1-1-2021

Variational Autoencoders and Wasserstein Generative Adversarial Networks for Improving the Anti-Money Laundering Process

ZhiYuan Chen
University of Nottingham Malaysia

Waleed Soliman
University of Nottingham Malaysia

Amril Nazir
Zayed University

Mohammad Shorfuzzaman
Taif University

Recommended Citation

Chen, ZhiYuan; Soliman, Waleed; Nazir, Amril; and Shorfuzzaman, Mohammad, "Variational Autoencoders and Wasserstein Generative Adversarial Networks for Improving the Anti-Money Laundering Process" (2021). *All Works*. 4305.
https://zuscholars.zu.ac.ae/works/4305

IEEE Access
Multidisciplinary ⁞ Rapid Review ⁞ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Variational Autoencoders and Wasserstein Generative Adversarial Networks for Improving the Anti-Money Laundering Process

ZhiYuan Chen[1], Member, IEEE, Waleed Soliman[1], Amril Nazir[2], and Mohammad Shorfuzzaman[3], Member, IEEE

[1]School of Computer Science, University of Nottingham Malaysia, Malaysia.
[2]Department of Information Systems, College of Technological Innovation, Abu Dhabi Campus, Zayed University, Abu Dhabi, United Arab Emirates
[3]Department of Computer Science, College of Computers and Information Technology, Taif University, Taif 21944, Saudi Arabia

Corresponding author: ZhiYuan Chen (e-mail: Zhiyuan.Chen@nottingham.edu.my).

This work was supported in part by the Taif University Researchers Supporting Project number (TURSP-2020/79), Taif University, Taif, Saudi Arabia

ABSTRACT There has been much recent work on fraud and Anti Money Laundering (AML) detection using machine learning techniques. However, most algorithms are based on supervised techniques. Studies show that supervised techniques often have the limitation of not adapting well to new irregular fraud patterns when the dataset is highly imbalanced. Instead, unsupervised learning can have a better capability to find anomalous and irregular patterns in new transaction. Despite this, unsupervised techniques also have the disadvantage of not being able to give state-of-the-art detection results. We propose a suite of unsupervised and deep learning techniques to implement an anti-money laundering and fraud detection system to resolve this limitation. The system leverages three deep learning models: autoencoder (AE), variational autoencoder (VAE), and a generative adversarial network. We preprocess the given dataset to separate the Transaction Date attribute into its base components to capture time-related fraud patterns. Also, Wasserstein Generative Adversarial Network (WGAN) is used to generate fraud transactions, which are then mixed with the base dataset to form a more balanced mixed dataset. These two datasets are used to train the AE and VAE models. We built two versions of the AE model (single-loss and multi-loss) besides a novel method of calculating the anomaly score threshold, called Recall-First Threshold (RFT), which helps enhance the model's performance. Experimental results demonstrated that the False Positive Rate (FPR) drops down to as low as 7% in the proposed multi-loss AE model. In comparison, we achieved an accuracy of 93%, with 100% of the fraud transactions recalled successfully.

INDEX TERMS Anti-money laundering (AML), Autoencoders, Anomaly detection, Deep learning, Fraud detection, GANs, Unsupervised learning.

I. INTRODUCTION

Money laundering involves concealing or disguising the origin of illegal profits that have been generated from criminal acts [1]. Banking products or services can be exploited to transfer criminal proceeds for terrorist financing and money laundering. These institutions become a direct or indirect victim of money laundering activity, which undermines the integrity of the financial system [2]. In light of this, the pressure on financial institutions and banks to improve their measures to fight money laundering is increasing. Similarly, central banks and finance-related laws have become stricter towards money laundering crimes such that banks need to follow specific rules; otherwise, they could be penalized or even closed [3]. One recent case includes the largest bank in Italy, Unicredit, which was fined $1.3 billion for using the US financial system to launder about $6.76 billion [4]. In another case, the UK-based banking giant, Standard Chartered, paid more than $1 billion in fines and settlements for helping in money laundering [5]. Lastly, as a result of compliance failures in the firm's anti-money laundering program, Morgan Stanley was fined $10 million [6].

Nevertheless, most banks still adopt systems that comprise a set of predefined if-then-else rules called "Rule-based systems" to detect incoming and outgoing suspicious transactions. This system requires a manual process of checking for each transaction that has triggered the static rules. Human experts define rule-based systems; hence, they embed their own working experience into the automated decision process. In future updates, more exceptions and rules are necassary, which may impair system performance. Additionally, those systems have a minimal ability to detect suspicious transactions by groups of people across different economic activities. This is because rule-based systems do not consider the economic activities of different people. Furthermore, acquiring economic knowledge about different groups of people can be tedious work on its own [7, 8].

While banks and financial institutions seek cost-effective means of complying with regulatory requirements, they face responsibility for evaluating larger, more complex, and faster-growing datasets, necessitating more powerful analytical tools to efficiently monitor the financial sector. Machine learning algorithms enable cheaper and more accessible tools that are increasingly powerful as they make sophisticated real-time insights on larger datasets possible. These algorithms and tools can be used in the anti-money laundering process by the anticipation and detection of fraud and suspicious transactions [9]. However, adopting machine learning to detect money laundering has long been in research, using different methods and techniques that will be covered in detail in the literature review section.

The current performance of machine learning techniques in the anti-money laundering field is acceptable. However, a lot of work is still required to enhance and optimize those models in terms of performance, namely the so-called "false-positive rate," which indicates the regular transactions that have been identified as fraud. The system will decline these transactions or delay them for further investigation. In some cases, false positives might be costing vendors much more than the fraud transactions themselves. It has been reported that even rule-based systems still struggle with about 20% false-positive rates wherein only 1 in 5 transactions marked by the system as fraud is genuinely fraud [10].

Our main contributions can be summarized as follows:

1. We design and implement deep learning models with promising results in terms of the FPR, RFT, and AUC for fraud detection.
2. We present recent state-of-the-art deep learning and unsupervised learning techniques, namely, the autoencoder (AE), variational autoencoder (VAE), and generative adversarial network (GAN) to improve the anti-money laundering (AML) process.
3. For the first time, we demonstrate the applicability and effectiveness of combining AE/VAE with WGAN methods. Particularly, the WGAN generates realistic synthetic fraud transactions to solve the issue of imbalanced class labels, and such additional transactions are then used by the AE/VAE to train the model. The results indicate that this approach offers significant improvements for fraud detection.

The rest of the article is organized as follows. Section 2 presents the related literature. Various deep learning architectures used in this study are described in Section 3. Proposed methodology and experimental results are presented in Sections 4 and 5, respectively. Finally, Section 6 concludes the article with a discussion of future work.

II. RELATED WORK

Decision Trees (DTs) are one of the common supervised learning algorithms that are used to identify money laundering cases. Rojas et al. [11] utilized DTs and Decision rules by selecting Random Forest (RF), Random-Tree, and J48graft from the DT algorithms group and decision table JRip from the Decision rules algorithms. MABS (Multi-Agent-Based Simulation) was used to generate synthetic data that simulates mobile money transactions. JRip generated about 0.999 true positives and only 0.012 false positives, which was one of the best accuracies obtained. Despite this accuracy, the research was based on synthetic data that may not reflect real suspicious case situations. The accuracy results may differ when used on real transaction data.

Sahin and Duman [12] proposed DT models such as C5.0, CART, and CHAID combined with SVM (Support Vector Machine), which utilizes various kernel functions, such as radial basis, linear, polynomial, and sigmoid. The proposed model was implemented in a credit card fraud detection system. These classification models were compared using a real dataset provided by a bank. However, due to the highly imbalanced records (i.e., a ratio of 20,000 normal transactions to 1 suspicious transaction), the author performed stratified sampling to under-sample the normal transactions. The result presented in the paper shows that both CART and C5.0 have the highest accuracy of detecting suspicious transactions at more than 90%. However, the research did not evaluate the false positive rate; furthermore, SVM offers 89% accuracy, but the author indicated that SVM tends to suffer from over-fitting.

Bitmap Index-based DT (BIDT) algorithm was implemented by Jayasree and Balan [13] to evaluate the adaptability risk for money laundering. Results of false positive and true positive rates, alongside the adaptability rate and risk identification time, showed that the proposed approach outperformed other methods. Also, the authors in [14] used DT to assign a risk score to each customer profile that represents their tendency to perform money laundering, using four types of attributes: industry, location, business size, and product type to build the decision tree. Each attribute, including the class label, can accept three risk values (high, middle, low). However, changes in the predefined risk values will cause the decision tree model to become inaccurate. Moreover, each type of attribute value must be assigned with a risk rank, and this will require domain experts to label those attributes correctly. Otherwise, any changes to the training set will require the decision tree to be trained again.

Recently, SVM [15] and ANN (artificial neural networks) [16] were used and compared against RF and other algorithms. Experimental results show that ANN performed better when compared to other algorithms. Radial-Basis function network (RBFN) is another approach that is used to examine suspicious transactions. Lin-Tao et al. [17] proposed an updated version of RBFN utilizing the APC-III algorithm to optimize parameter learning in the hidden layer. Additionally, RLS (Recursive Least Square) algorithm was introduced to improve model convergence. A real bank dataset containing 70 suspicious instances was used to train the network. The experiment resulted in a low false-positive rate, close to 0%, and a detection rate higher than 80%. Although this implementation shows an excellent false positive rate, the model's accuracy can further be enhanced.

Benford's Law and machine learning algorithms (ANN, DT, RF) were used to investigate money laundering patterns in real Spanish court cases [18]. The authors used Benford's law to map accounting records for each supplier to 21-dimensional space. Results showed that even more companies could be marked as a risk, but this approach still required a domain expert in accounting to do the feature engineering. Chouiekh and Haj [19] proposed a deep convolution neural network (DCNN) to detect fraud cases and obtained results outperforming the traditional machine learning techniques such as SVM and RF.

Due to the lack of genuinely suspicious transaction data and the sensitivity of these data, many researchers have resolved to use synthetic data or simulated data in the training set to reduce the class imbalance issue. However, such an approach may not truly reflect real-world money laundering cases, potentially causing a generalization issue. Supervised techniques require a domain expert to label the data and to help in feature engineering. Therefore, more researchers have recently turned to unsupervised learning methods to deal with the money laundering implementation problem.

Zhang et al. [20] utilized a clustering algorithm to detect money laundering. The authors extracted all the suspicious individuals (n) related to suspicious cases identified by an investigator. Then the author assembled the transactions that those individuals made in n+2 dimensional Euclidean space, where time represents the first dimension and transactions represent the second dimension. Then, to reduce the clustering problem, the timeline was discretized into various time instances. By doing so, each transaction is viewed as a node in one-dimension time-space. To make the problem even more straightforward, the transaction frequency or the money amount was accumulated in each timeline instance. Finally, the histogram segmentation was conducted using a k-means algorithm where each segmented histogram represents a single cluster k. The abnormal hills in the histogram are used to identify suspicious cases. Using only the transactions data, the proposed method managed to match the different transactions with their peers without other features, such as occupation or business size. However, the segmented histograms are only limited to transactions that occurred on the same time instance. The histograms are not able to uncover activities of money laundering that may occur through multiple time instances. Capturing those time instances can be a difficult task in such an approach.

Lune et al. [21] used the K-Nearest Neighbor (k-NN) approach, which has shown a good performance. A public domain dataset was used that was generated from a BTS (Banking Transaction Simulator) to simulate shell companies' behavior. These are companies that seem to be genuine, while their primary objective is to launder money. The author assigned an anomaly score for each data point called LOF (Local Outlier Factor), which is the data point's ratio and its average density of the k-NNs. This approach assumes that an outlier would be significantly lower than its nearest neighbors while the genuine data point would have a similar density. Finally, they set the LOF threshold to 0.9, which will mark all data points above it as a shell company. The problem with this approach is the sensitivity to the outliers, where it can cause variation in density for the data points.

Claudio and Balsa [22] chose to use numerical and nominal attributes in K-means cluster development despite K-means performance on nominal attributes being inefficient in its use of squared Euclidian-distance to calculate proximity. However, the data were clustered by customer attributes to build a customer profiles table, and then the PART algorithm was used for rule generation. The initial 3 month period produced unsatisfactory results. After expanding the client profiles to cover one year and including more attributes, the algorithm showed a better result. Nevertheless, the authors did not mention how they deal with the imbalanced data as k-NN does not perform well on an unbalanced dataset. Another research [23] tried to produce clusters that are more understandable. The authors attempted to add a meaningful description before clustering by following the Apriori and LINGO algorithm implementations to identify fraud in credit card transactions. Following this, they compared the results from both algorithms with other clustering algorithms such as k-NN. Using simulated test transactions, their results showed that the LINGO algorithm quickly generated more meaningful patterns that can be used in near real-time transactions.

Using one-class SVM, Tang and Yin [24] proposed another unsupervised approach to recognize normal and suspicious human transaction behaviors. An improved RBF kernel-based function was implemented over 1.2 million records obtained from Wuhan Agriculture Bank, China, with 30 simulated suspicious transactions. Results showed that the proposed RBF kernel enhanced the algorithm speed and accuracy. However, the proposed solution has only 69.13% accuracy in detecting doubtful cases, which may indicate impracticality when applied in the real world. Furthermore, the suspicious cases are synthetic records that may not fully reflect real suspicious cases.

Recent research [25] tried to avoid the sensitivity of OC-SVM (one-class SVM) for the noise and outliers existence in the dataset by introducing a sparse and robust methodology of fraud detection. The authors introduced the Ramp-loss function to the original OC-SVM. Hence, they called it Ramp-OCSVM. The advantage of implementing the ramp-loss function's non-convexity nature and the concave-convex procedure was the proposed algorithm's ability to solve non-convex, non-differentiable optimization problems. When they compared the proposed approach against other methods, such as OC-SVM and ROCSVM, the results showed that their system presented the best performance within an acceptable false-positive rate. Another research [26] proposed a special case that tried to overcome the OC-SVM shortcoming of ignoring the training data's inner-class structure. The proposed method attempted to minimize the scatteredness of the training points; hence, the points can be easily separated from the origin. The modified version is called OC-WCSSVM (within-class scatter OC-SVM), a typical OC-SVM except that it's $\beta = 0$. The result showed that the proposed method is more accurate for anomaly detection than other approaches such as PCA and Geometrical Driven Diagnosis (GDD).

Wilson and Martinez [27] proposed the usage of an improved RBF (Radial Basis Function) kernel-based function that uses various distance metric functions. They introduced three distance functions: HVDM (Heterogeneous Value Difference Metric), IVDM (Interpolated Value Difference Metric), and WVDM (Windowed Value Difference Metric). These functions can be used with k-NN for a wide range of implementations. Results showed that WVDM and IVDM produced higher accuracy than HVDMs.

Chitra and Subashini [28] estimated the proportion for each bank customer using EM (Expectation Maximization) algorithm. They used the probability density function Gaussian-Mixture Model to model the previous transaction's behavior for each bank customer and compare them against the current transaction's behavior. The main issue with this method is that it requires the assumption that statistical distribution (i.e., Gaussian distribution) of the dataset is used. Furthermore, for the EM algorithm to work in the first place, we need to define the number of clusters required and estimate and maximize the different clusters' data points. For instance, in the two clusters experiment, the EM algorithm assumed that every single cluster represents a different Gaussian distribution with its own function parameters.

