DOKUZ EYLÜL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

A NEW AND EFFICIENT METHOD FOR SYNTHETIC DATA GENERATION WITH GENERATIVE ADVERSARIAL NETWORKS

by

Okan DÜZYEL

January, 2023

İZMİR

A NEW AND EFFICIENT METHOD FOR SYNTHETIC DATA GENERATION WITH GENERATIVE ADVERSARIAL NETWORKS

A Thesis Submitted to the Graduate School of Natural and Applied Sciences of Dokuz Eylül University In Partial Fulfillment of the Requirements for the Degree of Master of Science in Electrical and Electronics Engineering Program

by

Okan DÜZYEL

January, 2023

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"A NEW AND EFFICIENT METHOD FOR SYNTHETIC DATA GENERATION WITH GENERATIVE ADVERSARIAL NETWORKS"** completed by **OKAN DÜZYEL** under supervision of **PROF.DR. MEHMET KUNTALP** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Mehmet KUNTALP

Supervisor

Prof.Dr. Mehmet Süleyman Ünlütürk	Doç. Dr. Hatice doğan
(Jury Member)	(Jury Member)

Prof. Dr. Okan FISTIKOĞLU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to thank my advisor Prof. Dr. Mehmet KUNTALP, who made this study possible. His guidance, advice, motivation, and enthusiasm helped me in writing my M. Sc. thesis.

I am grateful to the committee members for their encouragement, valuable comments, suggestions, and hard questions.

I would like to express my sincere gratitude to Assoc. Prof. Dr. Damla KUNTALP. Her math course, that is Random Variables and Stochastic Processes aided me in understanding the idea of probability and statistics for Artificial Neural Networks.

Finally, I would like to give my endless gratitude to my whole family members, especially my mother and father, who have always supported me throughout my entire life.

Okan DÜZYEL

A NEW AND EFFICIENT METHOD FOR SYNTHETIC DATA GENERATION WITH GENERATIVE ADVERSARIAL NETWORKS

ABSTRACT

Machine learning models, especially deep neural networks, need sufficient data for successful training. In case of insufficient data, the training of the model fails. Moreover, imbalanced datasets are also an important problem that reduces the performance of the classification system. In such cases, we need to increase the data in the dataset of the classes which have few data. Classical data augmentation methods do this by changing the data that we have. Existing data is duplicated by some techniques such as rotating, scaling, etc. However, no new data is synthesized by this approach. In areas where the dataset is very constrained, this solution would not work. We need to generate new data based on limited data in such cases. It is possible to generate synthetic new data using the Generative Adversarial Networks (GAN). Thus, sufficient training data can be provided for the deep learning model developed.

In this thesis, new synthetic ECG data were produced for MIT-BIH Arrhythmia Database using a new GAN-based method. The proposed method considers the cluster structure of individual classes. As a result of the study, the dataset was increased with new synthetic data. The performance of the developed systems was examined with different experiments which were conducted by using different training sets but the same test data. It has been observed that the success of the deep learning system has increased using the proposed GAN-based synthetic data augmentation method when compared with classical GAN-based method.

Keywords: Dataset, data augmentation, generative adversarial neural networks, artificial neural network, ecg

ÇEKİŞMELİ ÜRETİCİ AĞ KULLANILARAK SENTETİK DATA ÜRETMEDE YENİ VE VERİMLİ BİR METOD

ÖZ

Yapay zekâ modelleri, özellikle derin sinir ağları, eğitim esnasında yeterli sayıda veriye ihtiyaç duyarlar. Verilerin yetersiz olduğu durumlarda modelin eğitimi başarısız olur. Ayrıca sınıflar arası veri dengesizliği de sınıflandırıcı sistemin eğitiminde başarısızlığa yol açan önemli bir problemdir. Böyle durumlarda bu sınıflar için veri kümesindeki verileri arttırmamız gerekmektedir. Klasik veri arttırma yöntemleri elimizdeki verileri değiştirerek yapılır. Mevcut verilerin benzerleri oluşturulur, yeni veriler üretilmez. Veri kümesinin çok kısıtlı olduğu alanlarda bu çözüm işe yaramaz. Böyle durumlarda elimizdeki sınırlı verilerden yeni veriler üretmemiz gerekir. Çekişmeli Üretici Ağ (ÇÜA) kullanılarak sentetik yeni veriler üretmek mümkündür ve böylece oluşturulan derin öğrenme modelinin çalışması için yeterli eğitim verisi sağlanabilir.

Bu tezde, MIT-BIH Arrhythmia veri tabanı üzerinde yeni bir ÇÜA yaklaşımı kullanılarak sentetik ECG verileri üretilmiştir. Özellikle az kayıt içeren veri sınıfları için sentetik veri üretilmesi amaçlanmıştır. Önerilen bu yeni yaklaşımda belli bir sınıfa ait verilerin öbek dağılımları baz alınmıştır. Çalışma sonucunda elde edilen sentetik veriler ile eksik veri kümeleri arttırılmıştır. Geliştirilen sistemlerin başarımları her birinde farklı bir eğitim verisinin kullanıldığı farklı deneylerle incelenmiştir. Önerilen yeni yaklaşımla veri artırımı sonrası elde edilen derin öğrenme başarısının klasik GAN tabanlı yönteme göre arttığı gözlemlenmiştir.

Anahtar Kelimeler: Veri kümesi, veri arttırma, çekişmeli üretici ağ, yapay sinir ağı, ekg

CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

Page

CHAPTER 1
INTRODUCTION

Today, with the enormous increase in the processing power of computer processor units (CPU) and graphics processor units (GPU), the use of artificial intelligence in our daily lives has become widespread. In addition, with the development of the internet and massive storage media, it has become easy to obtain and collect data from numerous sources including electronic sensors, cameras, etc. In some cases, data collection can be problematic. For example, in medical arena, patients' data are not shared for privacy and security purposes. Therefore, in such areas, the inability to find sufficient data on diseases is a big problem for artificial intelligence-based projects. This is mostly due to the fact that enormous amount of data is needed to successfully train AI models. If our dataset is insufficient, the model's success rate against the test data remains low. In such cases, it is necessary to increase the existing data set with data augmentation methods. (Hoelzemann, Sorathiya & Van Laerhoven, 2021; Perez & Wang, 2017).

Another problem that negatively affects the successful training of machine learning algorithms is related to data imbalance problem. The large number of differences between data samples in different classes in the dataset leads to this problem. This problem causes the over-numbered classes to be learned more effectively during the training phase and dominate the classes fewer data samples. As a result, the classifier algorithms cannot adequately learn the problem producing erroneous results. The study in (Sun, Wong & Kamel, 2009) provides information about what the imbalance problem is and how to fix it. Moreover, the imbalance problem is not specific to classification algorithms. In a study by (López, Fernández, García, Palade & Herrera, 2013), the problems that imbalance data poses for data mining are discussed. In another article (Tanha, Abdi, Samadi, Razzaghi & Asadpour, 2020), different approaches for solving the imbalance problem are compared in multi class datasets.

In classical data augmentation methods, similar data is produced by changing the data in the dataset (Mikolajczyk & Grochowski, 2018). Various augmentation methods are selected and used according to the type and structure of the data. The data can be one-dimensional for audio files, while for images, the data is two-dimensional. It is multidimensional in tabular data. Some methods have also been applied for data augmentation of audio files (Nanni, Maguolo & Paci, 2020). More common studies have been done for images (Wong, Gatt & Stamatescu, 2016) and some studies have been done on the text (Bayer, Kaufhold & Reuter, 2022; J. Wei & Zou, 2019).

In literature, the common point of all shared studies is the importance of data augmentation. Despite the development of modern data augmentation methods, research on classical methods continues today. Another recent study has compared these classical methods to images (Nanni, Paci, Brahnam & Lumini, 2021). As with images, classical data augmentation methods have been used and compared in another study (S. Wei et al., 2020) for one-dimensional signals such as audio files. In another study (Lashgari, Liang & Maoz, 2020) one-dimensional signals were generated made for electroencephalography (EEG) signals. In artificial intelligence, the model's response to the data set is very important. Small differences in the data set greatly affect the model's running performance. In another study (Cubuk, Zoph, Mane, Vasudevan & Le, 2019), new algorithms applied automatically instead of manually performing data augmentation methods have been developed.

Modern approaches for data augmentation methods are mostly based on Variational Autoencoders (VAE) and Generative Adversarial Neural networks (GAN). VAE can be used not only for data augmentation but also for classification and feature extraction purposes (Metlapalli, Muthusamy & Battula, 2020). As with classical methods, the data model in the input may vary in modern methods. In a study (Metlapalli et al., 2020), image-based data was processed. In (Zhu, Wu, Latapie, Yang & Yan, 2021), the cross-correlation of audio and video together was investigated by changing the encoder and decoder structure of the VAE. GAN is generally more successful in data synthesis than VAE, but VAE is still being used in many data synthesis applications. For example, in (Saldanha, Chakraborty, Patil, Kotecha, Kumar & Nayyar, 2022), biomedical sound signals that were obtained from lung sounds from ICBHI respiratory sounds database (Rocha, Filos, Mendes, Serbes, Ulukaya, Kahya, Jakovljevic, Turukalo, Vogiatzis, Perantoni, Kaimakamis, Natsiavas, Oliveira, Jácome, Marques, Maglaveras, Pedro Paiva, Chouvarda & de Carvalho, 2019) were synthesized with several VAE models to produce new data.

GAN is a generative machine learning model. After the first GAN paper (Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville & Bengio,2014) was published, many studies have been done on GAN, and GAN has been adapted to many fields. In addition to image synthesis applications, it has been the subject of many studies from audio synthesis to tabular data synthesis. A study (Donahue, McAuley & Puckette, 2018) has been done to synthesize data from raw audio files. In this work, WaveGAN and SpecGAN are introduced. It creates images in the frequency domain from audio files and synthesizes new data from these images. Then it returns to the time domain again and new sound files are created. In addition, other studies (Wu, Chrysos & Cevher, 2022) are currently being conducted on raw sound synthesis. In addition, the study by (Xu et, Skoularidou, Cuesta-Infante & Veeramachaneni, 2019) has introduced conditional tabular GAN (CTGAN) in synthesizing tabular data with GAN. When synthesizing tabular data, it does not take into account previous values. In this case, if data is continuous and sequential like an ECG signal then it cannot be synthesized with CTGAN. For such continuous sequential data, recurrent neural network (RNN) or long short-term memory (LSTM) based GAN models should be used (Mogren, 2016; Ouyang, Zhang, Ma & Agam, 2018).

In this thesis study, a new GAN-based method is developed to synthesize ECG signals with LSTM-GAN architecture using MIT-BIH Arrhythmia Database (Moody & Mark, 2001). The results show that the ECG data can be classified more successfully with the 1D convolutional neural network (CNN) classifier by training it on data sets which are augmented by the proposed method.

This thesis consists of five main parts. Chapter 1 is the introductory part. In this section, the aim of the thesis, literature research, and suggestions are given. In Chapter 2, data augmentation methods are examined based on classical and modern approaches. Chapter 3 describes the GAN structure. In Chapter 4, the new data augmentation method is presented, and the conducted experiments and results obtained are given. Chapter 5 is the discussion part. The conclusion of this thesis is made in Chapter 6.

CHAPTER 2
DATA AUGMENTATION

The main purpose of data augmentation is to support the model with enough data to make more accurate predictions. Where the data set is insufficient, the model cannot learn enough and gives low accuracy results for test data in real life. It is said that the model cannot be generalized enough. In such cases, data augmentation methods should be applied to the dataset.

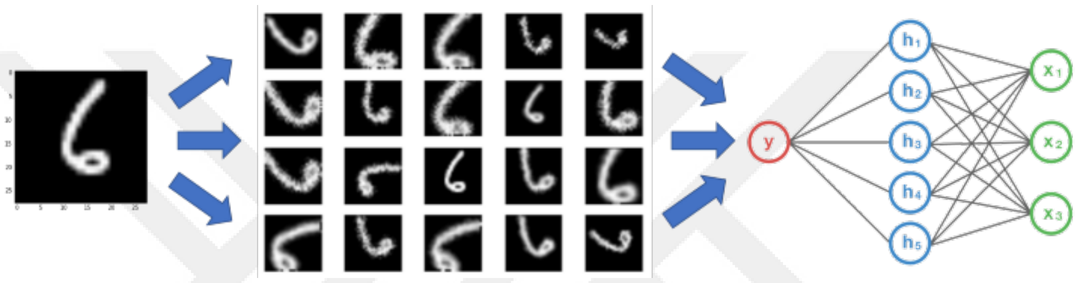


Figure 2.1 Data augmentation visualization for MNIST handwritten digits database (Gandhi, 2021)

The previous chapter mentioned that data augmentation methods would be examined under two main headings. The first of these is classical data augmentation methods and the second is modern data augmentation methods. The easiest way to describe data augmentation methods is to make use of data augmentation methods used for images. Because the process on the images gives visual outputs and the obtained results are more understandable for the readers. However, since ECG signal synthesis will be examined in this thesis, signal augmentation methods should also be mentioned. Thus, image and signal augmentation methods are explained in this chapter.

2.1 Classical Data Augmentation Methods for Image Processing

Classical data augmentation methods obtain different data by only changing the existing data. In this method, new data is not synthesized. Hence, this application for use in deep learning influences the results, but it is not always as effective as modern methods. Representing of grayscale digital images is defined in a 2-D array. $f(x,y)$ function defines the image that has M rows and N columns. Where (x, y) defines the discrete coordinates of the image and each element is called a pixel.

$$f(x,y) = \begin{bmatrix} f(0,0) & \cdots & f(0,N-1) \\ \vdots & \ddots & \vdots \\ f(M-1,0) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{2.1}$$

In the color image (RGB), spatial domains of Red, Green, and Blue are defined separately for each function.

$$f_R(x,y) = \begin{bmatrix} f(0,0) & \cdots & f(0,N-1) \\ \vdots & \ddots & \vdots \\ f(M-1,0) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{2.2}$$

$$f_G(x,y) = \begin{bmatrix} f(0,0) & \cdots & f(0,N-1) \\ \vdots & \ddots & \vdots \\ f(M-1,0) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{2.3}$$

$$f_B(x,y) = \begin{bmatrix} f(0,0) & \cdots & f(0,N-1) \\ \vdots & \ddots & \vdots \\ f(M-1,0) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{2.4}$$

By performing mathematical operations on the pixel values of functions, changes in the image are obtained. These operations are flip, rotation, translation, scale, crop, and Gaussian Noise.

