Leveraging Generative Adversarial Networks for Unsupervised Fraud Detection

Sunil Pradhan Sharma*, **Elakkiya Daivam****
*Senior Lead Software Engineer, Capital One Services, LLC
**Lead Software Engineer, Capital One Services, LLC

***Abstract*—** This study looks into two main areas related to detecting fraud without using labeled data. The first area focuses on using evaluation methods from Generative Adversarial Networks (GANs) to spot fraud or outliers by calculating the differences between samples from normal (non-fraudulent) data and generated data. The second area explores whether the discriminator of a GAN can be used as a feature space for calculating these differences. Since fraudulent examples are not available during training, the problem is approached as finding outliers. The GAN is trained using only non-fraudulent data, which helps create a feature space in the discriminator. Then, the study calculates the differences between known non-fraudulent data and unknown samples to identify potential fraud. The study also includes three experiments to test how factors like the type of GAN model, dimensionality reduction, and the percentage of fraud in the data affect the performance of fraud detection. The results offer valuable insights into how GAN-based metrics and feature spaces can be used to detect fraud even without labeled fraudulent data.

***Index Terms*— Fraud, Financial, Generative Adversarial Networks, Outliers**

I. Introduction

This study explores two primary topics. The first topic focuses on evaluating fraud detection methods using measures associated with Generative Adversarial Networks (GANs) [1], [2], [3], [4], [5], [6]. These measures assess the difference, or distance, between two groups of data: one group representing the original target population and another made up of data generated by a GAN. The second topic examines whether the discriminator component of a GAN can be effectively used as a feature space. In this feature space, the distance measures mentioned earlier could be calculated to aid in fraud or anomaly detection. The research takes place in the context of unsupervised fraud detection. This approach assumes there are no labeled examples of fraudulent data during the training phase. As a result, the problem is treated as one of identifying outliers. Outliers are data points that differ significantly from the normal pattern. In this setting, distance measures are calculated for new data points, and these distances help determine whether the data points are outliers. If they are classified as outliers, they are considered fraudulent. However, the process begins by training GAN models on the available data, which is assumed to consist only of non-fraudulent observations. The goal of this training is to use the discriminator component of the GAN to create a feature space. The discriminator is a part of the GAN that learns to differentiate between real and generated data during training. In the created feature space, the distances between known non-fraudulent data and unknown observations are calculated. These unknown observations may include both non-fraudulent and fraudulent data. By analyzing these distances, the method aims to identify potential fraud.

Three main experiments are conducted to study how different factors influence the behavior of these distance measures. The study investigates how the choice of the GAN model affects the accuracy and reliability of the distance measures in detecting outliers. Secondly, high-dimensional data can be challenging to work with, as it may lead to noisy or less interpretable results. Dimensionality reduction techniques are applied to simplify the data, and the impact of these techniques on distance measures is evaluated. Thirdly, the proportion of fraudulent observations: The presence of fraudulent data within the dataset could influence how well the model detects outliers. The experiments explore how varying the proportion of fraudulent data affects the calculated distances and the model's ability to classify fraud accurately. By focusing on these factors, the study aims to enhance the understanding of how GAN-based methods can be applied to fraud detection in an unsupervised setting.

The study is as follows; the relevant works will be shown in the next section. The materials and method are described in Section III. The experimental analysis and result analysis are carried out in Section IV, and in Section V, we wrap up the study with some conclusions and recommendations for further research.

