Investigating Deep Convolution Conditional GANs for Electrocardiogram Generation

Deepankar Nankani and Rashmi Dutta Baruah
Computer Science and Engineering Department
Indian Institute of Technology Guwahati
Guwahati, Assam, India-781039
{d.nankani, r.duttabaruah}@iitg.ac.in

***Abstract*—Cardiac arrhythmia is the leading cause of death worldwide that occurs due to irregular heartbeats obtained using an electrocardiogram (ECG) signal. The occurrence of irregular beats among the normal beats is very rare thereby creating the problem of data imbalance. This paper proposes a deep convolution conditional generative adversarial network model for ECG generation that owns the imbalanced distribution among different beat classes, namely normal, supraventricular ectopic beats, ventricular ectopic beats, and fusion beats as recommended by the Association for the Advancement of Medical Instrumentation (AAMI). Convolution neural network based generator and discriminator models are investigated that incorporates the class label information in addition to the conventional input for effective generation of minority class of beats. Different loss functions and noise distributions have been individually explored for obtaining an effective model. Moreover, the model convergence was improved with the introduction of the parameter *twist* that allows the gradient to backpropagate at a faster pace during early stages of training. The effectiveness of the model was tested on a standard benchmark MIT-BIH dataset and the quality of generated signals was measured using seven quantitative measures.**

***Index Terms*—Electrocardiogram, conditional generative adversarial network, deep convolution generative adversarial network.**

I. Introduction

Cardiac arrhythmia causes millions of deaths annually around the world [1]. The major reason behind the arrhythmia is the occurrence of irregular heartbeats obtained from an electrocardiogram (ECG) signal. These beats can be diagnosed through supervised machine learning methods [2]–[4]. A major drawback of these methods is the requirement of labelled data in huge quantity that is difficult to obtain in practice [5]. Although the open source data solves this problem but it sometimes do not satisfy the criteria for a study and often misses out on relevant information. Recently, the researchers sought off to synthetically generate ECG data using predefined synthesisers and machine learning techniques including generative adversarial networks (GANs) [6]–[16].

A. Need for GANs

GANs [17] learn to generate synthetic data by mapping latent input noise to the data distribution. GANs have depicted exceptional performance in the domain of computer vision by generating very real looking synthetic images [18]–[20]. GANs have also been used to generate synthetic music and speech signals [21] using an algorithm called WaveNet. Donahue et al. produced raw waveform audio (WaveGAN) using DCGAN [22] and generated different sound signals using unconditional autoregressive models. GANs have also been employed for the generation of single-channel electroencephalogram (EEG) signal [23].

B. Related Work

Initially data augmentation was performed using interpolation and extrapolation but these methods were limited by the density of the given data. Mc Sharry et al. produced synthetic ECG signals [6] with user defined heart rate characteristics using three coupled ordinary differential equations. Clifford et al. [7], [8] presented a nonlinear model that generated electrocardiogram signal using a 3-D vectorcardiogram formulation. Cao et al. [9] generated conventional twelve lead ECG signals. The problem with the aforementioned mathematical models is the reliability on independent systems and predefined functions making it difficult to gather enough information about the global characteristics of an ECG signal.

Recently, researchers have been employing GANs for biomedical signal generation. Brophy et al. [15] used Wasserstein GANs (WGANs) [24] with gradient penalty to synthetically generate sinusoidal, photoplethysmogram, and ECG signals by converting the time series signal to rasterized images and feeding it to GANs. The new images generated from GANs are converted back to time series signals. However, the generated signal quality is limited by the image resolution. Haradal et al. [10] used LSTM model to augment ECG and EEG datasets and compared the performance with other existing data augmentation methods and conveyed that GANs are more suitable than conventional methods for augmenting minority classes. Zhu et al. [12] used bidirectional LSTM and CNN for ECG generation. Zhou et al. [13] proposed Beat GAN for Anomalous Rhythm Detection and provided explainable results to pinpoint the anomalous segment but used the latent input noise obtained using the data itself, making it biased. Delaney et al. [14] generated synthetic sinusoidal and normal ECG using two bidirectional LSTMs and CNN in addition to a minibatch discrimination layer. Wang et al. [16] employed an auxiliary classifier GAN to generate normal, left, and right bundle branch block, premature atrial contraction , and premature ventricular contraction beats for single and

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

consecutive heartbeat abnormality detection. The problem in the aforementioned methods is that these methods have not generated the class of beats recommended by AAMI [25]. Hence, the minority class data still needs to be augmented. We came across only one work by Golany et al. [11] in which they proposed a personalized GAN that synthesizes minority class beats such as supraventricular and ventricular ectopic beats, and fusion beats in which they employed a combination of two losses, namely mean square and binary cross entropy loss using an LSTM model. The current literature still lacks the quantitative metrics needed to verify the generated signal quality.

C. Our Contributions

We aim to generate AAMI [25] recommended four beat classes, namely normal (N), supraventricular ectopic beats (SVEB), ventricular ectopic beats (VEB) and fusion of ventricular and normal beat (F) using a deep convolution conditional generative adversarial network. CNNs are preferred over recurrent neural networks as they provide better results over time series signals in the literature [5]. We investigated two different loss functions, cross entropy and mean square loss independently in the discriminator. Uniform and gaussian noise were also provided as an input to the generator for generating synthetic signal. A new parameter *twist* was also introduced for faster loss convergence during the training phase. We evaluated the generated beats quantitatively using seven evaluation metrics namely, Frechet Inception Distance (FID), Dynamic Time Warping (DTW), Euclidean Distance (ED), Pearson Correlation (PC), Maximum Mean Discrepancy (MMD), Kullback Leibler Divergence (KLD), and Time Warp Edit Distance (TWED). The results depicted that our model effectively generated different classes of beats.

