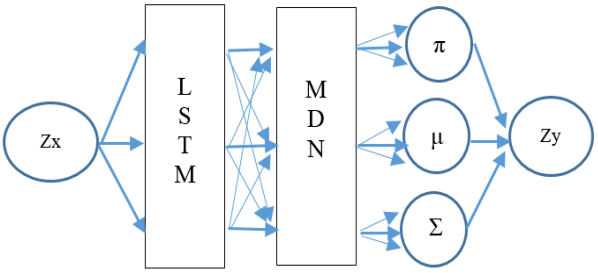


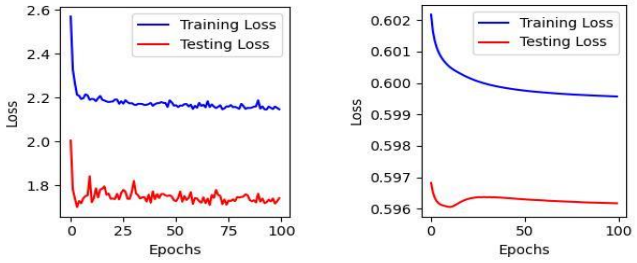
Fig. 4. Flow diagram of MDRNNs for the memory process.

One observation consists of some latent samples. In a sequence, one latent sample is related to other latent samples. MDRNNs training recognize the underlying generator of a sequence by making relationships between one sample and another using a time-series regression approach. Some samples are treated as independent variables, while the dependent variable is taken from the independent variable itself which has been shifted in a sequence. In this case, the independent variable is Zx while Zy is constructed from shifted Zx with a sequence length of four.

LSTMs have been applied to realize this time-series regression. Because LSTMs can memorize previous input, then for each latent sample, MDNs would encode the output of LSTMs into a mixture component that was a combination of individual distributions in the form of a mixture distribution. To identify that MDRNNs training has been able to recognize the underlying generator of a sequence, it was evaluated using the logsumexp function. The logsumexp function is a smooth approximation to the maximum function in a logarithmic scale. This provides a numerically stable estimation of PDFs for each sequence and sample. The logsumexp is used to measuring the similarity between the sequence and sample by maximizing the log-likelihood.

MDRNNs are built with one input layer, a single hidden LSTM layer, the output layer of LSTMs as the input of MDNs, a sequence length is four at training time, and a

mixture distribution from three Gaussian distributions as a mixture model. Three Gaussian distributions were chosen because the input distribution is not complex, it only consists of three feature vectors. One data distribution of the input layer comes from two hundred latent vectors. A single hidden layer with sixteen units is enough to perform computations on the input which consists of two hundred latent vectors. MDNs aim to model a sequence of latent samples that can be drawn from one of several possible distributions with a certain probability. Several possible distributions come from three Gaussian distributions. Each parameter of three Gaussian distributions needs two hundred vectors, so MDNs parameterize 3×3×200 = 1,800 parameters. These parameters will generate two hundred latent vectors as the output of MDRNNs (Zy). MDRNNs training uses the data of VAEs training so that the performance of MDRNNs training follows of VAEs training such as shown in Fig. 5.



a. The performance of VAEs training b. The performance of MDRNNs training

Fig. 5. The performance of VAEs and MDRNNs training.

E. The Density Process

The density process is the last part of data-generating process that play a role in producing the density (logl) from a sequence of latent samples. The density process works to enhance the density of the memory process. The memory process is designed to encode a sequence of latent samples, not the density. It is not sufficient to produce reliable density because a standard MDN as part of the memory process is prone to mode collapse [25]. Mode collapse happens when a standard MDN fail to generate data samples from the underlying probability distribution of Zx. It will turn multimodal learning [38] into unimodal so that all of the generated samples are very similar [25]. To overcome the problem, a joint distribution was built on Zx and Zy, denoted p(Zx, Zy). In this case, Zx and Zy represent the input and target variables respectively. The relationship between the input and target variables has been initialized by LSTMs in the memory process through time-series regression. It is the conditional distribution, which is the probability that Zy happens given Zx has happened, denoted p(Zy|Zx). It is created to build a more proper regression that is the joint probability density functions (joint PDFs). All the statistical information of the input and target variables is stored in the joint PDFs [37]. Furthermore, learning the joint PDFs of the input and target variables is a form of supervised learning [12]. It is a statistical approach to transform from unsupervised learning of the event process, denoted p(Zx), into supervised learning of the density process, denoted p(Zx, Zy). The joint PDFs and the conditional distribution are the statistical basis for multimodal regression.

An important factor in the joint PDFs is to predict target variables from input variables. A type of supervised learning algorithm that has been used to predict target variables from input variables as part of the joint PDFs is Gaussian mixture regression (GMR) [11], [12], [20], [36], [37], [40]. GMR is a regression approach that models probability distributions of latent samples from the event process and the memory process into multimodal regression using Gaussian mixture models (GMMs). It can be used to predict distributions of variables Zy by computing the conditional distribution p(Zy|Zx). The first process in GMR is to learn the joint PDFs through EM, then compute the conditional distribution to predictions. Training is the same procedure as in GMMs [11].

GMMs and GMR can seamlessly work together called the Gaussian mixture model for multimodal regression (GMMR), which applies two functions in one: the prediction function and the score function. The prediction function to predict target variables from input variables, this is the role of GMR. The score function to estimate the new joint PDFs of the input and class label, this is the role of GMMs. The realization of GMMR is in two steps. Step 1 is to predict target variables Zr from input variables (Zx, Zy) or Zxy using the prediction function. This target variable Zr is also used to predict class labels. Step 2 is to produce log-likelihood (logl) from bivariate data (Zx, Zr) or Zxr using the score function. Therefore, GMMR specifically is designed to enhance the reliability of the log-likelihood estimation produced by world models [14]. Flow diagram of GMMR as seen in Fig. 6.

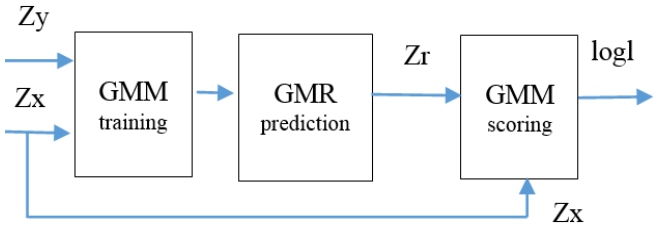


Fig. 6. Flow diagram of GMMR for the density process.