II. Related Works

Generative models are advanced tools in Machine Learning (ML) designed to generate data points based on a target distribution. For instance, they can create samples that mimic a target data distribution according to [7]. Additionally, they can extend to generate data conditioned on labels following the distribution. Broadly, generative models are classified into two main types: explicit [8] and implicit [8] models. Explicit models explicitly define the data distribution using a parametric form. They often include a log-likelihood function that measures how well the data fits the model. The goal is to maximize the likelihood or a close approximation of it. Some explicit models have tractable density functions, making them easier to optimize, while others use approximations to handle intractable densities. For instance, PixelRNN [9]v and PixelCNN [10]: These models focus on generating images pixel by pixel. Each

image is treated as a grid of pixels, which is rearranged into a sequence. These models estimate the joint probability distribution of all pixels using the product of conditional probabilities. In both approaches, images are generated sequentially, starting from one corner. Similarly, Variational Autoencoders (VAEs) [11] are similar to autoencoders but impose additional constraints to learn a structured latent representation of the data. A standard autoencoder compresses input data into a smaller representation and then reconstructs it. VAEs, however, map the input into a latent space described by a mean and variance. To generate data, VAEs sample from this latent distribution and use a decoder to map these samples back to the original data space. If trained effectively, observations close in the latent space correspond to similar observations in the original data. A key technique in VAEs is the reparameterization trick, which ensures gradients are differentiable during training. Unlike explicit models, implicit models do not define a specific probability distribution or log-likelihood function. Instead, they rely on stochastic procedures to generate data directly. GANs [12] are a popular type of implicit model. They involve two neural networks: a generator and a discriminator. The generator transforms random noise into samples resembling the target distribution, while the discriminator tries to distinguish between real and generated data. Over time, the generator learns to create more realistic samples, effectively mimicking the target distribution. Since their introduction in 2014, GANs have seen numerous variations and improvements. These advancements target different aspects, such as stability during training and better-quality outputs. While GANs are powerful, they are also challenging to train due to issues like mode collapse and instability [13], [14], [15], [16], [17], [18], [19], [20], [21].

Evaluating generative models is crucial, particularly to measure the quality and diversity of generated data. For image-based models like GANs, evaluation methods are generally categorized as quantitative or qualitative. Quantitative numerically assess the performance of generative models. One popular metric is the Inception Score (IS), which evaluates the quality of generated images based on a pre-trained image classification model. While widely used, IS has limitations, such as its inability to account for similarities between the generated and real data distributions. Whereas, the qualitative measures involve human judgment or visual inspection of the generated samples. However, they are subjective and may lack consistency. To address these limitations, some measures focus on comparing the distributions of real and generated data. For instance, metrics like the Fréchet Inception Distance (FID) compute the distance between feature representations of real and generated images. A lower FID indicates closer similarity between distributions. Generative models, particularly GANs, have demonstrated impressive results in areas like image generation [22], text synthesis [23], and fraud detection [24]. However, challenges persist, especially during the training process. For instance, achieving stable training in GANs requires careful tuning of hyperparameters and loss functions. For tasks such as fraud detection, where the generated data's distribution must align closely with the real data, evaluation metrics that compare distributions are more appropriate.

III. Materials and Methods

This section outlines the experiments conducted to explore various behaviors and gain insights into specific research questions. The experiments were designed to assess the performance of models and investigate different aspects of distance metrics. Before delving into the experimental details, it is important to describe the environment in which they were performed. This includes the hardware and software configurations used, providing clarity about the computational resources and facilitating reproducibility. The experiments were conducted on a desktop computer with the hardware such as i) CPU: AMD Ryzen 2600x, ii) RAM: 32GB (2 x 16GB at 3200MHz), iii) GPU: Nvidia GTX1080 with 8GB VRAM, and iv) Storage: Samsung 970 Evo 500GB NVMe SSD. The machine ran on Microsoft Windows 10 and utilized Nvidia's CUDA toolkit (Version 11) and the CUDA Deep Neural Network library (cuDNN, Version 8) to enable GPU-accelerated computing. The experiments were implemented in Python (Version 3.7) with the key libraries such as i) TensorFlow 2[1]: Used for building and training neural network models. It also facilitated parallel computation on the GPU, ii) NumPy[2]: Used for efficient matrix operations, iii) Pandas[3]: Used to create and manipulate data frames for organizing results, iv) Matplotlib and Seaborn[4]: Employed for visualizing experimental results, and v) Python Optimal Transport[5]: Implemented the Wasserstein distance[6] metric for the analysis. Additional tools like Graphviz[7] and PyDot[8] were used to generate visual representations of GAN architectures but were not essential for running the experiments. To understand the behavior of distance measures such as Wasserstein distance, Kernel Maximum Mean Discrepancy[9] (MMD), and Frechet distance[10], an initial experiment was conducted using controlled multivariate datasets. The experiment used four multivariate distributions, each with three variables. The first two variables shared identical Gaussian distributions, while the third variable followed a distinct Gaussian distribution for each dataset. A key focus of the experiments was to evaluate how feature spaces influence distance metrics. Data exists within a multi-dimensional space, where each dimension represents a feature or variable. For certain data types, such as images, raw features (e.g., pixel values) carry limited meaning. Transforming this data into a meaningful feature space helps extract relevant characteristics for analysis.