Cao and Do [29] attempted to attack money by moving money in a circular pattern between accounts. They used the CLOPE (clustering with sLOPE) algorithm to check small amounts of money distributed to various recipients. It also checks a single account for collecting money from different senders. Moreover, the CLOPE algorithm's main characteristic is the acceptance of nominal variables. Hence, continuous variables such as the transaction-amount need to be discretized and assigned to a meaningful label. The research used a dataset consisting of 12,350 normal records from an unspecified bank to measure CLOPE's performance in detecting money laundering. Furthermore, 25 simulated suspicious records were inserted into the dataset to test the algorithm. The experimental result showed that the detection rate was about 100%, with only 25% of the false-positive rate. Despite this, each cluster produced by CLOPE must be thoroughly examined to determine which cluster belongs to which type of money laundering case, which would require intervention from domain experts. Furthermore, data discretization requires a user to provide the number of bins, and the author did not mention which method they used to get the optimal bin number.

Zaslavsky and Strizhak [30] employed SOM (Self-organizing map) to detect credit card fraud transactions. Specifically, the authors used SOM to create a customer behavior model on credit card transactions. The idea behind the proposed model is to detect suspicious transactions when a customer deviates from his usual transaction behavior. In this approach, two profiles are created from the SOM algorithm, namely the normal behavior model and the fraudster behavior model, Each incoming transaction is then compared with both models, and, subsequently, the transaction similarity score is calculated for both models. The issue here is that a predefined threshold must be set to compare it against the similarity score. Also, to keep the models updated, newly encountered behaviors (i.e., either suspicious or normal) are used to re-train both models. This, in turn, may cause over-fitting for these models. Another research [31] proposed an improved version of SOM to overcome the large presence of outliers in the dataset. The author then compared his proposed method result against the K-prototypes algorithm. The research concluded that the improved SOM is better than the K-prototypes algorithm as it gives better results, especially in handling the outliers. However, interpreting results from SOM is a complicated process as it is not transparent.

To identify suspicious transactions, the authors in [32] proposed a sequence-matching algorithm. The idea of this algorithm is to extract a sequence of daily transactions within a certain peer group. Then, using a probabilistic model, it identifies the high-risk sequence within the extracted sequence. Later these sequences are compared against the transaction's history for each account. Each high-risk sequence is given a similarity score by implementing Euclidean similarity distance. Those assigned scores are then separated based on manual threshold scores to extract the suspicious sequence. However, having a predefined threshold is not an optimal solution as it may vary between different accounts. Moreover, in the real world, the number of suspicious sequences is unidentified. Hence, having a high threshold value might lead to a low false positive rate and might miss some suspicious transactions. Alternatively, having a low threshold might increase compliance officers' workload to verify each case and increase the false positive rate.

Another study [33] leveraged the semi-supervised learning approach, which uses both supervised and unsupervised algorithms. The proposed framework used artificial neural network (ANN) and k-NN clustering to investigate money laundering in an investment bank. The framework first consolidated the transactions on a monthly, weekly, and daily basis. By performing k-NN clustering over these transactions to locate the suspicious transactions, each suspicious transaction was labeled "suspicious" while others are marked as "normal." Following this, the ANN is trained using these labeled transactions to generate the model. To obtain enough suspicious transactions, the author used a genetic algorithm to generate more synthetic suspicious transactions like those that were detected from the k-NN clustering process. Once the training phase is done using these suspicious transactions, the trained model is then used to test any new transaction to determine whether it is suspicious or not. However, this approach still requires a domain expert to identify and label the suspicious transactions during the clustering phase. A heuristic approach is used to define the number of clusters during the clustering phase.

Shabat et al. [34] proposed two algorithms: geometry-based extraction, called Diffusion Maps (DM), and matrix decomposition. They deal with high-dimensional big data (HDBD), which is critical in cybersecurity. The result showed that the proposed approach could outperform the nearest neighbor-based (k-NN) and the clustering-based (uCBLOF) algorithms. However, a massive dataset is required for this approach to be efficient.

To identify the relationship between different accounts involved in the money laundering process, Shaikh and Nazir [35] implemented clustering using social networks analysis (SNA) that determines specific relations among illegal transactions and suspicious customers. However, the authors used fixed conditions and criteria to identify various types of relationships, which may not be ideal for generalization. Therefore, these conditions will need to be modified and updated for each geo-social zone.

Colladon and Remondi [36] proposed a similar approach to build a risk profile by using multiple networks during the experiment. However, they focused only on factories and the business sector, which may lead to less generalization when applied to personal bank networks. Also, they neglect certain features from their analysis, such as the size and the age of the firms. Another related approach proposed by Molloy et al. [37] used graph analytic and BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies). The proposed method used the SCC (Strongly Connected Component) to reduce the false-positives and efficiently identify suspicious transactions. SCC theory assumes that transactions within an SCC are less likely to be fraudulent than the transactions that span two SCCs. Although the proposed method showed good discrimination between normal transactions and suspicious ones, the implementation still requires high computational cost.

A powerful unsupervised deep learning approach was recently proposed based on variational autoencoders (VAE) for anomaly detection [38]. The VAE's main advantage over PCA and the standard autoencoder is that it delivers a probability measure as an anomaly score rather than a reconstruction error. The result showed that the proposed method performed better than PCA and standard autoencoder-based methods. Furthermore, given its generative nature, analyzing the anomaly's underlying cause is also possible through data reconstruction. However, reconstruction probability still requires a fixed threshold, and it can be easily affected by outliers. Furthermore, it still needs to be validated against real money laundering cases. In another similar effort [39], an autoencoder-based data augmentation technique was presented for unsupervised anomaly detection. Babaei et al. [40] proposed a prune-based outlier factor (PLOF) approach for the detection of point outliers which can significantly reduce the execution time of local outlier factor (LOF) while maintaining performance.

Another research [41] proposed unary classification with deep autoencoder, which used the OCC (One Class Classification) to identify only one class among all data objects. Results showed better accuracy and performance over the other traditional machine learning algorithms. However, as it is only one class, it is hard to identify the attribute that contributes the most to the separation of positive and negative classes.

Pumsirirat and Yan [42] used the Restricted Boltzmann Machine (RBM) and autoencoders to detect credit card fraud. By using RBM, the model can reconstruct the normal transactions to locate fraud. Having both algorithms enabled them to investigate the real-time transactions, the experiments were conducted over three datasets from Australia, Germany, and Europe. The results showed a low false-positive rate besides a good performance.

Paula et al. [43] used autoencoders to investigate fraud and money laundering in Brazilian exports. The authors used a dataset containing 820 thousand records and conducted the experiments using PCA and autoencoders. Results showed that autoencoder could detect fraud even with high latent dimensions while PCA could not achieve the same effect.

In conclusion, clustering approaches are simple but still require a domain expert to determine the number of clusters and analyze each cluster's members to determine the suspicious ones. However, clustering algorithms focus on grouping similar transactions based on each transaction's characteristics, so the imbalance dataset issue does not heavily impact it (e.g., a ratio of 20,000 normal transactions to 1 suspicious transaction). Additionally, recent advances in deep learning techniques such as autoencoders and their promising results in anomaly detection make it an excellent candidate for implementation in this research.

III. DEEP LEARNING MODELS

In this section, we describe autoencoders (AEs), Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), and Wasserstein GANs (WGANs). Autoencoders are an unsupervised learning method that is mainly used for feature extraction. They use a feedforward, non-recurrent neural network to perform representation learning. An autoencoder will learn the representation or code by trying to copy the input to output. However, using an autoencoder is not as simple as copying the input to output; otherwise, the neural network would not uncover the hidden structure in the input distribution. An autoencoder will encode the input distribution into a low-dimensional tensor, which usually takes the form of a vector. This will approximate the hidden structure that is commonly referred to as the latent representation, code, or vector. This process constitutes the encoding part. The decoder part will then decode the latent vector to recover the original input. As a result of the latent vector being a low-dimensional, compressed representation of the input distribution, it should be expected that the output recovered by the decoder can

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3086359, IEEE Access

only approximate the input. The dissimilarity between the input and the output can be measured by a loss function.

A. AUTOENCODERS (AEs)

An autoencoder consists of input, hidden (or bottleneck), and output layers. Although it is a single network, as Figure 1 shows, it is a virtual composition of two components [44]:

- Encoder: This transforms the input (*x*) into a low-dimensional latent vector bottleneck, *z* =*f* (*x*). Since the latent vector is of low dimension, the encoder is forced to learn only the most important features of the input data.
- Decoder: This tries to recover the input from the latent vector *g*(*z*)=*x*'. Although the latent vector has a low dimension, it has a sufficient size (*m* < *n*) to allow the decoder to recover the input data. Simultaneously, it restricts the encoder function to approximate *x* so that it is forced to learn only the most salient properties of *x* without copying it exactly.

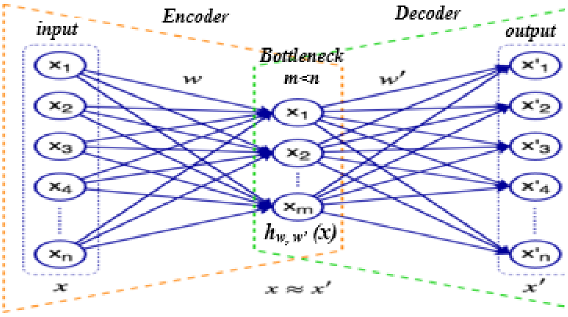


FIGURE 1. **Representation of an autoencoder**.

The autoencoder can be trained by minimizing the loss function known as the reconstruction error, *L*=(*x*, *x*'). It measures the distance between the original input and its reconstruction. It can be minimized in the usual way with gradient descent and backpropagation. Popular loss functions such as mean square error (MSE) or binary cross-entropy (like cross-entropy, but with only two classes) can be used as reconstruction errors, as in equation (1).

$$L(x,x') = MSE = \frac{1}{m}\sum_{i=1}^{m}(x_i - x_i') \qquad (1)$$

The reconstruction error in the equation above is used as an anomaly score for the autoencoders' fraud detection implementation, as will be explained later in the methodology.

B. VARIATIONAL AUTOENCODERS (VAEs)

By architecture, AEs tend to memorize the input, especially if the dimension of the latent code is significantly bigger than the number of features. To encourage the model to generalize better, various techniques can be used, such as Denoising AEs, Sparse AEs, or VAEs.

VAEs are the stochastic version of AEs as they can describe the latent representation in probabilistic terms [45]. Instead of discrete values, there will be a probability distribution for each latent attribute, making the latent space continuous. This makes random sampling and interpolation easier. In terms of structure, VAEs bear a resemblance to an autoencoder; they are also made up of an encoder (also known as recognition or inference model) and a decoder (also known as a generative model). Both VAEs and autoencoders attempt to reconstruct the input data while learning the latent vector. However, unlike autoencoders, the latent space of VAEs is continuous, and the decoder itself is used as a generative model.

VAEs can be expressed as follow: the encoder *q_ϕ* (*z*|*x*) where *ϕ* are the weights and biases of the network, *x* is the input, and *z* is the latent space representation. Here, instead of being a discrete value, the encoder output is a distribution (for example, Gaussian) over the possible values of *z*, which could have generated *x*.

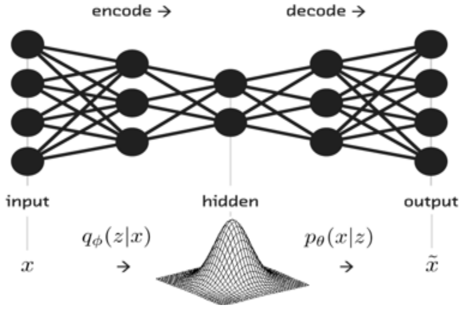


FIGURE 2. **Variational autoencoder representation**.

The VAE stochastically (randomly) samples *z* from the distribution, then it sends the sample through the decoder *p_θ* (*x*|*z*) where *θ* is the decoder weights and biases. The decoder output, in turn, is a distribution over the possible corresponding values of *x*, as Figure 2 shows.

By doing this kind of sampling from a distribution, VAEs have two different types of losses. The first of these is the Kullback-Leibler divergence (KL) between the probability distribution *q_ϕ* (*z*|*x*) and the expected probability distribution, *p_θ* (*x*|*z*). It measures how much information is lost when *q_ϕ* (*z*|*x*) is used to represent *p_θ* (*x*|*z*) (in other words, how close the two distributions are). It encourages the autoencoder to explore different reconstructions. The second is the reconstruction loss, which measures the difference between the original input and its reconstruction. The more they differ, the more it increases. Therefore, it encourages the autoencoder to reconstruct the data better. These two losses can be expressed as follows:

$$L(\theta,\phi;x) = -D_{KL}(q_\phi(z|x)||p_\theta(z)) + E_{q_\phi(x)}\left[log\, log\, \left(p_\theta(z)\right)\right] \qquad (2)$$

To implement this, the bottleneck layer will not directly output the latent state variables. Instead, it will output two vectors, which describe the mean and variance of each latent variable's distribution, as shown in Figure 3.

Once the mean and variance distributions are obtained, a state z can be sampled from the latent variable distributions, and it can be passed through the decoder for reconstruction. However, this sampling process has one issue during training such that Backpropagation gradients do not work over random processes (stochastic layer) like the one described above [46].

The solution to this problem is to push out the sampling process as the input, which can be done by using an innovative technique, called the reparameterization trick. First, a random vector ε is sampled, with the same dimensions as z from a Gaussian distribution (the ε circle in the figure below). Then, it is shifted by the latent distribution's mean μ, and is subsequently scaled by the latent distribution's variance σ, as shown in Figure 4 [47].

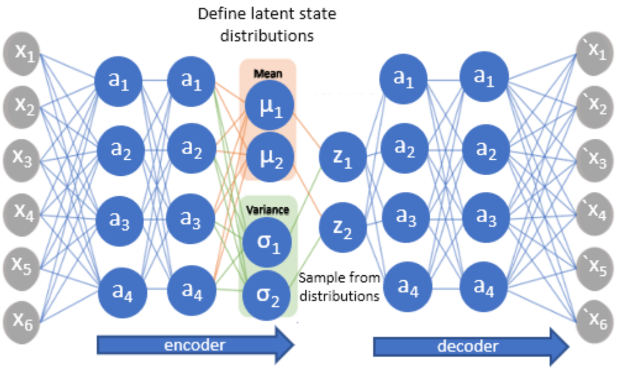


FIGURE 3. **VAE sampling process.**

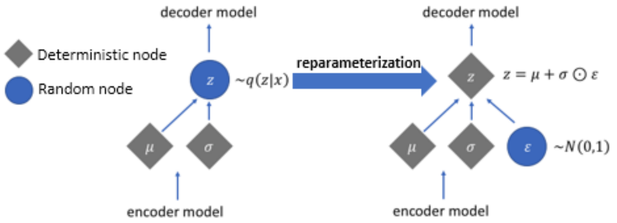


FIGURE 4. **VAE reparameterization trick.**

By doing this, the random generator is omitted from the backward pass, and the sampled data will have the properties of the original distribution. The updated sampling process now can be expressed as follows:

$$z = \mu + \sigma \odot \varepsilon \tag{3}$$

In the fraud detection domain, VAEs represent a powerful technique. The encoder would produce a distribution of possible encodings describing the transaction's essential characteristics, yet it will keep the generalization intact.

C. GENERATIVE ADVERSARIAL NETWORKS (GANS)

GANs were introduced by Ian Goodfellow and his fellow researchers at the University of Montreal in 2014 [48]. A GAN consists of two neural networks, as Figure 5 shows [49]:

- **Generator:** This is the generative model. It takes a probability distribution (random noise) as input from a latent space and tries to generate a realistic output sample. Its purpose is similar to the decoder part of the VAE.
- **Discriminator:** This is sometimes known as a “critic,” which takes two alternating inputs: the real samples of the training dataset or the generated fake samples from the generator. It tries to determine whether the input sample comes from the real samples or the generated ones.