In step 1, GMMR has utilized the mixture of experts (MoE) [20] for developing the GMR prediction. MoE gives a simple approach to combine parametric and nonparametric regression methods by taking the analytic advantages of parametric and the flexibility of nonparametric [12], [19], [37]. The parametric method has been realized by a mixture of linear models [36] and the nonparametric method has been realized with divide-and-conquer principles [20]. In fact, the model log-likelihood function disregards the global data density [32], because the global maximum that indicates the global data density does not smoothly regress all local maxima. Then, MoE replaces a single global model with a weighted sum of local models (experts) [11], [12], [20], [40] to achieve the stability of multimodal regression as follows:

$$p(Zx, Zy) = \sum_{k=1}^{K} \pi_k N_k(Zx, Zy|\mu(Zxy_k), \Sigma(Zxy_k)) \quad (1)$$

$$p(Zy|Zx) = \sum_{k=1}^{K} \pi(Zy|Zx_k) N_k(Zy|\mu(Zy|Zx_k), \Sigma(Zy|Zx_k)) \quad (2)$$

Eq. (1) is the joint PDFs of GMMs via the EM algorithm [9]. $N_k(Zx, Zy|\mu(Zxy_k), \Sigma(Zxy_k))$ are Gaussian distributions with means $\mu(Zxy_k)$, covariances $\Sigma(Zxy_k)$, five Gaussian components (K=5). Weights $\pi_k \in [0, 1]$ are priors that sum up to one. EM is an iterative method for fitting GMMs to the negative class label of latent samples (Zx, Zy) or Zxy in an OCC setting. Eq. (2) is GMR model via the MoE to predict distributions of variables Zr by computing the conditional distribution p(Zy|Zx). The conditional distribution of each Gaussian distribution is $N(Zx,Zy|\mu(Zxy),\Sigma(Zxy))$ then the conditional distribution of a mixture of Gaussian distributions is $N_k(Zy \mid \mu(Zy \mid Zx_k), \Sigma(Zy \mid Zx_k))$ with means $\mu(Zy|Zx_k)$ and covariances $\Sigma(Zy|Zx_k)$. Weights $\pi(Zy|Zx_k) \in [0, 1]$ are priors that sum up to one.

In step 2, GMMR has utilized EM [9] for developing the score function. EM has a role for fitting GMMs to the negative class label of latent samples (Zx, Zr) or Zxr in an OCC setting again. The joint PDFs are used to store the statistical information of the input and target variables such in Eq. (3). The target variable is assumed class labels. In this case, class labels for the global model are missing, then EM is used to compute the parameters of each mixture component and estimate the maximum log-likelihood of GMMs as the score function such in (4).

$$p(Zx, Zr) = \sum_{k=1}^{K} \pi_k N_k(Zx, Zr|\mu(Zxr_k), \Sigma(Zxr_k)) \quad (3)$$

$$logl = \log p(Zx, Zr) \quad (4)$$

Fig. 7 demonstrates GMMR with five local models, which yield the predicted outcome Zr_pred in the output space y-axis when conditioning by Zx on the input space x-axis. GMMR has worked well to combine linear and non-linear models. The key model of GMMR is p(Zy|Zx) which produces p(Zx, Zy) from p(Zx) that can be well represented by a set of Gaussians [36] so that GMMR can be analyzed easily using a mixture of linear models. For this reason, the evaluation of GMMR is carried out on the linear regression model. It is linear in the parameters. The linearity of parameters is shown by a linear relationship between the predictor (Zx), observed outcome (Zy), predicted outcome (Zr) in performing local regression [19], [20], [36] at the query point on demand. The query point on demand using: (1) the nearby training observations [19], (2) a nested sequence of regions [20], and (3) a small set of close points [36], to build a mixture of linear models. From global regression into local regression referred to as a memory-based procedure [19] of experts and gates. It can predict the outcome according to the memory of experts, each expert specializes in one local regression, and the gate defines the regions where the memory of an individual expert are trustworthy.

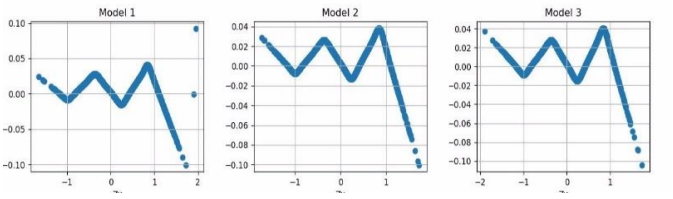


Fig. 7. GMMR with five local models.

The mechanism of GMMR with the local models has been explained specifically in hierarchical mixtures of experts (HME) [20]. The mechanism works on a set of experts and gates collaborating to solve a nonlinear function by dividing the input space into a nested sequence of regions [20], [40]. The experts learn the simple parameterized surfaces in these partitions of these regions, and the gate makes a soft split of the input space. The simple parameterized surface in both experts and gates can be learned using the EM algorithm [9].

The common metric to evaluate a regression model is root mean squared error (RMSE) and R-squared (R^2) [22]. RMSE is a function of the model residuals which is the difference between Zr_true and Zr_pred: in [0, ∞], the smaller the better. R^2 can be interpreted as the proportion of the variance in Zr_pred which is explained by the model: in [−∞, 1], the closer to 1 the better. R^2 is a measure of correlation, not accuracy [22]. The other metric is mean absolute error (MAE) which is the average of the absolut difference between Zr_true and Zr_pred: in [0, ∞], the smaller the better. The performance of GMMR for three selected models is shown in Table IV.

TABLE IV. MEASURING PERFORMANCE OF GMMR

Model	RMSE↓	R^2↑	MAE↓
1	0.0022	0.9752	0.0010
2	0.0010	0.9928	0.0009
3	0.0011	0.9926	0.0009

F. The Bias-Variance Alignment

The center point of cyber resilience model is in GMMs which build the fundamentals of representational and inference models. GMMs work on a probabilistic model to estimate maximum log-likelihood via the EM algorithm [9]. GMMs are a simple generative model that utilizes a mixture of Gaussian distributions to build a weighted sum of PDFs but GMMs have high flexibility in the predictive and inference modeling. The weakness is that GMMs fail in high-dimensional data [4]. To address the problem, the three methods take into account. The first method, dimensionality reduction to summarize important information from feature vectors into latent samples. The method uses VAEs for analyzing latent samples and MDRNNs for generating sequences of latent samples. The second method, multimodal regression to transform the non-linear function of latent samples into a mixture of linear models so that latent samples are easier to analyze just using linear regression but more stable in handling the density with local regression. The method uses MoE to combine parametric and nonparametric estimates of the model for a regression function. The third method, decision boundary optimization to conclude log-likelihood. The method uses the bias-variance alignment described in this subsection and likelihood ratios described in the next subsection.