[1] https://www.tensorflow.org/install
[2] https://numpy.org/
[3] https://pandas.pydata.org/
[4] https://techifysolutions.com/blog/seaborn-vs-matplotlib/
[5] https://pythonot.github.io/
[6] The Wasserstein distance, which emphasizes geometric relationships, quantifies the least amount of work needed to convert one probability distribution into another.
[7] https://graphviz.org/
[8] https://pypi.org/project/pydot/
[9] By comparing the means of two probability distributions in a replicating kernel Hilbert space, MMD calculates the difference between them.
[10] Frechet distance calculates the least amount of work required to align two curves while maintaining their order, thereby indicating how similar they are.

A. Dataset Analysis

This experiment utilizes the MNIST[11] dataset, a collection of black and white images representing handwritten digits. Each image is 28 x 28 pixels in size with one depth dimension, giving each observation a shape of (28, 28, 1). The digits range from 0 to 9, and their frequency is evenly distributed. The dataset comprises two parts: 50,000 training samples and 10,000 test samples. The purpose of this study is to analyze how different distance measures behave in various feature spaces. To make the comparison more interesting, we expand the dataset by creating two additional datasets: i) Shuffled Pixel Dataset: Derived from the MNIST test data, where the pixel positions are randomly rearranged, and ii) Random Pixel Dataset: Composed of images with pixel values generated randomly from a uniform distribution. These datasets allow us to explore how pixel arrangements and distributions impact distance metrics. For humans, it's easy to distinguish between original and shuffled images based on pixel arrangement. However, the pixel value distributions in both datasets are identical, making the difference invisible without considering their positions. Meanwhile, the randomly generated dataset has a completely different distribution. We compare distance measures in two main feature spaces: i) Original Pixel Space: Directly uses the pixel values of the images, and ii) Feature Spaces from GAN Discriminator Networks: The penultimate layers of GAN discriminators generate these feature spaces. GAN discriminators are trained to differentiate real images from fake ones. Their learned features could result in distinct activation patterns for the original and shuffled datasets, though this is not guaranteed and requires investigation. Since GANs function as black boxes, we aim to uncover whether they detect meaningful differences in pixel arrangements. By comparing the sensitivity of distance measures in these feature spaces, we can better understand their effectiveness. To calculate distances, images need to be converted into a 2D matrix format. Each image is flattened into a single row vector containing all its pixel values. These vectors are stacked to form a 2D matrix where each row represents an observation and each column corresponds to a pixel (variable). For GAN-generated feature spaces, the activation values from the discriminator layers are already in vector form and are similarly arranged into rows. We also explore Principal Component Analysis[12] (PCA) and Uniform Manifold Approximation and Projection[13] (UMAP) to reduce the feature space dimensions. High-dimensional data can slow down distance metric calculations, so investigating how much information is retained after dimensionality reduction is important.

