Guided Generative Adversarial Neural Network for Representation Learning and Audio Generation Using Fewer Labelled Audio Data

Kazi Nazmul Haque, Rajib Rana, Jiajun Liu, John H. L. Hansen, *Fellow, IEEE*, Nicholas Cummins, Carlos Busso, *Senior Member, IEEE*, and Björn W. Schuller, *Fellow, IEEE*

***Abstract*—The Generation power of Generative Adversarial Neural Networks (GANs) has shown great promise to learn representations from unlabelled data while guided by a small amount of labelled data. We aim to utilise the generation power of GANs to learn Audio Representations. Most existing studies are, however, focused on images. Some studies use GANs for speech generation, but they are conditioned on text or acoustic features, limiting their use for other audio, such as instruments, and even for speech where transcripts are limited. This paper proposes a novel GAN-based model that we named Guided Generative Adversarial Neural Network (GGAN), which can learn powerful representations and generate good-quality samples using a small amount of labelled data as guidance. Experimental results based on a speech [Speech Command Dataset (S09)] and a non-speech [Musical Instrument Sound dataset (Nsyth)] dataset demonstrate that using only 5% of labelled data as guidance, GGAN learns significantly better representations than the state-of-the-art models.**

***Index Terms*—Audio Generation, Disentangled Representation Learning, Guided Representation Learning, and Generative Adversarial Neural Network.**

I. Introduction

THE generation power of a Generative Adversarial Neural Networks (GANs) [1] is useful for learning a meaningful representation [2]–[5] from an unlabelled dataset. However, the success of a GANs can mainly be found in image generation; it does not perform equally well for audio generation, as it requires to generate complex waveforms. Encouragingly, recent studies, such as for Parallel WaveGAN [6], and GAN-TTS [7] have shown success of GANs for audio generation. Most of these GAN models are, however, based on speech audio generation and conditioned on text or acoustic features, therefore, they can not be generalised for other audio domains, even for speech audios where transcripts are limited. Some studies such as WaveGAN [8], GANSynth [9], and TiFGAN [10] have shown intriguing results for text/acoustic feature independent audio generation using GAN based models. In these studies, researchers focused on strategies to generate low-dimensional acoustic features/representation, such as audio spectrograms, rather than generating raw waveform. The spectrograms are then converted back to audio. Most of these models are based on the conventional DCGAN architecture [11], which leaves room to explore high performing GAN based architecture such as BigGAN [4] and StyleGAN [3]. In this paper, we focus on the BigGAN architecture.

Like any GAN based model, BigGAN shows good performance for conditional generation using labelled data (categorical labels). These categorical labels add useful side information for BigGAN, which help it to generate correct samples. BigGAN requires an enormous amount of labelled data [12]. However, for audio, acquiring labels is very expensive and error-prone. Therefore, using a BigGAN for conditional (categorical) audio sample generation with fewer labelled data is a challenge.

This paper tackles this challenge by proposing the new model called Guided Generative Adversarial Neural Network (GGAN), which is based on the BigGAN architecture, but can generate conditional audio samples successfully using a small number of categorical labels.

The proposed GGAN offering not only to produce high-quality generation, but also to learn useful representations for two reasons: First, BigGAN, which is the core of GGAN, is useful for representation learning [5]. Second, Locatello *et al.* show that unsupervised representation learning is fundamentally impossible because many variational factors of the data distribution are dependent on the human perception/bias, which cannot be discovered from a dataset without any supervision

Manuscript received June 1, 2020; revised December 28, 2020, April 8, 2021, and June 20, 2021; accepted July 4, 2021. Date of publication July 21, 2021; date of current version August 13, 2021. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Yu Tsao. *(Corresponding author: Kazi Nazmul Haque.)*

Kazi Nazmul Haque and Rajib Rana are with the University of Southern Queensland, Darling Heights, QLD 4350, Australia (e-mail: shezan.huq@gmail.com; Rajib.Rana@usq.edu.au).

Jiajun Liu is with the Distributed Sensing Systems Group, Data61, CSIRO Australia, Pullenvale QLD 4069, Queensland, Australia (e-mail: liujiajun.nju@gmail.com).

John H. L. Hansen and Carlos Busso are with the University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: john.hansen@utdallas.edu; busso@utdallas.edu).

Nicholas Cummins is with the King's College London, WC2R 2LS London, U.K. (e-mail: nick.cummins@kcl.ac.uk).

Björn W. Schuller is with the GLAM – Group on Language, Audio, and Music, Imperial College London, London SW7 2BX, U.K., and also with the Chair of Embedded Intelligence for Health Care and Wellbeing, University of Augsburg, 86159 Augsburg, Germany (e-mail: schuller@tum.de).

This article has supplementary downloadable material available at https://doi.org/10.1109/TASLP.2021.3098764, provided by the authors.

Digital Object Identifier 10.1109/TASLP.2021.3098764

towards that bias [13]. This observation implies that learning a good representation using BigGAN should require some form of supervision. GGAN using guidance from labelled data coincide with this finding.

We evaluate the performance of conditional (categorical) audio sample generation and representation learning the quality of the GGAN model using the widely used Speech Command Dataset (S09) and the Musical Instrument Sound dataset (Nsyth). A comparison with the existing studies shows that the GGAN performs significantly better than the state-of-the-art (SOTA) [8], [10] methods.

II. Background and Related Work

In this section, we discuss related studies in audio generation and representation learning.

A. Audio Generation

Generating human-like speech audio with a neural network (NN) is an active area of research. Successful audio generation with NNs is hard, as it depends on generating coherent periodic waveforms maintaining the regularity in the periodicity, where the temporal resolution (sampling rate) and time scale (audio length) are high with short and long-term dependencies [9], [14]. Human hearing is highly sensitive to irregularities and discontinuities in the periodic nature of any audio [9], which makes it even harder for any NN to generate human-like speech audio. However, in recent years, researchers have achieved extraordinary success using neural networks for generating speech audio from text [15]. Most of these successes are dominated by the autoregressive models. Oord *et al.* [16] proposed "WaveNet" as a powerful autoregressive model for Text-To-Speech (TTS) synthesis for both English and Mandarin. Later, Oord *et al.* improved this work by proposing "Parallel WaveNet," which is 20 times faster [17]. This research proposed a new method for training a parallel feed-forward network from a trained WaveNet. Further improvement of WaveNet was proposed in the CLARINET [18], where they introduced a parallel wave generation method using Gaussian inverse autoregressive flow. Researchers have also used seq2seq-based models for acoustic modelling (text to acoustic features) such as TACOTRON (1 & 2) [19], [20] and Char2Wav [21]. Here, TACOTRON 1 adopts Griffin-Lim [22] for vocoding (acoustic features to waveform), TACOTRON 2 adopts WaveNet for vocoding, and Char2Wav adopts SampleRNN [23] for vocoding. Eunwoo *et al.* [24] proposed a Long-Short-Term-Memory (LSTM) based Recurrent Neural Network for TTS. Furthermore, many researchers have used other Neural Networks for TTS [25], [26]. These autoregressive models directly generate raw audios, which makes them expensive and slow.

As an alternative approach (non-autoregressive models), many researchers are currently focusing on generating low-dimensional acoustic features/representation, such as audio spectrograms, rather than generating a raw waveform. The generated spectrograms are then either converted back to audio using low computational methods such as IFFT/PGHI [8]–[10] or using a neural network [14]. Recently, GAN-based models are becoming popular to provide a non-autoregressive solution. A GAN generally consists of two neural networks, the Generator and the Discriminator, both trained using an alternating minimax-game optimisation process. During training, the Discriminator tries to distinguish between real samples from a data distribution and fake samples generated from the Generator, while the Generator tries to fool the Discriminator by producing samples closer to the real sample [1]. Parallel WaveGAN [6], generates high-quality speech using a GAN-based technique where the Generator uses a WaveNet-like architecture. Kumar *et al.* [14] have proposed the MelGAN framework, which generates waveform samples using mel-spectrograms. Moreover, other studies propose GAN-based solutions for speech generation such as GAN-TTS [7]. In the GAN-TTS paper, the authors propose a text-conditional feed-forward Generator using multiple discriminators which evaluate the generated audio based on multi-frequency random windows. Other studies have utilised GANs for spectral conversion in speech data [27], [28]. The above GAN-based Parallel WaveGAN, GAN-TTS, and MelGAN are conditioned on text or acoustic features and mainly focus on speech generation. Therefore, these models cannot be generalised directly to other audio domains. They cannot even be used for speech generation, where text is not available.