The organization of the paper is as follows. Section II provides the database description and the preprocessing steps. Section III provides an overview of the ECG generation methodology. Section IV provides the experimental setup. Section V discusses upon the results and section VI provides conclusion with the future scope.

II. Data Description

Massachusetts Institute of Technology-Beth Israel Hospital (MIT-BIH) Arrhythmia database [26] is used for experimental purpose. It consists of annotated normal and clinically significant arrhythmic beats sampled at 360 Hz. We used the modified limb lead II signal for this work. Paced beat recordings were discarded, namely 102, 104, 107, and 218. The annotations provided in the dataset categorize the beats into 15 different classes (f, /, N, L, e, R, j, Q, A, J, a, V, S, E, F), each denoting a different type of beat. We follow the AAMI [25] recommended beat types, namely N including (N, L, e, R, j), SVEB including (A, J, a, S), VEB including (V, E), F, and Q. Q class was discarded since it was marginally represented and represents the heartbeats that are unclassified as mentioned in the database.

A. Preprocessing

The MIT-BIH records were contaminated with noises such as Power Line Interference (PLI), i.e., high-frequency noise (50 or 60 Hz) and Baseline Wander (BW), i.e., low-frequency noise. Hence, a notch filter was employed to remove PLI, and a mean median filter was employed to remove BW [27]. The beats of length 260-time ticks were extracted that represent 234 ms and 486 ms before and after the R-peak, respectively at 360 Hz sampling rate. No resampling was performed to minimize the loss of information. The minority class beats (SVEB, VEB, F) were augmented using window warping technique as proposed in [13]. Then uniform noise $\mathcal{U}[0.1, 0.15]$, was added to increase the number of beats and make them equal to the normal beats. We represented a single heartbeat as $hb(i) = \{v_1, ..., v_{260}\}$, where v_t denotes voltage at time t.

III. ECG Generation Methodology

A. Generative adversarial networks

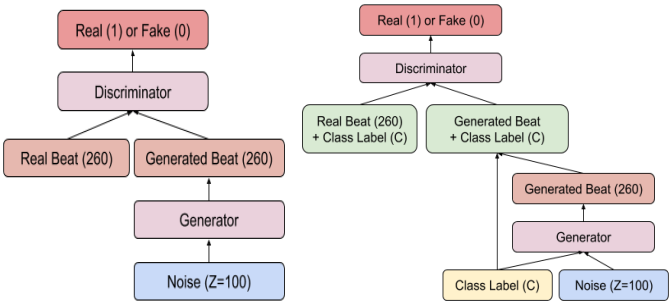
Generative adversarial networks (GANs) proposed by Ian Goodfellow [17] in 2014, belong to the class of generative models that deploy a neural network for both the discriminator and the generator. In 2015, Radford et al. proposed the deep convolutional GAN (DCGAN) to generate synthetic images [28] using CNN to improve stability in training. GANs consist of a generator network $G(\mathbf{z})$, where $\mathbf{z}$ represents the latent input noise and a discriminator network $D(\mathbf{x})$, where $\mathbf{x}$ represents the input to the discriminator. GANs learn to generate synthetic data until the Nash equilibrium between the two networks is attained via adversarial training. $G(\mathbf{z})$ maps the latent input noise to the probability distribution of real data P_r and the discriminator network learns to discriminate between the real and generated samples. Formally, G and D play a minmax game with two players where the evaluation function is defined using Eq. 1.

$$\begin{aligned}\min_G \max_D V(D,G) = E_{x\sim p_r(x)}[\log D(x)] \\ + E_{z\sim p_z(z)}[\log(1 - D(G(z)))]\end{aligned} \tag{1}$$

where $p_r(x)$ and $p_z(z)$ are the distributions of the real and synthetically generated data, respectively. Both, generator and discriminator are learned by alternately maximizing and minimizing $V(G, D)$ until $p_z(z)$ is indistinguishable from the true connectivity distribution. Figure 1a represents basic GAN.

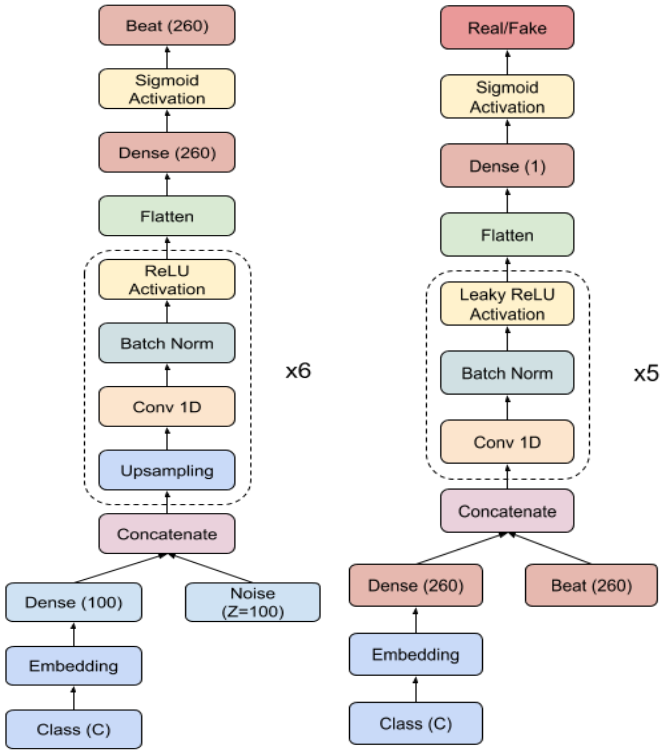
B. Proposed DCCGAN Model

We employed deep convolution GAN (DCGAN) [28] because of its stable training in addition to the superior performance on several tasks as compared to conventional GANs. Since our task is to generate class-specific heartbeats (N, SVEB, VEB, and F), the conventional architecture of GAN had to be modified. A new condition, namely class information, was incorporated in GAN, which led to the use of Conditional GANs (CGANs) [29]. The conditional training of