B. Model Analysis

GANs are powerful tools for generating data, but their performance can vary based on the architecture and training methods used. This study aims to explore how these differences influence the values calculated in the activation layer feature space. To achieve this, we will work with three distinct GAN models: Deep Convolutional GAN (DCGAN), Least Squares GAN (LSGAN), and Wasserstein GAN (WGAN), each employing unique architectures and training techniques. The goal is to analyze the discriminators of these models after training and assess how their architectural differences affect the feature space. Each model is trained using batch training with a batch size of 128 observations. The DCGAN model we use is based on the designed with upsampling, convolutional, batch normalization, and ReLU activation layers. These layers are repeated twice, creating a network structure that progressively transforms random noise into realistic data samples. Whereas, the discriminator includes convolutional layers, LeakyReLU activation, dropout layers, and batch normalization. Its primary role is to differentiate between real and generated data by analyzing patterns in the input. LSGAN architecture is based on the traditional GANs, it uses a least-squares loss function to improve training stability and encourage the generator to produce samples close to the real data distribution. WGAN is designed to address instability and mode collapse issues in traditional GANs by replacing the standard loss function with a Wasserstein distance metric. All three GAN models are trained using batch training with a batch size of 128 observations. This method processes small groups of data samples at a time, allowing the models to learn effectively while managing computational resources. The focus of this investigation is on the activation layer values within the discriminators of the three GAN architectures. By comparing these values, we aim to determine how differences in architecture and training methods influence the feature space. This analysis is critical because the activation layer feature space plays a significant role in how well the discriminator can differentiate real from generated data.

IV. Experimental Analysis

This study explores how distance metrics can be applied to detect fraud. The experiments utilize the CIFAR10[14] and CIFAR100[15] datasets, which contain 32 x 32-pixel color images. While CIFAR10 includes images from 10 categories, CIFAR100 spans 100 distinct categories with no overlapping subjects. Despite some similarities between categories, like a bus and a tractor (both vehicles), they remain fundamentally different. For the experiments, the CIFAR10 dataset represents "real" or "non-fraudulent" data, and the CIFAR100 dataset represents "fake" or "fraudulent" data. The CIFAR10 data is divided into two subsets: i) Trueknown: Used for training GAN models, generating feature spaces, and creating reference distributions, and ii) Trueunknown: Supplies new "real" observations for testing. The CIFAR100 data is labeled as unknown, simulating fraudulent data, and excluded from training. The goal is to evaluate how distance measures behave within feature spaces to distinguish between fraudulent and non-fraudulent observations. DCGAN is employed for feature generation due to its superior image quality in prior tests. Additionally, an InceptionV3 model, pre-trained on ImageNet, provides another feature space for comparison. Two types of fraud detection are examined: i) Multiple Observations:

[11] https://www.kaggle.com/datasets/hojjatk/mnist-dataset

[12] By finding orthogonal components that maximize variance, PCA decreases the dimensionality of data while maintaining important properties in complicated datasets.

[13] A dimensionality reduction method called UMAP maintains both local and global data structures for display.

[14] https://www.kaggle.com/c/cifar-10/

[15] https://www.kaggle.com/datasets/fedesoriano/cifar100

Analyzing groups of samples to detect fraud, and ii) Single Observation: Investigating fraud within individual observations. The experiment tests the sensitivity of distance metrics to varying proportions of fraudulent data in a sample. This experiment aims to identify patterns, such as linear or exponential relationships, between fraud proportions and distance values. These insights help evaluate the feasibility of fraud detection models based on distance metrics. For individual observations, the distance metrics require data in a multivariate format. To achieve this, single observation feature vectors are transformed into matrices through: i) Splitting: Dividing the vector into smaller parts to form a matrix, and ii) Repeating with Noise: Duplicating the vector with slight variations, though this approach is complex and may distort information.