For generating audio/speech without text/acoustic features as input, researchers are also using GAN-based solutions. In the SpecGAN work, the authors generate spectrograms and then convert them back to audio [8] using the Griffin-Lim algorithm [22]. Marafioti *et al.* [10] in their TiFGAN paper showed a significant improvement using the Phase Gradient Heap Integration (PGHI) [29] algorithm for converting spectrograms to audio. As the PGHI algorithm can reconstruct audio from a spectrogram with minimal loss in the perceptual quality [10], generating real-like spectrograms should result in a high-quality audio generation. The high performing GAN architectures, such as Style GAN [3], or BigGAN [4] usually work fine for spectrogram generation. The BigGAN model has been successfully explored in the field of image datasets. However, it is an open research question to explore its benefits in the audio domain, which we address in this paper.

B. Audio Representation Learning

In any high dimensional data distribution, the variational factors of the data are entangled and cannot be easily identified. Representation learning aims to disentangle these variational factors by mapping the high dimensional data to a low dimensional latent/representation space [13]. For any particular domain, learning representation from vastly available unlabelled data can improve the post-use case scenario where the availability of the labelled data is limited. Researchers are currently using deep neural networks to learn successful representation from any labelled or unlabelled audio dataset [30]–[36].

Among the unsupervised representation learning techniques, self-supervised learning is dominant in the field of computer vision [37]–[40]. In self-supervised learning, an alternative supervised signal is created from the information that exists in the unlabelled data to train the model for representation learning.

An example is predicting the rotation angle of images where the rotation angle is the supervised signal for training the model [41]. Similarly, audio researchers have achieved successful results using self-supervised representation learning. Van den Oord *et al.* [42] proposed a model for learning representation predicting future latent observations. In another study, De Chaumont Quitry *et al.* [43] have learnt representation by predicting the instantaneous frequency. Likewise, other researchers have achieved improvement in performance using different self-supervised techniques [43]–[48].

For self-supervised learning, we have to manually design the supervised signal, which is a major drawback [49]. This leads the researchers to focus on fully unsupervised representation learning techniques. Most of the unsupervised representation learning studies use autoencoders [50], [51]. Several researchers have utilised Variational Autoencoders (VAEs) [52] to learn a useful representation from an unlabelled speech dataset. Recent literature has used GANs for learning meaningful representations in an unsupervised manner [53], [54].

Unsupervised representation learning looks intriguing as it can utilise an enormous amount of unlabelled data. However, recent work of Locatello *et al.* have shown that entirely unsupervised representation learning is not possible without any short of supervise signal [13]. Furthermore, learning representation in an unsupervised manner does not guarantee its usability in the post-use case scenario. Therefore, we propose the Guided Generative Adversarial Neural Network (GGAN), which can learn an useful representation from an unlabelled audio dataset according to the categorical supervision given from a limited amount of labelled data.

C. Related Guided Generative Adversarial Neural Network Architectures

In Spurr *et al.* [55] proposed a method to guide an InfoGAN network. They use a small number of labels to help the InfoGAN to capture a specific representation. However, it fails to perform well for complex datasets such as SVHN [56], CelebA [57], and CIFAR-10 [58]. Springenberg *et al.* [59] suggested learning a classifier from a partially labelled dataset and then used that classifier for semi-supervision in a GAN architecture. They empirically evaluated their method on synthetic data as well as on challenging image classification tasks. Kumar [60] proposed two discriminators in a GAN architecture where one discriminator learns to identify real or fake samples from the unlabelled dataset, and another discriminator learns to identify real or fake samples with their labels from some labelled dataset. A recent study from Lucic *et al.* [12] explored different semi-supervised methods for images. They predict the missing labels of the dataset with the help of a small labelled dataset. First, they train with self-supervision and then fine-tune the classifier with a small labelled dataset. Subsequently, they predict the labels for other missing labelled datasets. They also propose a co-training method for this task, where they train this classifier on top of the discriminator during the training.

The review of the related guided GANs shows that most of the studies are focused on images generation. Also, representation learning has received a small focus from yhe community. We propose the GGAN, capable of learning a meaningful representation from an unlabelled audio dataset with some guidance from a minimal amount of labelled samples. The newly introduced GGAN can also generate higher quality audio samples than the state-of-the-art [8], [10] models given the small guidance.

III. Architecture of GGAN

A. Overview of the GGAN

In our proposed model, we have eight networks. These are the Encoder E, Generator G, Feature Extractor F, two Classifiers C_e and C_x, and three Discriminators D_1, D_2, and D_f. Fig. 1 shows the essential connections between different parts of the model.

The Generator of any unsupervised GAN usually takes a random latent space as input and generates samples from the real data distribution. It learns to map any latent space to the data distribution. In the data distribution, the factors of variation are entangled and not easily separable, but in the latent space, these categories are disentangled, and can be easily separated. The BiGAN [61] framework uses an extra network to learn the reverse mapping from the data distribution to the latent space. In our GGAN model, rather than feeding a random latent space to the Generator, we feed it with a generated latent space $u(z)$, which is easily classifiable into n categories. Our aim is that the Generator learns to map different latent space categories to different data categories, according to our guidance. However, the challenge is to force the Generator to create such different samples for different categories of latent space. To guide the Generator, we use a small percentage of labelled samples from the training datasets. This is how we achieve semi-supervised learning.

The first task is to generate a latent space $u(z)$, which can be divided into our desired n categories. In the GGAN model, the encoder E learns to generate a new latent space $u(z)$ from any known distribution $p(z)$ and categorical distribution $Cat(c, k = n)$, where C_e can classify $u(z)$ into n categories.

The second task is to force the Generator G to generate different categories of data samples from different latent space categories. We utilise the classifier C_x to guide the Generator G to generate accurate samples for different latent space categories. Note that we only have a small percentage of labelled samples from the training data, which is not enough to successfully train the Classifier C_x. To address this problem, we use these generated samples and the labelled samples from different categories to train the Classifier C_x. In the beginning, G produces incorrect samples, as C_x learns to classify only some samples correctly based on the used labelled samples. Then, C_x forces the Generator G to generate those samples correctly. Gradually, the Generator G learns to generate new samples with new characteristics, matching some characteristics from those correct categories. These news samples with correct categories improve the learning of C_x, which recursively improves G.

When our Generator G can successfully map n categories of latent samples $z_e \in u(z)$ to the n categories of data samples, the latent space becomes a useful representation of the data

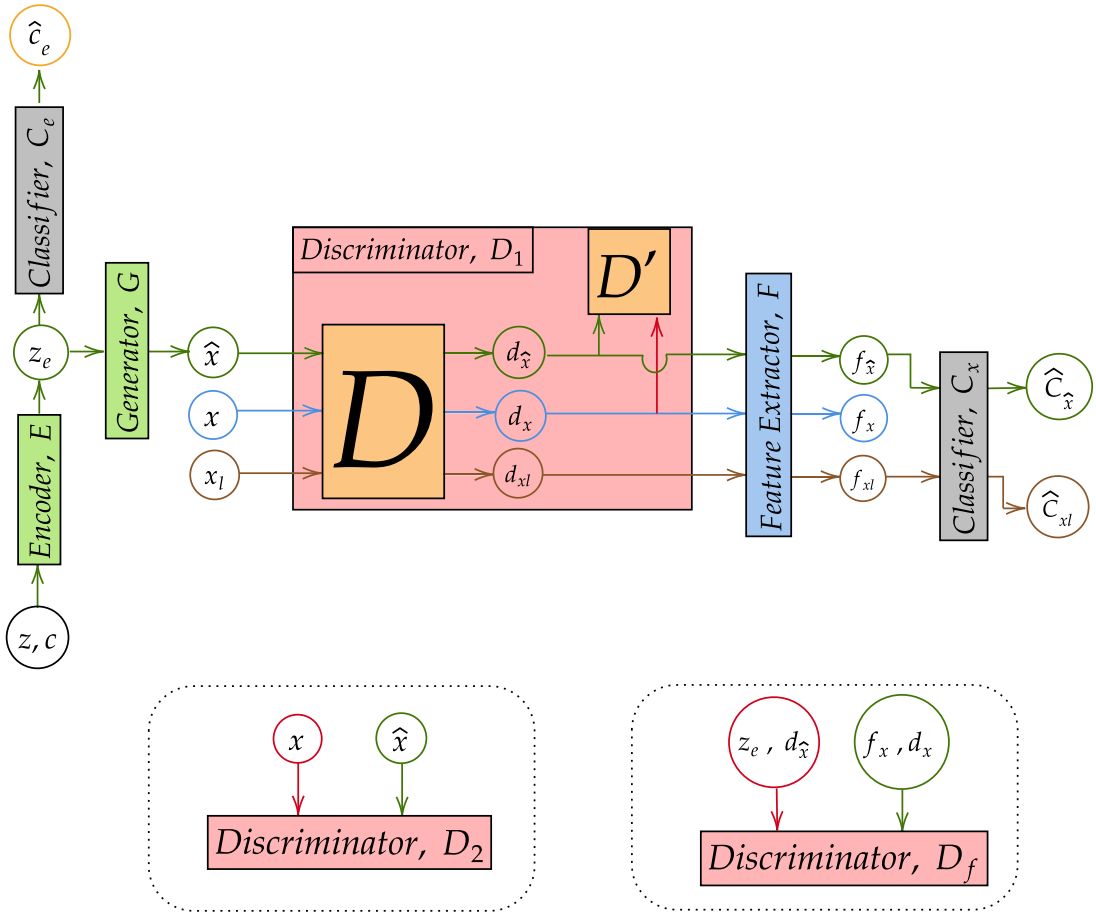


Fig. 1. The architecture of the proposed GGAN model. The model consists of the Encoder E, Generator G, Feature Extractor F, two Classifiers C_e and C_x, three Discriminators D_1, D_2, and D_f Networks. Here, z is the random sample from a continuous distribution, c is the random sample from a categorical distribution, $\hat{x}$ is the generated sample, x is the sample from real train data distribution, x_l is the labelled sample from the real train data distribution. Different colour of the arrow shows the direction from the input to output. For the Discriminator D_1, D_2, and D_f, the red colour indicates the real samples, where the green colour indicates fake/generated samples.

distribution. We then connect a Feature Extractor network F, which learns to map real data samples to the latent space $u(z)$. The output of the feature extractor then essentially becomes our "learnt representation."