2.1.1 Flip

With this data augmentation method, the image is flipped vertically or horizontally. As a result of these translations, two different images are obtained. To flip the image vertically, the following mathematical operation is applied (2.2). Flipping the image horizontally is applied (2.3). The result obtained is shown in Figure 2.2.

$$\sum_{(x,y)\in S_{xy}} f_{Hflipped}(x,y) = f(\mathrm{M} - \mathrm{x} - 1, \mathrm{y}) \tag{2.5}$$

$$\sum_{(x,y)\in Sxy} f_{Vflipped}(x,y) = f(\mathrm{x}, \mathrm{N} - \mathrm{y} - 1) \tag{2.6}$$

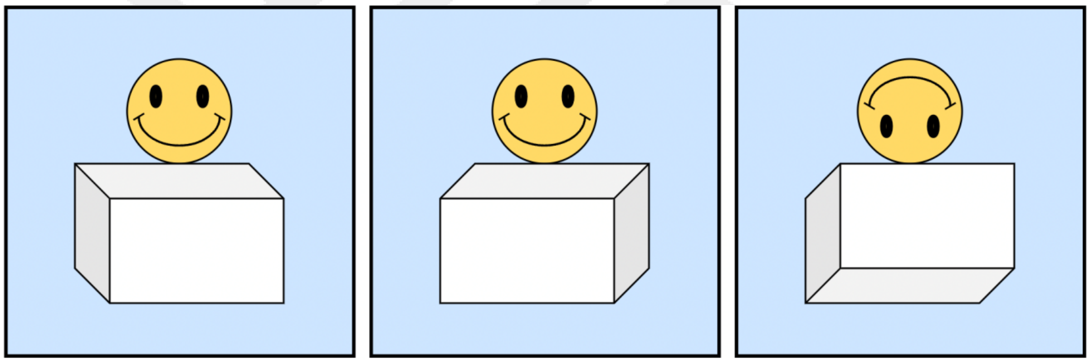


Figure 2.2 From the left, the original image, flipped horizontally, and flipped vertically

2.1.2 Rotation

Rotation is done by rotating a pixel in the desired direction, then positioning it in its new place and applying it to the entire image. The point to be noted here is that if the image is not square, there will be losses in the image. The mathematical structure of the rotation process and its results on the images are shown below.

$$(x,y) = R\{(v,w)\} \tag{2.7}$$

$$x = v\cos\theta + w\sin\theta$$
$$y = -v\sin\theta + w\cos\theta \tag{2.8}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \\ 1 \end{bmatrix} \tag{2.9}$$

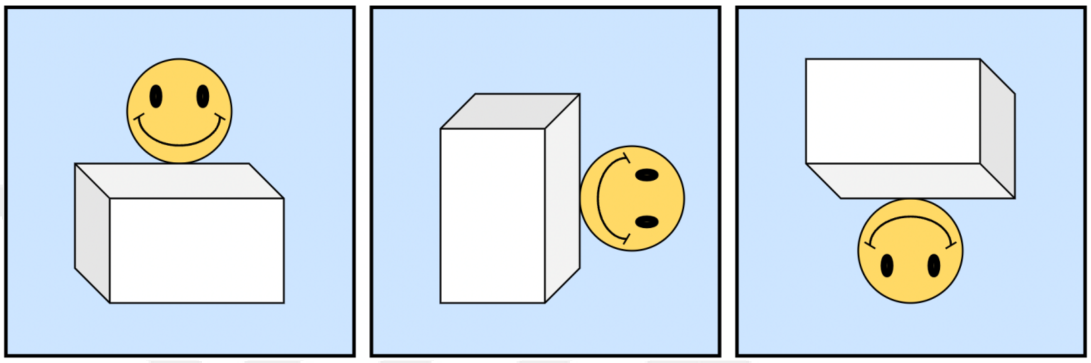


Figure 2.3 The images are rotated by 90, 180, and 270 degrees clockwise

2.1.3 Translation

The image is introduced to another region without being rotated in the translation process. How much transport will be made is decided by t_x and t_y values. The main thing to note in this operation is that the resulting image should be larger than the input image to avoid loss if there is a piece of information at the edge of the image.

$$x = v + t_x$$
$$y = w + t_y \tag{2.10}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \\ 1 \end{bmatrix} \tag{2.11}$$

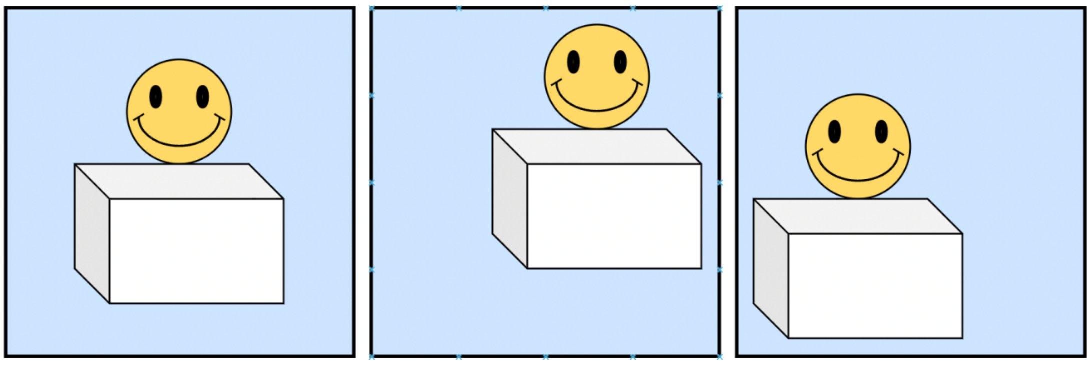


Figure 2.4 From the left of the original image, translated to the right, and upwards

2.1.4 Scale

The image is scaled in two directions, outward or inward. When scaling outward, the output image size will be larger than the original image size and the overflowing parts of the image will be cut off. Therefore, some data loss may occur. Inward scaling, on the other hand, will reduce the output image, so the outer parts of the image should be filled with black pixels.

$$x = c_x v$$
$$y = c_y w \tag{2.12}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \\ 1 \end{bmatrix} \tag{2.13}$$

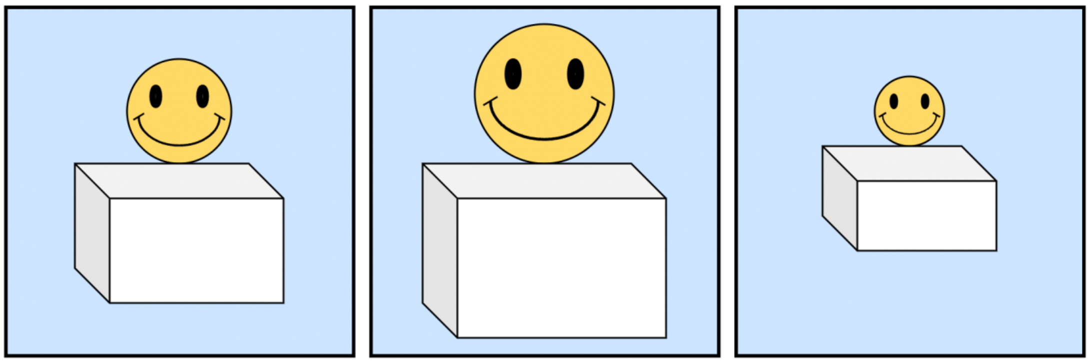


Figure 2.5 From the left of the original image, scaled two images

2.1.4 Crop

Instead of scaling, a random point is selected in the image, and scaling is made at the origin of this point. Part of the outward growing image is cut off. This process allows us to obtain different variations from the same image. Its mathematical expression is the same as scaling.

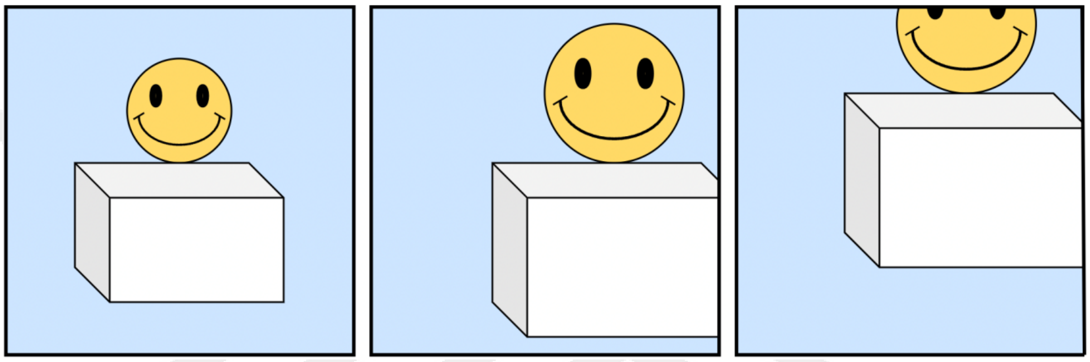


Figure 2.6 From the left original image, cropped from top left of, and the middle

2.1.4 Gaussian Noise

In cases where the data set was insufficient, it was examined that the existing data should be increased by changing it. In some cases, the data is so clear and smooth that in such cases, deep learning models memorize the data more than necessary and lose the ability to generalize for test data. It is an undesirable situation. It is a data augmentation method to add some noise to the seamless data and change its structure. Gaussian noise is generally preferred for this process due to its balanced distribution. On the next page, both Gaussian noise and Salt and paper noise are added to the original image.

$$g(x, y) = f(x, y) + n(x, y) \tag{2.14}$$

$$n(x, y) = \frac{1}{\left|\sqrt{2\pi\Sigma}\right|} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (y-\mu)} \tag{2.15}$$

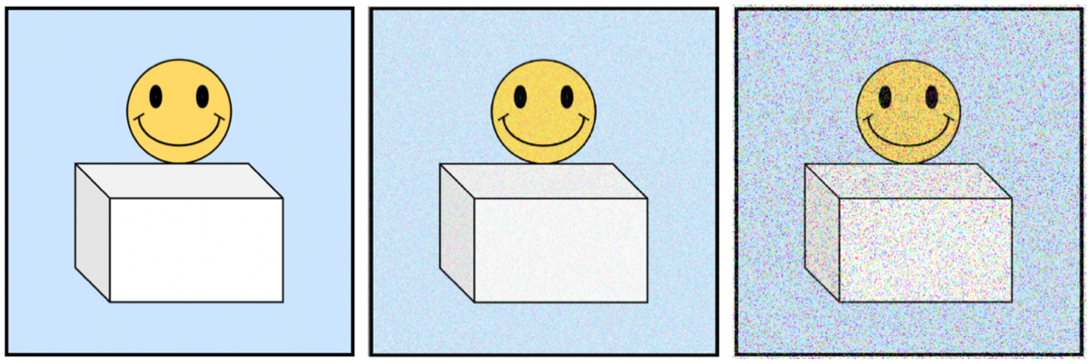
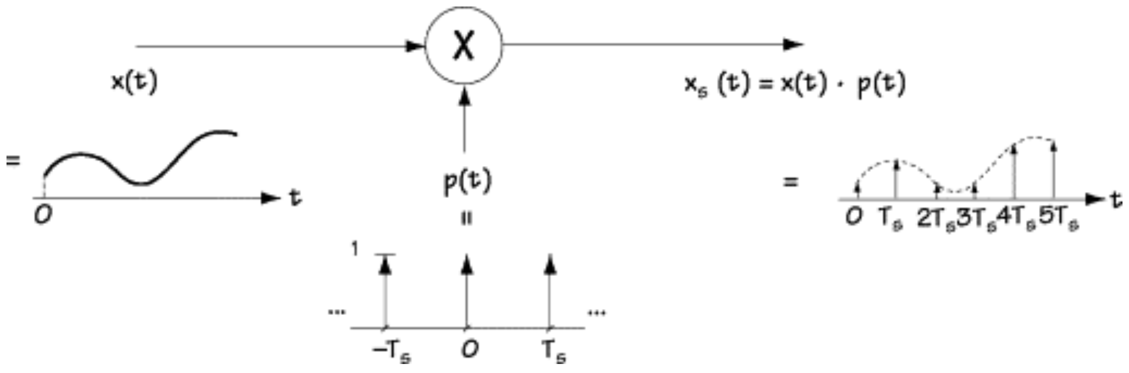


Figure 2.7 From the left of the original image, gaussian noise, and salt and paper noise

2.2 Classical Data Augmentation Methods for Signal Processing

Digital signals are represented by logic 1s and 0s, but analog signals cannot be represented by only these two values. Because they have intermediate values between these. Therefore, they must first be converted to quantized signal form. For this work, a method called sampling is used. Signals are generally one-dimensional structures obtained by sampling at a predetermined frequency. Audio signals, ECG signals, Voltage signals, etc. are in this structure.



x(t)
=
0
t
X
p(t)
=
1
...
...
$-T_s$
0
T_s
$x_s(t) = x(t) \cdot p(t)$
=
0 T_s $2T_s$ $3T_s$ $4T_s$ $5T_s$
t

Figure 2.8 Sampling Method (Nassar, 2001)

$$p(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT_s) \tag{2.16}$$

$$x_s(t) = x(t).p(t) = x(t). \sum_{k=-\infty}^{\infty} \delta(t - kT_s) \tag{2.17}$$

$$x_s(t) = \sum_{k=-\infty}^{\infty} x(kT_s)\delta(t - kT_s) \tag{2.18}$$

First, an impulse train is created as in (2.16) and then this generated signal is multiplied by the input signal. As a result, the output in (2.18) occurs. Thus, the analog signal is transferred to digital. The following shows how an audio signal looks in a digital environment.

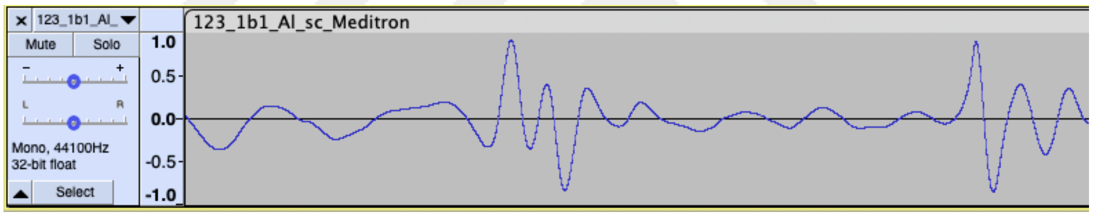
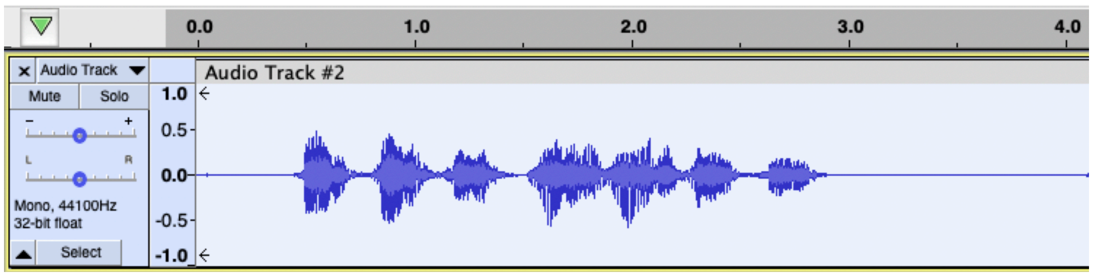


Figure 2.9 An audio signal that is sampled by 44.1Khz

Digital signals are kept in a 1D array in the computer environment. These signals, whose sampling frequency is known beforehand, can be visualized with the values in the array as in Figure 2.9. After this stage, data augmentation methods can be applied to digitized signals. These methods are time-shifting, time stretching, pitch scaling, noise addition, low/high/band-pass filters, and polarity inversion

2.2.1 Time-shifting

Time-shifting is the simplest of signal augmentation methods. The signal is only shifted in the time axis left or right and new modified data is obtained. The original signal and time shift are shown in Figure 2.10.



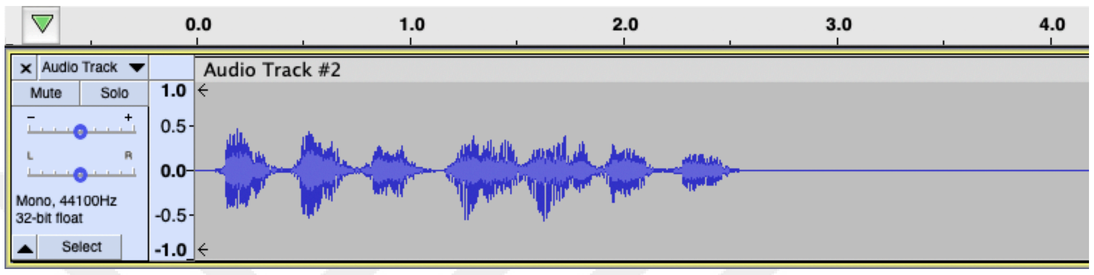


Figure 2.10 An original signal on top and a time-shifted signal on the bottom

2.2.1 Time stretching

With this operation, the time component of the signal is changed, and the pitch component is not touched. The signal is either slowed down or sped up over time. This is the inverse operation of pitch scaling.

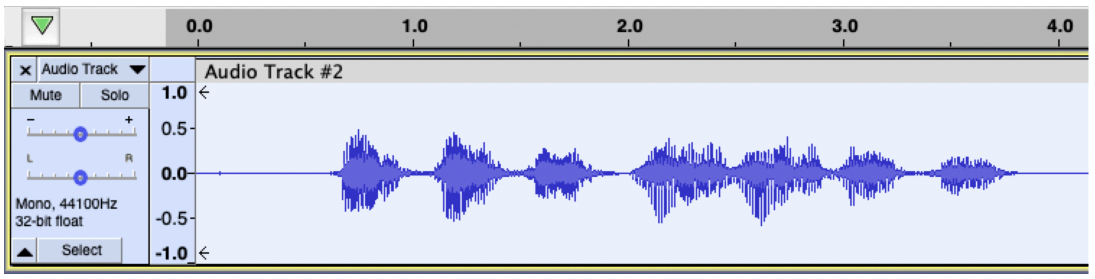


Figure 2.11 Time stretched audio signal

2.2.3 Pitch scaling

The pitch of a signal is changed with this method. Considering that this process is applied to the audio signal, the musical notes in the signal do not change in time. But

Music notes are changed at the specified level. That is, this method changes the frequency of a signal without changing the time information. A pitch-shifted audio signal is shown below.