The bias-variance tradeoff and alignment have similarities in decomposing the generalization error into bias and variance. The bias-variance tradeoff is often used to analyze

the generalization error in a regression setting [19], [22], while the bias-variance alignment is more suitable used to analyze the generalization error in a classification setting [7]. Quantitative measures of the generalization error in regression are derived from the mean squared error (MSE) which also can be calculated by squaring RMSE. The expected test MSE can be decomposed into bias and variance [19], [22] as follows:

$$E[MSE] = \sigma^2 + (\text{model bias})^2 + \text{model variance} \quad (5)$$

The first part (σ^2) is an irreducible error. It cannot be eliminated in predictive modeling. It shows intrinsic noise in the model due to unknown variables. The second part is the squared bias of the model. It shows the relationship between the predictor (Zx) and the outcome (Zr_true and Zr_pred) of the model. High bias means the model is unable to relate accurately between the predictor and the outcome, Zr_pred is very different from Zr_true that indicates underfitting. The third part is the model variance. It shows the sensitivity of predictive modeling to different datasets. High variance means the model learns more about the noise than the underlying patterns in datasets. More fitting in training data but poor in testing data indicates overfitting. The expected test MSE simultaneously achieves low bias and low variance.

Eq. (5) also works to multimodal regression via MoE. The variance model of multimodal regression can be expressed as the sum of two parts: the first part is related to the variance of the expert networks and the second part is related to the covariance of the expert networks [18]. This shows that a model that can be analyzed using a regression function means it can also be analyzed using bias-variance tradeoff, including GMMR.

TABLE V. TWO SAMPLES FOR CLASSIFICATION

Model	N(Zn)	N(Zp)	H
1	10	90	Hp
2	50	50	Hp
3	90	10	Hp
1	10	90	Hn
2	50	50	Hn
3	90	10	Hn

Table V describes two samples for classification. The samples size is one hundred for one observation. Positive samples denoted Zp and the size denoted N(Zp). Negative samples denoted Zn and the size denoted N(Zn). The hypothesis of classification models denoted H. There are two hypotheses: a positive hypothesis (Hp) and a negative hypothesis (Hn). A positive hypothesis is an observation that tests log-likelihood in an expected finding of the presence of an anomaly if the hypothesis is true. Quantitative measures of the presence of an anomaly are indicated by the positive likelihood ratio (PLR). Low log-likelihood is an instance from the presence of an anomaly. A negative hypothesis is an observation that tests log-likelihood in an expected finding of the absence of an anomaly if the hypothesis is true. Quantitative measures of the absence of an anomaly are indicated by the negative likelihood ratio (NLR). High log-likelihood is an instance from the absence of an anomaly.

The idea of using two samples for classification is motivated by the one-class classification method that suffered spurious detection [29] and perfect density models that could not guarantee anomaly detection [23]. One-class classification in the density process aims to learn a one-class decision boundary by GMMR that minimizes false positive rate (FPR) [35]. In this problem, the underlying patterns of negative samples is well-recognized by the model. However, because positive samples were not trained on the model, the model did not know the underlying patterns of positive samples. As a consequence, it suffers from spurious detection. The next problem, even though the data-generating process produces reliable log-likelihood, it does not mean that low log-likelihood is identical to the presence of an anomaly and vice versa. If the inference model of log-likelihood is not well-defined, high log-likelihood may be interpreted as the presence of an anomaly [8], [27]. The low and high log-likelihood of the representational model need to be proven quantitatively as the decision boundary that discriminates the two [33]. Therefore, anomaly detection using a single-sample distributional test is impossible [41]. The existency of positive and negative samples is a must for developing two competing hypotheses [33], [41]. The samples may be derived from ground truth or without. The model will learn from the samples and justified by two competing hypotheses. However, one-class classification is still required to define initial assumptions about the samples before modeling two competing hypotheses.

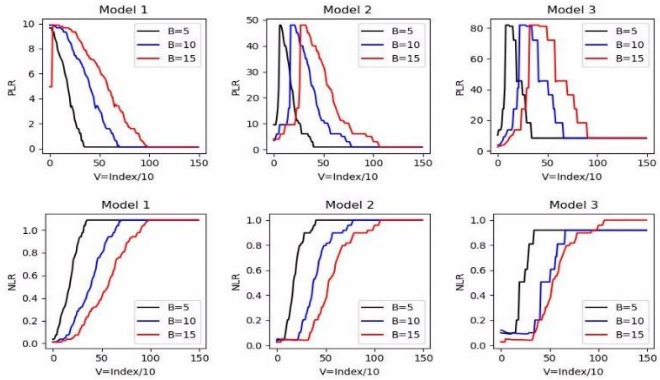


Fig. 8. The bias-variance alignment.

Fig. 8 describes the bias-variance alignment. The bias-variance alignment plays a role in decision boundary optimization to conclude log-likelihood. GMMR as part of data-generating process that produces log-likelihood is not free from the generalization error. The generalization error causes the estimated log-likelihood to shift from the true log-likelihood. This is called misleading and weak log-likelihood. However, the bias-variance tradeoff does not completely explain the phenomenon in a classification setting [7] as follows:

$$(\text{model bias})^2 \approx \text{model variance} \quad (6)$$

Eq. (6) about the phenomenon, which is different from bias-variance tradeoff in general that models of low capacity have high bias but low variance and vice versa. While in Eq. (6) shows the flexibility of bias-variance tradeoff. The model that has high bias does not tend to have low variance or vice versa. Eq. (6) suggests that the bias-variance alignment is specific to large neural networks. Meanwhile, for the small model as in Fig. 8 shows the same results that bias and variance are aligned at a sample level although squared bias does not approximately equal variance, as seen in a full and not-full concave curve of optimal likelihood ratios. It means that the effect of bias-variance tradeoff does not lost at all in a classification setting. Fig. 8 shows that decision boundary in a classification setting has optimized by bias-variance alignment. The relation between decision boundary and bias-variance alignment is as follows:

$$\text{aligned\_logl} = \text{logl-B}\times\mu(\text{Ze})\times\text{logl} \quad (7)$$

$$T = \mu(\text{aligned\_logl})\text{-V}\times\Sigma(\text{aligned\_logl}) \quad (8)$$

$$\text{y\_pred} = \begin{cases} 1, & \text{aligned\_logl} < T \\ 0, & \text{aligned\_logl} \geq T \end{cases} \quad (9)$$