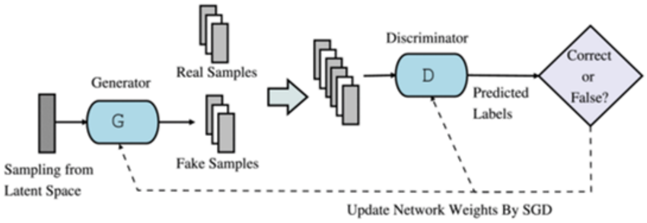


FIGURE 5. **The architecture of generative adversarial network.**

These two cooperating (and competing) networks are trained together as one system wherein the discriminator tries to get better at distinguishing between the real and fake samples. The generator tries to output more realistic examples to deceive the discriminator into thinking that the generated example is real. That’s why it is called “adversarial.” The system's ultimate goal is to make the generator so good that the discriminator would not be able to distinguish between the real and fake samples. Even though the discriminator does classification, a GAN is still unsupervised since it does not need labels for the samples.

The discriminator is a classification neural network, and it can be trained the usual way by using gradient descent and backpropagation. However, the training set is composed of equal parts real and generated samples. Therefore, the loss function can be minimized as follows:

$$L^{(D)}\left(\theta^{(G)},\theta^{(D)}\right) = -E_{x\sim P_{data}}\, log\, log\, D(x) - E_z\, log\, log\, \left(1 - D\left(G(z)\right)\right) \tag{4}$$

The equation is just the standard binary cross-entropy cost function. The loss is the negative sum of the expectation of correctly identifying real data, $D(x)$, and the expectation of 1.0 minus correctly identifying synthetic data, 1-$D(G(z))$. GAN considers the total of the discriminator and generator losses as a zero-sum game to train the generator. The generator loss function is simply the negative of the discriminator loss function [46]:

$$V^{(G)}\left(\theta^{(G)},\theta^{(D)}\right) = -L^{(D)}\left(\theta^{(G)},\theta^{(D)}\right) \quad (5)$$

Thus, the GAN minimax loss objective function can be written as [50]:

$$\min_{G} \max_{D} V(G,D) = E_{x\sim P_{data}}\, log\, log\; D(x) + \; E_z \; log\, log\; \left(1 - D\left(G(z)\right)\right) \quad (6)$$

The solution to the minimax game is called the Nash equilibrium. A Nash equilibrium happens when one of the actors does not change its action, regardless of what the other actor may do. A Nash equilibrium in a GAN framework happens when the generator becomes so good that the discriminator is no longer able to distinguish between the generated and real samples. However, the gradient descent algorithm is designed to find the minimum of the loss function rather than the Nash equilibrium. As a result, sometimes the training may fail to converge, but, due to the popularity of GANs, many improvements have been proposed.

D. WASSERSTEIN GANS (WGANS)

GANs can be very difficult to train and are prone to mode collapse. Mode collapse is when the generator produces outputs that look the same even though the loss functions are already optimized. Wasserstein GAN [46, 51] proposed an implementation that can avoid a mode collapse issue; that is, by replacing the GAN loss function based on the Wasserstein 1 or Earth-Mover distance (EMD). In our case, this is where the “critic” discriminator is calculating the Wasserstein distance between the real and fake samples. As the loss function decreases in the training process, the Wasserstein distance becomes smaller. Hence, the generator generates samples closer to the real ones.

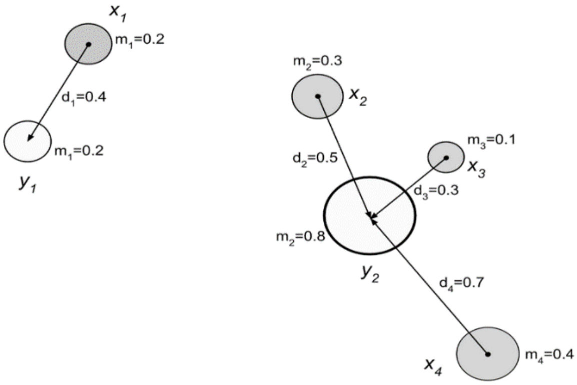


FIGURE 6. **A pictorial representation of the Earth-Mover distance computation [46].**

The intuition behind EMD is that it measures how much mass $\gamma(x,y)$ should be transported by d = ‖x-y‖ for the probability distribution *p_data* to match the probability distribution *p_g*, as shown in Figure 6 [46]. $\Gamma(x,y)$ is also known as a transport plan to reflect the strategy for transporting masses to match the two probability distributions, which can be expressed as the following equation:

$$W\left(p_{data}, p_g\right) = inf_{\gamma\in\prod(p_{data},p_g)} E_{(x,y)_{\sim\gamma}}\left[\left|\left|x-y\right|\right|\right] \quad (7)$$

When using EMD or Wasserstein 1 as the loss function, the generator will try to minimize, while the discriminator tries to maximize, it can be expressed as follow:

$$L^{(D)} = -E_{x_{\sim p_{data}}} D_w(x) + E_z D_w\left(G(z)\right) \quad (8)$$

$$L^{(G)} = -E_z D_w\left(G(z)\right) \quad (9)$$

In the generator loss function $L^{(G)}$, the first term disappears since it is not directly optimizing with respect to the real data. Moreover, the discriminator is not trying to tell whether the samples are real or fake anymore. Instead, it is using K-Lipschitz function to calculate the Wasserstein distance between the real and fake samples. As the loss function in the training process decreases, the Wasserstein distance becomes smaller. Hence, the generator generates samples closer to the real ones [52], which can be described by:

$$W\left(p_{data}, p_g\right) = \max_{w\in W} E_{x_{\sim p_{data}}}\left[D_w(x)\right] - E_z\left[D_w\left(G(z)\right)\right] \quad (10)$$

IV. METHODOLOGY

This section explores the workflow that the research follows towards the model implementation. It describes different techniques and methods that have been used in each one of these steps, such as the data preparation and preprocessing techniques, model building, and performance evaluation.

A. WORKFLOW

As shown in Figure 7, once the raw data is obtained, some time is invested in understanding the data in order to describe it and discover any underlying relations. Following this, different data preprocessing techniques are used to prepare the data for model training and evaluation. The output from the data preprocessing is separated into two different datasets. The first dataset, called “base,” is used to train and test the autoencoder models (AE and VAE). AE has two different versions: single-loss function and multi-loss function.

The second dataset, called “merged,” is used to train and test the WGAN model, which generates more fake fraud transactions. These transactions are then mixed with the merged dataset to produce the “mixed” dataset. Finally, the mixed dataset is used to train the autoencoder models one more time. The idea behind this approach is that by having more fraud transactions, the model performance is expected to increase, as will be explained later. All the models are then compared using the different evaluation techniques to obtain the best performing AE model.

B. DATASET DESCRIPTION

1) RAW DATA

The data is obtained from the research project that was undertaken in 2014 between the School of Computer Science, University of Nottingham (Malaysia campus) and a local Malaysian Bank. The original dataset that was obtained in 2014 contains about 30 million transactions (records) for the period from 2012 until 2013 [7]. However, for privacy reasons, the full dataset is not accessible anymore. Instead, this research obtained access to a subsection of the dataset as summarized in Table 1.

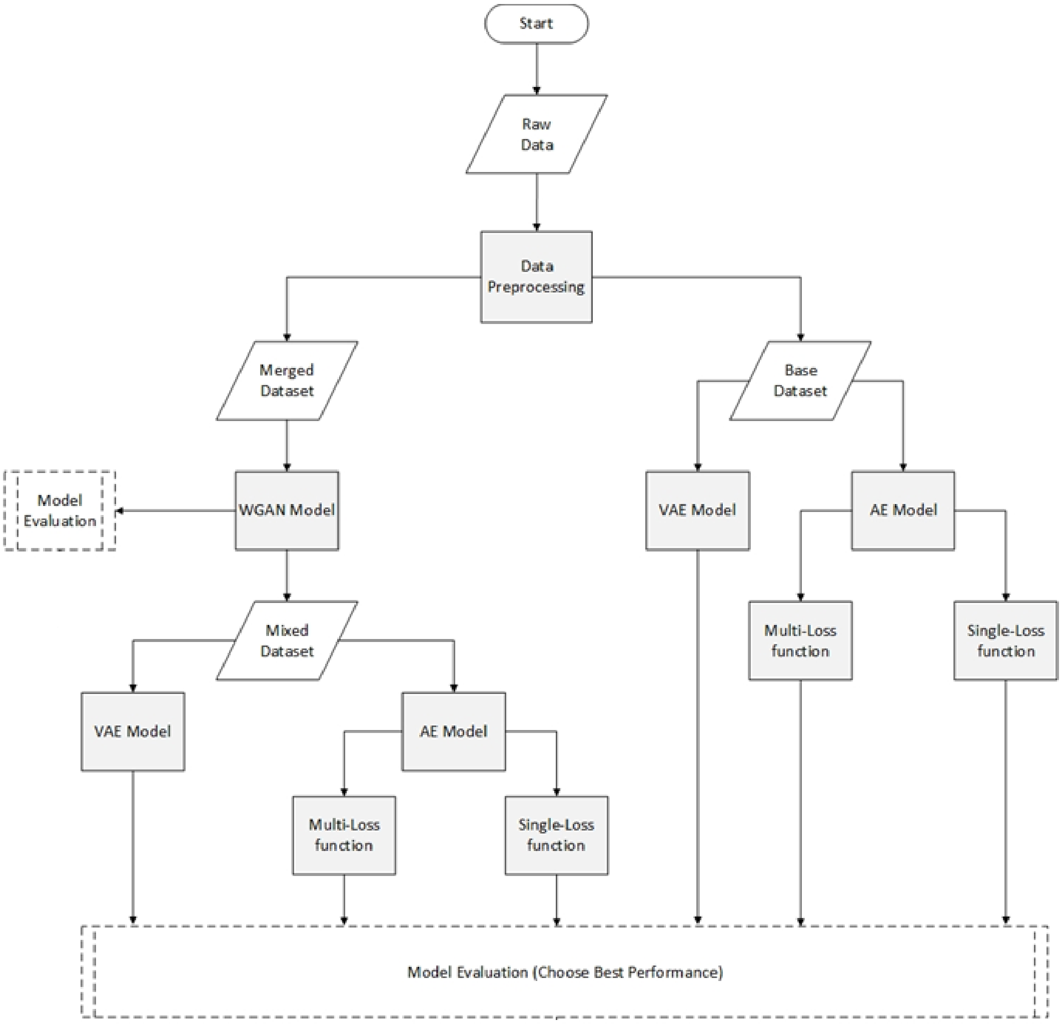


FIGURE 7. **The workflow of the model implementation. The raw data is split into two segments: merged and base datasets. The merged dataset is used as input for the WGAN model. The WGAN model produces and generates the mixed dataset which is then used to train the AE and VAE models. The base dataset is used as input to train both VAE and AE models. The AE model uses two loss functions, namely the multi-loss and single-loss function.**

TABLE I

RAW DATASET DESCRIPTION. THE RAW FINANCIAL TRANSATIONS ARE PRE-PROCESSED ON DIFFERENT TIME HORIZONS, NAMELY DAY, WEEK, AND MONTH.

Name	Rec.	Attr.	Normal	Fraud	Null Values	Duplicates
Day	2706	69	2661	45	3	0
Week	1490	71	1446	44	3	0
Month	693	71	649	44	2	0

The whole subsection dataset contains a total of 4889 transactions that are consolidated based on the time intervals in 3 different files (Day, Week, and Month) and labeled under the class attribute by a domain expert to be either (0 = normal) or (1 = fraud). The number of attributes (or "features") varies between these groups based on the time interval, as the Day group has 69 attributes including the class as it is the base time interval. In contrast, the other two groups have two extra attributes with a total of 71 attributes. These two extra attributes are (tran_date_to and Week), in the case of the Week group, and (tran_date_to and Month), in the case of the Month group.

The Day group has the most transactions, with a total of 2706 transactions, where 45 of them are fraud and the rest are normal with only 3 Null values under the P2 attribute and no duplicates. While the Week group has 1490 transactions, 44 of them are fraudulent ones, with three Null values under the P2 attribute and no duplicates. Finally, the Month group has 693 transactions, with 44 of them being fraud, two Null values under the P2 attribute, and no duplicates. In this research, the Day group is used as the base dataset, as it has the most normal transactions that will be required to train the autoencoders model and it requires less preprocessing work.

2) MERGED DATA

We notice from the dataset above that it does contain sufficient fraudulent transactions. However, training the WGAN model requires enough fraud transactions to produce more realistic fraud transactions in turn.

Therefore, to obtain enough transactions to train the WGAN, the three groups of the dataset in Table 1 are merged into one dataset. However, two issues need to be solved to perform this merging successfully. The first issue is that different datasets have different numbers of attributes. The Week and Month datasets have two extra attributes - one attribute is the time interval, and the other attribute is the ending date for that interval, as mentioned above. To solve this issue, the time interval will be reconstructed into the base unit for all three datasets, which will require feature engineering for the transaction date tran_date attribute that will be discussed in the next section. Therefore, the (tran_date_from and Week) attributes will be dropped from the Week dataset. Also, the (tran_date_from and Month) attributes equally will be removed from the Month dataset while keeping the tran_date_to attribute since it represents the end of the time interval for both datasets. Thus, the processed datasets will have equally 69 attributes and can be merged.

TABLE II

MERGED DATASET DESCRIPTION. THE NEW DATASET IS SORTED AND RE-INDEXED AFTER THE DATASETS ARE MERGED. THE DATASET CONTAINS 133 FRAUD TRANSACTIONS, AND, AS IT WAS EXPECTED, 7 DUPLICATES WERE FOUND.

Name	Rec.	Attr.	Nor mal	Fraud	Null Values	Duplicates
Merged	4889	69	4756	133	8	7

The second issue is the possibility of having duplicates. However, this issue will be discussed and solved in the next section. Table 2 summarizes the merged dataset. After the datasets are merged, the new dataset is sorted and re-indexed. The dataset now contains 133 fraud transactions, and as was expected, seven duplicates were found, which will be handled next.

C. DATA PREPROCESSING

Data preprocessing is one of the key steps towards any successful machine learning implementation. It helps to remove the noise data and irrelevant information from the dataset that prevents the knowledge discovery and can hurt the generalization. In the next sections, we will cover some of the data preprocessing techniques such as transformation, normalization, data cleaning, and feature extraction that were implemented in this research.

1) FEATURE DROPPING

This is the first technique that can help in dimension reduction. Keeping irrelevant attributes could hurt the model performance and cause overfitting, but by removing the unnecessary or redundant features, the model is expected to perform and generalize better. It will also help cut down the computing power required to train and run the model. In this research, two techniques were used to identify such features in the dataset: zero-sum and automatic generated.

Attributes that have the same value for every record instance do not add any extra knowledge to the model as it cannot enhance the prediction; rather, it can hurt the model. Significantly, if the total value for that attribute for the whole dataset is zero, this attribute is dropped during the data preprocessing step. This is the case for some features in the dataset such as (rl0003, rl0012, rl0013, rl0014, etc.).

There are two attributes directly related to the customer in the given dataset. These two features are customer identifier, cif_id, and account number, account_no. The bank system automatically generates both these attributes. Some fraud detection implementations are mainly built on such attributes as the graph analysis and the social network analysis, where the customer account number is considered to be a ‘node’ and his transaction an ‘edge’. Then certain weights and techniques are applied to evaluate whether this account is doing money laundering or not.