A. Result Analysis

In this experiment, we calculated the distances between pairs of samples from four different multivariate distributions: XAX_A, XBX_B, XCX_C, and XDX_D. To better understand the results, we analyzed the distributions of values for the X3X_3 variable in all four samples. These distributions are shown in Fig. 1. The key findings from the experiment are summarized in Table I, which provides the average distances between the distributions for 100 samples along with their 95% confidence intervals. The results mostly followed expected patterns. The average distance was smallest between the distributions that were most similar and largest between those that were the most different. However, there was an exception with the Kernel MMD metric. For this metric, the average distance between the distributions N(1,2)N(1, 2) and N(4,2)N(4, 2) was greater than the average distance between N(1,2)N(1, 2) and N(−2,4)N(-2, 4). For the other distance metrics, the distances consistently increased as the distributions became more distinct. One interesting observation was that, across all metrics, the average distance grew more significantly when the mean changed by 2 (as in N(1,2)N(1, 2) to N(4,2)N(4, 2)) compared to when the standard deviation increased by 2 (as in N(1,2)N(1, 2) to N(1,4)N(1, 4)). This suggests that a shift in the mean has a greater impact on how different the distributions appear. This makes sense because when the mean shifts, a larger portion of the distribution moves outside the original range. In contrast, an increase in the standard deviation mostly stretches the distribution while keeping much of it within the same range. Examining the 95% confidence intervals provided additional insights. For the Wasserstein and Kernel MMD distances, the confidence intervals for the first three distributions did not overlap, indicating clear distinctions between these distributions. However, for the Frechet distance, only the first two distributions had non-overlapping confidence intervals, suggesting that the Frechet distance might be more variable compared to the other two metrics. For all three distance measures, the confidence intervals for the last two distributions overlapped, indicating less distinction between them. This overlap could explain the unusual behavior observed with the Kernel MMD metric. It is also important to consider that the samples in this experiment came from multivariate distributions. The random variables X1X_1, X2X_2, and X3X_3 in each sample were also generated randomly, and their values could have influenced the distance measurements. This variability in the underlying data adds complexity to interpreting the results but also highlights the dynamic nature of these metrics in capturing differences between distributions.

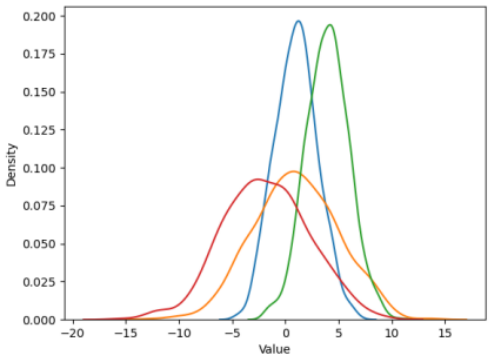


Fig. 1. The X3 random variables' various distributions

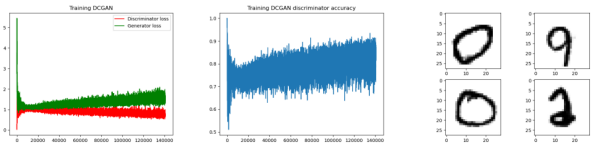
TABLE I
DISTANCE METRICS BETWEEN DISTRIBUTIONS

Distance	Wasserstein	kMMD	Frechet
N(1, 2) to N(1, 2)	0.89 (0.791, 1.034)	0.077 (0.041, 0.132)	0.229 (0.058, 0.572)
N(1, 2) to N(1, 4)	1.839 (1.403, 2.345)	0.254 (0.173, 0.341)	4.167 (2.168, 7.082)
N(1, 2) to N(4, 2)	3.045 (2.591, 3.651)	0.517 (0.438, 0.618)	9.152 (6.51, 13.168)
N(1, 2) to N(−2, 4)	3.320 (2.63, 4.17)	0.455 (0.344, 0.560)	13.639 (8.199, 21.965)