In Eq. (7), aligned_logl has related to a bias factor (B) and the mean of the predictors at testing time μ(Ze). Meanwhile, a decision boundary threshold (T) has related to a variance factor (V), means μ(aligned_logl), and standard deviations Σ(aligned_logl). Eq. (7) shows that log-likelihood needs to be aligned so that it shifts closer to the true log-likelihood, then aligned_logl performs as a parameter in determining the variability of thresholds together with a variance factor.

In Eq. (9), the threshold T creates a decision boundary that classifies log-likelihood into two classes: low log-likelihood if aligned_logl < T and high log-likelihood if aligned_logl ≥ T. Low log-likelihood being labeled positive samples (predicted positive/PP) and high log-likelihood being labeled negative samples (predicted negative/PN). Eq. (9) applies if the bias-variance alignment has achieved the optimal likelihood ratio.

G. Likelihood Ratios

A likelihood (L) is the relative probability of the observed data given a hypothesis parameter value [3]. It refers to the probability or density of a sample under a distribution [41]. It is related to the observed data, statistical models, and statistical hypotheses. It indicates the hypothesis for the goodness of fit of a model to the observed data. With these roles, the data-generating process is designed to produce the likelihood datasets that construct hypothesis. A hypothesis is a prediction for the observed data that can be tested for truth. Hypothesis testing is an important step to ensure that a model being built fits the observed data. Together with the observed data, statistical models, and statistical hypotheses are actual components to be able to draw inferences. The likelihood ratio (LR) is the ratio of two likelihoods for parameter values for two different hypotheses H_1 over H_2 given the observed data x [3] that can be written as follows:

$$LR = L(H_1|x)/L(H_2|x) = P(x|H_1)/P(x|H_2) \quad (10)$$

The likelihood ratio LR represents and measures statistical evidence. The LR is not a probability but a relative measure of evidence for competing two hypotheses. The likelihood of a hypothesis H given the observed data x (L(H|x)) is proportional to the probability of the observed data x under a hypothesis H (P(x|H)). L(H|x) builds a likelihood model. As a consequence, the LR builds a model to represent statistical evidence. Then, the LR model performs as an evidence measure for the observed data. Two different evidences will be measured: evidence from ground truth datasets (y_true) as explained in the subsection III.B and evidence from the bias-variance alignment (y_pred).

In the LR, hypotheses can be easily derived from the model and vice versa. Eq. (7) represents the observed data in a model and (9) is the hypothesis of the model. Eq. (9) formulates the positive hypothesis H_1 (align_logl<T) and the negative hypothesis H_2 (align_logl≥T). In contrast, the hypotheses formulated in (9) together with ground truth datasets build a new model, as seen in Fig. 9(b). The LR model measures statistical evidence by support, denoted S. Statistical evidence for two hypotheses on the graded scale can be seen in Table VI.

TABLE VI. INTERPRETING SUPPORT [3]

LR (1/LR)	Support (log LR)	Interpretation H_1 over H_2
1 (1.00)	0	No evidence
2.7 (0.37)	1	Weak evidence
7.4 (0.14)	2	Moderate evidence
20 (0.05)	3	Strong evidence
55 (0.02)	4	Extremely strong evidence

Table VII shows likelihood intervals or support intervals and their corresponding frequentist confidence intervals (CI) for the standard Gaussian distribution. Similar to confidence intervals, likelihood intervals describe the accuracy and reliability of estimation obtained from the observed data. Likelihood intervals measure the level of confidence in the estimation that expected parameters fall within a certain range. Likelihood intervals are a need to interpret likelihood clearly that the likelihood evidence points are confidence within a certain range.

TABLE VII. LIKELIHOOD INTERVALS [3]

S	LR	1/LR	% CI
1.6	4.95	0.202	92.6
2	7.39	0.135	95.4
3	20.09	0.050	98.6
4	54.60	0.018	99.5
5	148.4	0.007	99.8
6	403.4	0.0025	99.95
7	1096.6	0.0009	99.98
8	2981.0	0.0003	99.99

The LR is about the relative strength of evidence for two competing hypotheses. In contrast, the frequentist approach use of type I and type II errors allows one to specify the probability of rejecting the null hypothesis when it is true, and of not rejecting it when it is false [3]. The LR was built to test two hypotheses: predicted label as an outcome and true label as a reference standard. The two hypotheses compete to measure the relative strength of evidence.

In this paper, a competition is performed by considering ground truth datasets as a reference standard to test the hypothesis in (9) as predicted label. It will produce the outcome of a test: (1) y_pred=1 for a positive label, (2) y_pred=0 for a negative label, and (3) the relative strength of evidence for the observed data. Therefore, the outcome of a test by the LR is not to reject or not reject the null hypothesis (H_0 or H_2). If the relative strength of evidence is not misleading and not weak, the hypothesis may be accepted.

The 2×2 contingency table describes a matrix format used to display a frequency distribution of two variables with the vertical columns denoting instances of reference standard, or true label and the horizontal rows denoting instances of outcome of a test, or predicted label [24]. The relative strength of evidence is measured using the 2×2 contingency table as seen in Fig. 9.

	align_logl<T	align_logl≥T
y_pred=1	TP	FP
y_pred=0	FN	TN

a. The 2×2 contingency table to evaluate the performance indicator of a model

	y_true=1	y_true=0
y_pred=1	TP	FP
y_pred=0	FN	TN

b. The 2×2 contingency table to assess the relative strength indicator of evidence

Fig. 9. The 2×2 contingency table for binary classification.