However, these implementations require the account that the transaction was sent from and the account that the transaction will be sent to. Unfortunately, the given dataset does not provide these attributes. Also, in terms of implementation for this research, including these customer-related features will have a negative impact on the model performance. The model will be used as a real-time fraud detection system during the inference phase, where even a single transaction can be evaluated from a totally new customer. Therefore, during the model training no customer-specific features will be included and both attributes are removed from the dataset.

2) DUPLICATE DROPPING

Even though the groups that were mentioned in Table 1 do not have duplicates, when these groups are combined together in the merged dataset, some duplicates were found. Hence, we check for duplicates in the merged dataset and drop them.

However, it is worth pointing out that the duplicates that have been dropped from the dataset have occurred because of the merging process. In other cases where duplicates represent an original part of the dataset, it is still acceptable to keep them.

3) BINARY ENCODING

Categorical attributes need to be converted into numbers, so the model will be able to work with them, and there are different types of encoding techniques. Among the most popular ones are One Hot Encoding and Binary Encoding. We compare these two encoding techniques in terms of their impact on the model accuracy and the number of output attributes that each technique produces. Binary Encoding will have the same impact on the model accuracy as One Hot Encoding but with less attributes, which is sufficient for this research. Three attributes (account_type, product_type, business_type) in the dataset need to be encoded. The binary encoder will encode the categories in each one of these attributes into binary code then split it into columns.

4) NULL VALUES

Having null or missing values in the dataset can lead to wrong predictions or even issues during model training. Therefore, filling these values is an important step during the preprocessing phase. In the given dataset, eight null values were found under the P2 attribute. These null values were handled by filling them with the mean attribute value.

5) LOG TRANSFORMATION

Data skew represents another challenge that needs to be fixed. Three attributes in the given dataset (credit_amount, debit_amount, debitpluscredit_amount) show an extensive range of differences within their values because the vast majority of the values are skewed towards a certain direction while the remaining few are skewed in the other direction. By applying the common scaling techniques directly to such attributes, the scaled data will not preserve the original data representation. Therefore, log transformation is required to fix the data skew as it pulled in the extremely high values relative to the median while stretching the low values back further away from the median. Moreover, the log transformation respects the positivity of the attribute, which is essential for the scaling techniques that will be applied to the data. By applying log transformation on the attributes, their distribution takes a better shape.

6) STANDARDIZATION

As the dataset contains a wide range of values, the normalization or standardization of data prior to the training phase is favorable because it can reduce the estimation errors and calculation time.

Normalization, which is also called Min-Max Scaling, can be achieved by scaling the attribute to a fixed range (0 and 1) through this equation:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (11)$$

However, in fraud detection models, it is important to preserve the original distance between data points. Therefore, standardization will be much more appropriate for implementation. Standardization scales the data based on its mean (μ) and the standard deviation (σ) from the mean. Having $\mu = 0$ and $\sigma = 1$ will center the data around 0 as in the following equation:

$$Z = \frac{X - \mu}{\sigma} \quad (12)$$

The standardization technique was applied over all the non-binary attributes in the dataset (credit_amount, debit_amount, p2,…, etc.).

7) FEATURE ENGINEERING ON DATES

One reason for having the raw dataset divided into three groups is to enhance the model accuracy by grouping the transactions within a specific period. Although this is still a valid approach, it can be improved even further.

This research introduces another approach that can better use the date attribute, engineering some new features based on the tran_date feature. Specifically, the tran_date feature is split into its base date components, then these new features are added into the dataset. These new features are described in Table 3.

TABLE III
DATE-BASED NEW FEATURES

Feature	Description
month	Month of the transaction
day	Day of the transaction relative to the month
quarter	Quarter of the year (1~ 4) of the transaction
dayofweek	Day of the week of the transaction
is_weekend	Whether the transaction occurred during the weekend or not

The idea behind introducing these features is to allow the model to capture any pattern within the data that is related to its date. As shown in Figure 8, transactions tend to have different data peaks from one feature to another, which the model may utilize to identify fraudulent behavior.

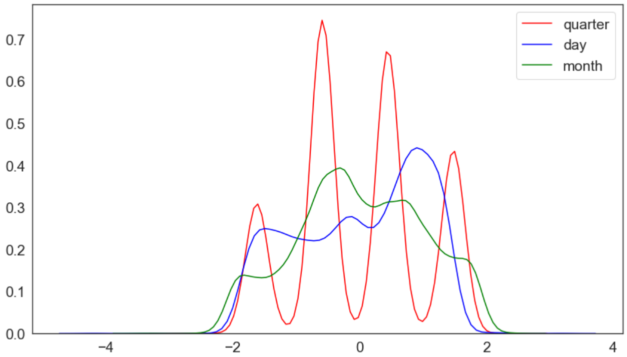


FIGURE 8. Data distribution for the new date related features based on different time horizons (i.e., day, month, and quarterly). The y-axis represents the density and the x-axis represents the range of values.

However, the given dataset does not provide any timestamp features. It could be instrumental in deducing even more information, such as whether the transaction occurred during daytime, night, morning, or afternoon, which may be useful for the model.

It is worth mentioning that there is a popular deep learning implementation that can handle time-series data and sequence data better than the non-recurrent neural networks, which is called LSTM (Long Short-Term Memory). However, this approach is not useful for this research because fraud behavior does not follow a particular sequence; one transaction cannot be used to predict the next one. Moreover, this research aims to build a real-time fraud detection application that may operate on one transaction rather than a batch of transactions.

D. PREPROCESSED DATASET

The preprocessing phase outputs two datasets; the base dataset will be used to train the autoencoder models, and the merged dataset used to train the WGAN. This section describes these two datasets as they are now ready to be used. Table 4 shows the description of these datasets.

The different number of attributes in the two datasets are due to one extra attribute, rl0030, that was dropped from the base dataset because it has a zero-sum value. However, this attribute holds some value for instances in the other two groups when the groups are merged together. Although the number of Fraud transactions is low compared to the normal transactions in the base dataset, the autoencoder implementation will overcome this issue. As the WGAN will need the fraud transactions for the training, the merged dataset is used as it has more fraud transactions than the base dataset.

TABLE IV
PROCESSED BASE AND MERGED DATASETS

Name	Rec.	Attr.	Normal	Fraud	Null Values	Duplicates
Base	2706	43	2661	45	0	0
Merged	4882	44	4749	133	0	0

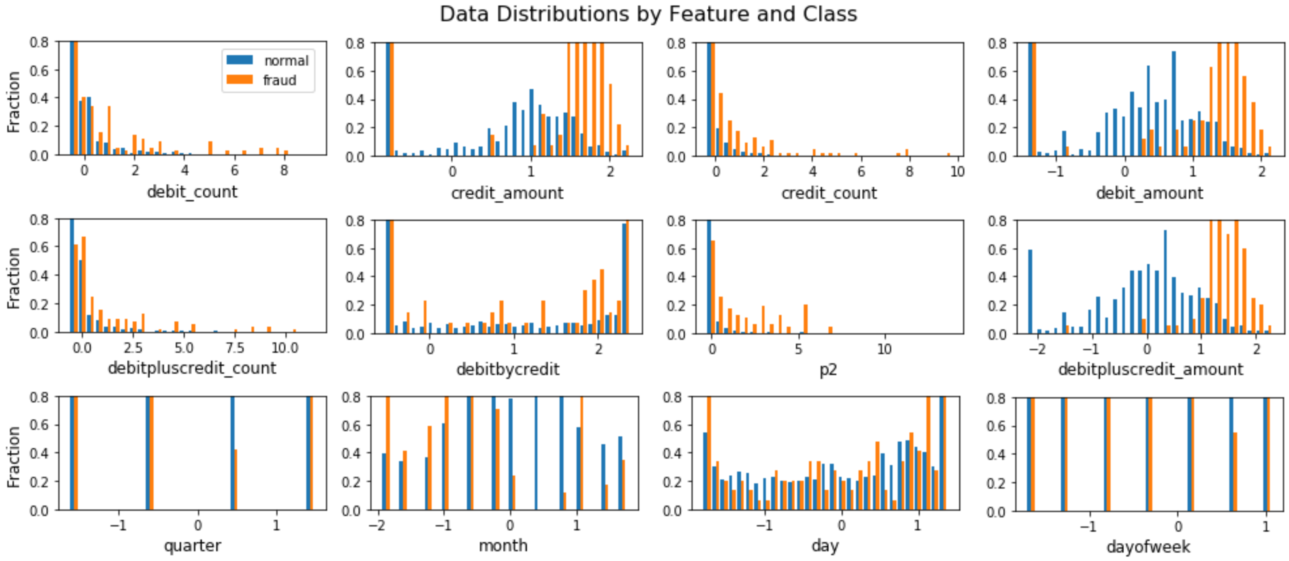


FIGURE 9. **Attributes distribution by class (Normal and Fraud). The y-axis represents the fraction of transactions, and the x-axis represents the time horizons (i.e., quarter, month, day, dayofweek etc.)**

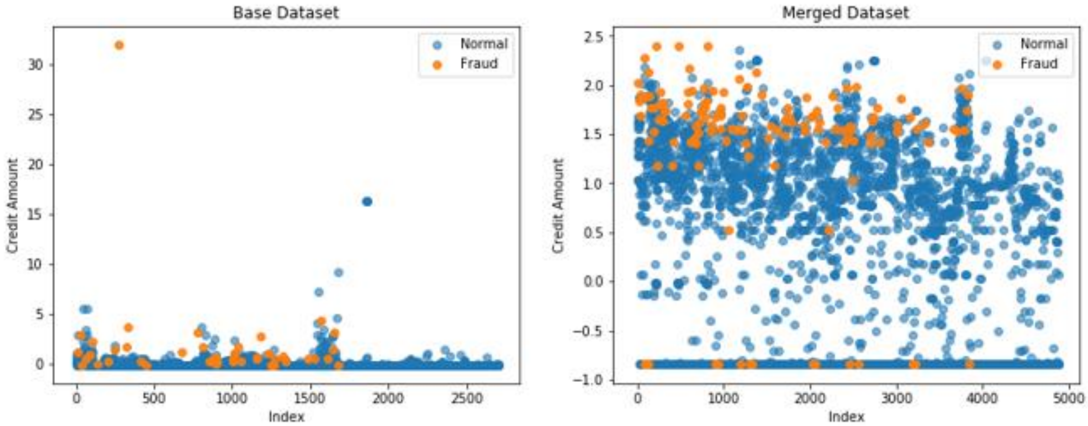


FIGURE 10. **Attributes distribution by class (Normal and Fraud). Normal transactions are represented as 'blue' whereas fraud transactions are represented as 'orange'**

In both datasets, fraud and normal transactions are overlapped in almost every feature, except in certain features such as credit_amount, debit_amount, and depitpluscredit_amount, where fraud and normal transactions can be slightly separable, as Figure 10 shows. Nevertheless, when these barely separable features are investigated further, their distribution shows a high level of mixing, as Figure 9 shows in the case of credit_amount. Therefore, a machine learning implementation is necessary for better normal-fraud class classification.

E. MODEL IMPLEMENTATION

1) AE

The autoencoder is mainly used to learn the important features; then, it utilizes that knowledge to reconstruct the data to be as similar as possible to the original data. However, in fraud detection implementation, the autoencoder's output is not the focus. Instead, the most important part is the knowledge that the model gains in the latent vector. That knowledge can be evaluated through the reconstruction error, as mentioned earlier.

In this implementation, the autoencoder will be only trained over the normal transactions. Thus, the model is expected to learn the normal transactions' important features and then reconstruct these transactions. However, during the testing phase, the model will be tested against both normal and fraud transactions. That's when the reconstruction error is used. If the tested transaction is normal the model will be able to reconstruct it with the minimum error. However, if the transaction is fraud, the reconstruction error will be relatively significant. Moreover, to determine whether the error is big or small, a predetermined threshold value is used, on which the anomaly score is given to each transaction. The threshold determination method will be discussed in the next sections.

MODEL ARCHITECTURE

The first component in the autoencoder implementation is the input layer, Model_Input, which has 42 neurons. Each neuron represents one attribute in the base dataset except the class attribute. No activation function is used for this layer as no prior weights exist; hence, it merely passes the values to the network's next component.

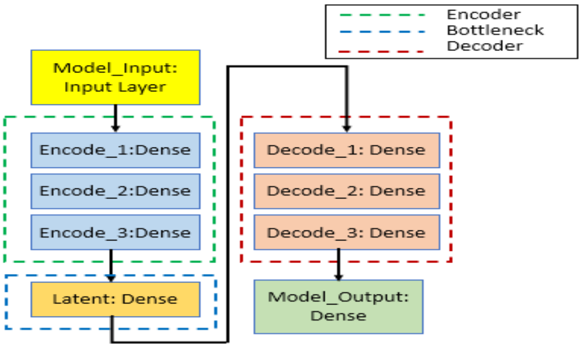


FIGURE 11. **Autoencoder model structure.**

The next component of the network is the encoder. It consists of 3 dense layers (Encode_1, Encode_2, Encode_3), and the number of neurons in each one is almost half of the number of its previous layer. Thus the autoencoder is forced to learn only the important features. The bottleneck layer is the next component; it has the minimum number of neurons, which is eight, that will hold the latent vector weights.

Then, the network passes the values to the decoder, which in turn consists of three dense layers (Decode_1, Decode_2, Decode_3). However, the number of neurons in each dense layer in the decoder is almost double the number of its previous layer to build towards restoring the same number of features as the original data. The last component of the network is the Model_Output layer, which has 42 neurons representing the same number of neurons as the original input. Figure 11 shows the architecture of the AE model.

The activation function in the first layer in both the encoder and the decoder is *tanh* as it will ensure the output values for neurons in these layers will always be between (-1, 1). This fits nicely because of the data standardization in the preprocessing phase. The other layers in both of these components use the ReLU activation function, which will force the output to be positive, or else zero. Finally, a sigmoid activation function in the Model_Output layer will produce output within the range of (0, 1).

The model uses Mean Squared Error (MSE) as a loss function as most of the input values are a spectrum rather than binary. The MSE computes the average of the square difference between the actual input value and the predicted value. Therefore, the objective of the optimizer is to minimize that loss function. The output of MSE is a positive value. However, as sigmoid was used as an activation function in the last layer of the autoencoder, it is expected to have a Binary Cross-Entropy (BCE) loss, which will be discussed in the next section.

Lastly, the gradient-based optimization optimizer Adam is used by the model to minimize the loss function as it is invariant to the gradients' diagonal rescaling and capable of handling a wide range of nosiy data.

MULTI-LOSS FUNCTION

Given that the input features have both binary and non-binary data, and sigmoid is used as the model output activation function, this paper implemented another variant of the AE model. Instead of having one loss function, this variant has two loss functions, MSE and BCE. BCE is capable of handling the binary values and has a bounded output of [0,1]. The idea of combining these two losses is to have a smooth and stable loss value that will handle both binary and non-binary values during the optimization. The implementation adds these two losses then returns their mean value.

TRAINING AND HYPERPARAMETERS TUNING

Several experiments have been undertaken to determine the values of the hyperparameters (such as the learning rate, number of epochs, etc.). After reaching a stable performance, the learning rate value, which determines the gradient optimizer's step size, is set to be (1e-3), and the number of epochs is set to be 300. To optimize the training process, an early stopping technique is used to terminate the training if the loss does not decrease beyond 1e-5.

As the dataset is not large enough, we used a cross-validation technique where 20% of the data is used for testing, and 80% is used in the training process. Each batch size is set to be 80, selected based on a random seed. Finally, to avoid the network's tendency to memorize the training data and fail to generalize, the model uses an activity regularizer in the encoder's first layer. This allows the application of penalties over this layer during the optimization and adds those penalties to the loss function.