1) Feature Space Selection Experiment

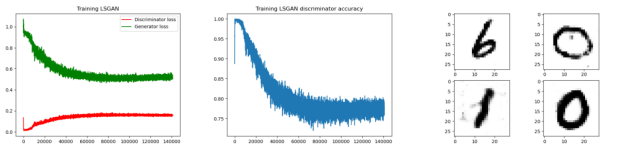
In this experiment, we analysed the distribution of distances between known and unknown samples within the feature spaces of different GAN discriminator networks. These results were compared to the distributions of distances between known-to-random and known-to-shuffled samples. To create feature spaces for this analysis, we trained three distinct GAN models, each producing a unique discriminator feature space. The GAN models do not need to generate flawless new data for this experiment. Instead, the focus is on training the discriminator network sufficiently to identify and learn features from the dataset. A perfectly trained generator is not essential since the discriminator's feature space is the key component in our analysis. Once the GAN can produce outputs that resemble the original dataset to some degree, we assume the discriminator has been trained well enough to create a usable feature space. Training GAN models to perfection is notoriously challenging. However, the less stringent requirements for this experiment make it more manageable to develop an adequate network. All three models were trained using the same dataset and for an equal number of epochs to maintain consistency in the training steps. We used the MNIST dataset, which contains 60,000 images, as the input for training. The dataset was passed through the networks 300 times during training. Fig. 2 highlights the training process of the DCGAN model. The loss values for both the generator and discriminator networks dropped significantly at the beginning of training but later started to increase gradually. The discriminator accuracy

fluctuated initially but eventually stabilized and showed a slow upward trend. Sample images generated by the DCGAN looked relatively good and closely resembled the original MNIST dataset. Fig. 3 shows the training summary for the LSGAN model. Unlike DCGAN, the loss values for LSGAN followed a smoother trend and remained consistent in the latter half of training. The accuracy of the discriminator had less variation compared to DCGAN. However, it followed a downward trend and eventually stabilized between 0.75 and 0.81. While the generated examples were less convincing than those from the DCGAN model, they still retained some similarity to the target dataset. Fig. 4 outlines the results of the WGAN model. The training process for WGAN differed from the other models, as its loss values diverged rather thans stabilizing. The discriminator's accuracy varied significantly during the early stages of training and briefly increased before following a downward trend. The generated samples were noticeably different from the outputs of the other two models, with gray backgrounds instead of white. Additionally, the quality of these samples was relatively poor. From this experiment, it is evident that the quality and consistency of the training outcomes varied among the three models. The DCGAN performed the best in generating realistic samples and maintaining stable accuracy trends. The LSGAN model showed smoother losses and consistent accuracy, but its generated outputs were less accurate. The WGAN model faced challenges with divergence in losses and produced lower-quality outputs.



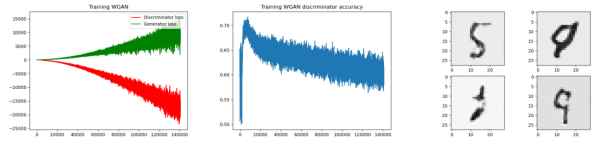
(a) Network losses (b) Discriminator Accuracy (c) Generated Examples

Fig. 2. Summary of DCGAN training



(a) Network losses (b) Discriminator Accuracy (c) Generated Examples

Fig. 3. Summary of LSGAN training



(a) Network losses. (b) Discriminator Accuracy. (c) Generated Examples

Fig. 4. Summary of WGAN training

V. Conclusion and Future Works

The first experiment, called the feature space experiment, didn't provide much insight. There was little difference between the distances calculated in the pixel space and the GAN discriminator feature space. We expected the distances to increase between known and shuffled samples, but this didn't happen, possibly due to issues with the dataset or the shuffled observations containing enough features to reduce the distance. The second experiment, focused on multiple observation fraud detection, was more promising. The GAN discriminator feature space produced distance results similar to the well-known Inception feature space, suggesting that the GAN discriminator learns useful features for analysis. The distance between real and fake observations increased as the proportion of fake data grew, which was expected. Different distance metrics showed varying results, with Wasserstein distance being least affected by dimensionality reduction, while Kernel MMD and Frechet distances performed poorly with reduced data. UMAP dimensionality reduction also negatively impacted the results. The single-observation fraud detection experiment didn't work well, as the distance measures didn't distinguish between true and false observations. The method used to transform individual observations into multivariate samples was too simple.