B. Detailed Architecture of the GGAN

1) Encoder and Classifier (C_e): The Encoder, E, learns to map any sample z and c to $z_e \in u(z)$, where $u(z)$ is any continuous distribution generated by E, $z \in p(z)$, and $c \in Cat(c, k = n)$; p_z is a random continuous distribution, e. g., a continuous uniform distribution and $Cat(c, k = n)$ is a random categorical distribution with n number of categories.

When E learns to map z and c to z_e, it can easily ignore the categorical distribution. To address this problem, we introduce a classifier network C_e, which takes z_e as input and outputs the predicted class $\hat{c_e}$, where the true label is the given categorical sample c. By using the network C_e, we force E to maximise the mutual information between c and z_e. Therefore, E learns to create a sample space $u(z)$, which can be classified into n categories by the network C_e.

2) Generator: Like any other GAN framework, we have a Generator G, which learns to map z_e into sample $\hat{x} \in P_G$, where P_G is the generated sample distribution. One of the major goals of G is to generate P_G so that it matches the true data distribution P_{data}. Another goal of G is to maximise the mutual information between $\hat{x}$ and the given random condition c (random categorical sample) ensuring that there are categorical variations in the generated samples. This formulation ensures that G generates $\hat{x}$ according to the input condition c.

3) First (D_1) and Second (D_2) Discriminators: The Discriminator plays primary roles in the GGAN model. Like all GAN models, it forces the Generator to generate samples from the training distribution. In the GGAN model, we have two sample Discriminator D_1 and D_2. The Discriminator D_1 has two parts: a feature extraction part D, and the real/fake sample identification part D'. From D, we obtain the features $d_{\hat{x}}$, d_x, and d_{x_l} for $\hat{x}$, x and x_l, sampling from the data distribution respectively, where a real data sample is $x \in P_{data}$ and a labelled data sample $x_l \in P_{ldata}$; P_{ldata} is the labelled data distribution. Here, P_{ldata} can be the subset of the P_{data} or any other data distribution. For a real sample x and a fake sample $\hat{x}$, we obtain the output d'_x and $d'_{\hat{x}}$ from D' respectively. In our D_1 Network, D optimises the feature learning for both the classification and the discrimination tasks, making the optimisation task considerably

more complex. So, D_1 cannot focus only on discrimination. Therefore, we have added another Discriminator D_2 to focus solely on the sample discrimination task.

4) Feature Extractor, Classifier (C_x), and Third Discriminator (D_f): The feature extractor network F learns the representation $f_{\hat{x}}$, f_x, and f_{xl} for $\hat{x}$, x, and xl, respectively. To map $\hat{x}$, x and xl to the feature space, we need a network similar to D. So, rather than introducing another network, we use D to get a lower dimensional representation ($d_{\hat{x}}$, d_x and d_{x_l}) of the samples. Then, F is trained to map $d_{\hat{x}}$, d_x and d_{x_l} to the latent space $u(z)$. As we do not use another network similar to D, it reduces the computation as well as it helps F from overfitting as the extracted representation/features from D is constantly changing during the task of discriminating real training samples and generated/fake samples.

To ensure that F can generate f_x from the known distribution $u(z)$, we have another discriminator D_f following the implementation from the "Bidirectional Generative Adversarial Networks" framework [61]. For any GAN based architecture, the generated samples are considered as fake samples and the training samples as real samples. Similarly, as z_e is a sample from $u(z)$ and $\hat{x}$ is a generated sample from z_e, D_f learns this $(z_e, \hat{x})$ pair as a real sample. We want F to map real sample x to f_x according to the relationship presents between z_e and $\hat{x}$. Therefore, the (f_x, x) pair is considered as fake sample for the discriminator D_f.

The second Classifier C_x learns to classify labelled data as well, as it learns to classify the generated sample $\hat{x}$ according to a given categorical, conditional class sample c. For the classification of $\hat{x}$ and xl, we feed the features generated from F to C_x.

C. Losses

1) Encoder, Classifier (C_e), and Generator Loss: For the Encoder E and the Classifier C_e, the classification loss is EC_{loss}. We have $z_e = E(z, c)$, where z is sampled from $p(z)$ and c is sampled from $Cat(c, k = n)$. Therefore,

$$EC_{loss} = -\sum c \log(C_e(z_e)). \tag{1}$$

For the Generator G, we have two generation losses coming from the discriminators D_1 and D_2. Our GGAN model is inspired by the BigGAN architecture [4], which uses hinge loss. We therefore use the hinge loss for both of our Generator and Discriminator. We have $\hat{x} = G(z_e)$ and $d_{\hat{x}} = D(\hat{x})$. Therefore,

$$G_{loss1} = -D'(d_{\hat{x}}), \tag{2}$$

$$G_{loss2} = -D_2(\hat{x}). \tag{3}$$

The Generator G has another loss for the classification of the samples. We have $f_{\hat{x}} = F(d_{\hat{x}})$, so the Classification Loss, GC_{loss} for G is,

$$GC_{loss} = -\sum c \log(C_x(f_{\hat{x}})). \tag{4}$$

Mode Divergence Loss. Similar to other GAN architectures, GGAN faces the challenge of mode collapse [62], where the Generator only produces samples from fewer modes (variational factor) of the data distribution. Moreover, the latent input variable has a minor impact on the generated samples from the Generator. To address mode collapse, we design a loss named "Mode Divergence Loss," MG_{loss}, which is inspired by the study of Mao *et al.* [62]. Our, Mode Divergence Loss forces the Generator to generate samples from different modes by penalising the Generator for generating similar samples. For calculating this loss, we take two random inputs z_1, z_2 sampled from $p(z)$, and the same conditional code c sampled from $Cat(c, k = n)$. For z_1, z_2, we get $\hat{x_1} = G(E(z_1, c))$ and $\hat{x_2} = G(E(z_2, c))$, respectively. We also take two random samples x_1 and x_2 from the real data distribution p_{data}, where $x_1 \neq x_2$. We calculate the loss based on the feature extracted from D. Let $d_{x1} = D(x_1)$, $d_{x_2} = D(x2)$, $d_{\hat{x1}} = D(\hat{x_1})$, $d_{\hat{x2}} = D(\hat{x_2})$, and α has a small value such as 0.0001. So we get,

$$MG_{loss} = \max\{1, \textstyle\sum (|d_{x1} - d_{x2}|) / (\sum (|d_{\hat{x1}} - d_{\hat{x2}}|) + \alpha)\}. \tag{5}$$

Here, d_{x1}, d_{x2}, $d_{\hat{x1}}$ and $d_{\hat{x2}}$ are the extracted features from the D network. To identify the real and fake samples, D' is trained on the representation from the D network. Therefore, D has to learn the key attributes of the data distribution. Now, $|d_{x1} - d_{x2}|$ calculates the differences between the attributes/characteristics of the real samples in the representation space. Similarly, $|d_{\hat{x1}} - d_{\hat{x2}}|$ calculates the difference for the generated samples. Therefore, $\sum (|d_{x1} - d_{x2}|) / (\sum (|d_{\hat{x1}} - d_{\hat{x2}}|)$ calculates the ratio between the difference of the attributes for real and generated samples. The target of the Generator G is to minimise this ratio, which implies that G needs to maximise the term $|d_{\hat{x1}} - d_{\hat{x2}}|$ as it does not have any control over $|d_{x1} - d_{x2}|$. Now, to maximise $|d_{\hat{x1}} - d_{\hat{x2}}|$, G has to create two samples $(\hat{x_1}, \hat{x_2})$ with different attributes for different samples (z_1, z_2) from the $p(z)$ distribution. Since the Encoder E is also trained to minimise the MG_{loss} loss, it is also forced to create two different samples z_{e1}, z_{e2} for two different samples z_1 and z_2. Otherwise, G can not generate two different samples from z_{e1}, z_{e2} where $z_{e1} = z_{e2}$.