2) VAE

The variational autoencoder follows the same idea for implementation as for the AE. That is, only the normal transactions are used during the training phase. Then an anomaly score is assigned with each transaction during the test phase for both normal and fraud transactions by comparing the reconstruction loss against a predefined threshold. The difference between VAE and AE is the latent space; as in VAE, it is represented by a distribution rather than data values.

MODEL ARCHITECTURE FOR VAE

The model starts with the input layer Model_Input, which has 42 neurons with no activation function, and the encoder, consisting of two dense layers (Encode_1, Encode_2) where they have (22 and 12) neurons, respectively. The Encode_2 layer, in turn, outputs two different vectors, Mean and Log Variance. Each one of these two vectors is mapped to its own layer. Log Variance here is used because of its more numerical stability than the standard deviation, which will be calculated later in the Sigma layer.

However, before recovering the standard deviation in the Sigma-dense layer, both Mean and Log Variance are passed to a custom layer, KLDivergenceLayer. This layer calculates the distribution loss using a KL divergence function, then adds this loss to the total model loss. Finally, it returns the inputs (Mean and Log Variance) unchanged to the next layer.

Next, the Sigma-dense layer receives the values from KLDivergenceLayer and recovers the Standard Deviation. Subsequently, we implemented the reparameterization trick by introducing a separate dense layer, Epsilon, which uses the Monte Carlo sampling technique to draw a random sample from a normal distribution with the same latent vector dimension. This sample represents Noise, which is then multiplied by Sigma, and the product is forwarded to the next layer.

The latent space Z-dense layer receives the sampled vector standard deviation multiplied by epsilon and also receives the Mean from KLDivergenceLayer. Then it adds them together and outputs the result to the model decoder. The decoder consists of two dense layers: (Decode_1, Decode_2), and the number of neurons in each of them is and 12 and 22, respectively. Finally, the Model_Output layer receives the decoded values and outputs 42 features. In general, the model uses the ReLU activation function in both the encoder and the decoder layers. Figure 12 shows the architecture of the VAE model.

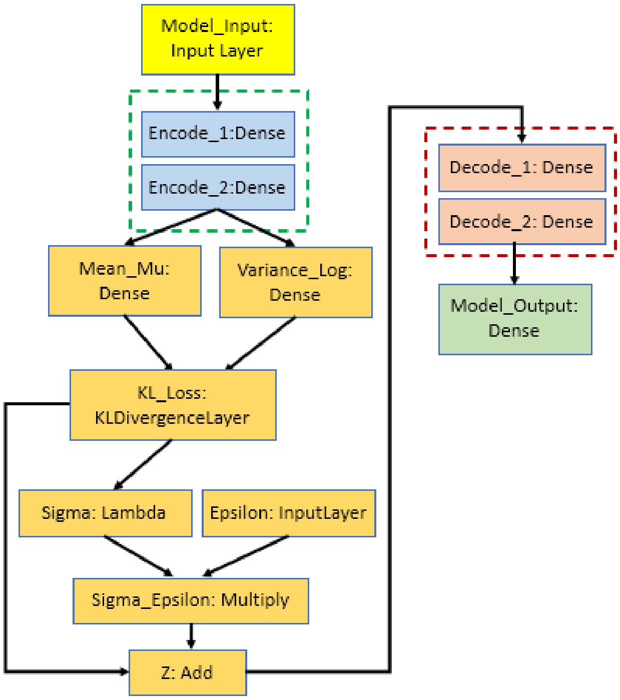


FIGURE 12. **VAE model architecture.**

TRAINING AND HYPERPARAMETERS TUNING FOR VAE

The initial learning rate is set to be (1e-3), while the model will be trained for 300 epochs. Cross-validation is used where training and test datasets correspond to 80% and 20% of the original dataset, and batch size is set to be 128.

The RMSprop optimizer is used to minimize the loss function because it limits the vertical direction fluctuations. This allows increasing the learning rate, allowing the gradient to take larger steps for faster convergence.

3) WGAN

The fraud detection models' problem is that their datasets are always unbalanced, given that the fraud behavior infrequently occurs. The same case applies to the merged dataset as it has only 133 fraud transactions. Therefore, WGAN will generate more fraud transactions, enhancing or solving the unbalanced dataset problem, hence enhancing the model's performance.

As the model will generate fraud transactions, the merged dataset is used to train the model. After the training is done, the model is used to generate new fraud transactions. These newly generated fraud transactions are mixed with the dataset to formulate the mixed dataset.

MODEL ARCHITECTURE FOR WGAN

As training GANs is not easy in terms of stability [50], the model implementation considers that adding extra layers such as the Dropout layer and the LeakyReLU activation layer could make the model more stable.

In general, the model starts with the Model_Input layer, which receives 43 features mapped to its 43 neurons. Then the model separated into two different networks: one is the generator, and the other one is the discriminator. The generator network has three dense layers (Gen_1, Gen_2, Gen_3) plus the Gen_Out layer. Because the generator generates fake transactions, it does not need to follow a specific structure in each layer's neuron number. However, as almost all GANs literature is based on image processing, it follows a binary multiplication system. To follow that practice in this network, the number of neurons in the generator layers are 128, 256, and 512, respectively. However, for the Gen_Out layer, the number of neurons is 43, as it will be the same number of features as the real transaction.

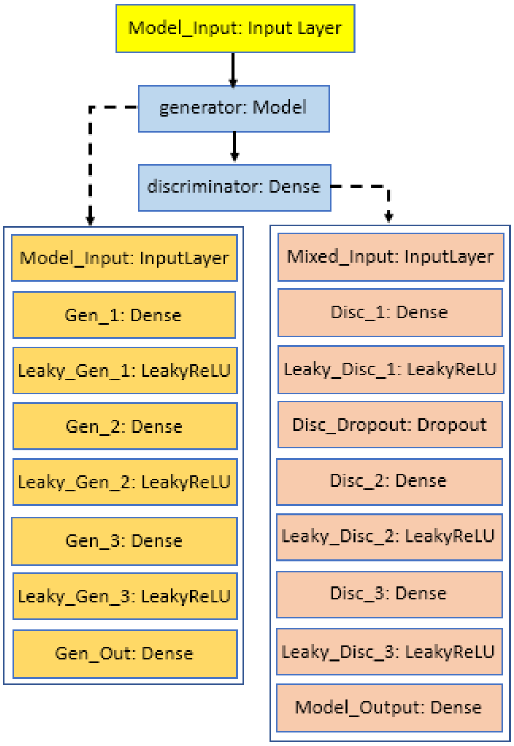


FIGURE 13. **WGAN model architecture.**

After each layer in the generator except for Gen_Out, there is a LeakyReLu activation layer that replaces the standard layer activation and could handle values better than the standard ReLU.

The discriminator network, on the other hand, starts with the Mixed_Input layer receiving input from both the generator, which will produce the fake transactions and also a randomly selected batch from the real transactions. These two sources are then mixed and passed to the next layers. The dimensions of the generated fake transactions and the real transactions are the same (i.e., 43 features). Hence, the number of neurons in this layer is 43 as well.

The discriminator consists of 3 dense layers (Disc_1, Disc_2, Disc_3) beside the Model_Output layer. The dense layers are based on the binary system where the number of neurons is 512, 256, and 128, respectively. The output layer will have only one neuron. The same technique of using LeakyReLu is used here, so each dense layer except the output is followed by the LeakyReLU activation layer. Moreover, to solve the problem of overfitting and stability issue, a Dropout layer is added after the first dense layer activation, which could help to regularize the network.

Finally, the discriminator output layer, Model_Output, has no activation function as it is implementing the Wasserstein distance. It will use the single neuron in the layer to output the distance of which the transaction is considered real or fake, rather than outputting 0 or 1, using a classic activation function such as Sigmoid. Figure 13 shows the architecture of the WGAN model.

TRAINING AND HYPERPARAMETERS TUNING FOR WGAN

GANs require a relatively long time to converge. Hence the number of epochs is set to be 50,000. The optimizer is set to be Adam with a learning rate of about (1e-3). The batch size for the real transactions' random sample is set to be 64, which the discriminator will use.

Moreover, a checkpoint is made to save the model and weights in every 100 epochs in addition to the loss values. Once training is done, an accuracy check iterates over all checkpoints to select the best version in terms of accuracy relative to its corresponding loss.

It is worth mentioning that the LeakyReLU layers have a hyper-parameter, called alpha, that determines the curve's negative slope. Here alpha is set to be 0.2, and the Dropout rate is set to be 0.1.

F. Performance Evaluation Methods

Various measures are used to evaluate the performance of proposed models. We start with a list of related terms that will be used in these measures. False positive (FP) refers to the number of normal transactions that are predicted as fraud. True positive (TP) is the number of fraud transactions that are predicted as fraud. False negative (FN) specifies the number of fraud transactions that are predicted as normal. Finally, true negative (TN) refers to the number of normal transactions that are predicted as normal.

In this research, a confusion matrix is used to report the model performance by combining the indicators mentioned above. This will help visualize how the model confuses the true class. Moreover, some other performance measures will be calculated using the confusion matrix, such as FP rate (FPR), accuracy (ACC), precision, recall, F1 score, and Receiver Operating Characteristics (ROC) curve.

FPR is highly important in fraud detection models, especially in this research, as it aims to minimize the value of FPR as much as possible. It is expressed as follows:

$$FPR = \frac{FP}{TN + FP} \tag{13}$$

Accuracy indicates the overall correct predicted transactions, whether it is TP or TN relative to the total instances. Precision identifies the correct fraud transactions rate relative to all transactions that are predicted as fraud. The recall is the rate of the correctly predicted fraud transactions relative to all of the actual fraud transactions. To summarize the model with only one single score, the F1 score is used as it considers both recall and precision in its formula as follow:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{14}$$

It is worthwhile to mention that both the Precision and F1 scores are relatively low in fraud detection models as the number of FPs is always more prominent than the TPs. Finally, ROC is used to visualize the True Positive Rate (TPR) and the FPR, and both are plotted on the *y*-axis and *x*-axis, respectively. Hence, it shows how the model reacts under all different combinations of thresholds.

An important indicator is calculated from ROC and called Area Under the Curve (AUC). AUC summarizes the whole model performance in one number, ranging from 0 to 1 with the best performance equal 1.

The optimizer's objective is to minimize the loss. However, this could lead to overfitting. Two model losses, such as Training Loss and Testing Loss, are reported to make sure that the model is not overfitting. Suppose the Training Loss is higher than the Testing Loss. In that case, the model is underfitting, and there is room for enhancement until the Training Loss is near or equal to the Testing Loss - which is perfect fitting. Once the Testing Loss exceeds the Training Loss, the model is overfitting, and it needs to be adjusted.

THRESHOLD OPTIMIZATION

As was mentioned before, fraud detection implementations require a predefined Threshold value to be able to assign an anomaly score to each transaction. However, given the business scope of these implementations, they should filter out all of the fraud transactions. Yet, they should maintain a good degree of efficiency by targeting a low FP rate.

To automate the process of determining the Threshold, and at the same time, aligned with the business scope target, this research defines the Threshold to be Recall-First Threshold (RFT). The Recall-First Threshold (RFT), is the value that will allow the recall of all fraud transactions with the highest precision possible, as Figure 14 shows. This can be used perfectly as a Threshold for our fraud detection implementation as its required to filter out all of the fraud transactions; nevertheless, it should maintain a good degree of efficiency by targeting low FP rate.

To calculate the RFT, precision-recall pairs for different probability thresholds are computed using the precision_recall_curve function in the sklearn library. Then the minimum value in the returned thresholds array is selected.

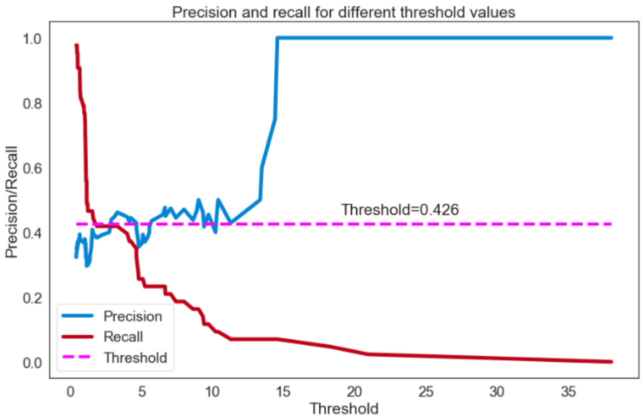


Figure 14. Recall-First threshold.

VI. RESULTS AND DISCUSSION

This section presents the results from different experiments. Firstly, it describes the mixed dataset, which is the result of the WGAN model. Secondly, it compares the AE model (single-loss and multi-loss) and VAE model under both datasets (Base and Mixed). Finally, the section concludes with a discussion of each model's performance using each dataset.

A. WGAN

WGAN training process is relatively tricky, and it often requires a significant number of epochs. Hence, the WGAN model with the configuration detailed in the previous section shows various accuracy levels during the training process, as Figure 15 shows. The accuracy refers to how much the generated samples are identical to the real samples. Thus, higher accuracy is an indication that the discriminator is no longer able to distinguish between the generated and real samples. During training, a checkpoint is saved for the model, and once the training is completed, an iteration is used to select the best version of the model based on its accuracy, which reached 99%. The optimal number of iterations is chosen when the accuracy has reached a plateau or degradation.

After training, the best model is utilized to generate fraud transactions. Specifically, it is used to generate about the same number of real fraud transactions in the merged dataset; that is, about 132 fraud transactions. These fake fraud transactions are then mixed into the merged dataset to result in the mixed dataset, as Table 5 shows.

TABLE V
MIXED DATASET DESCRIPTION

Name	Rec.	Attr.	Normal	Fraud	Null Values	Duplicates
Mixed	5014	44	4749	265	0	0

Subsequently, the mixed dataset is re-indexed and sorted to be used during the autoencoder models training, and the final dataset is shown in Figure 16.

B. AE SINGLE-LOSS (AE-S)

The autoencoder model with a single loss function (MSE) training generally shows an acceptable fitting level. The Training Loss stayed above but near the Testing Loss using the base dataset. On the contrary, the Training Loss goes below but near the Testing Loss when the mixed dataset is used, indicating a sort of overfitting, as Figure 17 shows.

When the base dataset is used, the Recall-First Threshold (RFT) was 0.216, which increased to reach 0.589 after the mixed dataset is used, as Figure 18 shows. The AUC was calculated to be 0.920 in the base dataset. However, it increased to reach about 0.963 once the mixed dataset is used, as shown in Figure 19.

As the RFT is already calculated, the reconstruction error can be assigned an anomaly score, as Figure 21 shows. All of the reconstruction error values located above the RFT are considered fraud; else, it is considered normal. However, the different color represents the actual points class.