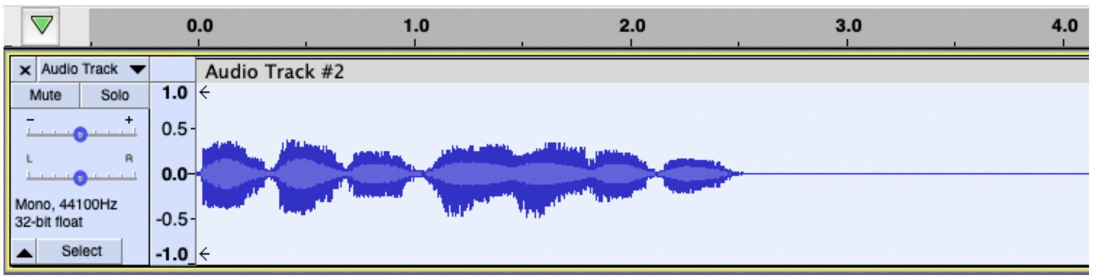


Figure 2.12 Pitch-shifted signal

2.2.4 Noise addition

As described in the method of adding noise for images, sometimes too clear information puts the model overfitting. This is true for signals. The model trained with perfect signals loses its generalization feature. Thus, the test performance decreases. In such cases, noise is added to the signals. An audio signal that is with added noise below.

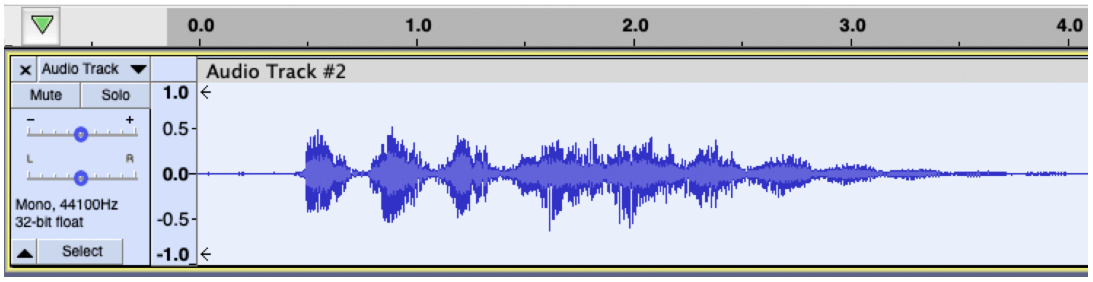


Figure 2.13 Noise added signal

2.2.5 Low/High/Band-pass filters

Except that the data is clean, the model cannot be adequately trained due to unwanted noise while collecting some data. In such cases, the data is filtered according to the type of noise. Below is a low-pass filter applied to an audio signal.

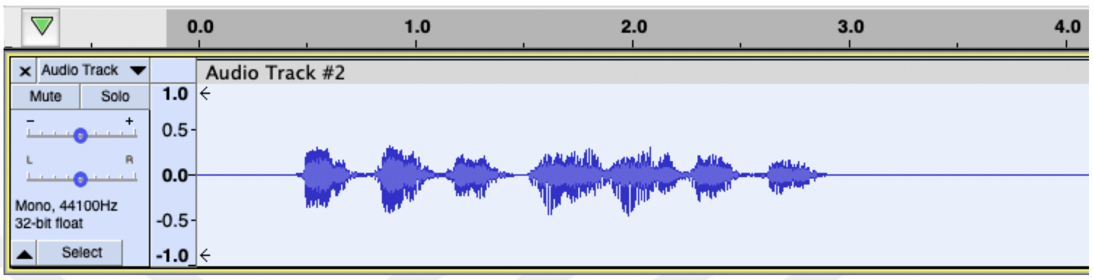


Figure 2.14 A signal that is applied to a low-pass filter

2.2.6 Polarity inversion

In this method, the input signal is multiplied by -1 to change the polarization. The signal is only inverted on the Y axis. There is no change in the time axis. In data such as audio signals, there is no difference in the sound heard, but the signal is changed in structure. Below is the block diagram explaining how this is done, and polarity inversion is applied to the audio signal.

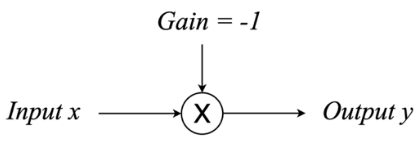


Figure 2.15 Block diagram of polarity inversion

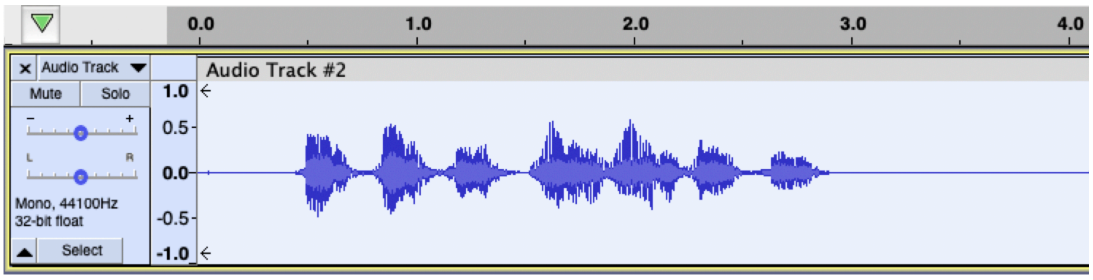


Figure 2.16 Polarity inverted signal below

2.3 Modern Data Augmentation Methods

With the methods described so far, the data set has been increased by making changes. Consider a deep learning model that aims to classify human photographs. In the current dataset, there are not any data for men or women with blond hair, wavy hair, large eyes, or small lips. This model can be trained with current data, but generalizability against external test data will be very poor. Because for the model, people can only have straight hair, small eyes, not blonde, and not small lips. The wavy-haired person may not perceive the hairstyle in the image or may interpret it differently. Human photographs with different hairstyles cannot be produced with classical data augmentation methods since this is a very complex task.

Another example can be given in music synthesis. For example, Gibson electric guitars and Fender Telecaster electric guitars have different sounds and tones even if they play the same notes. For an application where guitar sounds are tried to be separated, using only one brand or one model guitar sound causes the system to lose its generalization feature. Synthesizing new data in a tone that resembles guitar sounds other than the only brand-model guitar sound available is quite difficult.

VAE and GAN have capable of synthesizing new data. In this section, these modern data augmentation approaches are explained.

2.3.1 Variational Auto Encoders (VAE)

Variational Autoencoder (Kingma & Welling, 2013) is a kind of Autoencoder (AE) model. AE tries to compress and reconstruct the data. It does this with the Encoder and decoder layers. There is a hidden layer between these two layers. This layer is called the bottleneck layer. There is some loss as data is compressed and regenerated. The structure of an AE is given below.

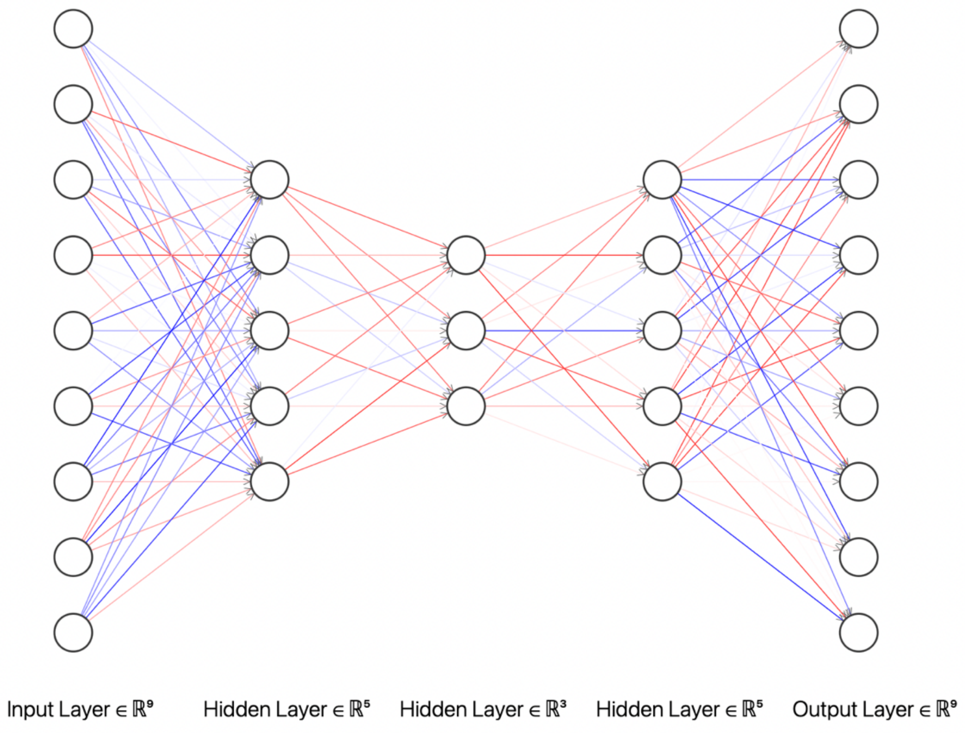


Figure 2.17 Example of Autoencoder structure

In the example image in Figure 2.17, the encoder part consists of an input layer with nine neurons, a hidden layer with five neurons, and a hidden layer (bottleneck) with three neurons. The decoder part is a hidden layer with three neurons, a hidden layer with five neurons, and an output layer with nine neurons. While 1% data loss is a big problem in some applications, 10% loss is not a problem in some applications. With this working method, AE can be used in feature extraction, dimension reduction, and image coloring/completion applications, but for applications where data loss is important, for example, file compression, etc. cannot be used.

The difference between VAE from AE is that it provides the opportunity to generate new data with layers that make two probabilistic distributions in the bottleneck layer. For VAE, this layer is called latent space. These layers extract the mean and variance vectors of the input. Then, after delivering this information to the sampled latent vector, they decode the data. Below is the structure of the VAE.

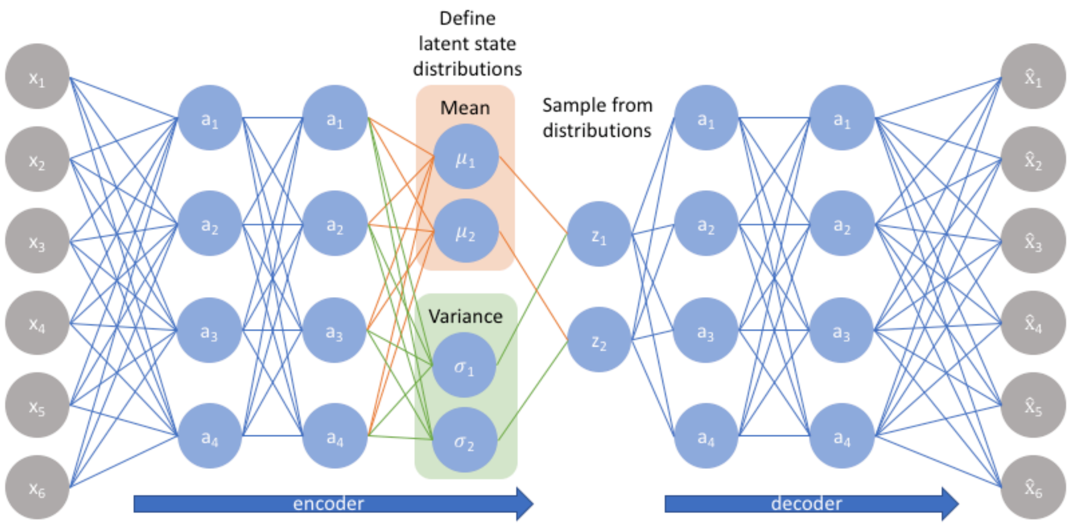


Figure 2.18 Example of Variational Autoencoder structure (Jordan, 2018)

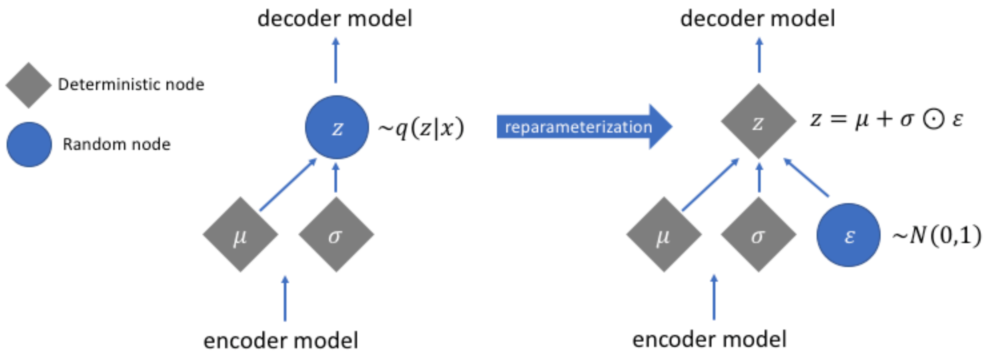


Figure 2.19 Variational Autoencoder latent space structure (Jordan, 2018)

The mean and variance values are deterministic, but the sampled latent vector is nondeterministic. This prevents the rearrangement of weights with the back-propagation algorithm. In order to eliminate this problem, it is necessary to produce it deterministic with the help of random noise that a random number produced from a standard normal distribution with ε=epsilon, epsilon mean 0 and variance 1, partial derivative in terms of Mean and Std with the back-propagation method.

$$\max_{\phi,\theta} \mathbb{E}_{q\phi(z|x)} [\log p_\theta(x|z)] \tag{2.19}$$

$$\log p\theta(x|z) = D_{KL}(q(z|x) \,||\, p(z) + \mathcal{L}(\theta, \phi; x, z) \tag{2.20}$$

$$\log p\theta(x|z) \geq \mathcal{L}(\theta, \phi; x, z) = \mathbb{E}_{q\phi(Z|x)}[\log p_\theta(x|z)] - D_{KL}(q(z|x) \,||\, p(z)) \tag{2.21}$$

Input x, which is converted to z format in Latent Space, is converted back with the decoder network. This process is the same as in autoencoder, but the difference is that VAE's loss function is standard MSE + Kullback-Leibler divergence (2.21). Thanks to this structure of the latent space, the data changes, and the VAE can synthesize new data. Figure 2.20 shows the images synthesized by VAE using the CelebA dataset (Liu et al., 2014).

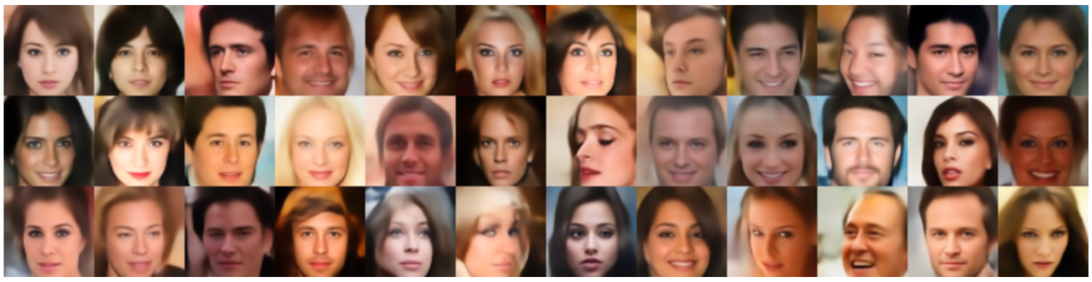


Figure 2.20 Variational Autoencoder-based synthesized human faces (Dai & Wipf, 2019)

2.3.1 Generative Adversarial Neural Networks (GAN)

One of the important developments in the field of artificial intelligence in the 21st century is GAN. When it was first published (Goodfellow et al., 2014), it revolutionized generative networks. Because although the results of VAE were successful, GAN gave much better results. VAE basically tries to compress the data and then reconstruct it. At this stage, depending on the size of the latent space, data is lost, albeit unintentionally. For example, the images that VAE synthesizes are blurry or the audio files it synthesizes have unwanted noise. With its structure different from VAE, GAN avoids this loss problem, which may cause problems for some applications. This is much more realistic images, sound, etc. It allows us to synthesize data. Below are human faces that have never lived on earth, synthesized with GAN.