Hence, we have a combined loss, ECG_{loss} for E, C, and G. We average the G_{loss1}, G_{loss2}, and MG_{loss} as all of these are losses for the generation of the sample. The E, C, and G networks are updated to minimise the loss ECG_{loss}.

$$ECG_{loss} = \alpha((G_{loss1} + G_{loss2} + MG_{loss})/3) + \beta(EC_{loss}) + \gamma(GC_{loss}). \tag{6}$$

Here, α, β and γ are the regularising hyperparameters.

2) Feature Extractor and Classifier (C_x) Loss: We have the feature generation loss, FG_{loss}, coming from the third Discriminator D_f, enforcing F to create features like z_e from real data.

$$FG_{loss} = -D_f(f_x, d_x), \tag{7}$$

where, $d_{\hat{x}} = D(\hat{x})$ and x is a sample from the real data distribution P_{data}.

We have two classification losses for C_x, one for the labelled sample, Cl_{loss}, and another one for the generated sample, Cg_{loss}. The label of the real sample is y. Now, we have $f_{xl} = F(d_{xl})$, where xl is sampled from the labelled data distribution

and $f_{\hat{x}} = F(d_{\hat{x}})$. Therefore,

$$Cl_{loss} = -\sum y \log(C_x(f_{xl})), \tag{8}$$

$$Cg_{loss} = -\sum c \log(C_x(f_{\hat{x}})). \tag{9}$$

Likewise, the total loss for F and C_x is FC_{loss}, which is the sum of the above losses:

$$FC_{loss} = Cl_{loss} + Cg_{loss} + FG_{loss}. \tag{10}$$

We update the F and C_x network to minimise the FC_{loss}.

3) Discriminators Loss: The D' part of D_1, and D_2 are two discriminators for identifying the real/fake samples generated from the generator. D_1 also has a part D, which is responsible for generating features for the F and C_x networks. It is also optimised to reduce the classification loss, Cl_{loss}, of the real labelled samples. Finally, the Discriminator D_f identifies the real or fake feature sample pairs from F. The discriminator losses D_{1loss}, D_{2loss}, and D_{floss} for D_1, D_2, and D_f, respectively, are given by,

$$\begin{aligned} D_{1loss} = &-min(0, -1 + D'(D(x))) \\ &-min(0, -1 - D'(D(\hat{x}))) - Cl_{loss}, \end{aligned} \tag{11}$$

$$\begin{aligned} D_{2loss} = &-min(0, -1 + D_2(x)) \\ &-min(0, -1 - D_2(\hat{x})), \end{aligned} \tag{12}$$

$$\begin{aligned} D_{floss} = &-min(0, -1 - D_f(f_x, d_x)) \\ &-min(0, -1 + D_f(z_e, d_{\hat{x}}))), \end{aligned} \tag{13}$$

where, x is a sample from the real data distribution P_{data}. Here, the discriminators' weights are updated to maximise these losses. The algorithm to train the whole model is given in Algorithm 1.

We present the architectural difference between GGAN and other related models in Table I. The GGAN model can be considered as an extension of the InfoGAN model [63]. We have added an extra Discriminator for latent space, a Classifier for latent space, and a Feature extractor for the real sample. These networks are added to facilitate the accurate conditional sample generation and guided representation learning using fewer labelled data. Comparison of the primary tasks of GGAN with other models are summarised in Table II.

Algorithm 1: Minibatch stochastic gradient descent training of the proposed GGAN. The hyperparameter k represents the number of times the discriminators are updated in one iteration. We use $k = 2$, which helped to converge faster.

1: **for** number of training iterations **do**
2: **for** k steps **do**
3: Sample minibatch of m, noise samples $\{\boldsymbol{z^{(1)}}, \ldots, \boldsymbol{z^{(2\,m)}}\}$ from p_z, conditions $\{\boldsymbol{c^{(1)}}, \ldots, \boldsymbol{c^{(m)}}\}$ from Cat(c), data points $\{\boldsymbol{x^{(1)}}, \ldots, \boldsymbol{x^{(2\,m)}}\}$ from p_{data} and labelled data points $\{\boldsymbol{xl^{(1)}}, \ldots, \boldsymbol{xl^{(m)}}\}$ from P_{ldata}.
4: Update the parts D, D' of discriminator D_1 by ascending its stochastic gradient:

$$\nabla_{\theta_d, \theta_{d'}} \frac{1}{m} \sum_{i=1}^{m} \left[\boldsymbol{D_{1loss}}^{(i)} \right].$$

5: Update the discriminator D_2 by ascending its stochastic gradient:

$$\nabla_{\theta_{d_2}} \frac{1}{m} \sum_{i=1}^{m} \left[\boldsymbol{D_{2loss}}^{(i)} \right].$$

6: Update the discriminator D_f by ascending its stochastic gradient:

$$\nabla_{\theta_{d_f}} \frac{1}{m} \sum_{i=1}^{m} \left[\boldsymbol{D_{floss}}^{(i)} \right].$$

7: **end for**
8: Repeat step [3].
9: Update the Generator G, Encoder E, and Classifier C_e by descending its stochastic gradient:

$$\nabla_{\theta_g, \theta_e, \theta_{c_e}} \frac{1}{m} \sum_{i=1}^{m} \left[\boldsymbol{ECG_{loss}}^{(i)} \right].$$

10: Repeat step [3].
11: Update the Feature Extractor F and Classifier C_x by descending its stochastic gradient:

$$\nabla_{\theta_f, \theta_{c_x}} \frac{1}{m} \sum_{i=1}^{m} \left[\boldsymbol{FC_{loss}}^{(i)} \right].$$

12: **end for**

IV. Data and Implementation Details

A. Datasets

We have validated GGAN based on two different scenarios: Speech audio and non-speech music audio tasks. For the speech audio dataset, we use the S09 [64] corpus and the Librispeech dataset [65]. For the non-speech music audio dataset, we use the popular Nsynth dataset [66].

In the S09 dataset, digits from zero to nine are uttered by 2618 speakers [64]. This dataset is noisy and includes 23 000 samples, where many of these samples are poorly labelled. Labels for the speakers and gender are not available in the S09 corpus. LibriSpeech is a corpus of approximately 1000 hours of 16 kHz read English speech. In the dataset, there are 1166 speakers, where 564 are female, and 602 are male [65].

The Nsynth dataset consists of 305 979 musical notes with four seconds duration from ten different instruments including acoustic, electronic, and synthetic ones [66]. For our experiment, we use three acoustic sources: Guitar, Strings, and Mallet from the Nsynth dataset.

B. Measurement Metrics

To evaluate the generation results, we use the Inception Score (IS) [67], the Fréchet Inception Distance (FID) [68], [69], Classification Accuracy [70] and Mean Opinion Score (MOS) [71].

TABLE I
DIFFERENCE BETWEEN THE CONSTITUENT NETWORKS OF THE GGAN MODEL AND THE OTHER RELATED MODELS FROM THE LITERATURE

Model	Sample Generator	Discriminator (Data Sample)	Discriminator (Latent Sample)	Feature Extractor	Classifier (Data Sample)	Classifier (Latent Sample)
Supervised GAN [1]	Y	Y	N	N	N	N
Unsupervised GAN [1]	Y	Y	N	N	N	N
InfoGAN [63]	Y	Y	N	N	Y	N
TiFGAN [10]	Y	Y	N	N	N	N
BiGAN [61]	Y	Y	N	Y	N	N
GGAN	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**

TABLE II
COMPARISON BETWEEN THE PRIMARY TASKS OF THE GGAN MODEL AND THE OTHER MODELS FROM THE LITERATURE

Tasks	Supervised GAN	Unsupervised GAN	InfoGAN	TiFGAN/ SpecGAN/ WaveGAN	BiGAN	**GGAN**
Conditional Sample Generation	Y	N	Y	Y	N	**Y**
Latent Space Generation	N	N	N	N	N	**Y**
Requires Labelled Data	Y	N	N	Y/N	N	**Y**
Guided Representation Learning	N	N	N	N	N	**Y**
Generalised Representation Learning	N	N	N	N	Y	**N**

1) Inception Score (IS): The IS score measures the quality of the generated samples as well as the diversity in the generated sample distribution. A pretrained Inception Network V3 [72] is used to get the labels for the generated samples. The conditional label distribution $p(y|\hat{x})$ is derived from the inception network, where $\hat{x}$ is the generated sample. We want the entropy to be low, which indicates a good quality of the sample. It is also expected that the samples are diverse, so the marginal label distribution $\int p(y|\hat{x})dz$ should have a high entropy. Combining these two requirements, the KL-divergence between the conditional label distribution and the marginal label distribution is computed as the IS score: $\exp(\mathbb{E}_{\hat{x}} KL(p(y|\hat{x})||p(y)))$. A 'higher' IS score indicates that the generated samples have good quality.