(a) GAN (b) Conditional GAN

Fig. 1: Basic Architecture of GAN and CGAN.



(a) Generator Network (b) Discriminator Network

Fig. 2: Illustration of Generator and Discriminator Network.

the DCGAN based models may be referred to as deep convolution conditional generative adversarial network (DCCGAN). In DCCGANs, G and D are provided with labels as well as the conventional input transforming $G(z)$ to $G(c, z)$ and $D(x)$ to $D(c, x)$, where c represents the class information. Figure 1b shows the structure of the proposed DCCGAN. The architecture details of G and D is illustrated in Figure 2a and Figure 2b. The underlying heartbeat distribution for a given arrhythmic beat class is denoted by the conditional probability $p(hb|c)$, where hb depicts the heartbeat and c depicts the arrhythmic beat class. Given a set of heart-beats $\{hb_1, ..., hb_K\}$ the generator tries to fit $p(hb|c)$ and generate new synthetic heartbeats that resemble to the real heartbeats of a desired class very closely in order to deceive the discriminator. Training of DCCGAN, details of Generator, and discriminator are provided in subsequent sections.

C. Generator

The generator receives a sequence $z = \{z_n\}_{n=1}^{N}$ where N is the length of latent input and a corresponding class label c, where $c \in \{\mathbb{Z}^+ \cup 0\}$. The latent input z of length N is randomly sampled from a noise (gaussian or uniform) distribution. The generator then produces a series $G(z_n) \in \mathbb{R}^k$ of fixed length k, where k is the same as the dimension of the real training signal. G tries to approximate underlying real heartbeat distribution provided class label $p(hb|c)$ and generates synthetic ECG heartbeats for a corresponding arrhythmic class c to deceive the discriminator. The objective of generator is to minimize log-probability of discriminator such that the D correctly assigns negative labels to the synthetically samples generated by the G. G is optimized by providing the generated signal (heartbeat) as input to D thereby optimizing its parameters so that D predicts synthetic signal (heartbeat) as a real signal. The generator loss is depicted by Eq. 2:

$$L(G) = -\log D(G(z)) \tag{2}$$

where, G_z is the generated heartbeat from the generator network. Figure 2a depicts the architecture of the generator used in this work.

D. Discriminator

The discriminator $D(hb)$ ingests real time series $x_n \in \mathbb{R}^T$ and fake time series $G(z_n) \in \mathbb{R}^T$, where, T represents the dimension of the training data, separately and outputs the probability that denotes whether hb originates from the original beats or synthetically generated beats. The discriminator loss is described using Eq. 3:

$$L(D) = -\log D(hb_{real}) - \log(1 - D(hb_g)) \tag{3}$$

where, hb_{real} and hb_g represents ECG heartbeat obtained from training and generator network, respectively.

E. Modified Loss Function

Since the DCCGAN consists of class label c, the modified loss function takes into account the original beats, synthetically generated beats, and class label is represented by Eq. 4:

$$\min_G \max_D V(D, G) = E_{c,x\sim p_r(c,x)}[\log D(c, x)] + E_{x\sim p_r(x), z\sim p_z(z)}[\log(1 - D(G(z, x), x))] \tag{4}$$

where, $p_r(x)$ and $p_z(z)$ are the data distributions of the original and synthetically generated beats, respectively.

IV. Experimental Setup

A. System Configuration

The experiments were performed on a workstation with Intel Core i5-6500 CPU with 3.6 GHz frequency, 16 GB RAM, and an Nvidia 1050 Ti GPU with 4 GB of VRAM. The models were developed using Python library keras as it is optimised for faster training on GPUs.

B. Generator Architecture

The input layer of generator takes two inputs, a latent input z from a noise distribution and a random integer for class label c, where $z = 100$, and $c \in \{0, 1, 2, 3\}$. Figure 2a represents the architecture of our generator and Table I depicts in-depth details of our generator. The label incorporation in the generator can be performed in many ways. We used an embedding layer followed by a dense (FC) layer of a size similar to the input and then concatenated it with the input data. The generator model is constructed using a CNN based architecture to adapt to time series nature of ECG. We used 1-Dimensional convolution layers in addition to upsampling that composedly work as a deconvolution layer. 1-D CNNs are employed because of their well suited behaviour towards time series signals [5]. Between the convolution layers, we have employed batch normalization [30], ReLU activation, and upsampling by a factor of two. In the last layer sigmoid activation is employed. Parameters represent the feature maps, filter size, and stride in case of convolution layers and number of neurons in case of dense or fully connected layers.