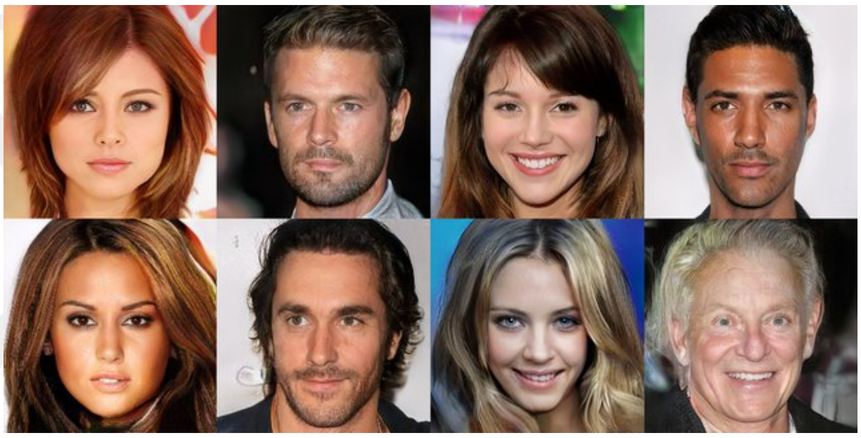


Figure 2.21 Human face generation with GAN (Karras et al., 2018)

If enough sample data is given to the GAN, it can make a new painting that does not exist on earth (Zhang, Gu, Zhang, Bao, Chen, Wen, Wang & Guo, 2021), compose music (Mukherjee & Mulimani, 2022), produce anime characters (Shang et al., 2022), pose guided person image generation (Siarohin, Sangineto, Lathuiliere & Sebe, 2017), convert an image into super-resolution (Wang, Jiang, Yi, Han & He, 2020), synthesize real images from photorealistic images (J. Zhang, Zhang, Li & He, 2021). It can also synthesize text into images (Tao, Tang, Wu, Jing, Bao & Xu, 2020). It produces a picture just like a painter with a few keywords. In the next chapter, the structure, types, and mathematical model of the GAN are examined in depth.

CHAPTER 3

GENERATIVE ADVERSARIAL NETWORKS (GAN)

A question people tried to solve in the 1950s can computers think like humans? The famous mathematician and computer scientist Alan Turing posed this question (Turing, 1950). Turing announced a game genre called "imitation game" in his paper. According to this game, a human and a computer are hidden in a room. In another room, there is a human subject. The computer or real human answers the questions asked by the subject. The subject tries to identify who gave this answer. If the subject does not notice the difference, the computer wins. In the future, with the development of technology and the acceleration of computers, computers have won this game many times.

The essentials of machine learning were laid out by Warren S. McCulloch & Walter Pitts in 1943 by mathematical modeling of the neural structure (McCulloch & Pitts, 1943). The foundations of machine learning were laid by McCulloch & Walter Pitts' mathematical neuron model. According to this model, each input is multiplied and summed by coefficients called synaptic weights. This total is given to the neuron. It is revealed whether it can activate the neuron after it is added with the bias value. If this sum is higher than the neuron's spark value, the neuron becomes active and produces an output. If it is less than the spark value, the neuron will not output. An image of this model is given in Figure 3.1.

$$g(x_1, x_2 x_3 \dots, x_n) = g(x) = \sum_{i=1}^{n} x_i \tag{3.1}$$

$$y = f\big(g(x)\big) = \begin{cases} 1 \; if \; g(x) \geq 0 \\ 0 \; if g(x) < 0 \end{cases} \tag{3.2}$$

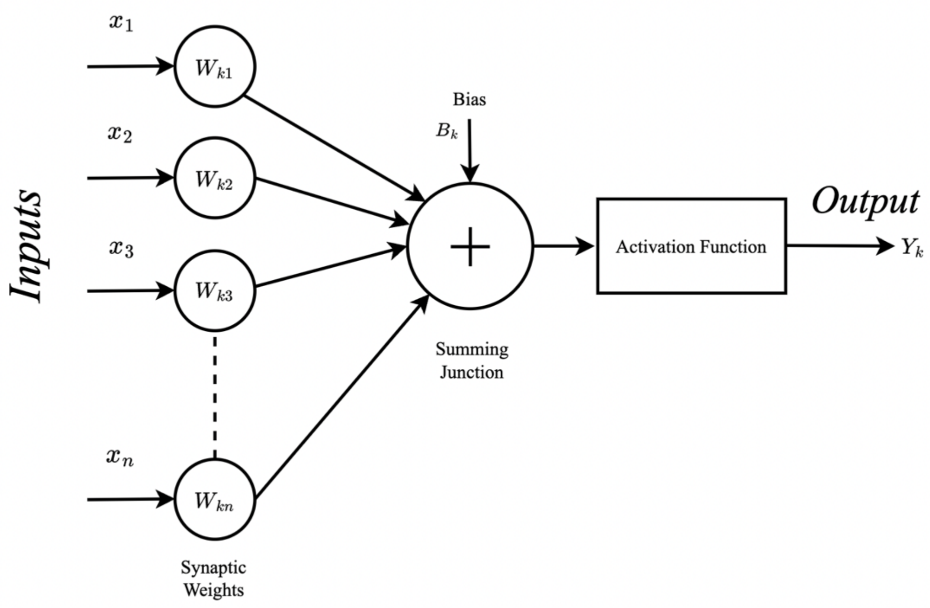


Figure 3.1 McCulloch-Pitts neuron model

The weakness of this mathematical model was that it could work for a single layer. Because in the training phase, the data must be given to the system many times and the weights must be rearranged. At that time, in multi-layer structures, it was not known how to do this for each layer. This problem was solved by a publication published in the 1970s (Linnainmaa, 1976). With the back-propagation algorithm, the weights in all layers could be updated backward. Currently, this algorithm is still used in most artificial intelligence applications.

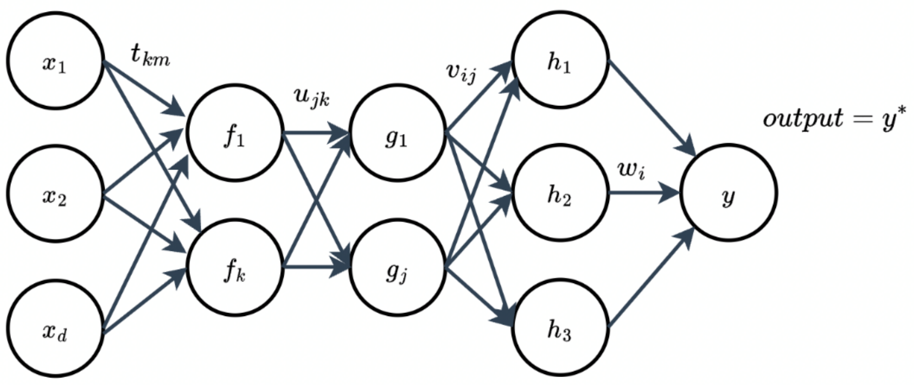


Figure 3.2 Multi-layer perceptron model

$$g_j = \sigma\left(u_{j0} + \sum_k u_{jk} f_k\right) \tag{3.3}$$

$$E = \sum_{j=1}^{n} \frac{1}{2}\left(t_j - y_j^*\right)^2 \tag{3.4}$$

$$\frac{\partial E}{\partial g_j} = \sum_i \sigma'(h_i) v_{ij} \frac{\partial E}{\partial h_i} \tag{3.5}$$

$$\frac{\partial E}{\partial u_j} = \frac{\partial E}{\partial g_i} \sigma'(g_j) f_k \tag{3.6}$$

$$u_{jk} \leftarrow u_{jk} - \eta \frac{\partial E}{\partial u_{jk}} \tag{3.7}$$

During the training, this algorithm calculates the total error (3.4) and layer errors (3.5) for each node by using backward partial derivatives to calculate (3.6) the error in each neuron after the feed-forward process (3.3) and updates the weights one by one (3.7) for billion times. For this reason, it needs computers with very high processing power. The development of computers in the last two decades has allowed artificial intelligence models to be trained enough to be used in everyday life. But artificial intelligence applications still use classification, prediction, noise removal, self-learning, etc. It was still not possible for computers to generate data. After the initial cessation of VAE, GAN broke ground in this area.

3.1 Original GAN

GAN is based on a very bright idea. It consists of two neural networks. These are generator and discriminator. The generator generates random data and sends it to the discriminator. Discriminator checks how much this generated data overlaps with the original data from dataset. If the generated data is too far from the original data, it learns the data in the dataset and trains itself while rearranging the weights of the generator with the back-propagation algorithm. In other words, two neural network models compete. This situation continues throughout the training period. When the randomly generated data starts to become like the real data in the dataset, new data is synthesized. Consider, for example, the famous handwriting dataset MNIST. All zero digits are trained by the discriminator. Random images produced by the generator are back-propagated according to the errors that occur after the classification error. The generated random image will gradually start to resemble the zero images in the dataset but will not be the same. Because the new zero image produced in the generator is like blending all the zero images in the dataset. Thus, new data is synthesized. The structure of the GAN is shown in Figure 3.3.

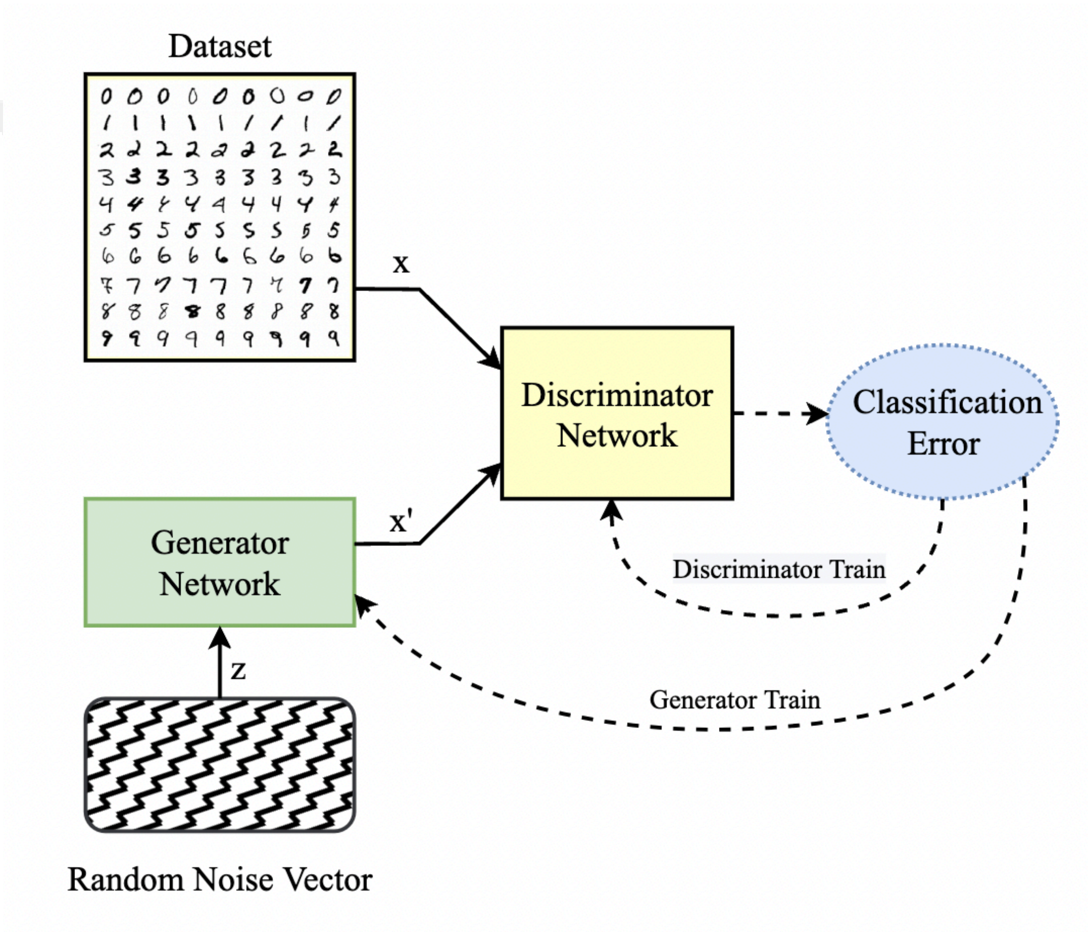


Figure 3.3 Generative Adversarial Network structure

In order to understand the mathematical structure behind the GAN, some definitions must first be made. Thus, mathematical expressions and their definitions are given in the table 3.1 below. After these definitions, GAN will be explained part by part.

Table 3.1 GAN's Formula Notation

Mathematical Notation	**Meaning**
x	Real data from dataset
z	Latent space vector
$G(z)$	Fake data
$D(x)$	Discriminator's evaluation for real data
$D(G(z))$	Discriminator's evaluation for fake data
$Error(x, y)$	Error Between x and y

3.1.1 The Discriminator

The general purpose of Discriminator is to label the generated fake data is false if it is false, and true if it is true. It does this by learning the data in the dataset. A loss function is required for this structure. This function is given below (3.8).

$$L_D = Error(D(x),1) + Error(D(G(z)),0) \tag{3.8}$$

The error in the function here calculates the error of two inputs statistically. If the error is acceptable, discriminator will label it true, otherwise false.

3.1.2 The Generator

The purpose of the generator is to deceive the discriminator with the fake data it generates. These two networks are constantly in conflict. This is where the name Adversarial comes from. The loss function for the generator is defined in (3.9).

$$L_G = Error(D(G(z)),1) \tag{3.9}$$

The generator starts with a random data at the beginning and is trained with the back-propagation algorithm in each iteration. The purpose of the Generator is to produce fake data as close to reality as possible. Therefore, this error is expected to decrease as the model is trained. Since the problem here is binary classification problem, it is necessary to define the above functions as binary cross entropy.

$$H(p,q) = \mathbb{E}_{x\sim p(x)}[-logq(x)] \tag{3.10}$$

There are two neural networks and thus two loss functions in the GAN structure. When (3.8) and (3.9) are combined with (3.10), the following nested loss function is formed.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(g(z)))] \tag{3.11}$$

As can be understood from (3.11), Discriminator and Generator compete constantly. While the discriminator tries to raise its own error, the generator tries to reduce its own error. This is fixed at an optimal point at the end of the competition. This point is called the Nash equilibrium. When this point is reached from the model, fake data is produced.

After GAN was first announced, it met with high interest by people and developed rapidly. Over time, thanks to new studies on GAN, this structure has changed and as a result, the success of the outputs it produces has increased. The evolution of the image data synthesized by GAN between 2014 and 2018 is shown in Figure3.4.

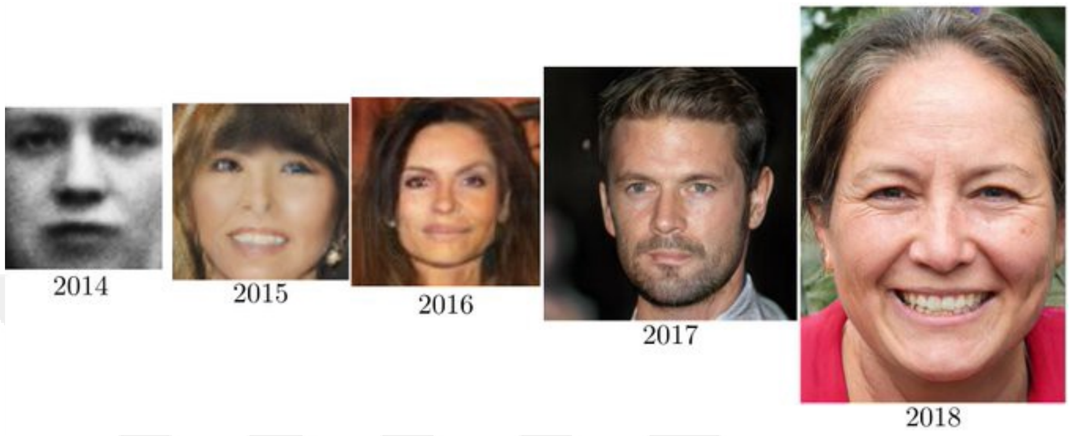


Figure 3.4 Progress of GAN generated synthetic images (Öngün, 2020)

3.2 Deep Convolutional GAN (DCGAN)

Original GAN that published in 2014 (Goodfellow et al., 2014) contains simple multi-layer perceptron models both discriminator and generator side. Thanks to the development of deep learning and the use of the convolution operator with machine learning, many difficult problems have been solved. For example, thanks to the features obtained with the convolution operator in image classification, the classification process is performed successfully. Convolution helps us not only for images, but also for signal or audio data. The mathematical formula of the discrete time convolution operator is given below (3.12).