Table VIII defines a measure factor in the 2×2 contingency table and its derivative. The limitation in the evidential approach is that no observed values are zero [3]. The generalization error controlled by the bias-variance alignment is only a reducible error part not all error conditions [19], [22]. A measure factor of TPR, TNR, FPR, and FNR is an error-based measure. A measure factor, such as TPR or TNR, has a test's ability to detect correctly a condition when it is present and to rule it out correctly when it is absent. A measure factor, such as FPR or FNR, has a test's propensity to detect incorrectly a condition when it is absent and not to detect it correctly when it is present [24]. Therefore, a measure factor of TP, TN, FP, and FN is considered to be not zero.

PLR measures the change in pre-test to post-test odds and diagnostic gain. It also combines information about TPR and FPR [24]. If a sample from population tested positive, PLR represents the relative strength of evidence that a sample is an anomaly given a positive test result. However, the relative evidence of PLR also depends on the explicit prior in the Bayesian approach which is not discussed in this paper. To reduce uncertainty due to factors that influence it, PLR utilizes information from NLR.

Ideally the PLR value is 1/NLR [3], but in practice, the PLR value is sometimes greater than 1/NLR so that the relative strength of evidence for the observed data is less properly. It is necessary to combine PLR and NLR for a single test value as the performance indicator of a model and the relative strength indicator of evidence. Basically, PLR and NLR are the ratio of paired measures [24]. Hence the approach taken is based on the harmonic mean method [43] to combine PLR and NLR that gets a new measure factor, namely, the harmonic likelihood ratio (HLR):

$$HLR = 2\times PLR/(1+(PLR\times NLR)) \quad (11)$$

TABLE VIII. A MEASURE FACTOR IN THE 2×2 CONTINGENCY TABLE AND ITS DERIVATIVE

Term	Denoted	Formula	Description
True positive	TP	TP > 0	Correct predictions of anomaly
True negative	TN	TN > 0	Correct predictions of normal
False positive	FP	FP > 0	Incorrect predictions of anomaly
False negative	FN	FN > 0	Incorrect predictions of normal
True positive rate	TPR	TP/(TP+FN)	The rate of an anomaly sample tested positive
True negative rate	TNR	TN/(TN+FP)	The rate of a normal sample tested negative
False positive rate	FPR	1-TNR	The rate of a normal sample tested positive
False negative rate	FNR	1-TPR	The rate of an anomaly sample tested negative
Positive likelihood ratio	PLR	TPR/FPR	The ratio of an anomaly sample tested positive and a normal sample tested positive
Negative likelihood ratio	NLR	FNR/TNR	The ratio of an anomaly sample tested negative and a normal sample tested negative

The idea of a self-supervised binary classifier is to utilize the structure within data from the samples (logl) to build a supervisory signal for the classifier. A supervisory signal comes from the underlying structure of the samples realized in producing two samples using (7), (8), and (9). A supervisory signal first finds the bias B and variance V that give the maximum likelihood ratios of the two samples. For simplicity, the bias B is assumed to be 10 refers to the B value in the description of the bias-variance alignment in Fig. 8 and the variance V is tried to find iteratively within a certain range on the PLR or HLR value is the maximum, denoted _PLR_ and _HLR_. This maximum value indicates the relative strength of evidence for two samples that the classifier has worked to classify the two samples properly.

The performance evaluation of classifiers involves three models, two groups of observation, and five indicators. Classification is designed in two stages. The first stage is about training samples with a one-class classification method for fine-grained negative samples from subdistributions 1, 2, and 3. Three types of training samples were realized with a negative sample size of 10, 50, and 90, respectively. Training samples were taken from ground truth datasets randomly.

Training samples were the initial assumptions about the prior distribution implicitly (the implicit prior). The second stage is about testing samples with a binary classification method for fine-grained negative samples from subdistributions 1, 2, and 3 and positive samples from subdistributions 4, and 5. Ninety-nine types of testing samples were realized with a negative sample size from 1 to 99 and a positive sample size from 99 to 1, respectively. Testing samples were taken from ground truth datasets randomly. The second stage applies (7), (8), and (9).

Table IX shows that 4 out of 5 performance indicators of the group of observation _HLR_ are better than the group of observation _PLR_. This proves that HLR as a single test value of performance indicators has worked well to evaluate the performance of binary classifier even though the absence of the explicit prior. This correspond to the result of the area under the receiver operating characteristic (AUROC). While the area under the precision-recall curve (AUPRC) is almost the same.

TABLE IX. THE METRIC TO EVALUATE THE PERFORMANCE OF BINARY CLASSIFIERS

Model	Model 1		Model 2		Model 3	
Group	_PLR_	_HLR_	_PLR_	_HLR_	_PLR_	_HLR_
V	1.148	0.576	1.556	0.741	1.352	0.683
T	3.815	5.127	3.868	5.114	3.908	5.067
PLR↑	36.128	22.432	36.664	20.053	37.811	22.591
NLR↓	0.207	0.076	0.211	0.082	0.200	0.079
HLR↑	11.608	16.072	11.041	14.805	10.966	15.492
AUROC↑	0.894	0.941	0.895	0.938	0.897	0.938
AUPRC↑	0.922	0.935	0.927	0.931	0.926	0.939

IV. EXPERIMENTAL RESULTS

The study in this paper was conducted using an experimental method. The experiment was conducted in four stages. The first stage was an experiment with raw datasets that produce ground truth datasets: true or to be true positive (TP) samples and true or to be true negative (TN) samples using the K-Means and EM algorithms.

The second stage was an experiment with the LR as the performance indicator of a model (HLR). The experiment uses data-generating process to produce the logl. The bias-variance alignment enhances likelihood by shifting the logl closer to true likelihood thereby producing the LR model. An indication that the logl closer to true likelihood if the LR model is not misleading and not weak refers to Table VI. The LR model constructs two hypotheses: a positive hypothesis (H_1) and a negative hypothesis (H_2). A measure factor HLR facilitates the resolution of two hypotheses to obtain the variance V properly based on the underlying structure of the two samples. The variance V is proper if the HLR has reached its maximum within a certain range of variance. Together with the bias B, the variance V constructs a supervisory signal for the classifier by a threshold T. Referring to (9), the LR model produces predicted datasets: predicted positive (PP) samples and predicted negative (PN) samples. In the LR, two samples present two models and two hypotheses. Otherwise, two models and two hypotheses represent two samples.

The third stage was an experiment with the LR as the relative strength indicator of evidence (S). Evidence comes from ground truth and predicted datasets which construct two hypotheses: a primary hypothesis (H_1) and a null hypothesis (H_0). Once again, a measure factor HLR makes it easier to resolve two hypotheses to get the relative strength of evidence.