TABLE I: Detailed Generator Architecture

Layer	Input Size	Parameters	Output size
Embedding	1, 100*1	50	1*50, 100*1
Dense	1*50, 100*1	100	100*1, 100*1
Reshape	100*1, 100*1	-	100*1, 100*1
Concatenate	100*1, 100*1	-	100*2
UP1	100*2	2	200*2
Conv1	200*2	32*16,10,1	191*512
BN-ACT-UP2	191*512	-	382*512
Conv2	382*512	32*8,4,2	191*256
BN-ACT-UP3	191*256	-	382*256
Conv3	382*256	32*4,4,2	191*128
BN-ACT-UP4	191*128	-	382*128
Conv4	382*128	32*2,4,2	191*64
BN-ACT-UP5	191*64	-	382*64
Conv5	382*64	32*1,4,2	191*32
BN-ACT-UP6	191*32	-	382*32
Conv6	382*32	1,4,2	191*1
BN-ACT-FLAT	191*1	-	191
Dense-ACT	191	260	260

C. Discriminator Architecture

Discriminator ingests an input layer with a dimension similar to beat length and a random integer for class label. Label is incorporated in a similar fashion to the generator. Figure 2b represents the architecture of our discriminator and Table II depicts in-depth details of our discriminator. 1-D CNN layers are used with batch normalization [30] and Leaky Rectified Linear Unit activation in between 1-D CNN layers except the last layer where sigmoid activation function is employed.

D. DCCGAN Architecture

DCCGAN draws k random numbers from a noise distribution N and a random integer for class label. DCCGAN is comprised of the generator followed by the discriminator. Figure 1b illustrates the specific structure of DCCGAN, where the details of G and D are depicted in Figure 2a and 2b.

TABLE II: Detailed Discriminator Architecture

Layer	Input Size	Parameters	Output size
Embedding	1, 260*1	50	1*50, 260*1
Dense	1*50, 260*1	260	1*260, 260*1
Reshape	1*260, 260*1	-	260*1,260*1
Concatenate	260*1,260*1	-	260*2
Conv1	260*2	32,4,2	130*32
BN-ACT	130*32	-	130*32
Conv2	130*32	32*2,4,2	65*64
BN-ACT	65*64	-	65*64
Conv3	65*64	32*4,4,2	33*128
BN-ACT	33*128	-	33*128
Conv4	33*128	32*8,4,2	17*256
BN-ACT	17*256	-	17*256
Conv5	17*256	32*16,4,2	9*512
BN-ACT	9*512	-	9*512
Flatten	9*512	-	4608
Dense-ACT	4608	1	1

E. Discriminator Training

The discriminator training involved two phases during each epoch: (i) training over m real ECG samples and (ii) training over m randomly generated samples via generator network. Firstly, the D was provided with a random batch of real data $X \in \mathbb{R}^{m \times T}$, that produces d_real as the loss obtained during the training over real samples. Secondly, the D was provided with a batch of fake data $Z \in \mathbb{R}^{m \times T}$, that produces d_fake as the loss obtained during the training over fake samples. The average of these two losses can be taken as the overall discriminator loss during that epoch. A "twist" phenomenon was also introduced during the training phase, where the generated sampled were assigned the label of 0.1 in place of 0, and similarly, the real data samples were assigned the label of 0.9 in place of 1. This allowed the model to converge faster by alleviating the problem of vanishing gradient during the early phase of training. Adam optimizer [31] was employed during the training with $LR = 0.0002$. as suggested in [28].

F. DCCGAN Training

After training the discriminator, the parameters of discriminator were frozen so that they do not get updated during the training of the generator. During the training of GAN, m random noise sequences $\{z^1, ..., z^m\}$ are sampled from noise distribution p_z and fed to G. G then generates fake samples that are provided with the class label of 0.9 ,i.e., labels corresponding to the original beats and are provided as input to the D. D returns high error that is backpropagated to generator network in order to optimise G weights. Now again the D is trained as explained in IV-E. The aforementioned steps are repeated for several number of batches and force the loss of generator and discriminator to alternately maximize and minimize the value function $V(G, D)$ as provided in Eq. 1 and achieve the Nash Equilibrium by approximating the generator probability distribution $p_z(z)$ to the original beats distribution. The training is not stopped by monitoring the losses as the GAN training is highly unstable in nature. Adam optimizer

[31] was employed during the training of discriminator with $LR = 0.0002$ as suggested in [28].

G. Evaluation Metrics

The model performance is measured by the quality and diversity of the generated signals. The generated signals corresponding to a class should persist low entropy (quality), whereas the entropy across signals of that class should be high (diversity). The former property forces the signal to resemble that particular class, and the later property alleviates the chances of mode collapse in the generator. Regarding the former property, we calculate the class-wise scores, and regarding the later property, we calculate the probability distribution of generated data P_g and real data P_r of each class individually. We used seven evaluation metrics namely, Frechet Inception Distance (FID), Dynamic Time Warping (DTW), Euclidean Distance (ED), Maximum Mean Discrepancy (MMD), Pearson Correlation (PC), Kullback Leibler Divergence (KLD) and Time Warp Edit Distance (TWED).

Frechet Inception Distance Evaluates synthetic signals based on the statistical information of generated beats compared to the statistical information of real beats [32]. FID can be calculated using Eq. 5.

$$d^2((\mu_1, \sigma_1), (\mu_2, \sigma_2)) = ||\mu_1 - \mu_2||^2 + Tr(\sigma_1 + \sigma_2 - 2 * \sqrt{\sigma_1 * \sigma_2}) \tag{5}$$

where, μ_1, μ_2 are the mean of real and synthetically generated beats. σ_1, σ_2 are the covariance matrix or the second moment for the real and synthetically generated beats. Tr refers to the trace linear algebra operation (element sum) along the main diagonal of the square matrix.