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{3.12}$$

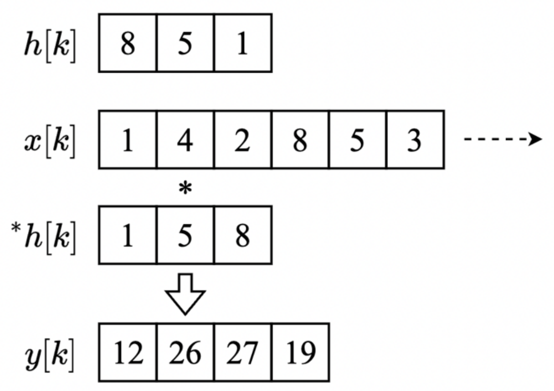


Figure 3.5 1D Convolution Example

Where x represents the input signal and h represents the kernel matrix. This matrix is predetermined in such a way that it can extract the desired feature. Sometimes the data set can be two-dimensional. In such cases, it is necessary to apply the 2D convolution process. Mathematical formula of 2D convolution is given in (3.13).

$$y[m,n] = x[m,n] * h[m,n] = \sum_{l=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} x[k,l]h[m-k,n-l] \qquad (3.13)$$

DCGAN (Radford et al., 2015) has a convolution structure in both the generator and discriminator parts. In this way, GAN incorporates some innovations provided by the convolution process. These are super-resolution, denoising and deconvolution. Figure3.6 shows how the convolution process is done for the generator.

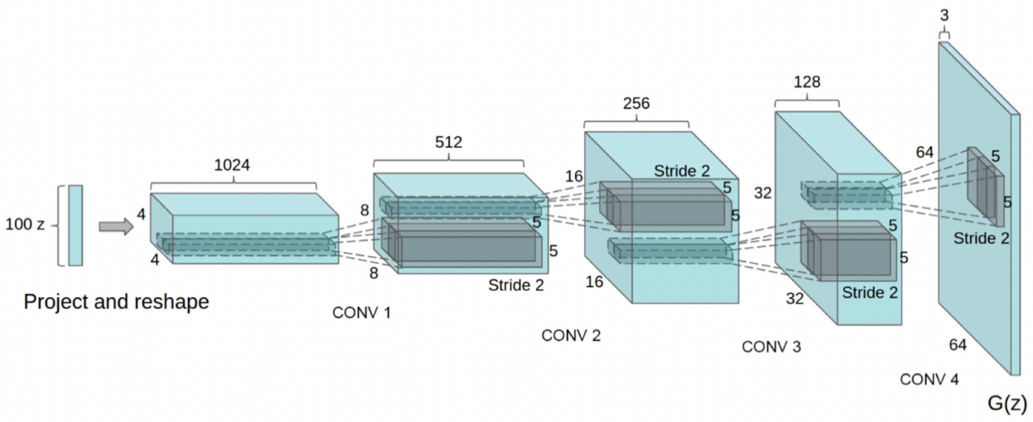


Figure 3.6 2D Convolution example for generator (Radford et al., 2015)

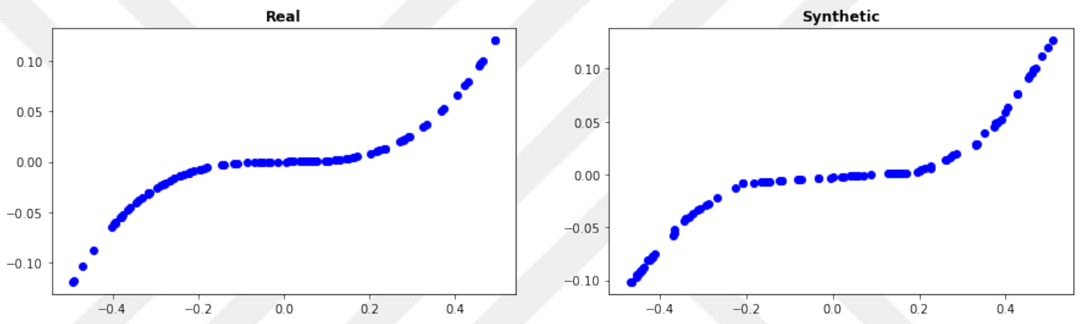


Figure 3.7 Real and Synthetic 1D signal generated by DCGAN

CHAPTER 4

METHODOLOGY AND RESULTS

In this thesis, new ECG data were synthesized from MIT-BIH Arrhythmia Database (Moody & Mark, 2001) using a new GAN-based method. Results were obtained using Python programming language and Kaggle working environment. The t-Distributed Stochastic Neighbor Embedding (t-SNE) method was used to visually examine the structure of the original and newly generated data. The cross-correlation between the newly produced ECG signals and the original signals were investigated to learn the similarities between them. In the MIT-BIH Arrhythmia Database, each record is represented by 187 samples and the 188th data is reserved for the label of that record. The MIT-BIH Arrhythmia Database is an unbalanced dataset. It consists of five classes in total. The classes and their data distributions are given in Table 4.1 and Figure 4.1.

Table 4.1 MIT-BIH Arrhythmia Database Classes

Class	**Number of data**
0- Normal	72471
1- Atrial Premature	2223
2- Premature ventricular contraction	5788
3- Fusion of ventricular and normal	641
4- Fusion of paced and normal	6431

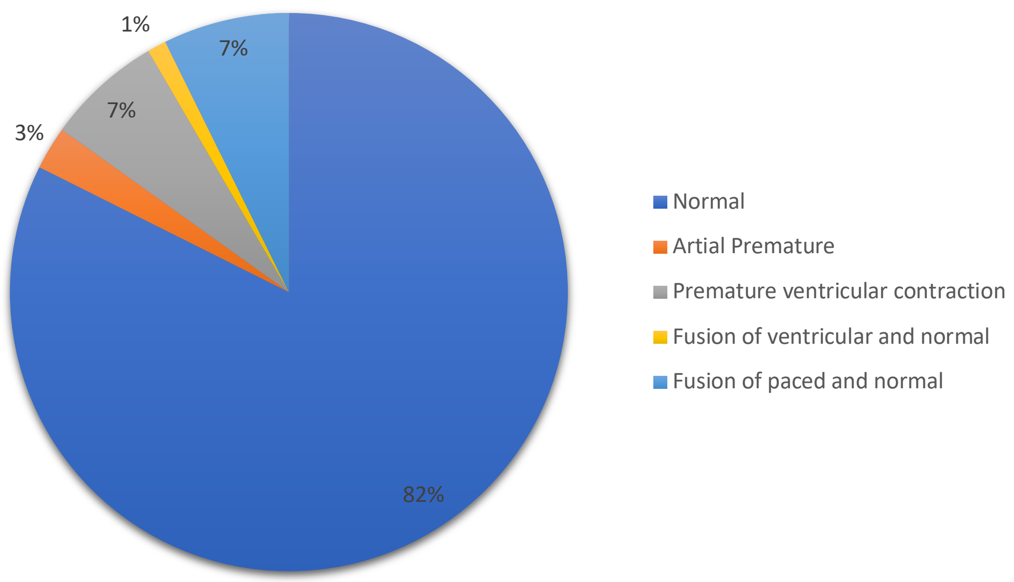


Figure 4.1 MIT-BIH Arrhythmia Database Classes

When the data set is examined, it can be understood that the Normal class has a large majority with 87% of data, while the Fusion of ventricular and normal Class has very few data (1%). This will create an underfitting problem for other classes except Normal, Fusion of ventricular and normal, and Fusion of paced and normal classes for a device that involves machine learning inside designed to process ECG signal. These few data classes need to be supported by generating synthetic data for them. This chapter consists of four sub-sections. These are ECG Synthesizing models and methods, proposed data generation method, Mathematical analysis for synthetic data and Classification performances and Results.

The data distribution structure of the data sets can be visually observed with t-SNE. This stochastic mathematical structure outputs probabilistic distributions of multidimensional data by reducing it into two dimensions. In this way, information about how much and by which method the data sets should be classified is revealed. When we look at the Dataset with t-SNE, it seems that some classes are nested. Nesting is observed in Premature ventricular contraction and Fusion of ventricular and normal classes. This is not a good situation for linear or nonlinear classifiers. The t-SNE output of the MIT-BIH Arrhythmia Database is shown in Figure 4.2.

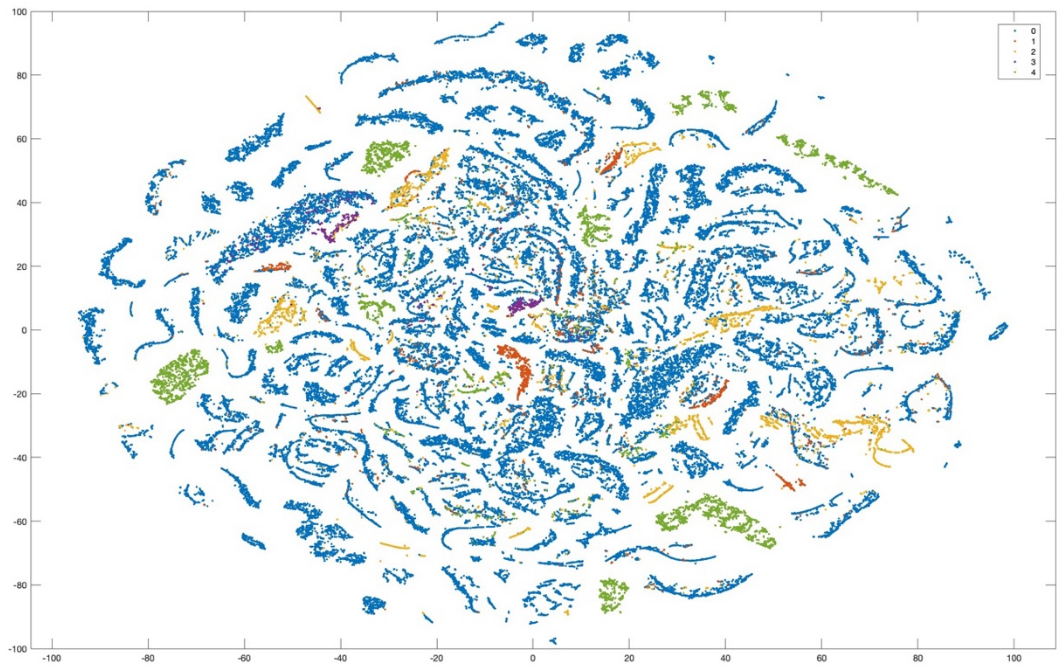


Figure 4.2 t-SNE output of MIT-BIH Arrhythmia Database

In this study, signals were synthesized using the LSTM-GAN structure (G. Zhu et al., 2019) shown in Figure 4.3. The generator structure of the model consists of an input layer with size 187, a hidden layer with size 128 and a single output layer. On the Discriminator side, an input layer size of 187, a hidden layer size of 256 and a single output layer are selected. Rectified linear unit (Relu) is used for activation functions in both parts.

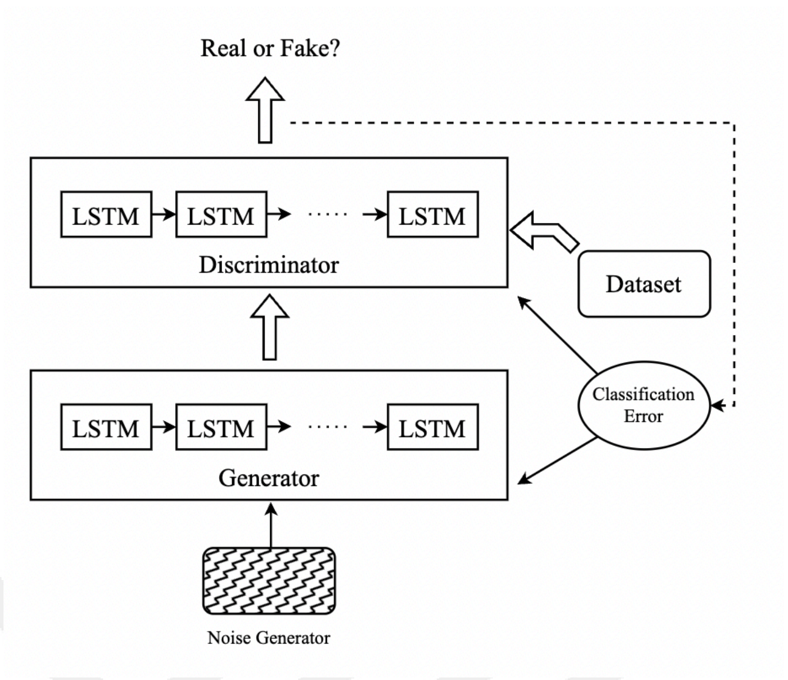


Figure 4.3 Structure of LSTM-GAN

For all studies, the GAN model was trained for individual classes using 3000 epochs on Kaggle platform. Examples of both the original signal and the generated synthetic signal for each class are presented side by side except Normal Class (Class 0) in Figure 4.4.

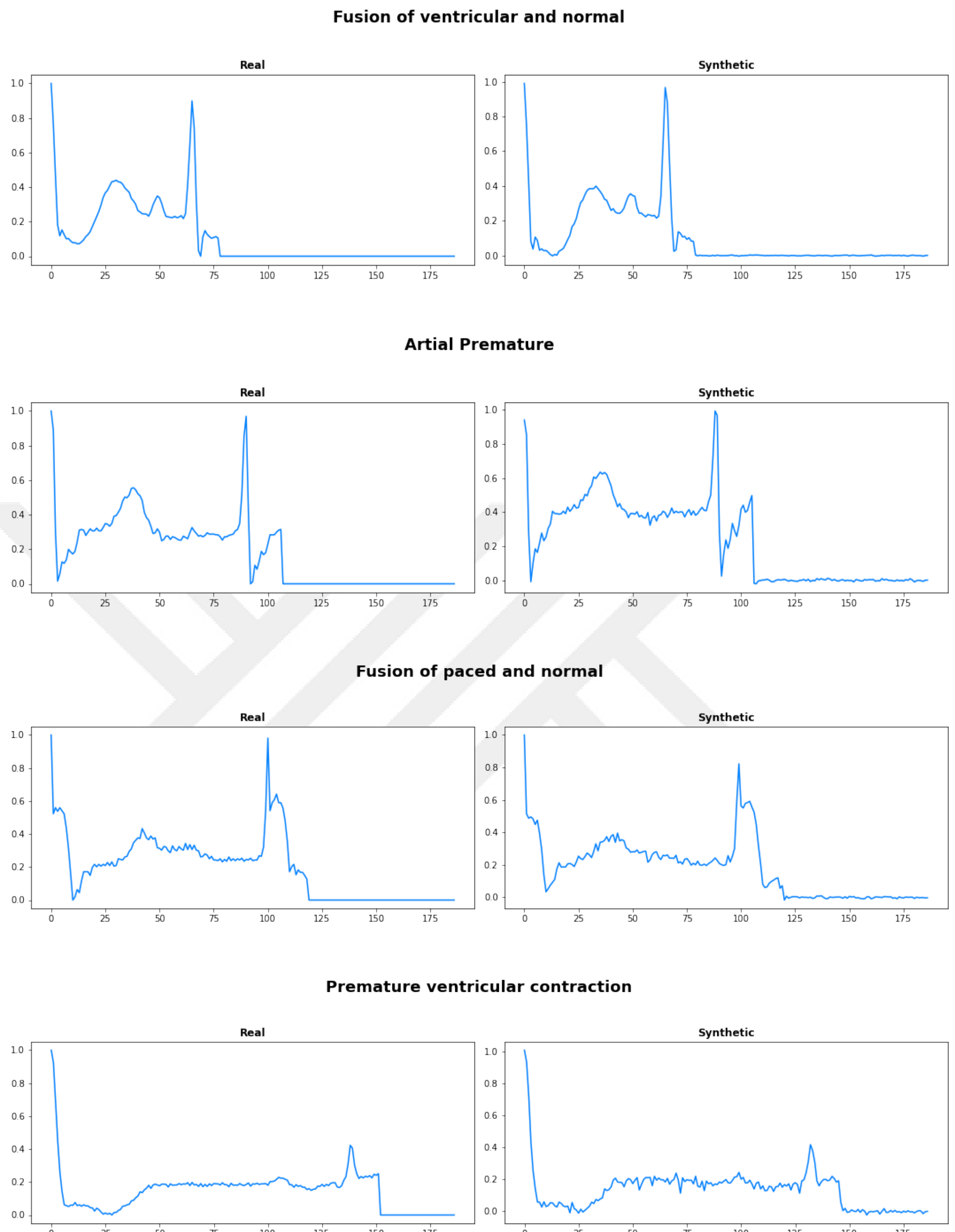


Figure 4.4 Real vs Synthetic data

4.1 Proposed Data Generation Method

It is a fact that in some data sets, there may be distinct clusters within a specific class or some of these clusters may be very close to some clusters of other classes. This situation can be easily seen from t-SNE output. In such a case, data separation becomes a difficult problem. As an example, consider a dataset of animal images containing classes, e.g. cats, foxes, dogs, seals etc. In the class of cats, however, there may be subclasses of different cat types such as Abyssinian Cat, American Bobtail Cat Breed, Chartreux Cat Breed etc. These are clusters within the same class. If the GAN is trained to produce new synthetic data using the cat class as a single entity, which is the classical use of GAN-based data generation, we will end up with new data that is somewhat a mix of all types. This is an undesirable situation because the newly synthesized data will have no resemblance to real cat data. This could lead to a potential problem. For example, a Somali cat is very similar to a fox; i.e. they are very close to each other in t-SNE output. It is therefore possible that the data synthesized from the cat class as a single entity will contain some samples that could be easily mixed with fox class.