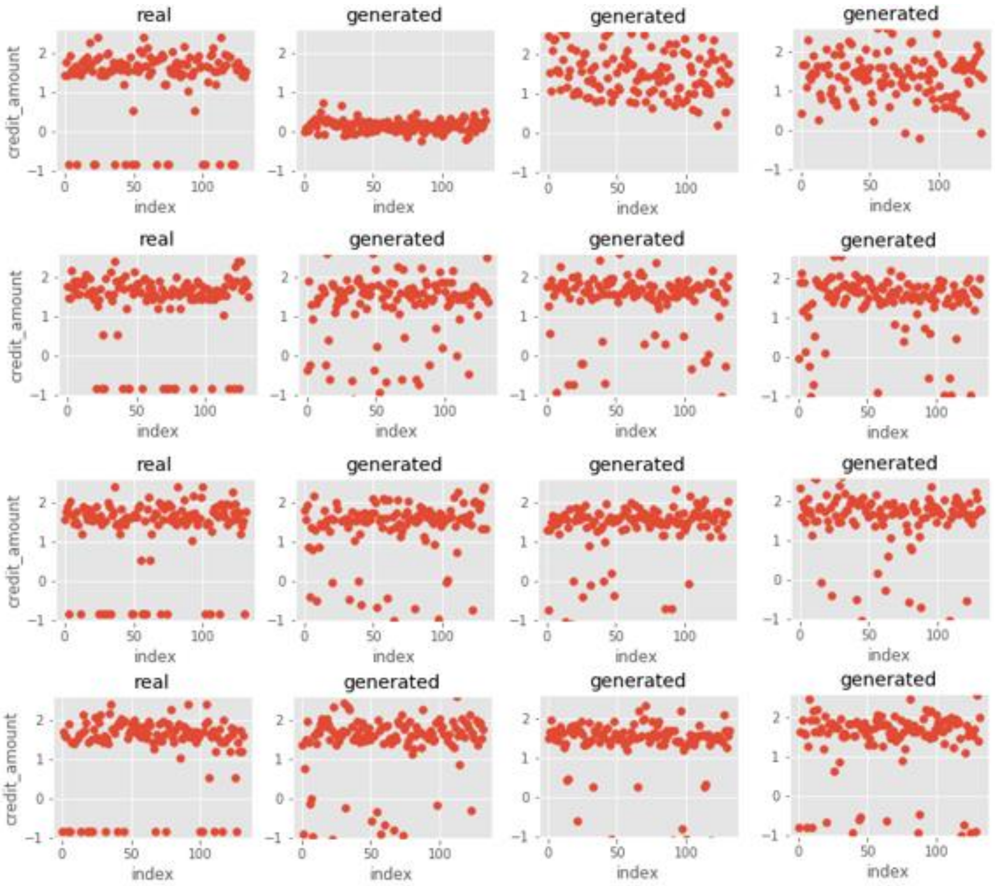


FIGURE 15. **WGAN classification accuracy at different epochs during training of the credit_amount attribute. The left side shows the distribution of the real dataset while the three remaining right sides show the distribution of the generated synthetic data for different epochs (i.e., 100, 200, and 300). It can be observed that the distribution of the synthetic data is very similar to the real dataset.**

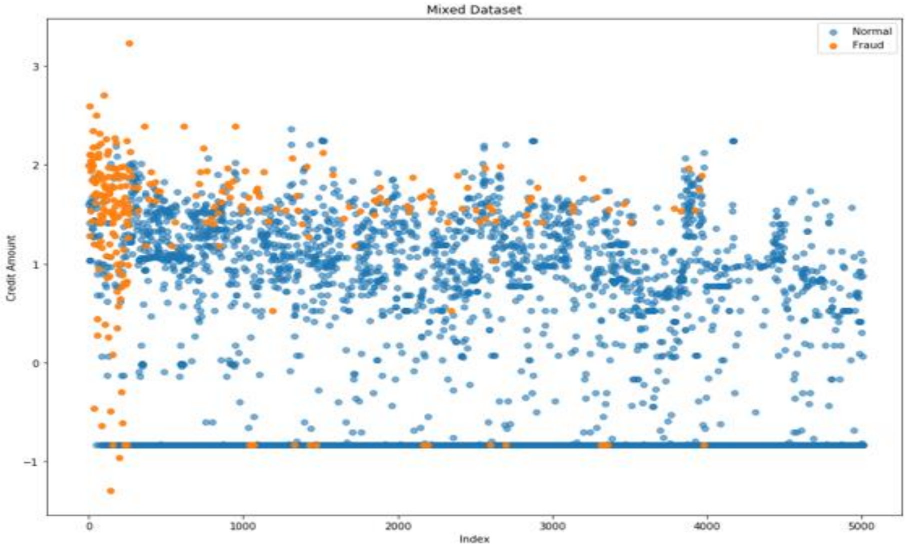


FIGURE 16. Credit_amount feature distribution in the mixed dataset by class (Normal and Fraud).

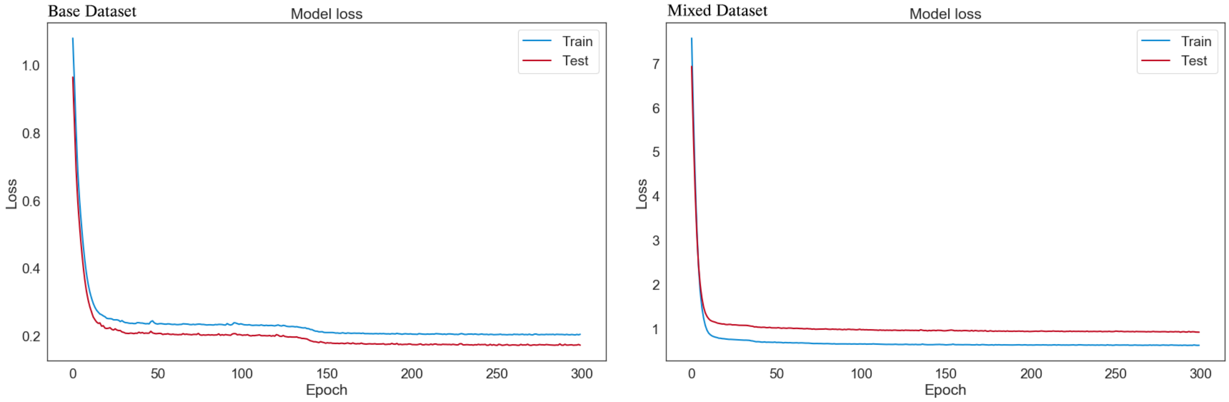


FIGURE 17. AE-S model loss.

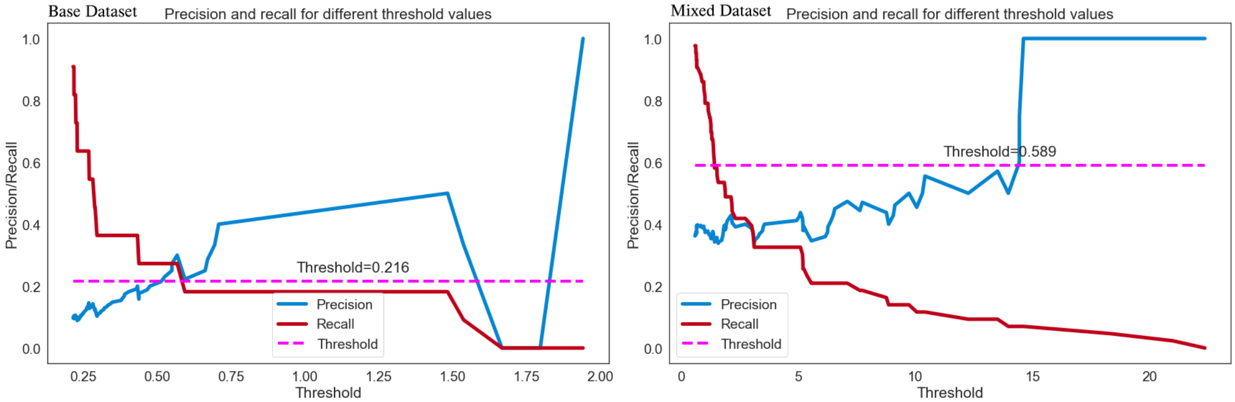


FIGURE 18. AE-S recall-first threshold. The precision/recall value is between 0 and 1.

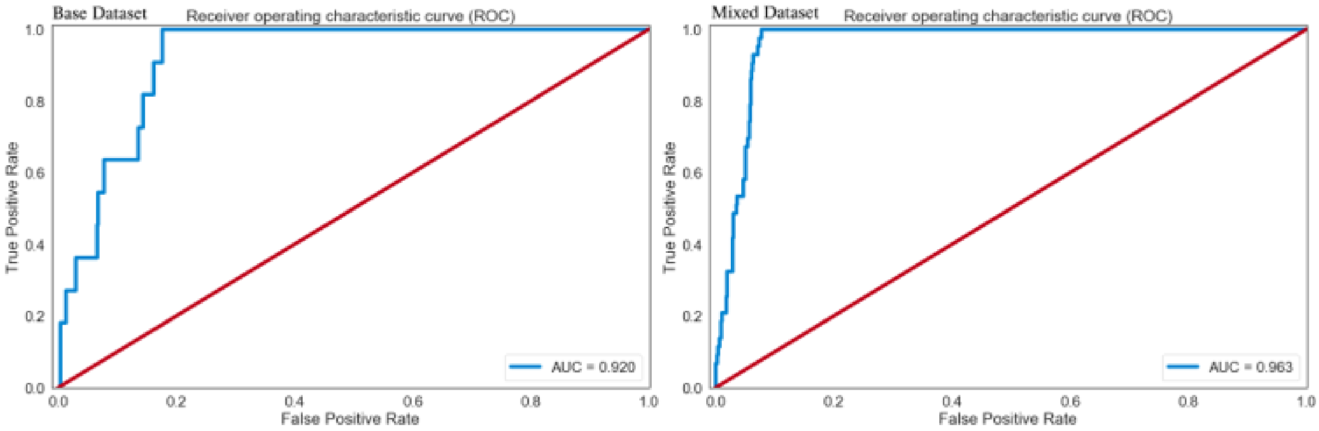


FIGURE 19. **AE-S receiver operating characteristic curve and AUC.**

Finally, the confusion matrix is depicted in Figure 20, based on the fraud scores assigned in the above step. All fraud transactions are identified correctly, which makes the recall 100% for the base dataset. The same goes for the mixed dataset, where all of the 43 fraud transactions are identified correctly.

However, the number of normal transactions that are incorrectly predicted as fraud was about 94 transactions in the base dataset, while decreased to 69 transactions after using the mixed dataset. Moreover, these predictions impacted other measures, such as the FPR, which reached 0.18 when the based dataset was used against 0.07 when the mixed dataset was used. In general, all of the measures are reported in Table 6.

C. AE MULTI-LOSS (AE-M)

As was proposed by this research, the AE-M uses both cross-entropy and MSE loss functions to evaluate the model loss. Results show that the model has a perfect fitting in the base dataset case as Training Loss and Testing Loss are positioned over each other. However, it was relatively over-fitted in the mixed dataset case, as Training Loss went below the Testing Loss, as Figure 22 shows. The RFT scored about 0.229 when the base dataset was used, while it scored 0.554 when the mixed dataset was used, as Figure 23 shows. AUC was lower in the base dataset than the mixed dataset case, as it is reported to be 0.915 and 0.965, respectively, shown in Figure 24.

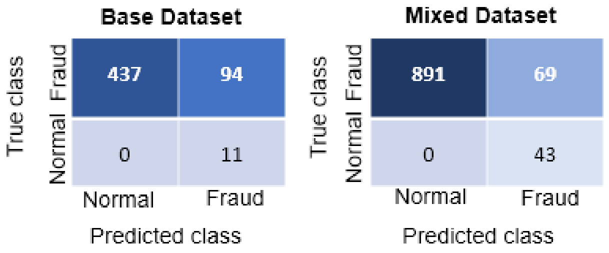


FIGURE 20. **AE-S confusion matrix.**

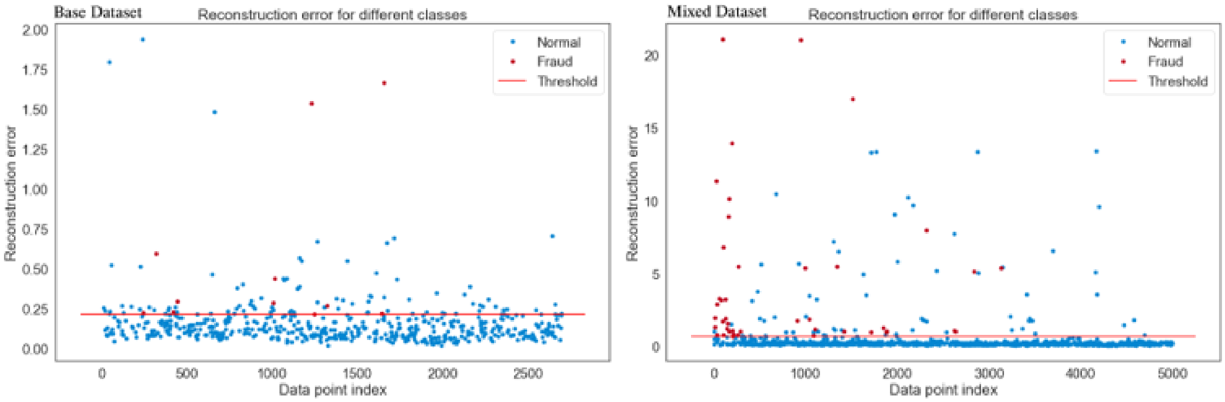


FIGURE 21. **AE-S reconstruction error fraud-score. The reconstruction error is between 0 and 25.**

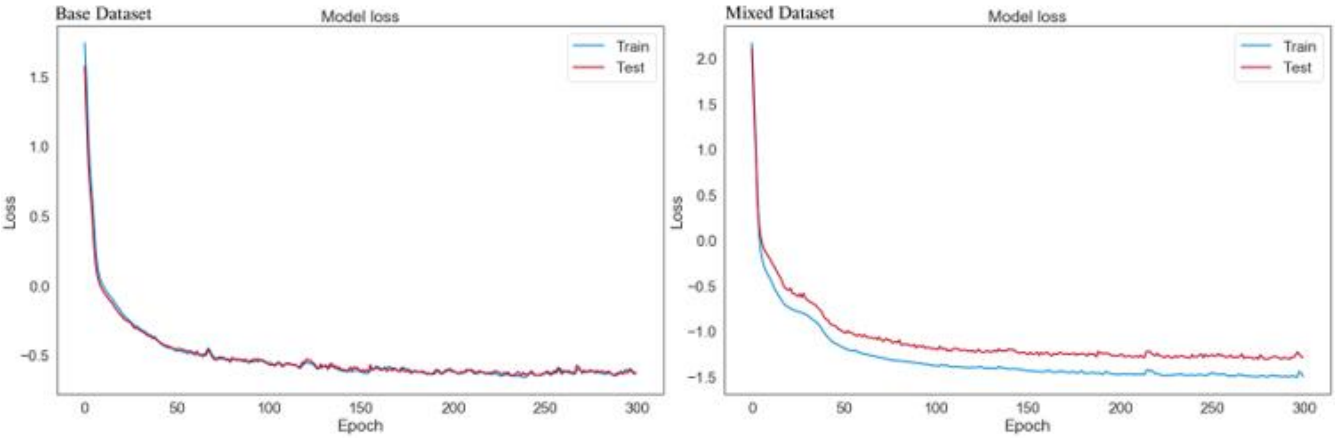


FIGURE 22. AE-M model loss.

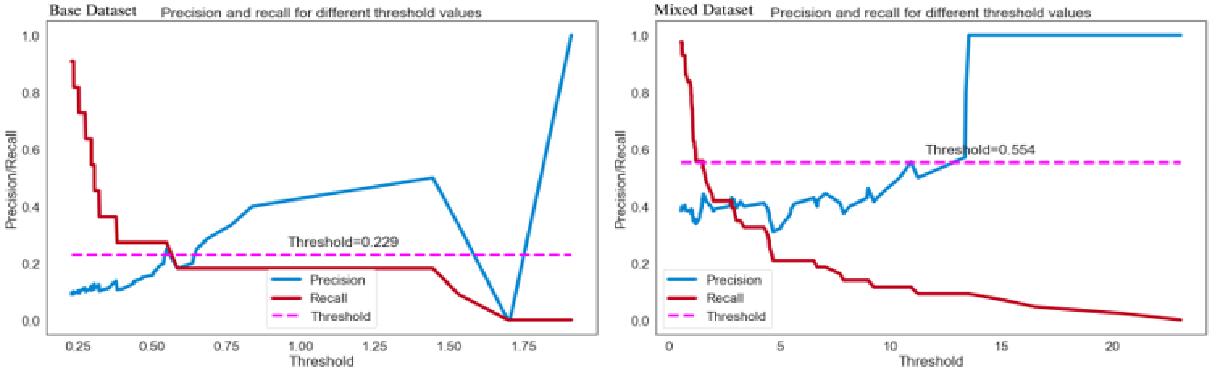


FIGURE 23. AE-M Recall-First threshold. The precision/recall value is between 0 and 1.