The fourth stage was an experiment to build up a self-supervised anomaly detection (AD) approach. The resilience scale (RS) and anomaly score (AS) can be written as follows:

$$RS = 2\times IPN\times S/(IPN+S) \quad (12)$$

$$AS = 4-RS \quad (13)$$

In (12), the support S is the natural logarithm of a measure factor HLR. It assumes the highest relative strength of evidence is 4 that refers to Table VI. Also, the highest index of PN is 4, because PN is in the range of 0 to 100, the index of PN (IPN) is PN/25. So that the IPN and S have the same ratio from 0 to 4, then both variables can be estimated smoothly using a harmonic mean method [43] to derive (12). Eq. (13) assumes the resilience scale and anomaly score are complements to the support S.

The experiments were realized using a Python programming language, a deep learning library Pytorch [30] to implement an artificial neural network, a Python module Scikit-learn [31] to implement a machine learning model, a Python library gmr [11] to implement a mixture of experts, a Python toolbox PyOD [42] for benchmarking anomaly detection methods, and a 2D graphics package Matplotlib [17] for the visualization of observational and experimental results.

TABLE X. THE LOG-LIKELIHOOD SCORE OF EACH SUBDISTRIBUTION

subd	1	2	3	4	5
Zx	-1.238	-1.199	-1.281	-1.372	-1.475
Zxy	-1.217	-1.193	-1.233	-1.278	-1.335
Zxr	5.078	5.104	5.057	4.985	4.921
Zxr, B=5	6.211	5.514	4.726	3.652	2.712
Zxr, B=10	7.344	5.925	4.395	2.318	0.502
Zxr, B=15	8.477	6.335	4.064	0.985	-1.707

Table X presents the log-likelihood score of each subdistribution for some latent variables and the bias factor that affects the log-likelihood score. The latent variable Zx is the latent samples generated by VAEs. The latent variable Zxy is the latent samples generated by VAEs and MDRNNs also known as world models [14]. The latent variable Zxr is the

latent samples generated by VAEs, MDRNNs, and GMMR also called the representational model.

Table X proves that the log-likelihood score of Zx and Zxy is too low, below zero. GMMR can increase the log-likelihood score significantly from Zxy to Zxr. However, the log-likelihood score generated by GMMR is not strong enough to be used as evidence. The log-likelihood scores of one subdistribution and the others are difficult to distinguish as normal and anomaly classes. It shows that the classification of the two samples is not clear. The reason is that the latent samples are generated by the model, while a model is not free from the generalization error.

In Table X, the bias factor B as one of a supervisory signal straightens the weak evidence by shifting the log-likelihood score closer to the true classification of the two samples. A higher B value has implications for a higher log-likelihood score of subdistributions 1, and 2 and a lower log-likelihood score of subdistributions 3, 4, and 5. This simple experiment explains that subdistributions 1, and 2 tend to be samples of normal class and subdistributions 3, 4, and 5 tend to be samples of anomaly class.

Fig. 10 gives three indicators of the cyber resilience model: the predicted negative PN, the support S, and the resilience scale RS. The predicted negative PN about two samples needed in a likelihood ratio model. One sample represents a reference standard used as an example (ground truth datasets). The other sample represents the observed data (predicted datasets). Ground truth and predicted datasets construct a likelihood ratio model. In the context of cyber resilience, the samples that need to be monitored are the predicted negative (PN) as part of the predicted datasets. PN is the total number of elements labeled as belonging to the negative class. PN is true negative (TN) + false negative (FN). True negative (TN) is as part of the ground truth datasets while false negative (FN) indicates incorrect predictions of the negative class. So, FN is the difference between PN and TN. The PN indicator proves the relationship between PN and the number of negative samples N(Zn), TN, FN.

The support S about the relative strength indicator of evidence. Two samples in a likelihood ratio model construct two competing hypotheses. A measure factor HLR makes it easier to resolve two competing hypotheses to get the support S. From the experiment, it is known that the support S is not affected by the number of negative samples linearly in the training and testing phases except for extreme numbers, such as less than 10 or more than 90 if the sample size is 100. As a consequence, the number of negative samples of 10 has met the requirement as training data.

The resilience scale RS about the resilience scale of the service. It is the indicator to measure the behavior of cyber resilience model. The cyber resilience model requires two data to realize (12): the predicted negative (PN) and the support (S). Because RS derives from the PN and S, which refers to Table VI, the resilience scale RS also has an interpretation that refers to Table VI.

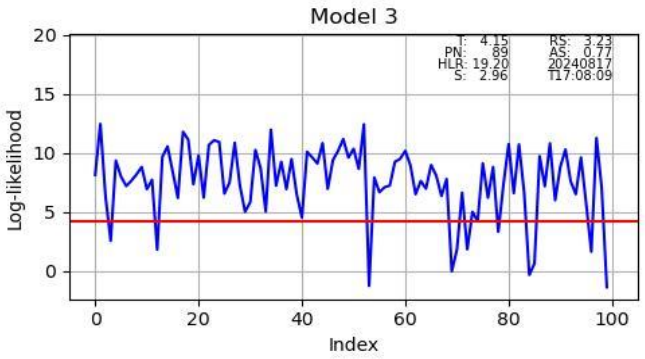


Fig. 10. The cyber resilience model.

Referring to this experimental results, it is known that the cyber resilience model is identical to a likelihood ratio model (LRM). A likelihood ratio model is effectively carried out based on anomaly detection. In anomaly detection, two samples are clearly defined. LRM summarizes complete information about the observed data, statistical models, and statistical hypotheses of DNS events. It identifies the individual and group behavior of each sample shown by the PN and S indicator.

The PN indicator can be used to estimate the status of the data. High PN shows that DNS events are normal and low PN shows otherwise. In addition, LRM provides information about the stability of the model in predicting the observed data. The stability of the model can be seen from the S indicator. A high S shows that the model has the ability to predict the data normally and a low S shows otherwise. Furthermore, the two indicators are used to test the hypothesis of DNS events. A high anomaly score or low resilience scale indicates a DNS anomaly. A low anomaly score or high resilience scale indicates a DNS normal. Table VI will help interpret both in a more understandable form. Therefore, the anomaly score AS is a complement to the resilience scale RS as has been formulated in (13).