2) Fréchet Inception Distance (FID): The IS score is computed solely on the generated samples. The Fréchet Inception Distance (FID) improves the IS score by comparing the statistics of generated samples to real data samples. First, the features are extracted for both real and generated samples from the intermediate layer of the inception network. Then, the mean μ_r, μ_g and covariance Σ_r, Σ_g for real and generated samples are calculated respectively from those features. Finally, the Fréchet Distance [73] between two multivariate Gaussian distributions (given by the μ_r, μ_g and Σ_r, Σ_g) is calculated using: $||\mu_r - \mu_g||^2 + \mathrm{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$. A 'lower' FID score indicates a good quality of the generated samples.

Note that the Inception V3 model is trained on the ImageNet dataset [74], which is completely different from audio spectrograms. Therefore, it will not be able to classify spectrograms into any meaningful categories, resulting in poor performance on the calculation of the IS and FID scores for our datasets. In the "Adversarial Audio Synthesis" [8] paper proposed by Donahue *et al.* instead of using the pretrained Inception V3 network to calculate the IS and FID scores, the authors trained a classifier network on the S09 spoken digit dataset and obtain good performance. We, therefore, use their pretrained model to calculate both the IS and FID scores for our generated samples. For the Nsynth dataset, there is no pre-trained classifier available. Therefore, we train a simple Convolutional Neural Network (CNN) on the training dataset.

3) Classification Accuracy: To evaluate the accuracy of the conditional sample generation, we use the approach similar to the study of Shmelkov *et al.* [70]. We train two CNN networks; the first one is trained with all the labelled training data, and the second one is trained with the generated samples from the GGAN model for different random conditions/categories. For training the second CNN network, the random conditions/categories are used as true labels. Now, the classification accuracy is calculated based on the test dataset for both of the CNN models. The classification accuracy of the second classifier is compared with the first classifier. Here, the second CNN model will perform better when the GGAN model can generate correct and diverse samples according to the given conditions/categories. This classification accuracy metric can evaluate both conditional sample generation as well as the sample diversity.

4) Mean Opinion Score (MOS): For subjective evaluation, we had 20 student volunteers (10 Males and 10 Females). The students were in their undergraduate studies and their medium of study was English. For the S09 and Nsynth datasets, we collected ten real and ten generated samples for each model. The students marked the audio samples in the range from 1 to 5 presented in Table III. Then, the scores were averaged for each of the datasets. We informed the students that some recordings are real, and others are generated using a computer program. Finally, for different models, we averaged the scores for each of the datasets. This average value is called as Mean Opinion Score (MOS) [71].

C. Data Preprocessing

We conduct experiments based on one-second audio segments with a sampling rate of 16 kHz. For the Nsynth dataset, we take the first one second from any audio clip. After investigating

TABLE III
MEANING OF THE DIFFERENT SCALES IN THE MEAN OPINION SCORE

Rating	Quality	Distortion
5	Excellent	Imperceptible
4	Good	Just perceptible, but not annoying
3	Fair	Perceptible and slightly annoying
2	Poor	Annoying, but not objectionable
1	Bad	Very annoying and objectionable

the Nsynth sample, we found out that the first second of the audio represents most of the sound from any instrument. For the Librispeech dataset, a one-second audio segment is sampled randomly from any audio clip. We convert the audio data to log-magnitude spectrograms with the short-time Fourier Transform. The generated log-magnitude spectrograms are converted to audio through the PGHI algorithm [29]. We followed the exact method from TifGAN [10] to calculate the log-magnitude spectrograms. We calculate the short-time Fourier Transform with overlapping Hamming window of size 512 ms, and a hopping length of 128 ms, which gives us a spectrogram of size 256×128 (matrix). The spectrogram is standardised with the equation $\frac{X-\mu}{\sigma}$, where X is the spectrogram, μ is the mean of the spectrogram, and σ is the standard deviation of the spectrogram. We limit the dynamic range of the log-spectrogram by clipping the values at $-r$. For the S09 and Librispeech datasets, we found 10 as a suitable value of r, and for the Nsynth dataset, we found 15 as suitable value. After the clipping, the values of the spectrograms are normalised between 1 and -1. This spectrogram representation of audio is used to train the Discriminators, the Feature Extractor, and the Classifier. The Generator generates the spectrogram with the values between 1 and -1. Then, the above steps are reversed before applying the PGHI algorithm.

V. EXPERIMENTAL SETUP, RESULTS, AND DISCUSSION

For our Generator and Discriminator network, we use the BigGAN [4] architecture for its unprecedented success. We maintain the same parameters as BigGAN, except we only change the input size of the Discriminator and output size of the Generator to accommodate the 128×256 size log-magnitude spectrogram. During the training of GGAN, we keep the learning rate of the generator and Discriminator equal. The architecture details are given in the supplementary document.

A. Impact of Labelled Data on Conditional Audio Generation

1) Setup: The conditional sample generation quality of the GGAN model for different percentage of labelled data (1% - 5%, 50%, 75%, 100%) as guidance is measured with the IS and FID score. The IS and FID scores are calculated based on 50 000 generated samples from the GGAN model [67]. To calculate the IS and FID scores, we first convert the generated log-magnitude spectrograms of the models to audio using the PGHI algorithm. We then convert the audio to spectrogram to pass through the pretrained classifier to calculate the IS and FID scores, as the input spectrogram format for the pretrained model is different. We follow this process for both the S09 and the Nsynth dataset.

For a given percentage of data, we train the GGAN model three times. Each training takes approximately 36 000 iterations with mixed-precision [75] for a batch size of 128. For training, we randomly sample the datasets, and repeat training only for three times. A single training run time on two Nvidia p100 GPUs is approximately 19 hours, making the total training time approximately $17 \times 3 \times 8$ (1-5%, 50%, 75%, 100% data) $\times$ 2 (two datasets) $= 816$ hours or 34 days.

2) Results and Discussions: Fig. 2 shows the impact of the percentage of labelled training data used as guidance and the IS and FID score of the GGAN model. For both datasets, we notice that the performance of the GGAN model improves along with the increase of the percentage of the data. We also note that with 5% of labelled data as guidance, GGAN can achieve IS and FID scores close to the score we achieved with 100% labelled data. Therefore, we compare the score of GGAN achieved with 5% of the labelled data with the results from other models.

B. Conditional Audio Generation Based on Guidance From the Same Dataset

1) Setup: In this section, we evaluate the quality of the generated samples with the FID, IS, Classification Accuracy, and MOS Score. Moreover, Classification Accuracy and a manual hearing test of the authors are used to evaluate the correctness of the conditional audio generation according to the given guidance from 5% labelled samples from the same dataset. For different categories, we generate audio samples and check manually to verify the quality of the conditional generation. For large-scale validation, we use classification accuracy from the CNN models trained on the generated audio samples.

We train the Supervised BigGAN, and an Unsupervised BigGAN [4] model from scratch for both of the datasets for comparison. Moreover, we compare our GGAN model with other existing models from the literature.

2) Results and Discussions: Test for the Quality of the Generated Samples: For the S09 dataset, using an unsupervised BigGAN model, we achieve an IS score of 6.17 ± 0.2, and an FID score of 24.72 ± 0.05, whereas we receive an IS score of 7.3 ± 0.01, and FID score of 24.40 ± 0.5 for the supervised BigGAN. For our GGAN model, with 5% labelled data as guidance, we achieve an IS score of 7.24 ± 0.05, and an FID score of 25.75 ± 0.1 for the generated samples, which is very close to the performance of the fully supervised BigGAN model and considerably better than unsupervised BigGAN model in terms of the IS score. For the FID score, the performance of the GGAN seems similar to the Unsupervised BigGAN. Comparison of the IS score and FID scores between different models are shown in the Table IV. We observe that the performance of the proposed GGAN is better than that of other models reported in Table IV. We also note similar performance for the Nsynth dataset in Table V. Note that the Nsynth dataset is being relatively new — many studies have not used it. Hence, we compare the performance of GGAN only with Supervised and Unsupervised BigGAN.