In order to test the efficiency of the proposed method, we need two classes with unbalanced data distribution. When each class in the MIT-BIH Arrhythmia Database is examined with the help of t-SNE as shown in Figure 4.5, it is seen that Premature ventricular contraction and Fusion of ventricular and normal classes (Class 2 and Class 3) are suitable for this experiment. It can also be seen that these two classes are very close to each other, even mix in some places. This is not a desirable situation for a classification system.

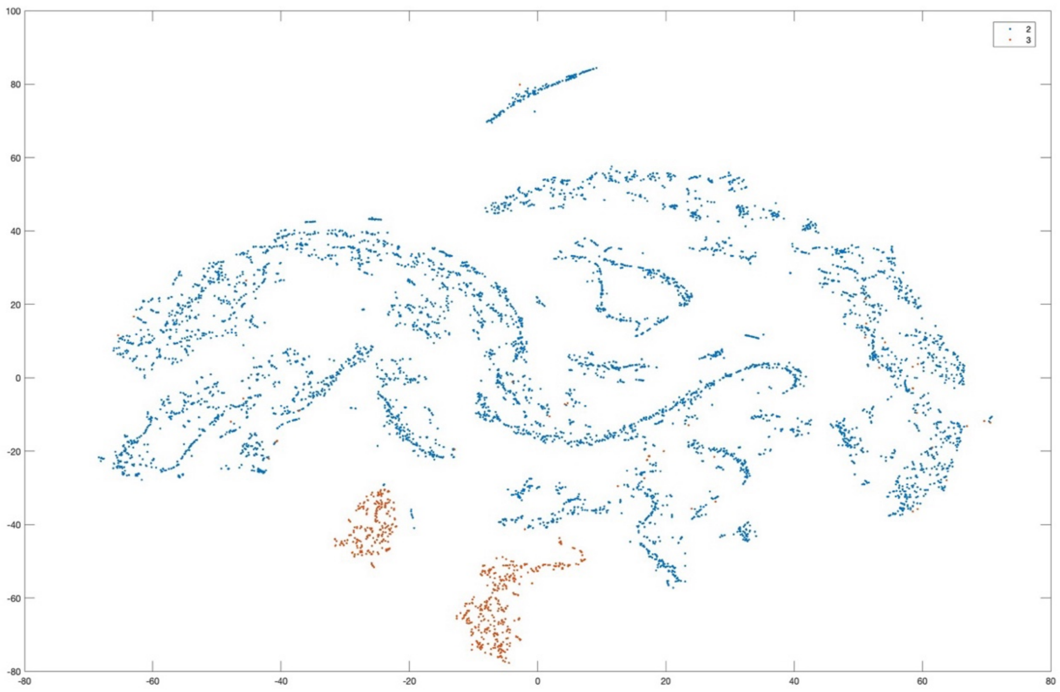


Figure 4.5 t-SNE output of original data in Class 2 and Class 3. Blue dots: Class2, Red dots: Class3

The amount of data in the Fusion of ventricular and normal class is 641, while that of Premature ventricular contraction class is 5788. Due to this unbalance situation, 600 new synthetic data from Fusion of ventricular and normal class were produced by the original GAN-based method, i.e. a single GAN for the entire class, and added to the dataset. When the number of synthesized data starts to exceed the amount of data in the class, the variance of these new synthesized data starts to increase, thus moving away from the original data. This reveals the confusion of classes. Figure 4.6 shows this problem clearly.

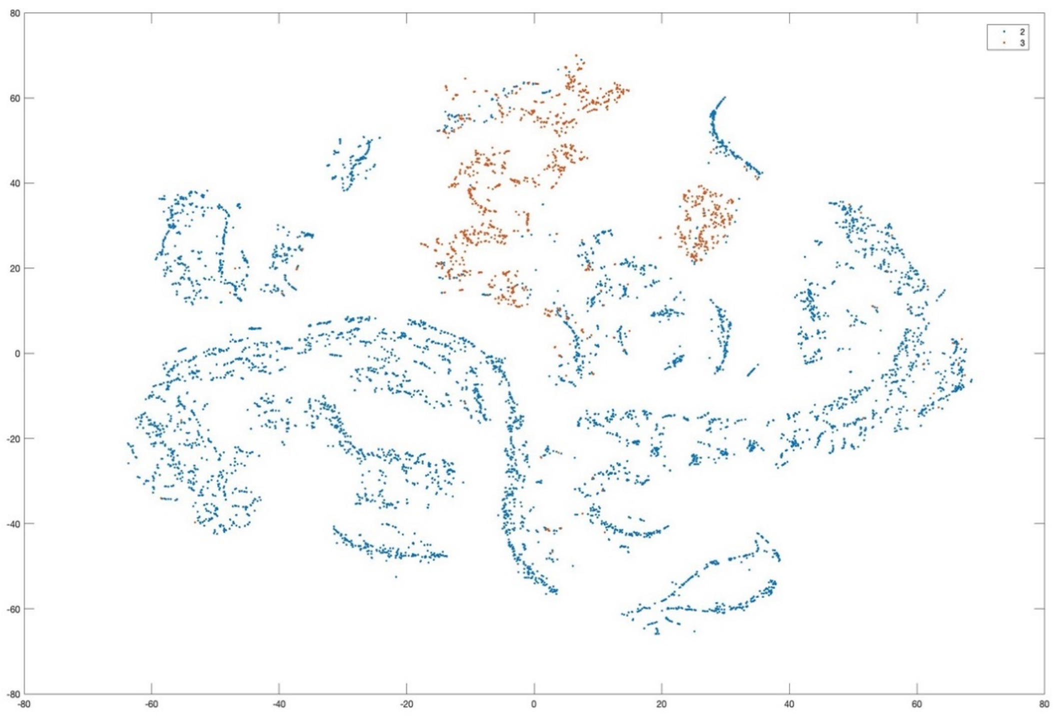


Figure 4.6 t-SNE output of Class 2 and Class 3 after data augmentation 3 by Original GAN method. Blue dots: Class2, Red dots: Class3

In order to solve this problem, possible clusters in each class should be examined. The t-SNE output of the original data in Fusion of ventricular and normal class (Class 3) is given in Figure 4.7. If the data distribution is examined, it is clearly seen that this class consists of two distinct clusters.

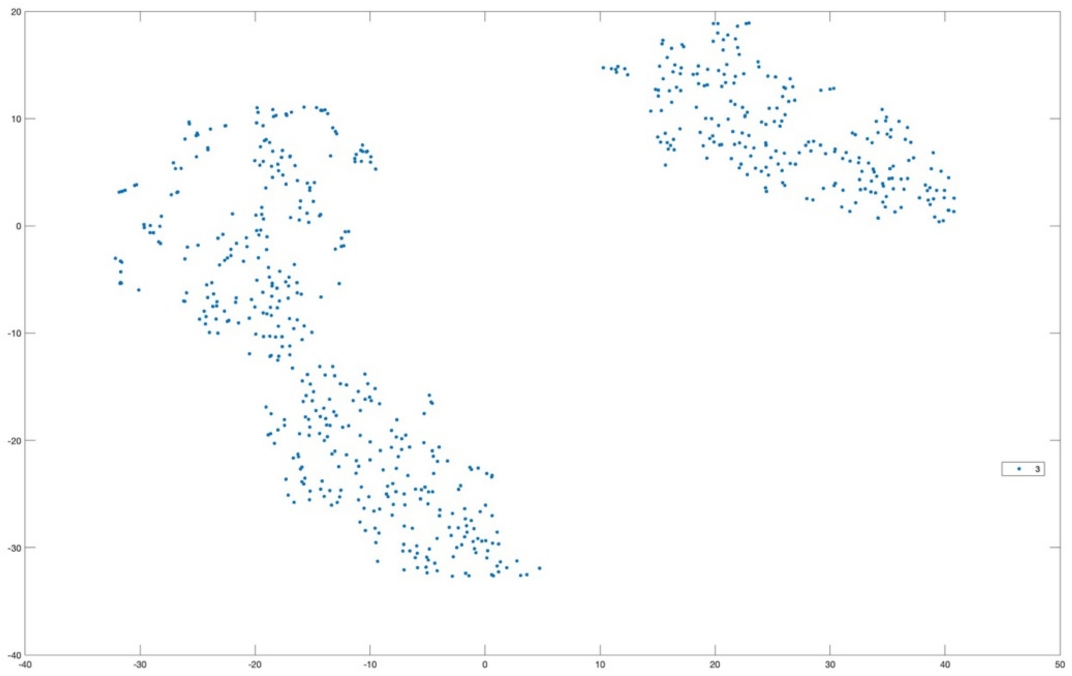


Figure 4.7 t-SNE output of original data in Class 3

If 600 new data are synthesized for this class using original GAN-based approach, the situation in Figure 4.8 emerges. As an empirical phenomenon, it has been observed that while too much data has been synthesized from one cluster, almost no data has been synthesized from the other cluster. This is due to the fact that one of the clusters dominate the other one in data size. In other words, an unbalanced data generation has occurred for these clusters. In addition, producing a large amount of data from only one cluster corrupts the variance of the data in that class. In order to overcome these problems, the classes in the dataset should be divided into clusters and new data should be synthesized for each cluster separately by a different GAN. For this process, Inverse t-SNE method (Tran et al., 2019) was used. According to this approach, it is necessary to find clusters with DBSCAN code in MATLAB, defining t-SNE indexes and go back from the t-SNE output to original data.

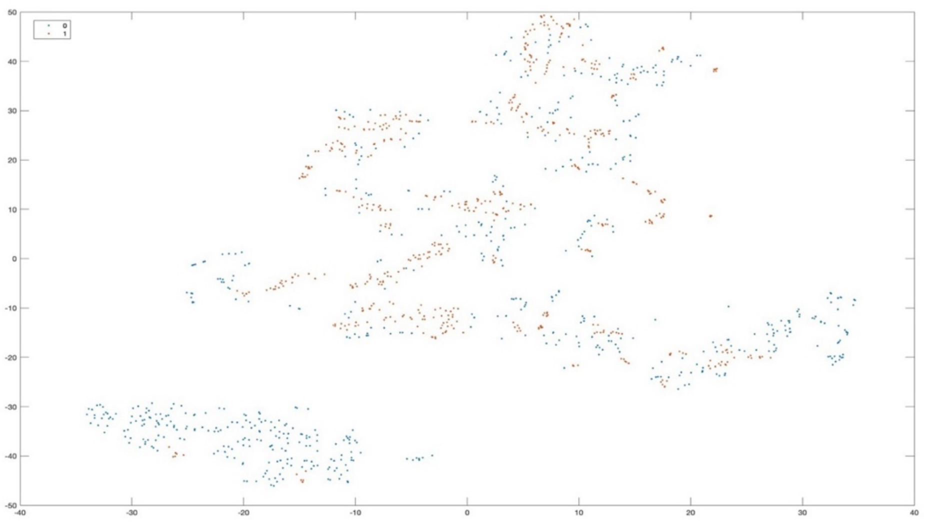


Figure 4.8 t-SNE output for Class 3 after data augmentation by Original GAN method. Blue dots: Class2, Red dots: Class3

After this separation is achieved and new data are synthesized with our proposed GAN method, i.e. a separate GAN for each cluster, the result in Figure 4.9 is obtained. This result is exactly as desired.

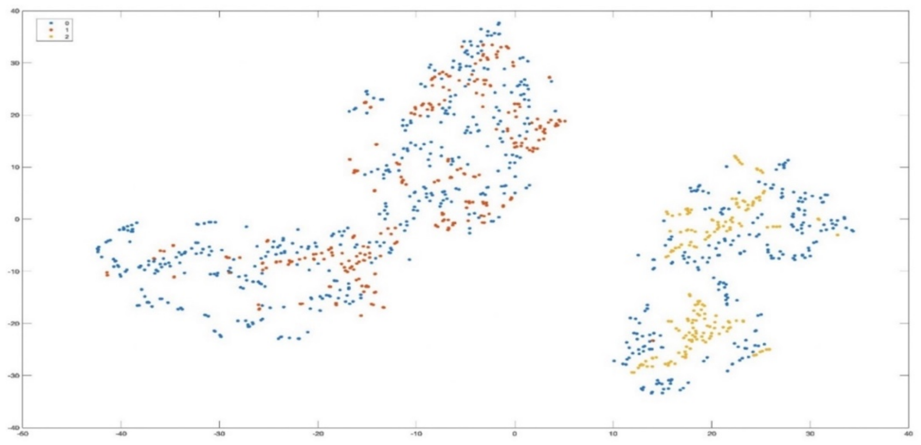


Figure 4.9 t-SNE output for Class 3 after data augmentation for 2 clusters. Blue dots: original data in Class3, Red dots: synthetic data for the 1st cluster, Yellow dots: synthetic data for the 2nd cluster

t-SNE output for Premature ventricular contraction and Fusion of ventricular and normal classes after data augmentation with the proposed method is given in Figure

4.10. If the results are examined carefully, it will be easily noticed that the classes are not mixed with each other. This will increase the success of classifier algorithms.