Anomaly detection is designed using a specific binary classification task to build a reliable anomaly detection method in dealing with dynamic models. The binary classification task goes through two stages. The first stage is fitting data using regression to obtain the initial value of the relationship between the model and the genuine labeled data (the implicit prior) and the second stage classifies the log-likelihood by a decision boundary threshold to define high and low log-likelihoods. The complete differences between some anomaly detection methods (PyOD) and the tested method (LRM) in the experiment can be seen in Table XI.

TABLE XI. THE DIFFERENCE OF PYOD AND LRM IN THE EXPERIMENT

PyOD	LRM
Input from the log-likelihood score of latent samples (logl) about 100 samples	Input from latent samples (Zxr) about 2×200×100 latent samples
Fitting and predicting in classification	Fitting in regression Predicting in classification
Fitting is done on the negative samples	Fitting is done on the negative latent samples
Predictors are the negative and positive samples	Predictors for regression are the negative latent samples Predictors for classification are the negative and positive samples
Not using latent samples but directly using the log-likelihood score as the observed samples	For each latent sample, the maximum log-likelihood score is taken as the observed samples

Some anomaly detection methods that have been compared: Angle-Based Outlier Detection (ABOD), Isolation Forest (IForest), K-Nearest Neighbors (KNN), and One-Class Support Vector Machines (OCSVM). The methods represent four different anomaly detection methods that have been implemented in PyOD [42]. The results of the benchmarking show that LRM has high performance and stability as an anomaly detection method as seen in Table XII.

TABLE XII. BENCHMARKING THE FOUR METHODS AND LRM

Model	Indicator	ABOD	IForest	KNN	OCSVM	LRM
1	HLR↑	6.003	14.236	14.236	17.626	17.626
	AUROC↑	0.856	0.933	0.933	0.944	0.944
	AUPRC↑	0.984	0.993	0.993	0.994	0.994
2	HLR↑	10.212	13.458	13.458	13.458	32.441
	AUROC↑	0.910	0.930	0.930	0.930	0.970
	AUPRC↑	0.928	0.943	0.943	0.943	0.980
3	HLR↑	10.862	10.124	10.124	9.478	19.203
	AUROC↑	0.961	0.956	0.956	0.950	0.994
	AUPRC↑	0.794	0.778	0.778	0.763	0.955

V. DISCUSSION AND FUTURE WORK

Two samples can be well-modeled with the LR model so that the interpretation of the observed data is more objective, depending only on the data itself. It is relevant to be used for anomaly detection. It is also relevant to be used for normal detection (resilience). Based on a self-supervised anomaly detection approach, self-labeling in a dynamic model can be realized with measurable supervision. Self-labeling directed by a supervisory signal consisting of a bias factor B, a variance factor V, and a threshold T. The evaluation of regression and classification models with results as follows: RMSE=0.0014, R^2=0.9869, MAE=0.0009 for fitting 3 regression models at training time and HLR=15.456, AUROC=0.939, AUPRC=0.935, S=2 (2.738) with the S-2 likelihood interval for 297 (3×99) classification models at testing time. It means that the LR model does not generally produce misleading and weak log-likelihood. In this LR, the relative strength of evidence is defined by a primary hypothesis that models self-labeling and a null hypothesis that models a standard reference of the samples. In the experiment, it has been proven that a primary hypothesis is accepted, S=2 with a 95% confidence level.

The difference index (DI) of likelihood in the EM as a sample classifier is the difference between a subdistribution classified as positive and negative samples. The larger DI, the larger the disjoint support between the distribution of positive and negative samples. In Table III, subdistribution 2 has a DI of 4.004, subdistribution 3 has a DI of 0.944, and subdistribution 4 has a DI of 3.210. It means the difference index of subdistribution 3 is small, then the disjoint support is also small. As a consequence, subdistribution 3 is less reliable as a standard reference of samples.

To address the problem of misleading and weak log-likelihood required aligning bias-variance and optimizing likelihood-ratios. In Table X, the bias factor B has been worked properly for subdistributions 1, 2, 4, and 5 but not for subdistribution 3. This is correlated with the analysis of Table III that the disjoint support of subdistribution 3 is small so subdistribution 3 produces anomaly interpretation: some parts tend to be negative samples and others tend to be positive samples.

Referring to the facts in Table III and X, it is necessary that a null hypothesis may need to be redesign. A null hypothesis represents ground truth datasets as examples of positive and negative samples that declare a reference standard of the observed samples in two competing hypotheses. The next study focused on enhancing the ground truth datasets, especially to address the problem of subdistribution 3 that produces a stronger relative strength indicator of evidence. Furthermore, predictive modeling only relies on examples of positive and negative samples from those trained to the model as the implicit prior.

VI. CONCLUSION

There is a relationship between the cyber resilience and likelihood ratio models. The likelihood ratio model (LRM) is useful in building the cyber resilience model. It meets the requirements needed to realize the representational and inference models in practice. The representational model has been realized with likelihood maximization inside data-generating process and the inference model has been realized with likelihood ratios inside two competing hypotheses. In the representational model, the Gaussian mixture model for multimodal regression (GMMR) enhances the ability of world models produces log-likelihood. The log-likelihood needs to be aligned by the bias-variance factor to be worthy of being used as evidence. In the inference model, evidence from ground truth datasets and evidence from the model build two competing hypotheses to handle anomaly detection tasks in self-supervised settings. Evidence from the model has been labeled through a self-labeling technique that supervised by three supervisory signals: a bias factor, a variance factor, and a threshold to optimize decision boundary in minimizing the generalization error of predictive modeling with the harmonic likelihood ratio (HLR).

ACKNOWLEDGMENT

This study was conducted in collaboration between Universitas Muhammadiyah Malang and Universiti Teknologi Malaysia. The authors fully acknowledge to Rector of Universitas Muhammadiyah Malang for the research grant (E.5.c/1693/UMM/XII/2020) that helps in funding the research works. Special thanks to all the reviewers for their valuable feedback.

REFERENCES

[1] C.M. Bishop, "Novelty detection and neural network validation," In: S. Gielen, B. Kappen (eds), ICANN '93, Springer, 1993.

[2] C.M. Bishop, Mixture density networks, Technical Report, Aston University, 1994.

[3] P.M.B. Cahusac, Evidence-based statistics: An introduction to the evidential approach-from likelihood principle to statistical practice, John Wiley & Sons, 2020.