We present the comparison of the MOS score between different models in Table VI. We observe that the GGAN model

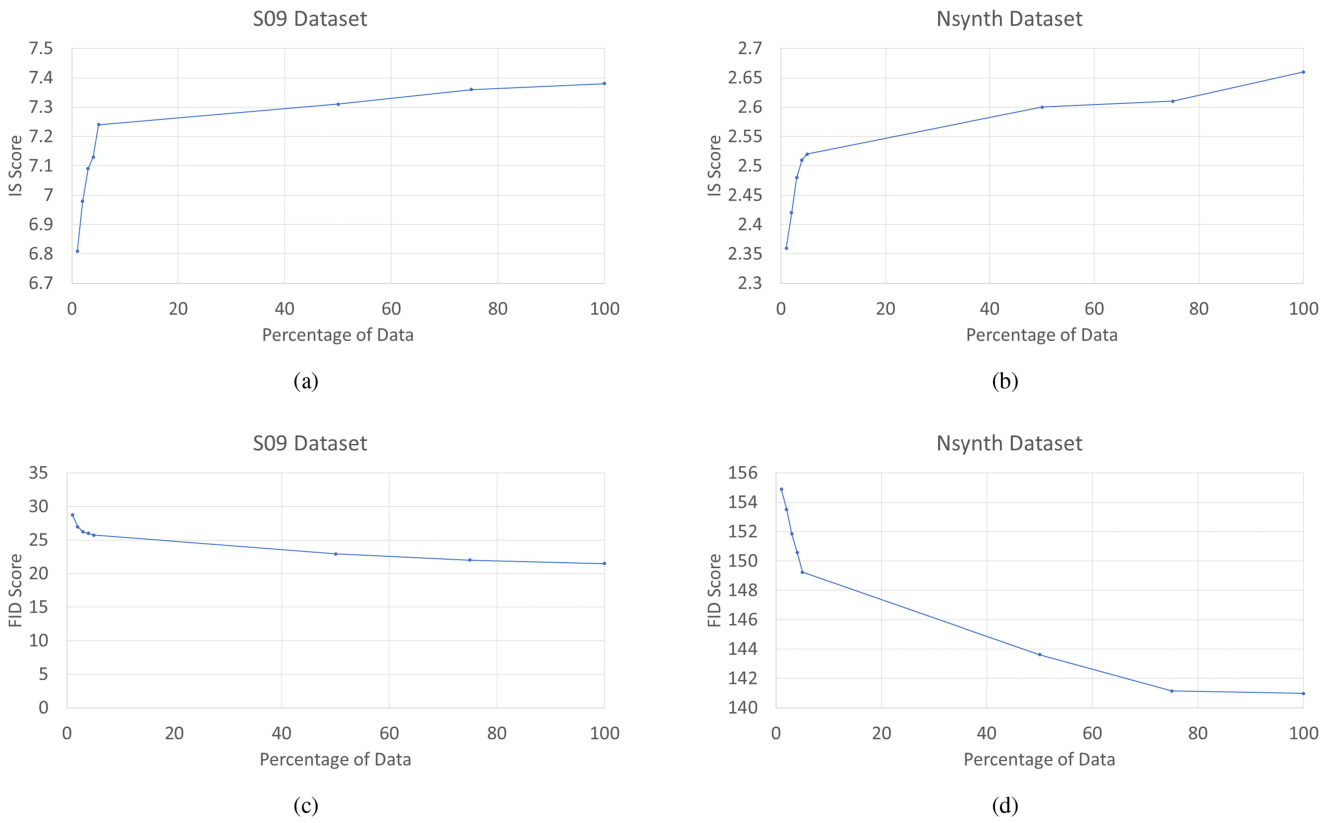


Fig. 2. The figure shows the relationship between the scores (IS and FID) and the percentage of data used as guidance for training GGAN model. Here, Left columns (a), (c) show the comparison for the S09 dataset, and the right columns (b), (d) show the comparison for the Nsynth dataset.

TABLE IV
COMPARISON BETWEEN THE PERFORMANCE OF THE GGAN MODEL AND THE OTHER MODELS ON THE S09 DATASET, IN TERMS OF THE QUALITY OF THE GENERATED SAMPLES, MEASURED WITH IS AND FID SCORE

Model Name	IS Score	FID Score
Real (Train Data) [8]	9.18 ± 0.04	-
Real (Test Data) [8]	8.01 ± 0.24	-
TiFGAN [10]	5.97	26.7
WaveGAN [8]	4.67 ± 0.01	-
SpecGAN [8]	6.03 ± 0.04	-
Supervised BigGAN	**7.33 ± 0.01**	**24.40 ± 0.50**
Unsupervised BigGAN	6.17 ± 0.2	24.72 ± 0.05
GGAN	7.24 ± 0.05	25.75 ± 0.10

TABLE V
COMPARISON BETWEEN THE PERFORMANCE OF THE GGAN MODEL AND THE OTHER MODELS ON THE NSYNTH DATASET, IN TERMS OF THE QUALITY OF THE GENERATED SAMPLES, MEASURED WITH IS AND FID SCORE

Model Name	IS Score	FID Score
Real (Train Data)	2.83 ± 0.02	-
Real (Test Data)	2.81 ± 0.12	-
Supervised BigGAN	**2.64 ± 0.08**	**148.30 ± 0.23**
Unsupervised BigGAN	2.21 ± 0.11	172.01 ± 0.15
GGAN	2.52 ± 0.06	149.23 ± 0.09

TABLE VI
COMPARISON BETWEEN THE PERFORMANCE OF THE GGAN MODEL AND THE OTHER MODELS ON BOTH THE S09 AND THE NSYNTH DATASET, IN TERMS OF MOS SCORE WITH 95% CONFIDENCE INTERVAL

Model Name	MOS (S09)	MOS (Nsynth)
Real Data	**4.1 ± 0.26**	**4.6 ± 0.24**
Supervised BigGAN	3.4 ± 0.47	4.1 ± 0.30
GGAN	3.2 ± 0.33	3.9 ± 0.38
Unsupervised BigGAN	2.9 ± 0.50	3.5 ± 0.42
TiFGAN	2.7 ± 0.51	-
WaveGAN	1.3 ± 0.20	-
SpecGAN	1.1 ± 0.14	-

outperforms the Unsupervised BigGAN and achieves a score close to the score given by the Supervised BigGAN.

The spectrogram of the generated samples from the S09 and Nsynth dataset are reported in Fig. 3. We observe some similarities between the generated samples and the real samples in the spectrogram space. As there is no one-to-one mapping relationship between the generated and real samples, the visual observation of the spectrogram cannot verify the quality of the spectrogram. Therefore, we have conducted further evaluations.

3) Results and Discussions: Manual Hearing Test: We manually checked the generated audios for different categories/condition. After reviewing 50 samples from each category, we note that the GGAN model can generate audio samples almost correctly for different categories for both the Nsynth and the S09 datasets. For the S09 dataset, we check the zero to nine

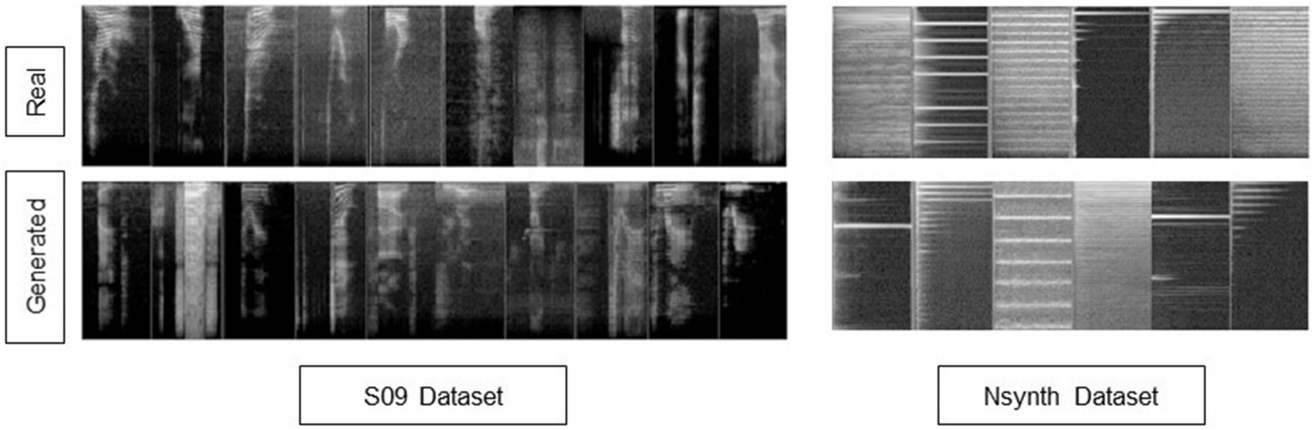


Fig. 3. This figure shows the generated samples from the GGAN and the real samples for both the S09 and the Nsynth dataset. Here, the left part shows the samples for the S09 dataset (ten digits categories), and the right part shows the samples for Nsynth dataset (three categories). The first row exhibits the real samples, and the second row exhibits the generated sample from GGAN.