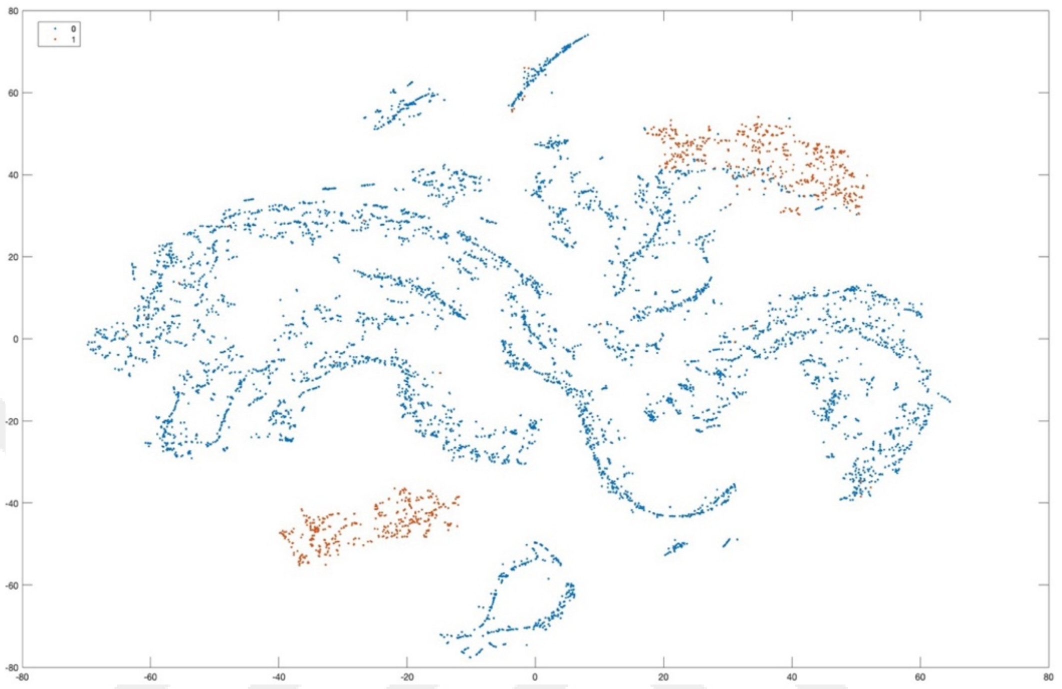


Figure 4.10 t-SNE output of data distribution of Class 2 and Class 3 after data augmentation by the proposed method. Blue dots: Class2, Red dots: Class3 augmented by proposed data generation method

Next the situation for more clusters is investigated. It is possible that the bigger cluster in Class 3 can be seen as consisting of two clusters although they are not exactly distinct. In this case, Class 3 consists of 3 clusters and 3 different GANs are created to generate new data for each of them. t-SNE output for the data distribution in this case is shown in Figure 4.11. When the t-SNE results are carefully examined, they are seen to be similar to the results in Figure 4.9.

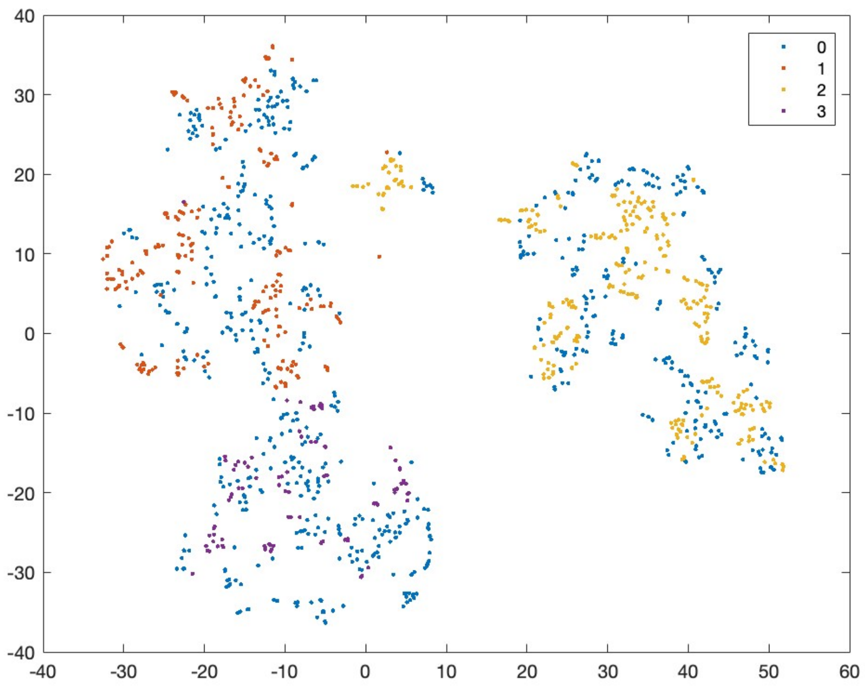


Figure 4.11 t-SNE output for Class 3 after data augmentation for 3 clusters. Blue dots: original data in Class3, Red dots: synthetic data for the 1st cluster, purple dots: synthetic data for the 2nd cluster, Yellow dots: synthetic data for the 3rd cluster

When these data are combined with Class 2, t-SNE output given in Figure 4.12 is obtained. When the data distribution in this case is examined, it is seen to be similar to the one in Figure 4.10. The conclusion to be drawn from here is that while synthesizing data separately for well separated clusters provides successful results, no benefits are obtained for non-distinct and/or close clusters.

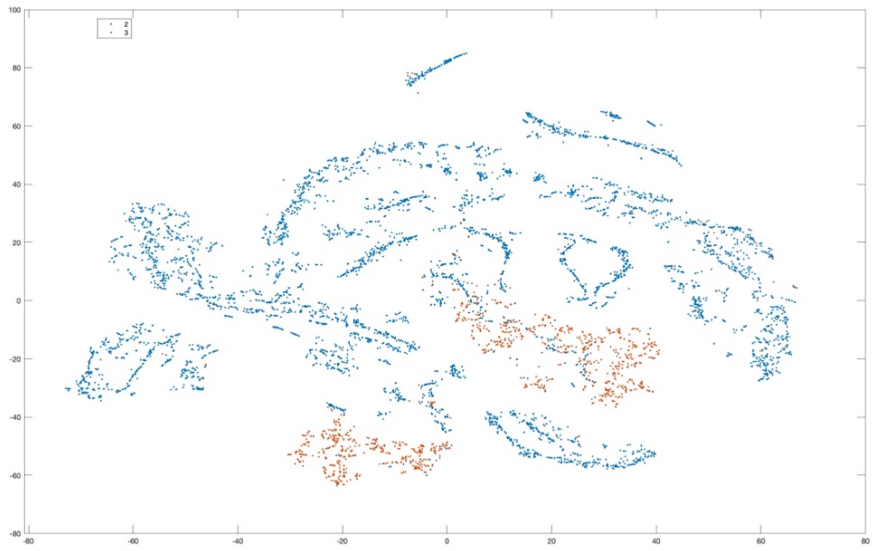


Figure 4.12 t-SNE output of 3 separated Class 3 and 2 with new approximation. Blue dots: Class 2, Red dots: Class 3

4.2 Classification Performances

Synthetic data generation serves to solve the unbalanced data and inefficient training problems of deep learning classifiers. After the examination of synthesized data with t-SNE visualization and mathematical analysis, the classification performance of the classifier which is trained with augmented data is examined. For this classification process, a single classifier model based on 1D CNN structure was used. The reason for using the same classifier in all these experiments is for making the results obtained to be independent from using different types of classifiers. The structure of the model is given in Figure 4.13.

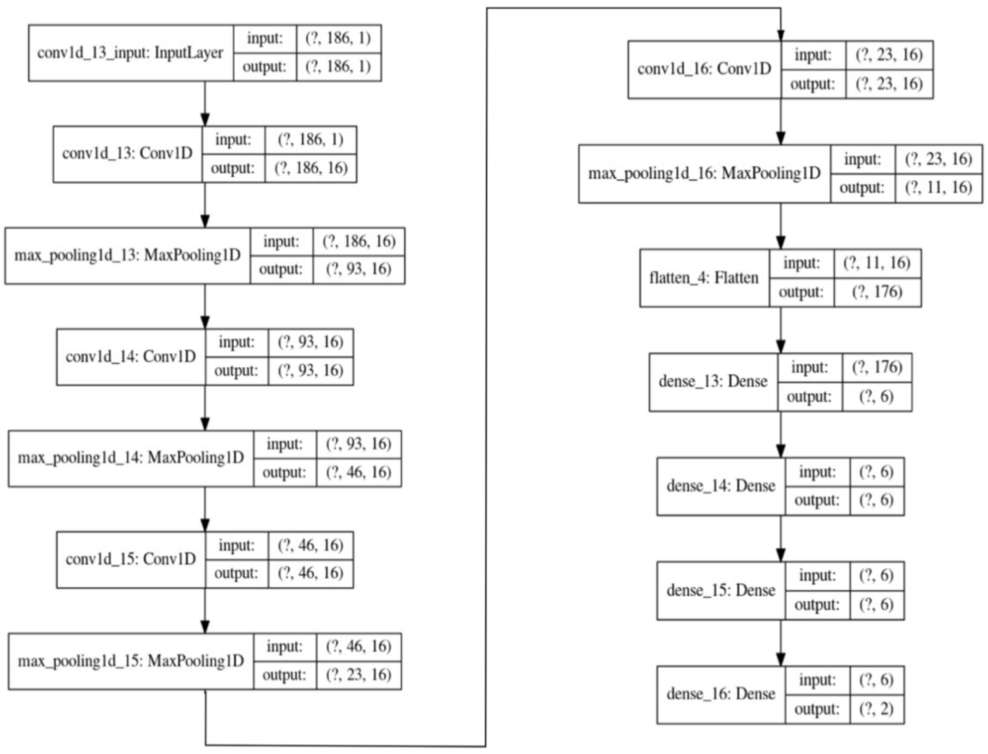


Figure 4.13 1D CNN Model

Four different experiments are conducted for the classification of Class 2 and Class 3 and the classification results of them are given in four tables below. The dataset of Class 2 is not touched in all these experiments. The test of the classifier in each experiment is done by using real data, not synthesized ones. Table 4.2 shows the results for the case where only original datasets are used (Experiment 1). The results of the case where the dataset of Class 3 is augmented by the original GAN-based method is represented in Table 4.3 (Experiment 2). Table 4.4 displays the results when the Class 3 dataset is assumed to contain two clusters and thus augmented by two different GANs (Experiment 3). The results of the experiment where the Class 3 dataset is augmented by three different GANs based on the assumption that it consists of 3 clusters (Experiment 4). The confusion matrices for all these four experiments are presented in Figures 4.14 to 4.17.

Table 4.2 Classification results of Experiment 1

	Precision	**Recall**	**F1-Score**	**Amount of Data**
Class 2	0.98	0.99	0.98	1448
Class 3	0.88	0.82	0.85	162
Accuracy			0.97	1610
Macro Avg	0.93	0.90	0.92	1610
Weighted Avg	0.97	0.97	0.92	1610

Table 4.3 Classification results of Experiment 2

	Precision	**Recall**	**F1-Score**	**Amount of Data**
Class 2	0.97	0.99	0.98	1448
Class 3	0.89	0.69	0.78	162
Accuracy			0.96	1610
Macro Avg	0.93	0.90	0.88	1610
Weighted Avg	0.96	0.96	0.96	1610

Table 4.4 Classification results of Experiment 3

	Precision	**Recall**	**F1-Score**	**Amount of Data**
Class 2	0.98	0.99	0.98	1448
Class 3	0.87	0.83	0.85	162
Accuracy			0.97	1610
Macro Avg	0.93	0.91	0.92	1610
Weighted Avg	0.97	0.97	0.92	1610

Table 4.5 Classification results of Experiment 4

	Precision	**Recall**	**F1-Score**	**Amount of Data**
Class 2	0.98	0.99	0.98	1448
Class 3	0.87	0.83	0.85	162
Accuracy			0.97	1610
Macro Avg	0.93	0.91	0.92	1610
Weighted Avg	0.97	0.97	0.97	1610