Maximum Mean Discrepancy Calculates the distance between the mean of P_r and P_g. However, the variance of probability distributions might be different [33]. Gaussian MMD can be calculated using Eq. 6.

$$MMD^2 = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq 1}^{n} \kappa(x_i, x_j) - \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \kappa(x_i, y_j) + \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq 1}^{m} \kappa(y_i, y_j) \tag{6}$$

where, $\kappa(x, x') = \sum_{j=1}^{k} e^{-\alpha_j \|x - x'\|^2}$ with bandwidth α equal to the pairwise distance between the joint data.

Dynamic Time Warping Estimates dissimilarity between two varying time series signals [34]. It optimally aligns the two time series signals and calculate the distance between them. Since the heartbeats become either fast or slow naturally, DTW becomes more suitable in these type of situations [13]. Accumulated cost is calculated using Eq. 7.

$$D_{i,j} = f(x_i, y_j) + min\{D_{i,j-1}, D_{i-1,j}, D_{i-1,j-1}\} \tag{7}$$

where $f(x_i, y_j) = (x_i - y_i)^2$, for $i = \{1, ..., N\}$ and $j = \{1, ..., M\}$ where N and M is the lengths of signal x and y. Fast DTW was employed for approximating DTW, owing to this the computational complexity is reduced to $O(N)$ [35].

Euclidean Distance Quantifies the distance between generated signal x and real signal y. ED can be calculated using Eq. 8.

$$d_{ED}(x, y) = \sqrt{\sum_{k=1}^{N} (x_k - y_k)^2} \tag{8}$$

where N is number of samples in signal.

Pearson Correlation Measures the strength of linear correlation between two signals. It varies from $-1 \leq PC \leq 1$. Higher the value of PC better the correlation between the signals. However, the drawback is its sensitivity towards outliers, as a single outlier reduces the correlation hugely. PC between signal X and Y is calculated using Eq. 9.

$$PC(X, Y) = \frac{\sum_{i=1}^{n} (X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \overline{X})^2} \sqrt{\sum_{i=1}^{n} (Y_i - \overline{Y})^2}} \tag{9}$$

where, X_i, Y_i are individual samples of signal X and Y. $\overline{X}$ and $\overline{Y}$ are the mean of X and Y and n is the length of signal.

Kullback Leibler Divergence KLD represents the amount of information loss while approximating one distribution with the other [36]. However, KLD is not symmetric. KLD between P_r and P_g defined on the same probability space is provided by the Eq. 10.

$$D_{KL}(P_r || P_g) = \sum_{i=1}^{N} P_r(x_i) . \log \frac{P_r(x_i)}{P_g(x_i)} \tag{10}$$

where, $D_{KL}(P_r || P_g)$ is often called the information gain achieved if P_g is used instead of P_r. $P_r(x)$ is one of the signal obtained from distribution P_r.

Time Warp Edit Distance It represents the elastic distance metric which enables warping in time domain and integrates the edit distance with $L_p - norms$ and is denominated using a stiffness parameter, ν and a mismatch penalty, λ [37]. TWED between signals x and y can be calculated using Eq. 11.

$$g(x_i, y_j) = \min\{g(x_i, y_j) + \delta(x_i, y_j), g(x_{i-1}, y_j) + \delta(x_i), g(x_i, y_{j-1}) + \delta(y_j)\} \tag{11}$$

where, $i = \{1, 2, ..., N\}$, $j = \{1, 2, ..., M\}$, $g(x_0, y_0) = 0$, $\delta(x_i, y_j) = cost(x_i, y_j) + cost(x_{i-1}, y_{j-1}) + 2\nu.|i - j|$, $\delta(x_i) = cost(x_i, x_{i-1}) + \lambda + \nu$, $\delta(y_j) = cost(y_j, y_{j-1}) + \lambda + \nu$, $g(x_i, y_0) = g(x_0, y_j) = \infty$, and $cost(x_i, y_j) = |x_i - y_j|^2$. The dissimilarity value $g(x_N, y_M)$ between P_r and P_g is defined as the TWED metric. Twenty four possible constant combinations for $\lambda = [0, 0.2, 0.4, 0.6, 0.8, 1]$ and $\nu = [0.001, 0.01, 0.1, 1]$ were computed to obtain a series of TWED values. The minimum of the obtained values is considered as the final TWED value.

A lower score of FID, MMD, DTW, ED, KLD and a high value of PC depicts that the two distributions are similar to each other and have a high correlation.

V. Results and Discussion

We mainly discuss two sets of experiments which involve the use of two different types of noises, gaussian and uniform noise in addition to two different loss functions, binary cross entropy and least square loss [38]. Firstly, we took the data of all four classes namely N, SVEB, VEB and F. Secondly, we used the data of only three minority classes namely SVEB, VEB and F. The purpose of breaking down the experiments in these two broad categories was to observe the impact of majority (normal) class over the minority (SVEB, VEB and F) class during data generation. The hierarchy of the conducted experiments is provided in figure 3. We split the experiments in three levels: class (three or four), type of input noise (uniform or gaussian), and the loss function (least square or log loss).