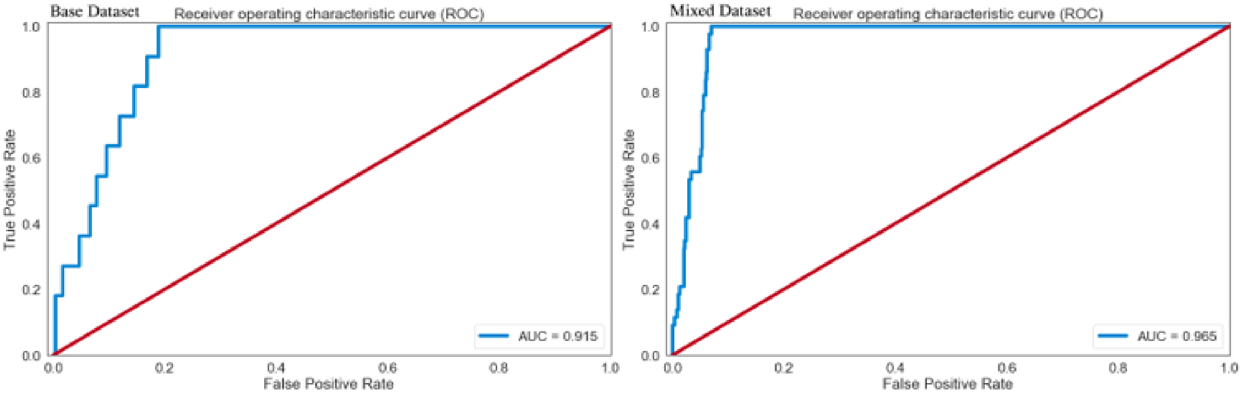


FIGURE 24. AE-M receiver operating characteristic curve and AUC.

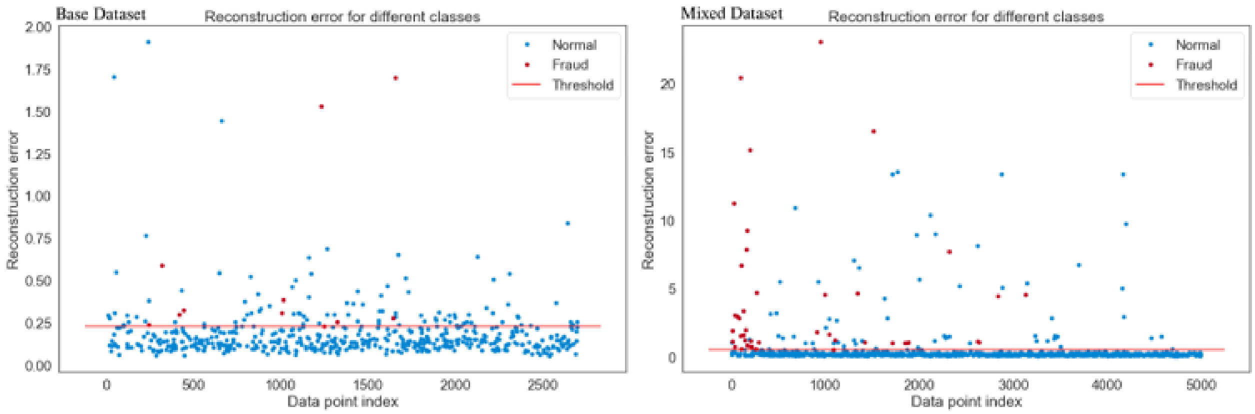


FIGURE 25. AE-M reconstruction error fraud-score. The reconstruction error is between 0 and 25.

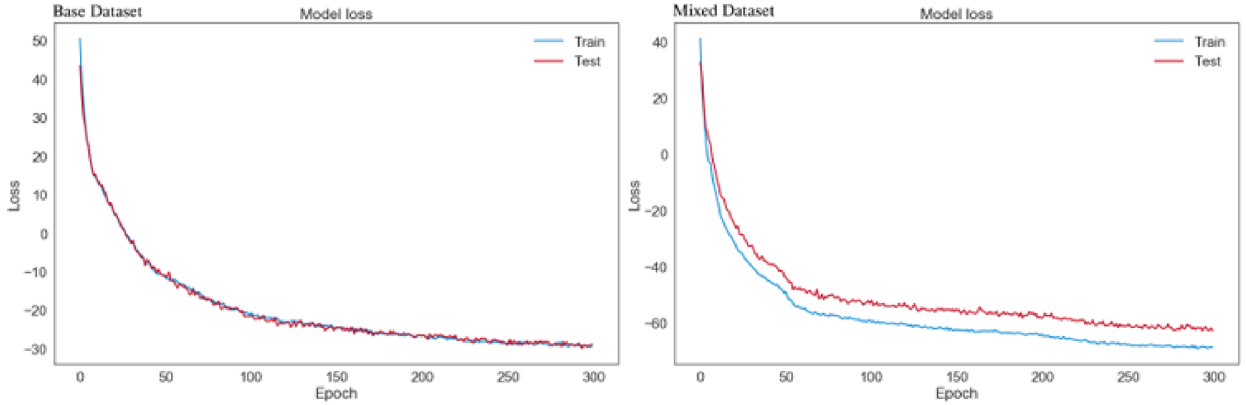


FIGURE 26. VAE model loss.

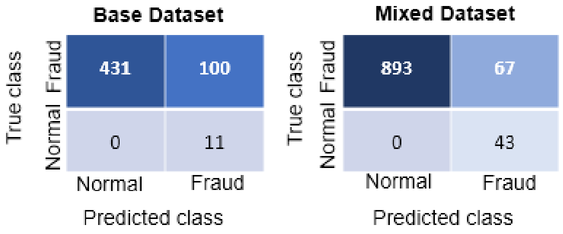


FIGURE 27. AE-M confusion matrix

When AE-M is used to classify the transactions based on their reconstruction error score against the calculated RFT value, results shows that the distribution of the error points was scattered in the base dataset compared to the mixed dataset case, as shown in Figure 25. After the fraud scores were assigned to the transactions, the confusion matrix in Figure 27 is constructed. In both datasets, all the fraud transactions were recalled correctly. In contrast, 100 normal transactions were predicted as frauds in the base dataset case, and 67 normal transactions were incorrectly

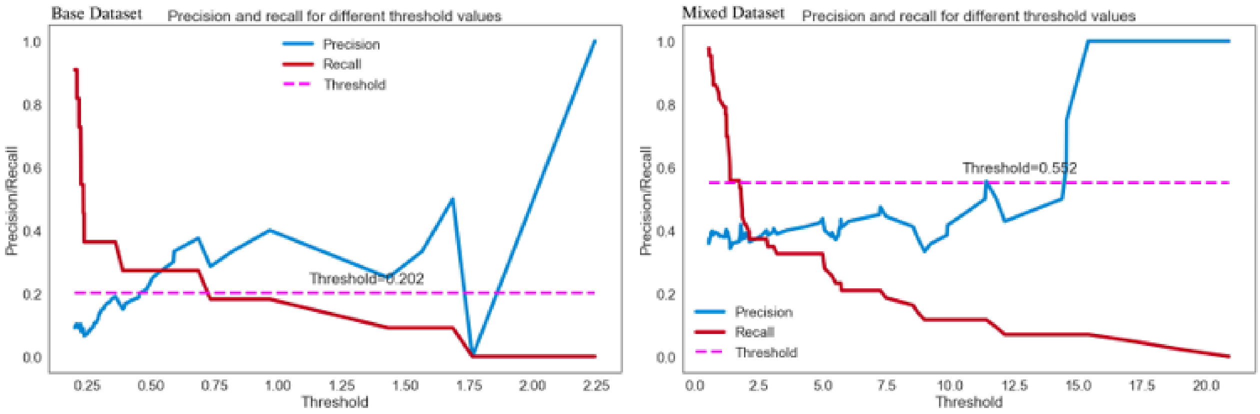
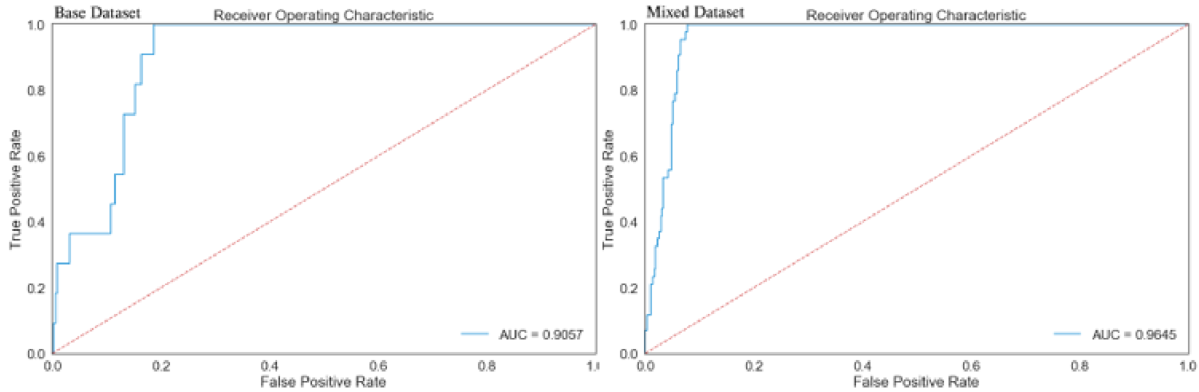


FIGURE 28. VAE recall-first threshold. The precision/recall value is between 0 and 1.



.

FIGURE 29. **VAE receiver operating characteristic curve and AUC.**

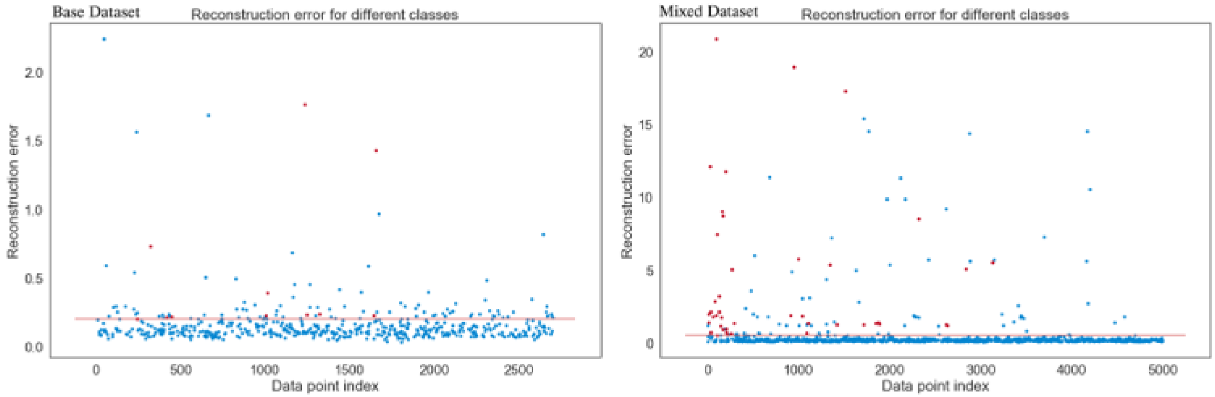


FIGURE 30. **VAE reconstruction error fraud-score.**

classified for the mixed dataset as well, as Figure 27 shows. Among other performance measures that are detailed in Table 6, FPR scored about 0.19 in the base dataset case while it scored 0.07 in the mixed dataset case.

D. VAE

Variational autoencoder showed perfect fitting in the base dataset case, yet the Training Loss went below the Testing Loss when the mixed dataset was used. Hence, it is overfitting, as Figure 26 shows. The RFT was calculated to be 0.202 for the base dataset and 0.552 for the mixed dataset to assign fraud scores to the predicted transactions, as shown in Figure 28.

As Figure 29 shows, AUC reached as high as 0.9645 when the mixed dataset was used, while it decreased to reach 0.9057 when the base dataset was used. Based on the RFT computed value, the fraud score was assigned, showing that the reconstruction error points are more condensed under the threshold when the mixed dataset was used. In contrast, they were scattered under the threshold when the based dataset was used, as Figure 30 shows.

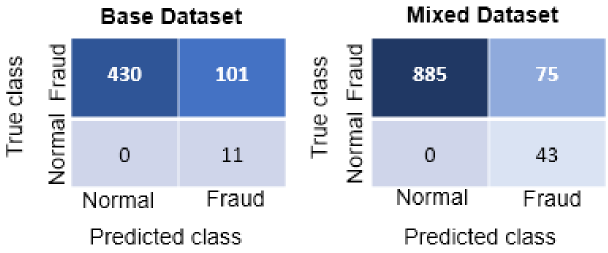


FIGURE 31. **VAE confusion matrix.**

Lastly, the confusion matrix was assembled based on the RFT, where it shows that all the fraud transactions were correctly predicted in both datasets. Moreover, the incorrectly predicted normal transactions decreased from 101 transactions for the base dataset experiment to be 75 transactions in the mixed dataset experiment, as depicted in Figure 31.

Accordingly, FPR decreased from 0.19 in the base dataset to about 0.08 in the mixed dataset experiment. Table 6 shows a summary of all the performance results in the next section.

E. DISCUSSIONS

The mixed dataset shows a better performance in almost every measure, and Table 6 shows the overall indicators in percentage for comparison.

The FPR dropped drastically from around 19% for the base dataset to around 8% for the mixed dataset. This was expected as the number of fraud class transactions in the mixed dataset is almost double. However, the interesting part is that the FP number itself decreased in nearly all experiments, which implies a better model's ability to separate the fraud class from the normal class. The reason for this is that the shape of the reconstruction error distribution that was nicely flattened under the threshold line. Moreover, the proposed threshold (RFT) proved to be a good estimate, as its value was almost doubled from around 0.2 for the base dataset, reaching approximately 0.5 in the mixed dataset case. This increased the model's ability to cover a broader range of the reconstruction error.

In anomaly detection research, a highly unbalanced dataset is always the case, which impacts the performance of certain measures such as precision and accordingly F1 measure; that is, the precision considers the FP count relative to the true positive count. Thus, although the result shows a better precision and F1 measure in the mixed dataset, reaching around 39% and 56%, respectively, these values are still low compared to other studies that are not in the anomaly detection domain. The better alternatives to the F1 measure that can be used in the anomaly detection field are ROC and AUC as the TP/FP ratio issue does not impact them. Therefore, we obtained an AUC value of 96.50% in the mixed dataset, which is still comparable across different research domains.

TABLE VI
OVERALL PERFORMANCE RESULTS IN PERCENTAGE FOR BOTH DATASETS. THE RESULTS SHOW THE MEAN OF RETURNS AVERAGED OVER SIX RUNS. THE STANDARD DEVIATION IS NOT REPORTED SINCE THE VARIATION IS INSIGNIFICANT (STD LESS THAN 0.01).