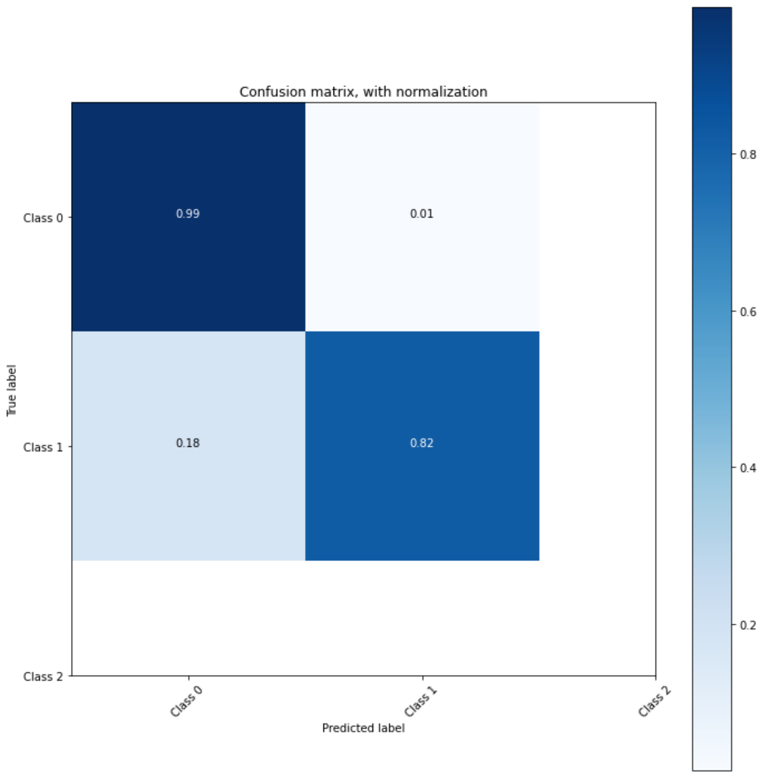


Figure 4.14 Confusion matrix of Experiment 1

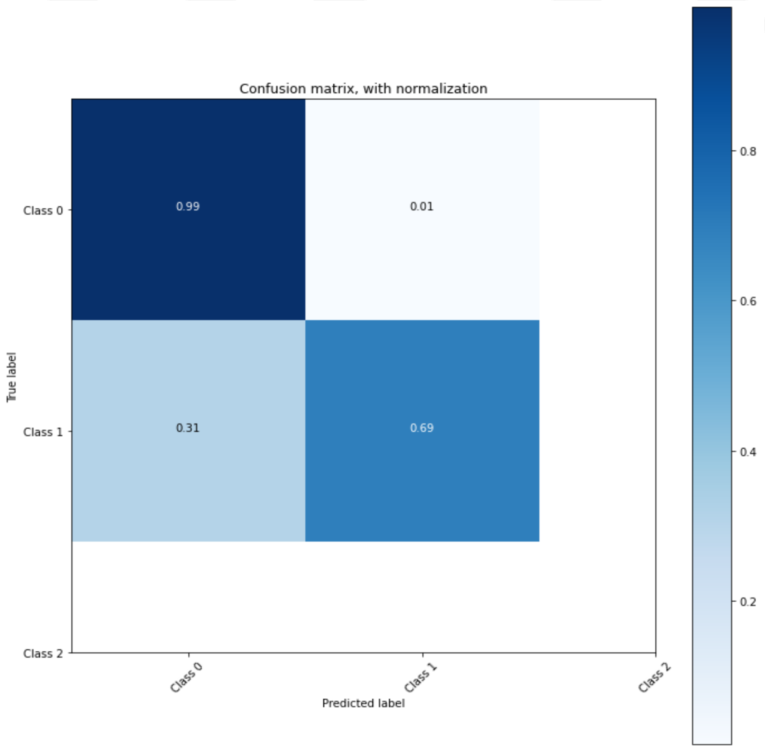


Figure 4.15 Confusion matrix of Experiment 2

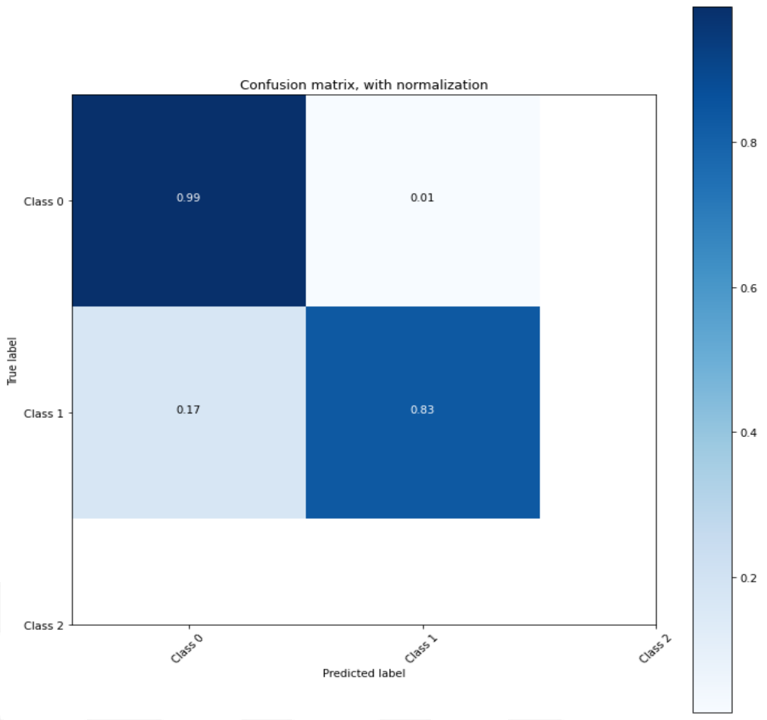


Figure 4.16 Confusion matrix of Experiment 3

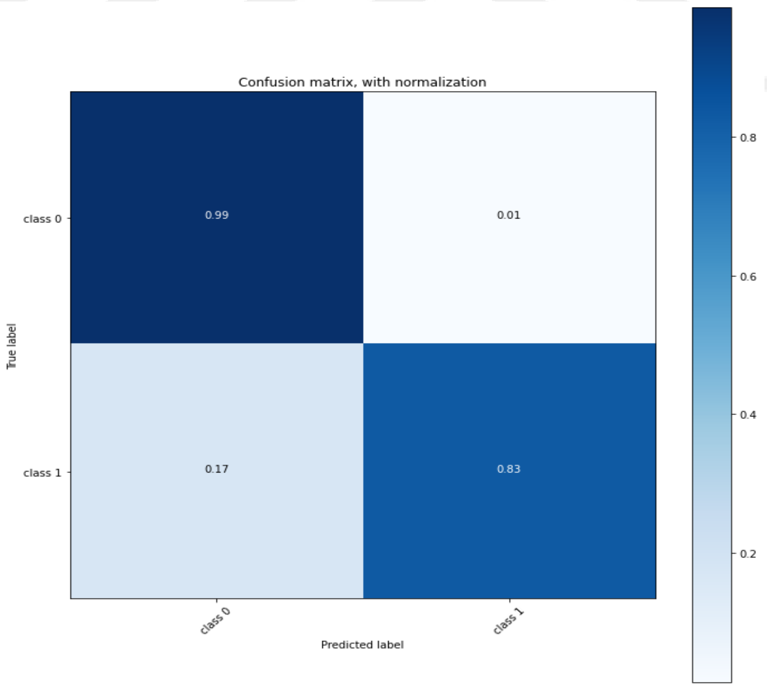


Figure 4.17 Confusion matrix of Experiment 4

If the results in Table 4.2 are carefully examined, it will be noticed there is an unbalanced dataset problem for Class 2 and Class 3. To solve this issue, when data is synthesized for Class 3 by a single GAN, the results in Table 4.3 are obtained. A decrease in classification performance can easily be observed. The reason for this is the mixed in the data in Figure 4.6. When the data is synthesized with the proposed method, the results in Table 4.4 appear. When the data is examined in detail, it is seen that the performance is as high as the original state while the unbalanced problem has also been resolved. Table 4.5 shows the classification results of separating a single cluster into more than one part and synthesizing separate data for each part separately. If the values in the table are examined carefully, it can be seen that results very similar to those in Table 4.4 were obtained and no significant improvement was observed in the performance metrics. This shows that the proposed data generation model could lead to successful results when the data is synthesized for distinct clusters in a class. Dividing single clusters into more than one part does not improve performance when this method is applied. In addition, the division of single clusters into more parts creates a disadvantage with the time and energy consumed during the training process.

CHAPTER 5
DISCUSSION

The proposed new data synthesis method uses inverse t-SNE method for effective synthetic data generation. This method provides two main benefits. First, the classes may contain distinct clusters and some of these clusters may outnumber others. In such cases, large clusters dominate the others; therefore, a small number of synthetic data is generated by GAN from the smaller clusters. In addition, some of the newly generated data could be different from data in that class. Such data synthesis obviously cannot represent real data properly. With the new data synthesis method proposed in this thesis, this problem could be avoided, and more realistic data be synthesized by preventing large clusters from dominating smaller ones. Second, sometimes there can be mixed data between two classes. A classifier probably would make some mistakes when separating these parts. If data from intermingled regions are synthesized and reproduced, the error rate will increase even more. Avoiding generating data from these regions would increase the success of the classifier. This is possible in our proposed method because we can choose for which clusters, we generate new synthetic data.

As a downside, the proposed data synthesis method may be difficult to apply to datasets containing large number of clusters due to time and computational complexities. In another situation, clusters of some classes may not have enough data to feed the GAN. In such cases, it would be more accurate to synthesize data from the entire class using a single GAN, i.e., with the original GAN-based data augmentation. When all classes of the MIT-BIH Arrhythmia Database are examined, it is revealed that there are 745 clusters. Separating these clusters one by one and synthesizing data with 745 GANs, one for each cluster, would take incredibly long time.

MATLAB, Python language and Kaggle environments were used in all studies. The same classifier model was used for all experiments. The aim is to make the performance of the proposed method independent from using different types of classifiers. The results in Table 4.3 are obtained when trying to separate Class 2 and Class 3 where Class 3 data is augmentation with original GAN-based approach, i.e., with a single GAN. When the values were carefully examined, it is observed that the classifier performance has decreased. The reason for this decrease is that one cluster in Class3, which have more data, dominate the other cluster, and thus the newly generated synthetic data loses its reality structure. The new data generation method proposed in this thesis solve this problem. With this model, Class3 is divided into two distinct clusters and data synthesis is made for each cluster separately by a different GAN. The result obtained are shown in Figure 4.9, which is the blue colored data in the image represent the original Class3 data, the other colored ones represent the newly synthesized data. When this image is carefully examined, it is noticed that the newly generated data produced now is very similar to the actual data of Class 3, and the new result is totally different than Figure 4.8.

Classifier algorithms can be used to test the dataset that was created with the newly synthesized data. In this study, the 1D Convolutional Neural Network model in Figure 4.13 was used. The success of the CNN model is as in Table 4.4 when it is desired to separate Class2 with this new data. As a result of this test, it has been shown that the imbalance problem in the dataset is resolved, and the success of the classifier has improved according to Table 4.3. Perhaps the following question can be asked here. So, does dividing the clusters into small pieces and synthesizing data from each piece increase my performance? The t-SNE result of this experiment is shown in Figure 4.11 that Class3 divided into three clusters and then, after merging Class2 with this new Class 3, the newly generated data and it's the t-SNE result are shown in Figure 4.12. When these two classes are tried to be classified with same CNN architecture, the results in Table 4.5 are obtained. No significant success was observed when the results were compared with Table 4.4. From this point of view, while obtaining clusters from classes and synthesizing data with GAN increases performance, obtaining smaller data sets from clusters and synthesizing data with GAN does not contribute to success.

CHAPTER 6

CONCLUSION

The aim of this thesis is to develop a new and efficient GAN-based data augmentation method to be used for the training of deep neural networks in both scarce data and unbalanced data set situations. The method, which is developed as an alternative to classical GAN-based data augmentation, uses the data structure of a given class which is observed by t-SNE method. As the application of the method, 1D ECG signals were synthesized by using the signals in MIT-BIH Database. The effect of the method on the performance of the classification system is tested and compared with each other for different augmented training data sets. It has been shown that there is an improvement in the classifier results when data is augmented by the proposed method. This method could be used by researchers for data augmentation purposes in many situations where a class consists of several distinct data clusters.

REFERENCES

Bayer, M., Kaufhold, M.-A., & Reuter, C. (2022). A survey on data augmentation for text classification. *Acm computing surveys*. https://doi.org/10.1145/3544558

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. v. (2019). Autoaugment: learning augmentation strategies from data. *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 113–123. https://doi.org/10.1109/CVPR.2019.00020

Dai, B., & Wipf, D. (2019). *Diagnosing and enhancing vae models*. http://arxiv.org/abs/1903.05789

Donahue, C., McAuley, J., & Puckette, M. (2018). *Adversarial audio synthesis*. http://arxiv.org/abs/1802.04208

Gandhi, A. (2021). *Data augmentation in play*. Nanonets. https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). *Generative adversarial networks*. http://arxiv.org/abs/1406.2661

Hoelzemann, A., Sorathiya, N., & van Laerhoven, K. (2021). Data augmentation strategies for human activity data using generative adversarial neural networks. *2021 IEEE International conference on pervasive computing and communications workshops and other affiliated events (percom workshops)*, 8–13. https://doi.org/10.1109/PerComWorkshops51409.2021.9431046

Jordan, J. (2018). *Example of variational autoencoder structure*. Jeremyjordan. https://www.jeremyjordan.me/variational-autoencoders/

Karras, T., Laine, S., & Aila, T. (2018). *A style-based generator architecture for generative adversarial networks*. http://arxiv.org/abs/1812.04948

Kingma, D. P., & Welling, M. (2013). *Auto-encoding variational bayes*. http://arxiv.org/abs/1312.6114

Lashgari, E., Liang, D., & Maoz, U. (2020). Data augmentation for deep-learning-based electroencephalography. *Journal of neuroscience methods*, *346*, 108885. https://doi.org/10.1016/j.jneumeth.2020.108885

Linnainmaa, S. (1976). Taylor expansion of the accumulated rounding error. *Bit*, *16*(2), 146–160. https://doi.org/10.1007/BF01931367

Liu, Z., Luo, P., Wang, X., & Tang, X. (2014). *Deep learning face attributes in the wild*. http://arxiv.org/abs/1411.7766

López, V., Fernández, A., García, S., Palade, V., & Herrera, F. (2013). An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics. *Information sciences*, *250*, 113–141. https://doi.org/10.1016/j.ins.2013.07.007

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133. https://doi.org/10.1007/BF02478259

Metlapalli, A. C., Muthusamy, T., & Battula, B. P. (2020). Classification of social media text spam using vae-cnn and lstm model. *Ingénierie des systèmes d Information*, *25*(6), 747–753. https://doi.org/10.18280/isi.250605

Mikolajczyk, A., & Grochowski, M. (2018). Data augmentation for improving deep learning in image classification problem. *2018 international interdisciplinary phd workshop (IIPhDW)*, 117–122. https://doi.org/10.1109/IIPHDW.2018.8388338

Mogren, O. (2016). *C-RNN-GAN: Continuous recurrent neural networks with adversarial training*. http://arxiv.org/abs/1611.09904

Moody, G. B., & Mark, R. G. (2001). The impact of the mit-bih arrhythmia database. *IEEE engineering in medicine and biology magazine*, *20*(3), 45–50. https://doi.org/10.1109/51.932724

Mukherjee, S., & Mulimani, M. (2022). Composeinstyle: music composition with and without style transfer. *Expert systems with applications*, *191*. https://doi.org/10.1016/j.eswa.2021.116195

Nanni, L., Maguolo, G., & Paci, M. (2020). Data augmentation approaches for improving animal audio classification. *Ecological informatics*, *57*, 101084. https://doi.org/10.1016/j.ecoinf.2020.101084

Nanni, L., Paci, M., Brahnam, S., & Lumini, A. (2021). Comparison of different image data augmentation approaches. *Journal of imaging*, *7*(12). https://doi.org/10.3390/jimaging7120254

Nassar, C. (2001). *Ideal sampling*. Sciencedirect. https://www.sciencedirect.com/topics/engineering/impulse-train

Ouyang, X., Zhang, X., Ma, D., & Agam, G. (2018). *Generating image sequence from description with lstm conditional gan*. http://arxiv.org/abs/1806.03027

Öngün, C. (2020). *Progress of GAN*. Cihanongun. https://cihanongun.medium.com/generative-adversarial-networks-gan-nedir-5cc6a48a6870

Perez, L., & Wang, J. (2017). *The Effectiveness of data augmentation in image classification using deep learning*. http://arxiv.org/abs/1712.04621

Radford, A., Metz, L., & Chintala, S. (2015). *Unsupervised representation learning with deep convolutional generative adversarial networks*. http://arxiv.org/abs/1511.06434

Rocha, B. M., Filos, D., Mendes, L., Serbes, G., Ulukaya, S., Kahya, Y. P., Jakovljevic, N., Turukalo, T. L., Vogiatzis, I. M., Perantoni, E., Kaimakamis, E., Natsiavas, P., Oliveira, A., Jácome, C., Marques, A., Maglaveras, N., Pedro Paiva, R., Chouvarda, I., & de Carvalho, P. (2019). An open access database for the evaluation of respiratory sound classification algorithms. *Physiological measurement*, *40*(3), 035001. https://doi.org/10.1088/1361-6579/ab03ea

Saldanha, J., Chakraborty, S., Patil, S., Kotecha, K., Kumar, S., & Nayyar, A. (2022). Data augmentation using variational autoencoders for improvement of respiratory disease classification. *PLOS ONE*, *17*(8), e0266467. https://doi.org/10.1371/journal.pone.0266467

Shang, Y., Miao, K., Chen, B., & Wen, Z. (2022). Transfer photo to anime with dual discriminators gan. *2022 2nd international conference on consumer electronics and computer engineering (ICCECE)*, 265–270. https://doi.org/10.1109/ICCECE54139.2022.9712766

Siarohin, A., Sangineto, E., Lathuiliere, S., & Sebe, N. (2017). *Deformable gans for pose-based human image generation*. http://arxiv.org/abs/1801.00055

Sun, Y., Wong, A. K. C., & Kamel, M. S. (2009). Classification of imbalanced data: a review. *International Journal of Pattern Recognition and Artificial Intelligence*, *23*(04), 687–719. https://doi.org/10.1142/S0218001409007326

Tanha, J., Abdi, Y., Samadi, N., Razzaghi, N., & Asadpour, M. (2020). Boosting methods for multi-class imbalanced data classification: an experimental review. *Journal of big data*, *7*(1). https://doi.org/10.1186/s40537-020-00349-y

Tao, M., Tang, H., Wu, F., Jing, X.-Y., Bao, B.-K., & Xu, C. (2020). *Df-gan: A simple and effective baseline for text-to-image synthesis*. http://arxiv.org/abs/2008.05865

Turing, A. M. (1950). *Mind a quarterly review of psychology and philosophy I.- Computing machinery and intelligence*. https://academic.oup.com/mind/article/LIX/236/433/986238

Wang, Z., Jiang, K., Yi, P., Han, Z., & He, Z. (2020). Ultra-dense gan for satellite imagery super-resolution. *Neurocomputing*, *398*, 328–337. https://doi.org/10.1016/j.neucom.2019.03.106

Wei, J., & Zou, K. (2019). *EDA: Easy data augmentation techniques for boosting performance on text classification tasks*. http://arxiv.org/abs/1901.11196

Wei, S., Zou, S., Liao, F., & lang, weimin. (2020). A comparison on data augmentation methods based on deep learning for audio classification. *Journal of physics: conference series*, *1453*(1), 012085. https://doi.org/10.1088/1742-6596/1453/1/012085

Wong, S. C., Gatt, A., Stamatescu, V., & McDonnell, M. D. (2016). *Understanding data augmentation for classification: when to warp?* http://arxiv.org/abs/1609.08764

Wu, Y., Chrysos, G. G., & Cevher, V. (2022). *Adversarial audio synthesis with complex-valued polynomial networks*. http://arxiv.org/abs/2206.06811

Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). *Modeling tabular data using conditional gan*. http://arxiv.org/abs/1907.00503

Zhang, B., Gu, S., Zhang, B., Bao, J., Chen, D., Wen, F., Wang, Y., & Guo, B. (2021). *Styleswin: transformer-based gan for high-resolution image generation*. http://arxiv.org/abs/2112.10762

Zhang, J., Dou, Q., Liu, J., Su, Y., & Sun, W. (2021). BE-ACGAN: Photo-realistic residual bit-depth enhancement by advanced conditional gan. *Displays*, *69*. https://doi.org/10.1016/j.displa.2021.102040

Zhu, G., Zhao, H., Liu, H., & Sun, H. (2019). A novel lstm-gan algorithm for time series anomaly detection. *2019 Prognostics and system health management conference (PHM-Qingdao)*, 1–6. https://doi.org/10.1109/PHM-Qingdao46334.2019.8942842

Zhu, Y., Wu, Y., Latapie, H., Yang, Y., & Yan, Y. (2021). Learning audio-visual correlations from variational cross-modal generation. *ICASSP 2021 - 2021 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, 4300–4304. https://doi.org/10.1109/ICASSP39728.2021.9414296