In our initial experiment, the DCCGAN generated similar type of signals for a class with no diversity in the generated signals regardless of the latent input provided to the model. This problem was bound to happen as our model was susceptible to mode collapse, i.e., even though the discriminator was classifying the real and generated signals correctly, it was unable to differentiate between multiple generated signals. We alleviated this problem by performing batch wise training using a batch size of 256. During batch wise training the model was updated only at the end of each batch providing the model a combined error of multiple samples. Using this the GAN was now able to differentiate between multiple generated signals with respect to different latent noise input.

Initially we used three different loss functions that includes: (i) wasserstein loss [24], binary cross entropy and least square loss [38]. Out of these, the wasserstein loss did not converged even after 400 batches whereas the log loss and least square converged after 100 and 60 batches, respectively. Hence, we did not consider the wasserstein loss for the rest of this work.

Input to the generator was also changed to observe the effect of noise on generated signals. For this purpose, we used two types of noises: Gaussian (Normal) and uniform noise distribution. The normal noise distribution is represented as $\mathcal{N}(\mu, \sigma)$ where $\mu = 0$ and $\sigma = 1$ and the uniform noise distribution is represented as $\mathcal{U}(a, b)$, where $a = 0$ and $b = 1$.

The model with uniform noise as input to generator and binary cross entropy loss in the discriminator is described as Uni_BCE. Model with uniform noise and least square loss is described as Uni_MSE. Model with gaussian noise and binary cross entropy loss is described as Gaus_MSE. Model with gaussian noise and least square loss is described as Gaus_MSE.

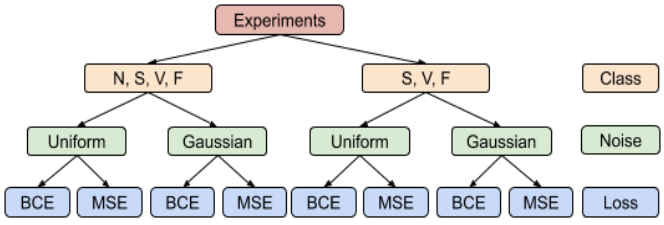


Fig. 3: Illustration of different Set of experiments performed.

The network size of both generator and discriminator was also increased by two layers of Conv($x * y$)-BN-ACT aka CBA($x * y$). Two CBA layers were appended after the second last activation layer in the discriminator with CBA(32*32)-CBA(32*64). Two CBA layers were appended after the concatenate layer in the generator with UP-CBA(32*32)-UP-CBA(32*64). The bigger models with uniform noise as input and binary cross entropy loss is represented as Big_BCE and with least square loss is described as Big_MSE. The training procedure of the original DCCGAN was also modified by allowing the generator to be trained twice for each batch. This training procedure with binary cross entropy loss is referred to as Gen2_BCE and the training procedure with least square loss is referred to as Gen2_MSE.

Due to the limited computational power available, it was not possible to calculate the evaluation metrics with the complete dataset after every iteration. Therefore, only 200 real and generated samples of each class were compared against each other after every five batches for model performance evaluation.

Data Generation for Four Classes

Table III depicts the average performance over all four classes (N, SVEB, VEB, and F) for all the models. Uni_BCE model outperforms all other models on all evaluation metrics except the metric DTW and TWED. Gaus_MSE performed better than Uni_BCE for DTW and TWED. The model size was increased to inspect whether the learning capacity of smaller models was sufficient to learn the real data distribution. But increasing the size of the model did not favour the data generation capability of our DCCGAN.

TABLE III: Evaluation Metrics for all Models for Four Classes

Model	FID	MMD	DTW	ED	PC	KLD	TWED
Uni_BCE	12.47	0.10	30.92	0.22	0.52	0.08	9.91
Uni_MSE	13.79	0.10	31.68	0.23	0.52	0.09	9.49
Gaus_BCE	22.35	0.16	38.22	0.29	0.39	0.13	7.52
Gaus_MSE	18.94	0.13	29.48	0.26	0.38	0.11	6.94
Big_BCE	25.74	0.18	42.70	0.31	0.10	0.14	6.98
Big_MSE	21.53	0.16	41.30	0.28	0.39	0.13	5.08
Gen2_BCE	24.60	0.18	42.60	0.30	0.16	0.14	7.66
Gen2_MSE	15.71	0.12	33.04	0.25	0.44	0.10	7.23

The models using MSE loss tend to perform better for all the metrics. Uni_BCE is an exception to this, as it performs better than its MSE counterpart. This might happen because discriminator only discriminates between the real and fake signal when BCE loss is employed. This means that BCE loss is only concerned with the labels (correct or incorrect) whereas MSE loss is concerned with how correct and incorrect the generated signals is as compared to the real signal. So MSE loss penalizes the large errors heavily that in turn results in a large update in generator weights as compared to BCE loss.

Big_BCE and Gen2_BCE models miserably failed to perform for any metric except the TWED. The reason behind this behaviour might be the more number of parameters in Big_BCE and overfitting in the case of Gen2_BCE.

As Uni_BCE model outperforms the other models in most of the evaluation metrics. We report the detailed class wise results for all metrics in Table IV. The results depict a very high pearson correlation of generated normal class (N) data as compared to the real data due to highly imbalanced class distribution in the dataset.