TABLE VII
THE COMPARISON BETWEEN DIFFERENT CNN CLASSIFIERS BASED ON THE TEST DATA CLASSIFICATION ACCURACY FROM THE S09 DATASET. THE CNN MODELS ARE TRAINED WITH THE GENERATED SAMPLES FROM DIFFERENT MODELS

Sample for Training	**Test Accuracy (in %)**
Train Data	95.52 ± 0.50
Supervised BigGAN	86.58 ± 0.56
GGAN	86.72 ± 0.47
Supervised BigGAN + Train Data	96.21 ± 1.34
GGAN + Train Data	$\mathbf{96.36 \pm 1.07}$

TABLE VIII
THE COMPARISON BETWEEN DIFFERENT CNN CLASSIFIERS BASED ON THE TEST DATA CLASSIFICATION ACCURACY FROM THE NSYNTH DATASET. THE CNN MODELS ARE TRAINED WITH THE GENERATED SAMPLES FROM DIFFERENT MODELS

Sample for Training	**Test Accuracy (in %)**
Train Data	92.01 ± 0.94
Supervised BigGAN	83.50 ± 0.62
GGAN	81.40 ± 0.48
Supervised BigGAN + Train Data	$\mathbf{93.91 \pm 0.37}$
GGAN + Train Data	93.82 ± 0.24

categories and for the Nsynth dataset, we check three categories: Guitar, Strings, and Mallet.

4) Results and Discussions: CNN Based Classification Accuracy: For the S09 dataset, the classification accuracy for the CNN model is $95.52\% \pm 0.50$ when it is trained on the whole training dataset. We receive an accuracy of $86.72\% \pm 0.47$ for the CNN model trained on the generated samples from the GGAN model (with 5% labelled data). Table VII shows the comparison with other models.

The GGAN generated samples belong to the training data distribution, and are not exactly the same as the available training data. Therefore, we can use these generated samples to augment the training dataset. To test the feasibility of data augmentation using the generated samples from GGAN, we mix together the train data and the generated data from GGAN and train another simple CNN model. The accuracy of the CNN model increases from $95.52\% \pm 0.50$ to $96.36\% \pm 1.07$. Therefore, the generated samples from the GGAN model can be useful at augmenting a dataset. We observe similar results for the Nsynth dataset (Table VIII). These results show that our GGAN model achieves superior performance in terms of generating correct samples for different categories.

C. Conditional Audio Generation Based on Guidance From Different Dataset

1) Setup: We also test the possibility of using the GGAN model to generate based on a given condition, where the guidance categories/condition comes from a non-related dataset. We use the whole S09 training data as the unlabelled dataset and the Librispeech as the labelled dataset for guidance on the gender categories. Note that gender information is not available for the S09 dataset. We take 10 male and 10 female speakers from the Librispeech dataset with labels. Our expected output from the GGAN is to produce the male and female spoken digits based on the guidance from the Librispeech dataset.

The pretrained network used before to calculate the IS and FID scores is no longer effective, as it was trained on the digit classification task. So, we train another simple CNN model for the gender classification task as we want to evaluate gender-based generation. This pretrained model is used for calculating IS and FID scores. To achieve this goal, we randomly select 15 male and 15 female speakers from the Librispeech dataset. We use ten males and ten females to train the CNN model and the others for testing. We achieve an accuracy of 98.3 ± 0.50. This pretrained model can be used for calculating the IS and FID

TABLE IX
COMPARISON BETWEEN THE PERFORMANCE OF THE GGAN MODEL (TRAINED WITH GENDER GUIDANCE) AND THE OTHER MODELS ON THE S09 DATASET. THE QUALITY OF THE GENERATED SAMPLES BASED ON THE GENDER ATTRIBUTES OF THE SPEAKER IS MEASURED WITH THE IS AND THE FID SCORE

Model Name	IS Score	FID Score
Train Data	1.92 ± 0.04	-
Test Data	1.91 ± 0.05	-
Unsupervised BigGAN	1.13 ± 0.89	56.01 ± 0.85
Supervised BigGAN	1.48 ± 0.56	35.22 ± 0.50
GGAN (Digit Guided)	1.58 ± 0.05	37.75 ± 0.10
GGAN (Gender Guided)	$\mathbf{1.73 \pm 0.08}$	$\mathbf{23.72 \pm 0.04}$

scores for different models to evaluate the sample generation quality for gender attributes.

2) Results and Discussions: After evaluating the generated samples of GGAN, we successfully generate samples of the 0-9 digits for male and female speakers. We achieve an IS score of 1.73 ± 0.08, and an FID score of 23.72 ± 0.04. From Table IX, we observe that GGAN performs well when it was guided with gender categories. These results prove that guidance from another dataset is effective for the GGAN model.

D. Guided Representation Learning

1) Setup: For representation learning, we compare the GGAN model with unsupervised the BigGAN and a supervised Convolutional Neural Network (CNN). Our primary goal is to learn representation from the unlabelled S09 training dataset so that we can get better classification result on the S09 test dataset. For any unsupervised GAN model, the latent space captures the representation of the training dataset. To map the real data samples to that latent space, we follow the strategy from the BiGAN study [61].

After training the unsupervised BigGAN, we train a feature extraction network to reverse map the sample to a latent distribution to get the representation for real samples. Then, we train a simple classifier at the top of the feature extraction network with different percentage of the randomly sampled labelled dataset (1% to 5%, 50%, 75%, 100%) from the training dataset and evaluate on the test dataset. The sampling of the training dataset is repeated three times, and the results are averaged. We use the same percentage of labelled data (1% to 5%, 50%, 75%, 100%) for the training of a CNN network and use the same CNN architecture as of the feature extraction network.

We take the pretrained D, F, and C_x networks from the GGAN models, and pass the test dataset through those pretrained networks to get the prediction for classes $C_x(F(D(x_{test})))$. Then we compute the classification accuracy on the test dataset. The C network is built on top of the F network. Therefore, if the F network of the GGAN model cannot learn the representation according to the given guidance, the C network will not perform better on the test dataset.

To investigate the quality of the learnt latent/representation space by the generator, we conduct a linear interpolation between two random points in the latent space as in the DCGAN work [11]. Furthermore, if the F network of the GGAN model can learn a representation according to the guidance, the guided categories should be easily separable in the latent space. To investigate this scenario, we visualise the learnt representation in the 2D plain. We take the representation/feature $F(D(x_{test}))$ of the test dataset passing through the trained D and F networks. Then, the higher dimensional features are visualised with the t-distributed stochastic neighbour embedding (t-SNE) [76] visualisation technique.

TABLE X
RELATIONSHIP BETWEEN THE PERCENTAGE OF THE DATA USED AS THE GUIDANCE DURING THE TRAINING AND THE S09 TEST DATASET CLASSIFICATION ACCURACY OF THE GGAN MODEL

Training Data Size (Percentage)	CNN Network	BiGAN	GGAN
1	82.21 ± 1.20	73.01 ± 1.02	84.21 ± 2.24
2	83.04 ± 0.34	75.56 ± 0.41	85.39 ± 1.24
3	83.78 ± 0.23	78.33 ± 0.07	88.25 ± 0.10
4	84.11 ± 0.34	80.03 ± 0.01	91.02 ± 0.50
5	84.50 ± 1.02	80.84 ± 1.72	92.00 ± 0.87
50	89.50 ± 0.68	83.25 ± 2.10	94.78 ± 0.32
75	93.50 ± 0.45	85.14 ± 0.94	95.03 ± 0.11
100	$\mathbf{95.52 \pm 0.50}$	$\mathbf{86.77 \pm 2.61}$	$\mathbf{96.51 \pm 0.07}$

TABLE XI
RELATIONSHIP BETWEEN THE PERCENTAGE OF THE DATA USED AS THE GUIDANCE DURING THE TRAINING AND THE NSYNTH TEST DATASET CLASSIFICATION ACCURACY OF THE GGAN MODEL

Training Data Size (Percentage)	CNN Network	BiGAN	GGAN
1	85.76 ± 1.10	82.21 ± 0.84	88.52 ± 0.32
2	89.79 ± 0.51	86.65 ± 0.57	91.69 ± 0.24
3	89.83 ± 0.49	87.21 ± 0.46	91.95 ± 0.20
4	90.52 ± 0.25	87.59 ± 0.41	92.16 ± 0.19
5	91.07 ± 0.31	87.95 ± 0.39	92.45 ± 0.14
50	91.31 ± 0.53	87.97 ± 0.31	92.61 ± 0.12
75	91.93 ± 0.78	88.03 ± 0.29	93.09 ± 0.11
100	$\mathbf{92.01 \pm 0.94}$	$\mathbf{88.09 \pm 0.24}$	$\mathbf{93.56 \pm 0.09}$

2) Results and Discussions: Classification Accuracy: A comparison on the test accuracy between the CNN, BiGAN, and GGAN is shown in Table X and XI. For the S09, with 5% labelled data, GGAN achieves an accuracy of $92.00 \pm 0.87\%$, which is close to the accuracy of the fully supervised CNN model ($95.52 \pm 0.50\%$). For the Nsynth dataset, the GGAN ($92.45 \pm 0.14\%$) outperforms the fully supervised CNN model (92.01 ± 0.94,%).