VI. Declarations

A. ***Funding:*** No funds, grants, or other support was received.

B. ***Conflict of Interest:*** The authors declare that they have no known competing for financial interests or personal relationships that could have appeared to influence the work reported in this paper.

C. ***Data Availability:*** Data will be made on reasonable request.

D. Code Availability: Code will be made on reasonable request.

References

[1] G. S. Kashyap, K. Malik, S. Wazir, and R. Khan, "Using Machine Learning to Quantify the Multimedia Risk Due to Fuzzing," *Multimed. Tools Appl.*, vol. 81, no. 25, pp. 36685–36698, Oct. 2022, doi: 10.1007/s11042-021-11558-9.

[2] S. Wazir, G. S. Kashyap, K. Malik, and A. E. I. Brownlee, "Predicting the Infection Level of COVID-19 Virus Using Normal Distribution-Based Approximation Model and PSO," Springer, Cham, 2023, pp. 75–91. doi: 10.1007/978-3-031-33183-1_5.

[3] G. S. Kashyap *et al.*, "Detection of a facemask in real-time using deep learning methods: Prevention of Covid 19," Jan. 2024, Accessed: Feb. 04, 2024. [Online]. Available: https://arxiv.org/abs/2401.15675v1

[4] N. Marwah, V. K. Singh, G. S. Kashyap, and S. Wazir, "An analysis of the robustness of UAV agriculture field coverage using multi-agent reinforcement learning," *Int. J. Inf. Technol.*, vol. 15, no. 4, pp. 2317–2327, May 2023, doi: 10.1007/s41870-023-01264-0.

[5] G. S. Kashyap *et al.*, "Revolutionizing Agriculture: A Comprehensive Review of Artificial Intelligence Techniques in Farming," Feb. 2024, doi: 10.21203/RS.3.RS-3984385/V1.

[6] S. Naz and G. S. Kashyap, "Enhancing the predictive capability of a mathematical model for pseudomonas aeruginosa through artificial neural networks," *Int. J. Inf. Technol. 2024*, pp. 1–10, Feb. 2024, doi: 10.1007/S41870-023-01721-W.

[7] A. Borji, "Pros and cons of GAN evaluation measures," *Comput. Vis. Image Underst.*, vol. 179, pp. 41–65, Feb. 2019, doi: 10.1016/j.cviu.2018.10.009.

[8] F. Chollet, *Deep Learning with Python, Second Edition*. Manning Publications Co. LLC, 2021. Accessed: Dec. 27, 2023. [Online]. Available: https://www.manning.com/books/deep-learning-with-python-second-edition

[9] A. Goyal, A. Sordoni, M. A. Côté, N. R. Ke, and Y. Bengio, "Z-

forcing: Training stochastic recurrent networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 6714–6724.

[10] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," *8th Int. Conf. Learn. Represent. ICLR 2020*, Jan. 2020, Accessed: Feb. 29, 2024. [Online]. Available: https://arxiv.org/abs/2001.04643v1

[11] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "MiDI-VAE: Modeling dynamics and instrumentation of music with applications to style transfer," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, International Society for Music Information Retrieval, Sep. 2018, pp. 747–754. Accessed: Oct. 12, 2024. [Online]. Available: https://arxiv.org/abs/1809.07600v1

[12] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680. doi: 10.3156/jsoft.29.5_177_2.