TABLE IV: Class wise Evaluation using Uni_BCE Model

C	FID	MMD	DTW	ED	PC	KLD	TWED
N	14.46	0.11	41.68	0.24	0.69	0.10	10.64
S	4.77	0.04	12.68	0.14	0.51	0.03	9.33
V	13.46	0.10	32.47	0.23	0.48	0.08	9.57
F	17.19	0.13	36.84	0.27	0.40	0.12	10.10

Figure 4 depicts the discriminator loss d_real, d_fake and gan curves for the Uni_BCE model when all the classes were used. Although we trained the models for more than 400 batches, but the optimum values for all the evaluation metrics were obtained at around 330^{th} batch. All three losses are showing a zigzag curve in the early phases of training before stabilizing for a short period of time at around batch 310 to 320. However, the losses again increased after that instance. All the losses were around 0.5 at batch 330 but the gan loss increased after that and d_loss stabilised after that. It is also worth mentioning that the nest metrics obtained were also generated during this batch itself.

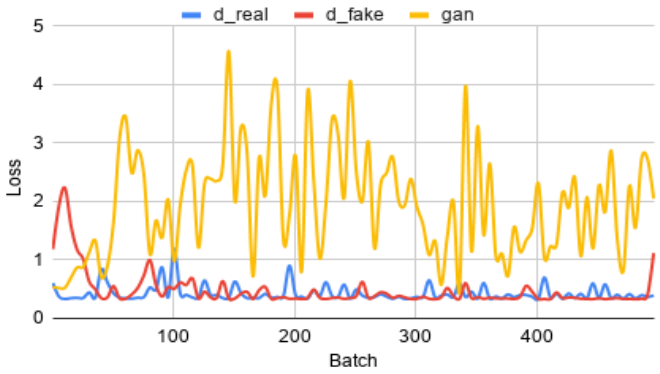


Fig. 4: Loss curves for all classes for Uni_BCE Model

Data Generation for Three Classes

Table V depicts the average performance of the models for three classes (SVEB, VEB, and F) on all the evaluation metrics. We can see that the model Uni_BCE, Gaus_MSE, and Gem2_MSE performed better among all other models for various evaluation metrics. The larger GANs were unable to perform at par as compared to their smaller counterparts due to limited data availability of minority classes.

The models using MSE loss tend to perform better for all the metrics except Uni_BCE with the same reason as before. We report the detailed class wise results for all metrics for

TABLE V: Evaluation Metrics for all Models on Three Classes

Model	FID	MMD	DTW	ED	PC	KLD	TWED
Uni_BCE	14.97	0.11	31.30	0.24	0.50	0.09	5.13
Uni_MSE	20.75	0.15	35.45	0.28	0.34	0.11	21.25
Gaus_BCE	21.47	0.16	38.13	0.28	0.04	0.12	17.28
Gaus_MSE	14.41	0.11	29.96	0.24	0.47	0.08	14.97
Big_BCE	21.51	0.16	40.17	0.28	0.08	0.12	10.87
Big_MSE	17.24	0.13	35.24	0.26	0.41	0.10	8.96
Gen2_BCE	20.85	0.15	36.93	0.28	0.22	0.11	15.90
Gen2_MSE	16.11	0.12	28.94	0.25	0.44	0.09	6.73

the classes SVEB, VEB, F using the model Gaus_MSE in Table VI. The results depict a very high PC of generated fusion beats (F) data as compared to the real data due to the data augmentation applied in II-A. A specific pattern can be observed among all the classes ,i.e., SVEB shows the best performance as compared to VEB and VEB better than F. This pattern arose because the data augmentation technique augmented the F beat data highest, then VEB data and SVEB the lowest.

TABLE VI: Class wise Evaluation using Gaus_MSE Model

C	FID	MMD	DTW	ED	PC	KLD	TWED
S	7.84	0.06	21.73	0.18	0.35	0.04	14.77
V	14.86	0.11	30.66	0.25	0.48	0.08	14.57
F	20.54	0.15	37.50	0.29	0.57	0.13	15.57

Figure 5 depicts the loss curves for the Gaus_MSE model when three classes were used. Although we trained the models for more than 400 batches, but the optimum values for all the evaluation metrics were obtained at around 71^{th} batch and after that the model performance deteriorated. An erratic loss curve can be observed for all batches except the range 60 to 100, at this range the GAN loss converged at provided us the best performance.

VI. Conclusions

We proposed a deep convolution conditional generative adversarial network model for generating normal as well as abnormal beats comprising of supraventricular ectopic beats, ventricular ectopic beats, and fusion beats. We experimented with different loss functions such as log loss, least square

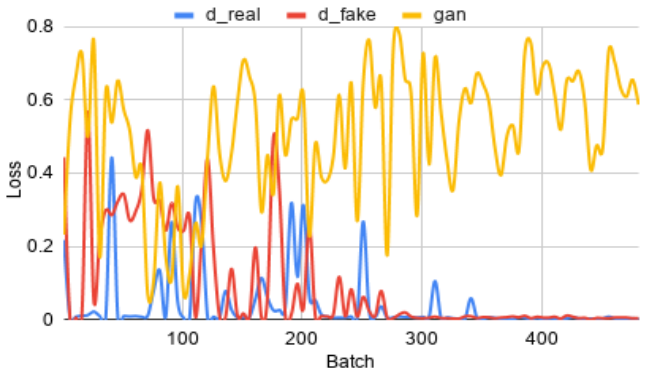


Fig. 5: Loss curves for 3 classes for Gaus_MSE Model

loss and wasserstein loss in the discriminator in addition to uniform and gaussian noise as an input to the generator. Soft labels were used for faster training of the model. The quality of generated beats was evaluated using seven quantitative indicators on the standard MIT-BIH benchmark database and found that our model effectively generates different classes of beats. Moreover, the loss curves were analysed to identify if the training was ideally successful. If the loss converges, the GAN succeeds in learning the desired data distribution during the training phase. As the loss curves exhibits an erratic behaviour during the training, further research is required to stabilise GANs during training.