[4] E.B. Cahyono, S.M. Sam, N.H. Hassan, N. Mohamed, N.B. Ahmad, and Y.M. Yusuf, "A review on cyber resilience model in small and medium enterprises," 4th International Conference on Smart Sensors and Application (ICSSA), pp. 114-119, 2022.

[5] J. F. Carías, M. Borges, L. Labaka, S. Arrizabalaga, and J. Hernantes, "Systematic approach to cyber resilience operationalization in smes," IEEE Access, vol. 8, pp. 174200-174221, 2020.

[6] J. F. Carías, S. Arrizabalaga, L. Labaka, and J. Hernantes, "Cyber resilience progression model," Applied Sciences, vol. 10, iss. 21, pp. 7393, 2020.

[7] L. Chen, M. Lukasik, W. Jitkrittum, C. You, and S. Kumar, "On bias-variance alignment in deep models," International Conference on Learning Representations, 2024.

[8] H. Choi, E. Jang, and A.A. Alemi, "WAIC, but why? Generative ensembles for robust anomaly detection," arXiv, abs/1810.01392, 2018.

[9] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the em," Journal of the Royal Statistical Society, Series B (Methodological), vol. 39, no. 1, pp. 1-38, 1977.

[10] K.O. Ellefsen, C.P. Martin, and J. Tørresen, "How do mixture density rnns predict the future?," ArXiv, abs/1901.07859, 2019.

[11] A. Fabisch, "gmr: Gaussian mixture regression," Journal of Open Source Software, vol. 6, no. 62, pp. 3054-3057, 2021.

[12] Z. Ghahramani, and M.I. Jordan, "Supervised learning from incomplete data via an em approach," Neural Information Processing Systems, 1993.

[13] A. Graves, "Generating sequences with recurrent neural networks," ArXiv, abs/1308.0850, 2013.

[14] D.R. Ha, and J. Schmidhuber, "World models," ArXiv, abs/1803.10122, 2018.

[15] S. Hochreiter, and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, pp. 1735-1780, 1997.

[16] H. Hojjati, T.K. Ho, and N. Armanfard, "Self-supervised anomaly detection: A survey and outlook," ArXiv, abs/2205.05173, 2022.

[17] J.D. Hunter, "Matplotlib: A 2D graphics environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

[18] R.A. Jacobs, "Bias/variance analyses of mixtures-of-experts architectures," Neural Computation, vol. 9, pp. 369-383, 1997.

[19] G.M. James, D.M. Witten, T.J. Hastie, and R. Tibshirani, An introduction to statistical learning with applications in r, Second Edition, Springer Texts in Statistics, 2021.

[20] M.I. Jordan, and R.A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," Neural Computation, vol. 6, pp. 181-214, 1993.

[21] D.P. Kingma, and M. Welling, "Auto-encoding variational Bayes," ArXiv, abs/1312.6114, 2013.

[22] M. Kuhn, and K. Johnson, Applied predictive modeling, Springer, 2013.

[23] C.L. Lan, and L. Dinh, "Perfect density models cannot guarantee anomaly detection," Entropy, vol. 23, no. 12, 2020.

[24] A. J. Larner, The 2x2 matrix: Contingency, confusion and the metrics of binary classification, Second Edition, Springer, 2024.

[25] O. Makansi, E. Ilg, O. Çiçek, and T. Brox, "Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction," IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7137-7146, 2019.

[26] P. Martin, GoPassiveDNS: Network-based dns logging in go, GitHub repository, https://github.com/Phillipmartin/gopassivedns, 2016.

[27] E.T. Nalisnick, A. Matsukawa, Y.W. Teh, D. Görür, and B. Lakshminarayanan, "Do deep generative models know what they don't know?," International Conference on Learning Representations, 2019.

[28] G. Pang, C. Shen, L. Cao, and A. Van Den Hengel, "Deep learning for anomaly detection: A review," ACM Computing Surveys, vol. 54, no. 2, pp. 1-38, Research Collection School Of Computing and Information Systems, 2022.

[29] J. Park, J. Moon, N. Ahn, and K. Sohn, "What is wrong with one-class anomaly detection?," ICLR 2021 Workshop on Security and Safety in Machine Learning Systems, 2021.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 33rd Conference on Neural Information Processing Systems, 2019.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, G. Louppe, P. Prettenhofer, R. Weiss, R.J. Weiss, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python", Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[32] P. Poklukar, "Seeing the whole picture instead of a single point: Self-supervised likelihood learning for deep generative models," 2nd Symposium on Advances in Approximate Bayesian Inference, 2019.

[33] J. Ren, P.J. Liu, E. Fertig, J. Snoek, R. Poplin, M.A. DePristo, J.V. Dillon, and B. Lakshminarayanan, "Likelihood ratios for out-of-distribution detection," 33rd Conference on Neural Information Processing Systems, 2019.

[34] D.J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," International Conference on Machine Learning, 2014.

[35] L. Ruff, J.R. Kauffmann, R.A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T.G. Dietterich, and K. Muller, "A unifying review of deep and shallow anomaly detection," Proceedings of the IEEE, vol. 109, no. 5, pp. 756-795, 2020.

[36] F. Stulp, and O. Sigaud, "Many regression algorithms, one unified model: A review," Neural networks: the official journal of the International Neural Network Society, vol. 69, pp. 60-79, 2015.

[37] H.G. Sung, Gaussian mixture regression and classification, PhD thesis, Rice University, 2014, unpublished,

[38] M. Suzuki, and Y. Matsuo, "A survey of multimodal deep generative models," Advanced Robotics, vol. 36, pp. 261-278, 2022.

[39] F. Weimer, "Passive dns replication," 17th annual FIRST conference on computer security incident, 2005.

[40] S.E. Yüksel, J.N. Wilson, and P.D. Gader, "Twenty years of mixture of experts," IEEE Transactions on Neural Networks and Learning Systems, vol. 23, pp. 1177-1193, 2012.

[41] L.H. Zhang, M. Goldstein, and R. Ranganath, "Understanding failures in out-of-distribution detection with deep generative models," Proceedings of machine learning research, vol. 139, pp. 12427-12436, 2021.

[42] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," Journal of Machine Learning Research, vol. 20, pp. 1-7, 2019.

[43] D. Ziou, "Pythagorean centrality for data selection," ArXiv, abs/2301.10010, 2023.