[13] F. Alharbi and G. S. Kashyap, "Empowering Network Security through Advanced Analysis of Malware Samples: Leveraging System Metrics and Network Log Data for Informed Decision-Making," *Int. J. Networked Distrib. Comput.*, pp. 1–15, Jun. 2024, doi: 10.1007/s44227-024-00032-1.

[14] G. S. Kashyap, D. Mahajan, O. C. Phukan, A. Kumar, A. E. I. Brownlee, and J. Gao, "From Simulations to Reality: Enhancing Multi-Robot Exploration for Urban Search and Rescue," Nov. 2023, Accessed: Dec. 03, 2023. [Online]. Available: https://arxiv.org/abs/2311.16958v1

[15] M. Kanojia, P. Kamani, G. S. Kashyap, S. Naz, S. Wazir, and A. Chauhan, "Alternative Agriculture Land-Use Transformation Pathways by Partial-Equilibrium Agricultural Sector Model: A Mathematical Approach," Aug. 2023, Accessed: Sep. 16, 2023. [Online]. Available: https://arxiv.org/abs/2308.11632v1

[16] F. Alharbi, G. S. Kashyap, and B. A. Allehyani, "Automated Ruleset Generation for 'HTTPS Everywhere': Challenges, Implementation, and Insights," *Int. J. Inf. Secur. Priv.*, vol. 18, no. 1, pp. 1–14, Jan. 2024, doi: 10.4018/IJISP.347330.

[17] P. Kaur, G. S. Kashyap, A. Kumar, M. T. Nafis, S. Kumar, and V. Shokeen, "From Text to Transformation: A Comprehensive Review of Large Language Models' Versatility," Feb. 2024, Accessed: Mar. 21, 2024. [Online]. Available: https://arxiv.org/abs/2402.16142v1

[18] H. Habib, G. S. Kashyap, N. Tabassum, and T. Nafis, "Stock Price Prediction Using Artificial Intelligence Based on LSTM– Deep Learning Model," in *Artificial Intelligence & Blockchain in Cyber Physical Systems: Technologies & Applications*, CRC Press, 2023, pp. 93–99. doi: 10.1201/9781003190301-6.

[19] S. Wazir, G. S. Kashyap, and P. Saxena, "MLOps: A Review," Aug. 2023, Accessed: Sep. 16, 2023. [Online]. Available: https://arxiv.org/abs/2308.10908v1

[20] G. S. Kashyap, A. Siddiqui, R. Siddiqui, K. Malik, S. Wazir, and A. E. I. Brownlee, "Prediction of Suicidal Risk Using Machine Learning Models," Dec. 25, 2021. Accessed: Feb. 04, 2024. [Online]. Available: https://papers.ssrn.com/abstract=4709789

[21] G. S. Kashyap, A. E. I. Brownlee, O. C. Phukan, K. Malik, and S. Wazir, "Roulette-Wheel Selection-Based PSO Algorithm for Solving the Vehicle Routing Problem with Time Windows," Jun. 2023, Accessed: Jul. 04, 2023. [Online]. Available: https://arxiv.org/abs/2306.02308v1

[22] M. T. Shaban, C. Baur, N. Navab, and S. Albarqouni, "Staingan: Stain style transfer for digital histological images," in *Proceedings - International Symposium on Biomedical Imaging*, IEEE Computer Society, Apr. 2019, pp. 953–956. doi: 10.1109/ISBI.2019.8759152.

[23] M. Diqi, M. E. Hiswati, and A. S. Nur, "StockGAN: robust stock price prediction using GAN algorithm," *Int. J. Inf. Technol.*, vol. 14, no. 5, pp. 2309–2315, Aug. 2022, doi: 10.1007/s41870-022-00929-6.

[24] B. Bhattarai, S. Baek, R. Bodur, and T. K. Kim, "Sampling strategies for gan synthetic data," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Institute of Electrical and Electronics Engineers Inc., May 2020, pp. 2303–2307. doi: 10.1109/ICASSP40776.2020.9054677.