ACKNOWLEDGMENT

The authors would like to thank Miss Pallabi Saikia for providing valuable suggestions and insights throughout the preparation of this manuscript.

REFERENCES

[1] S. Kiranyaz, T. Ince, and M. Gabbouj, "Personalized monitoring and advance warning system for cardiac arrhythmias," *Scientific reports*, vol. 7, no. 1, p. 9270, 2017.

[2] P. De Chazal and R. B. Reilly, "A patient-adapting heartbeat classifier using ecg morphology and heartbeat interval features," *IEEE transactions on biomedical engineering*, vol. 53, no. 12, pp. 2535–2543, 2006.

[3] C. Ye, B. V. Kumar, and M. T. Coimbra, "Heartbeat classification using morphological and dynamic features of ecg signals," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 10, pp. 2930–2941, 2012.

[4] M. A. Escalona-Morán, M. C. Soriano, I. Fischer, and C. R. Mirasso, "Electrocardiogram classification using reservoir computing with logistic regression," *IEEE Journal of Biomedical and health Informatics*, vol. 19, no. 3, pp. 892–898, 2014.

[5] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng, "Cardiologist-level arrhythmia detection with convolutional neural networks," *arXiv preprint arXiv:1707.01836*, 2017.

[6] P. E. McSharry, G. D. Clifford, L. Tarassenko, and L. A. Smith, "A dynamical model for generating synthetic electrocardiogram signals," *IEEE transactions on biomedical engineering*, vol. 50, no. 3, pp. 289–294, 2003.

[7] G. Clifford and P. McSharry, "Generating 24-hour ecg, bp and respiratory signals with realistic linear and nonlinear clinical characteristics using a nonlinear model," in *Computers in Cardiology, 2004*, pp. 709–712, IEEE, 2004.

[8] G. D. Clifford, S. Nemati, and R. Sameni, "An artificial vector model for generating abnormal electrocardiographic rhythms," *Physiological measurement*, vol. 31, no. 5, p. 595, 2010.

[9] H. Cao, H. Li, L. Stocco, and V. Leung, "Design and evaluation of a novel wireless three-pad ecg system for generating conventional 12-lead signals," in *Proceedings of the Fifth International Conference on Body Area Networks*, pp. 84–90, ACM, 2010.

[10] S. Haradal, H. Hayashi, and S. Uchida, "Biosignal data augmentation based on generative adversarial networks," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 368–371, IEEE, 2018.

[11] T. Golany and K. Radinsky, "Pgans: Personalized generative adversarial networks for ecg synthesis to improve patient-specific deep ecg classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 557–564, 2019.

[12] F. Zhu, F. Ye, Y. Fu, Q. Liu, and B. Shen, "Electrocardiogram generation with a bidirectional lstm-cnn generative adversarial network," *Scientific reports*, vol. 9, no. 1, p. 6734, 2019.

[13] B. Zhou, S. Liu, B. Hooi, X. Cheng, and J. Ye, "Beatgan: anomalous rhythm detection using adversarially generated time series," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 4433–4439, AAAI Press, 2019.

[14] A. M. Delaney, E. Brophy, and T. E. Ward, "Synthesis of realistic ecg using generative adversarial networks," *arXiv preprint arXiv:1909.09150*, 2019.

[15] E. Brophy, Z. Wang, and T. E. Ward, "Quick and easy time series generation with established image-based gans," *arXiv preprint arXiv:1902.05624*, 2019.

[16] P. Wang, B. Hou, S. Shao, and R. Yan, "Ecg arrhythmias detection using auxiliary classifier generative adversarial network and residual network," *IEEE Access*, vol. 7, pp. 100910–100922, 2019.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[18] J. T. Guibas, T. S. Virdi, and P. S. Li, "Synthetic medical images from dual generative adversarial networks," *arXiv preprint arXiv:1709.01872*, 2017.

[19] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4681–4690, 2017.

[20] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *arXiv preprint arXiv:1605.05396*, 2016.

[21] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[22] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," *arXiv preprint arXiv:1802.04208*, 2018.

[23] K. G. Hartmann, R. T. Schirrmeister, and T. Ball, "Eeg-gan: Generative adversarial networks for electroencephalograhic (eeg) brain signals," *arXiv preprint arXiv:1806.01875*, 2018.

[24] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, pp. 214–223, 2017.

[25] A. ECAR, "Recommended practice for testing and reporting performance results of ventricular arrhythmia detection algorithms," *Association for the Advancement of Medical Instrumentation*, p. 69, 1987.

[26] R. Mark, P. Schluter, G. Moody, P. Devlin, and D. Chernoff, "An annotated ecg database for evaluating arrhythmia detectors," vol. 29, no. 8, pp. 600–600, 1982.

[27] Z.-D. Zhao and Y.-Q. Chen, "A new method for removal of baseline wander and power line interference in ecg signals," pp. 4342–4347, 2006.

[28] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[29] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[32] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in neural information processing systems*, pp. 6626–6637, 2017.

[33] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, "A kernel method for the two-sample-problem," in *Advances in neural information processing systems*, pp. 513–520, 2007.

[34] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

[35] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[36] S. Kullback, *Information theory and statistics*. Courier Corporation, 1997.

[37] P.-F. Marteau, "Time warp edit distance with stiffness adjustment for time series matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 306–318, 2008.

[38] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794–2802, 2017.