Name	Acc.	Precision	Recall	F1	AUC	FPR
Base						
AE-S	83	10	100	19	92.00	18
AE-M	82	10	100	18	91.50	19
VAE	81	10	100	18	90.57	19
Mixed						
AE-S	93	38	100	55	96.30	7
AE-M	93	39	100	56	96.50	7
VAE	93	36	100	53	96.45	8

The proposed approach of having a multi-loss function for the autoencoder model shows the best overall result with the mixed dataset, as its FPR went as low as 7%, while its AUC scored 96.50%.

Although the proposed models' overall performance and results were good, some minor overfitting was reported in the mixed dataset even after applying cross-validation, regularization, and dropout techniques. Solving this issue may require access to a bigger dataset. However, in this research, the implementation is mainly focused on the latent vector and the reconstruction error rather than the actual output of the model. Hence, minor overfitting is not expected to have an impact on the model's overall performance.

In this paper, we have designed and implemented a deep learning model that gives state-of-the-art results, in terms of the FPR, RFT, and AUC, for improving the anti-money laundering (AML) process. We also explored recent state-of-the-art deep learning and unsupervised learning techniques such as autoencoder (AE), variational autoencoder (VAE), and generative adversarial network (GAN), and we showed that these techniques can enhance earlier results [7, 8].

Recent works such as Pumsirirat and Yan [42] and Paula et al. [43] both used autoencoders (AEs) to investigate fraud and money laundering. However, for the first time, we demonstrate the applicability and effectiveness of combining AE and VAE with WGAN methods. We use WGAN to generate realistic synthetic fraud transactions to solve the issue of imbalanced class labels, and such additional transactions are then used by the AE/VAE to train the model. Our results indicate that this approach achieves significant improvements for fraud detection.

VI. CONCLUSIONS AND FUTURE WORK

Money laundering is a serious global issue that needs to be addressed, especially considering the fast-growing datasets that need to be evaluated and analyzed. This research attempted to extend the work previously started in 2014 by applying deep learning and unsupervised techniques to improve the anti-money laundering process. More specifically, our system leveraged AE and VAE models. However, as access to the whole dataset is not available anymore, the current study relied on another advanced technique in deep learning called GAN to generate more fraud transactions to produce more reliable models.

To obtain a more balanced dataset, WGAN was used to generate more fraud transactions, which were mixed with the base dataset to produce the mixed dataset. This was then used to train the autoencoders. WGAN performance scored a very high accuracy, reaching about 99%. Hence, the generated fraud transactions were almost identical to the real fraud transactions.

Experimental results show that even with the base dataset, the proposed models performed better than the original research as it helped decrease the FPR to reach around 18%. However, using the mixed dataset, the results were even better as the FPR was reduced to 7%. Other measures were enhanced, such as accuracy, which increased to 93%, and AUC, which reached 96.50%. Results also show that the proposed multi-loss function autoencoder performed better than the other models.

It is worth mentioning that the model’s loss in the mixed dataset case was slightly over-fitted. Hence, additional data may be required to overcome this issue in the future. However, as this implementation mainly focused on the latent vector and the reconstruction error rather than the actual output of the models, minor overfitting is not expected to impact the model's overall performance.

Money laundering inherently possesses complicated characteristics, for example, the layering phase in which launderers distribute money between the different accounts while trying to hide their sources. Capturing such a pattern will require additional work such as the graph and social network analysis along with the deep learning and unsupervised techniques proposed by this study.

Despite the promising results, there is still some space for enhancement. This could be achieved if access to a bigger dataset is secured along with an in-depth interpretation for the dataset attributes.

REFERENCES

[1] Money Laundering, FATF. Accessed on: May 01, 2019. [Online]. Available: https://www.fatf-gafi.org/faq/moneylaundering.

[2] S. Surendran and M. Ramasamy. "Consequences of money laundering in banking sector," *Sains Humanika*, vol. 64, no. 2, 2013.

[3] I. E. Bekhouche, "Money laundering in Malaysia, regulations and policies," *Intl. J. of Law*, vol. 4, no. 2, pp. 22-26, 2018.

[4] K. Dugan, Italian bank will pay $1.3B fine for money laundering, New York Post. Accessed on: May 01, 2019. [Online]. Available: https://nypost.com/2019/04/15/italian-bank-will-pay-1-3b-fine-for-money-laundering.

[5] K. Dugan, UK bank hit with $1B in fines for helping Iran launder money, New York Post. Accessed on: May 01, 2019. [Online]. Available: https://nypost.com/2019/04/09/uk-bank-hit-with-1b-in-fines-for-helping-iran-launder-money

[6] Reuters. Morgan Stanley fined $10M for anti-money laundering failures, New York Post. Accessed on: May 01, 2019. [Online]. Available: https://nypost.com/2018/12/26/morgan-stanley-fined-10m-for-anti-money-laundering-failures.

[7] Z. Chen *et al.*, "Exploration of the effectiveness of expectation maximization algorithm for suspicious transaction detection in anti-money laundering," in *Porc. 2014 IEEE ICOS*, 2014, pp. 145-149.

[8] Z. Chen *et al.*, "Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: A review," Knowl. and Inf. Sys., vol. 57, no. 2, pp. 245-285, 2018.

[9] Financial Stability Board, "Artificial intelligence and machine learning in financial services – Market developments and financial stability implications,", Financial Stability Board, pp. 45, Nov 01, 2017. [Online]. Available: http://www.fsb.org/2017/11/artificial-intelligence-and-machine-learning-in-financial-service/

[10] R. Wedge *et al.*, "Solving the false positives problem in fraud prediction using automated feature engineering," in *Proc. Joint Eur. Conf. on Mach. Learn. and Know. Disc. in Databases*, pp. 372-388, 2018.

[11] L. Rojas, E. Alonso, and S. Axelsson, "Multi agent based simulation (MABS) of financial transactions for anti-money laundering (AML)," in *Proc. Secure IT Systems - 17th Nordic Conf.*, NordSec 2012, Karlskrona, Sweden, 2012.

[12] Y. Şahin and E. Duman, "Detecting credit card fraud by decision trees and support vector machines," in *Proc. IMECS*, Hong Kong, 2011.

[13] V. Jayasree and RV S. Balan. "Money laundering regulatory risk evaluation using Bitmap Index-based Decision Tree." *Journal of the Association of Arab Universities for Basic and Applied Sciences*, vol. 23, no. 2017, pp. 96-102, 2017.

[14] S. Wang and J. Yang. "A money laundering risk evaluation method based on decision tree," in *Proc. 2007 IEEE Int. Conf. on Machine Learning and Cyber.*, vol. 1, 2007, pp. 283-286.

[15] D. Savage, Q. Wang, P.L. Chou, X. Zhang, and X. Yu "Detection of money laundering groups using supervised learning in networks." *arXiv preprint arXiv:1608.00708*, 2016.

[16] Y. Zhang and P. Trubey, "Machine learning and sampling scheme: An empirical study of money laundering detection," *Comput. Econ*, vol. 54, pp. 1043-1063, 2019.

[17] L.T. Lv, N. Ji, and J. Zhang. "A RBF neural network model for anti-money laundering." in *Proc. 2008 Int. Conf. on Wav. Anal. and Patt. Recog*., Hong Kong, China, 2008, pp. 209-215.

[18] E. Badal-Valero, J.A. Alvarez-Jareño, and J.M. Pavía, "Combining Benford's Law and machine learning to detect money laundering. An actual Spanish court case." *Forensic Sc. Int.*, vol. 282, pp. 24-34, 2018.

[19] A. Chouiekh and E.H. Haj, "Convnets for fraud detection analysis," *Proc. Comp. Sc.*, vol. 127, pp. 133-138, 2018.

[20] Z. Zhang, J .J. Salerno, and P. S. Yu, "Applying data mining in investigating money laundering crimes," in *Proc. of the Ninth ACM SIGKDD*, 2003, pp. 747-752.

[21] D. K. Luna, G. K. Palshikar, M. Apte, and A. Bhattacharya, "Finding shell company accounts using anomaly detection," in *Proc. of the ACM India Joint Int. Conf. on Data Sc. and Manag. of Data*, 2018, pp. 167-174.

[22] A. Claudio and J. Balsa, "Client profiling for an anti-money laundering system," *arXiv preprint arXiv:1510.00878*, 2015.

[23] M. Hegazy, A. Madian, and M. Ragaie, "Enhanced fraud miner: credit card fraud detection using clustering data mining techniques," *Egyp. Com. Sc. Journal*, vol. 40, no. 3, pp. 72-81, 2016.

[24] J. Tang and J. Yin, "Developing an intelligent data discriminating system of anti-money laundering based on SVM," in *Proc. 2005 IEEE Int. Conf. on Machine Learning and Cyber.*, vol. 6, 2005, pp. 3453-3457.

[25] T. Yingjie *et al.*, "Ramp loss one-class support vector machine; A robust and effective approach to anomaly detection problems," *Neurocomputing*, vol. 310, pp. 223-235, 2018.

[26] W. An, M. Liang, and H. Liu, "An improved one-class support vector machine classifier for outlier detection," in Proc. of the Inst. of Mech. Eng., Part C: J. of Mech. Eng. Sc., vol. 229, no. 3, 2015, pp. 580-588.

[27] D. R. Wilson and T. R. Martinez, "Improved heterogeneous distance functions," *J. of AIR*, vol. 6, no. 1, pp.1-34, 1997.

[28] K. Chitra and B. Subashini, "Data mining techniques and its applications in banking sector," *Int. J. of Emerg. Tech. and Adv. Eng*., vol. 3, no. .8, pp. 219-226, 2013.

[29] D. K. Cao and P. Do, "Applying data mining in money laundering detection for the Vietnamese banking industry," in *Proc. of ACIIDS*, 2012, pp. 207-216.

[30] V. Zaslavsky and A. Strizhak, "Credit card fraud detection using self-organizing maps," *Information and Security*, vol. 18, pp. 48-63, 2006.

[31] S. Engardt, "Unsupervised learning with mixed type data for detecting money laundering," M.S. Thesis, School of Elect. Eng. and Comp. Sc., KTH Royal Institute of Technology, Stockholm, Sweden, 2018.

[32] X. Liu, P. Zhang, and D. Zeng, "Sequence matching for suspicious activity detection in anti-money laundering," in *Proc. Int. Conf. on Intell. and Sec. Inform.*, 2008, pp. 50-61.

[33] N. A. Le Khac and M. Kechadi, "Application of data mining for anti-money laundering detection: A case study", in *Proc. of ICDM*, 2010, pp. 577–584.

[34] G. Shabat, D. Segev, and A. Averbuch, "Uncovering unknown unknowns in financial services big data by unsupervised methodologies: Present and future trends," in *Proc. of KDD 2017 Workshop on Anomaly Detection in Finance*, vol. 71, 2018, pp. 8-19.

[35] K. A. Shaikh and A. Nazir, "A model for identifying relationships of suspicious customers in money laundering using social network functions," in *Proc. of the World Cong. on Eng*., vol. 1, 2018, pp. 1-4.

[36] A. F. Colladon and E. Remondi, "Using social network analysis to prevent money laundering," *Expert Systems with Applications*, vol. 67, pp. 49-58, 2017.

[37] I. Molloy *et al*., "Graph analytics for real-time scoring of cross-channel transactional fraud," in *Proc. of Int. Conf. on Fin. Crypt. and Data Sec.*, 2016, pp. 22-40.

[38] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2.1, pp. 1-18, 2015.

[39] B. Kasra, Z. Chen, and T. Maul. "Data augmentation by autoencoders for unsupervised anomaly detection," arXiv preprint arXiv:1912.13384, 2019.

[40] B. Kasra, Z. Chen, and T. Maul, "Detecting point outliers using prune-based outlier factor (PLOF)," arXiv preprint arXiv:1911.01654, 2019.

[41] A. Malathi, J. Amudha and P. Narayana, "A prototype to detect anomalies using machine learning algorithms and deep neural network," *Comp. Vis. and Bio Insp. Comp.*, pp. 1084–1094, 2018.
[42] A. Pumsirirat and L. Yan, "Credit card fraud detection using deep learning based on auto-encoder and restricted Boltzmann machine', *Int. J. of Adv. Comp. Sc. and Appl.*, vol. 9, no. 1, pp. 18–25, 2018.
[43] E. L. Paula, M. Ladeira, R. N. Carvalho, and T. Marzagão, "Deep learning anomaly detection as support fraud investigation in Brazilian exports and anti-money laundering," in *Proc. 15th IEEE Int. Conf. on Mach. Learn. and Appl.*, 2016, pp. 954–960.
[44] I. Vasilev *et al.*, "Python Deep Learning" Packt Publishing, 2019. [Online]. Available: https://www.packtpub.com/product/python-deep-learning-second-edition/9781789348460
[45] D. P. Kingma and M. Welling, "Auto-encoding Variational Bayes", in *Proc. ICLR*, 2014, pp. 1–14.
[46] R. Atienza, *Advanced Deep Learning with Keras*, Packt Publishing, pp. 369, 2018. [Online]. Available: http://www.packtpub.com.
[47] J. Jordan, Variational Autoencoders, Accessed on: August 15, 2018. [Online]. Available: https://www.jeremyjordan.me/variational-autoencoders.
[48] I. J. Goodfellow *et al*., "Generative Adversarial Networks", pp. 1–9, 2014. [Online]. Available: http://arxiv.org/abs/1406.2661.
[49] P. Spyridon and Y.S. Boutalis, "Generative adversarial networks for unsupervised fault detection," in *Proc. 2018 Eur. Control Conf.* (*ECC*), 2018, pp. 691–696.
[50] J. Klaas, Machine Learning for Finance, Packt Publishing, Accessed on: August 05, 2019. [Online]. Available: https://github.com/PacktPublishing/Machine-Learning-for-Finance.
[51] M. Arjovsky, C. Soumith, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, 2017, pp. 214-223.
[52] L. Weng, "From GAN to WGAN," Accessed on: August 18, 2019. [Online] Available: https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html
[53] Arjovsky, M., Chintala, S. and Bottou, L., 2017, July. Wasserstein generative adversarial networks. In International conference on machine learning (pp. 214-223). PMLR.

Zhiyuan Chen received the M.Phil. and a Ph.D. in Computer Science from the University of Nottingham in 2007 and 2011, respectively. Since 2012, she has been an Assistant Professor with the University of Nottingham Malaysia (UNM), School of Computer Science. She works as Principal Consultants for many industrial and research organizations. Before joining UNM, she has been a research associate in the UK Horizon Digital Economy Research Institute. Her research interests are in the area of Machine Learning, Data Mining, Deep Learning, and Anomaly Detection.

Waleed Mahmoud Soliman received his Ph.D. in economics & management from China Agricultural university in 2015. He received the M.S. degrees in agri. economics & management from Minia University in 2008 and the M.S. degrees in computer science from the University of Nottingham in 2020. His research interests include Macro Economics, Artificial Intelligence (AI), Machine Learning, Data Science, and Big Data.

AMRIL NAZIR is an Associate Professor at the College Technological Innovation, Zayed University, and the Consulting Director / Chief Architect at CODECOMPASS LLP. He was formerly a Senior Research Scientist for the Malaysian R&D institute for 9 years. His research interests include Artificial Intelligence (AI), Machine Learning, Data Science, and Big Data.

MOHAMMAD SHORFUZZAMAN is currently an Associate Professor with the Department of Computer Science, College of Computers and Information Technology (CCIT), Taif University, Taif, Saudi Arabia. He is also a member of the Big Data Analytics and Applications (BDAAG) Research Group, CCIT. His current research interests include applied Artificial Intelligence in the areas of computer vision, natural language processing, big data, and cloud computing.