3) Results and Discussions: Linear Interpolation of the Latent Space: After the linear interpolation in the S09 dataset, we observe a smooth transition in the generated spectrogram space and the audio after converting the spectrogram to audio. Here, if the interpolation is conducted between the same digits from different speakers, we notice the changes in voice characteristics. Moreover, if we interpolate between two different digits and different speakers, we notice changes in the voice characteristics. In the middle point, it sounds like mixed digits. The more we approach towards a digit in the interpolation space, the more it sounds like that digit.

For the unsupervised BigGAN, during the linear interpolation, we notice a smooth transition in the generated spectrogram

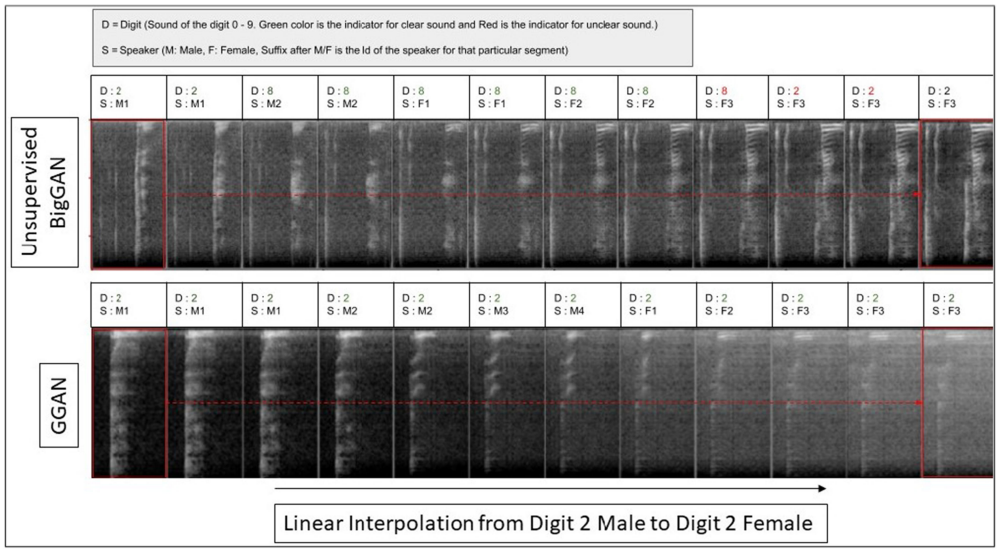


Fig. 4. Generated spectrogram from the interpolation of the latent space from the male sound of digit 2 to the female sound of digit 2 (left to right). The top row is the linear interpolation for the unsupervised BigGAN, and the bottom row represents the linear interpolation for the proposed GGAN.

space, but after converting them to the audio, we find that the transition becomes non-smooth. As the Unsupervised BigGAN is trained independent of any condition on the categorical distribution, it does not learn the relationship between different audio data categories. In contrast, our GGAN model can learn the categorical distribution of the dataset given guidance from a fewer percentage of labelled data; it learns the attributes related to categories in the latent space. This results in a smooth transition in both the spectrogram and the (temporal) audio space. For the Nsynth dataset, we observe similar results.

In Fig. 4, we show the linear interpolation for both the unsupervised BigGAN and the proposed GGAN. We avoid the latent space interpolation for the supervised BigGAN. The generator of the supervised BigGAN uses both conditions, c, and the latent space, z, during the sample generation, $G(z, c)$. As the condition code is given during the training, it makes the supervised BigGAN condition-independent, discouraging the Generator from learning any conditional characteristics in the latent space. It instead learns the common attributes in the latent space. In our S09 dataset experiment, the supervised BigGAN does not disentangle the digit categories (condition) in the latent space. It learns the common characteristics like gender, pitch, volume, noise, etc. in the latent space, and generates different digits for the same latent space given different conditions.

4) Results and Discussions: T-SNE Visualisation: In Fig. 5, we observe that the features of the similar categories are clustered together, and they are easily separable. This implies that in the latent space, the GGAN disentangles the categories successfully. We also observe similar behaviour for the Nsynth dataset.

E. Impact of the Hyperparameter

In the GGAN model, the Encoder E, Classifier C, and Generator G are core networks, and the weights of these networks are updated together based on equation 6. In this equation, we have losses G_{loss1}, G_{loss2}, and MG_{loss} responsible for sample

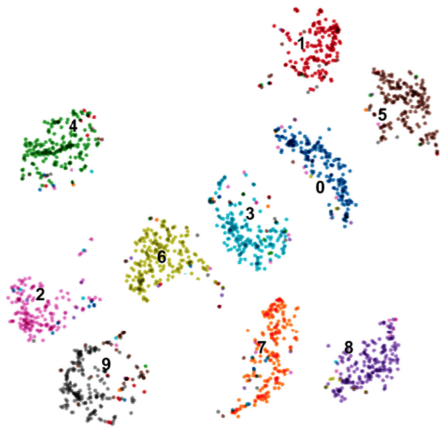


Fig. 5. t-SNE visualisation of the learnt representation of the test data of the S09 dataset. Here, different colours of points represent different digit categories. Representations of the different digit categories are clustered together.

generation. To give equal importance to each of the losses, we average these three losses (divided by three). We also have EC_{loss} and GC_{loss} in the equation. EC_{loss} ensures that the latent space is divided according to the given category, and GC_{loss} makes sure that the samples are generated according to the correct category.

We have three hyperparameters; α, β, and γ. The hyperparameter α is the weight for the overall G_{loss1}, G_{loss2}, and MG_{loss} losses in the context of the whole Equation 6. Therefore, α controls the generation quality of the GGAN model. If we increase the value of α, the model will reduce its focus on EC_{loss} and GC_{loss} and vice versa. Increasing the value of β will increase the focus of GGAN on the latent classification, and the purpose of the latent space for generating diverse samples will be lost. A higher β value forces the latent space to converge into a single point for any given condition. Lowering the β value will reduce the quality of encoding categorical distribution in the learnt latent space from the Encoder E. Finally, γ is responsible for

TABLE XII
ABLATION STUDY OF THE GGAN MODEL. A SCORE (IS, FID, AND CLASSIFICATION ACCURACY) IS CALCULATED AFTER REMOVING ANY COMPONENT FROM THE MODEL KEEPING OTHER COMPONENTS UNCHANGED

Removed Parts	IS	FID	Classification Accuracy
None (Full Model)	7.24	25.75	**92.00**
E and C_e	6.36	41.89	76.25
F and C_x	5.98	34.50	-
D_2	6.97	28.22	91.29
D_f	**7.27**	**24.57**	87.16

generating samples according to the given category/condition. If the value of γ is increased, the Generator will focus on sample classification rather than diversity. The model will, therefore, not be able to produce diverse samples for each category. If the value of γ is reduced, then the Generator will not be forced to generate samples according to the given categories. Considering these scenarios, we want to give equal weights to α, β, and γ, and we use a value 1 for all of these three hyperparameters. This essentially confirms that we provide equal weight to the losses, while considering $(G_{loss1} + G_{loss2} + MG_{loss})/3$ as a single loss.

VI. ABLATION STUDY

We conduct an ablation study to understand the significance of the components of the GGAN model. The study is conducted on the S09 dataset with 5% labelled data as guidance. Table XII shows the IS score, FID score, and Classification Accuracy after removing related components from the GGAN model. After removing D_f, we notice that the IS and FID scores improve, but the classification accuracy decreases. Note that, the third discriminator D_f forces to reverse map the generated sample to the latent space. Also note, the Feature Extractor F is built on top of the Discriminator D_1, which focuses on both feature learning and discriminating real or fake samples from the Generator. When D_f is removed, D_1 can only focus on the generated samples to discriminate, thus improving the generation of the Generator. However, when D_f is removed, the Feature Extractor F cannot utilise the learnt latent space from Encoder E and thus, the classification accuracy drops. Here, we use the extra discriminator D_2 in our model because D_1 cannot focus only on discrimination only. Without D_2, the performance of our model drops significantly, which is evident from Table XII.

VII. CONCLUSION

In this paper, we proposed the novel Guided Generative Adversarial Neural network (GGAN), where we guide an unsupervised GAN network with only 5% of the labelled data. This method allows the network to learn specific class dependent attributes while learning the representation of the dataset. We showed that the GGAN can learn powerful representations as well as generate good-quality samples given a small amount of labelled data as guidance. As we guide the model during the training process according to a post-task, the proposed GGAN can be used to learn task-specific representations.

The key challenge we faced is related to the sample generation of the model. In particular, there was an issue of mode collapse, which we could not sufficiently address implementing different techniques in the literature [67]. However, inspired by the Mode Seeking GAN [62], we modified the loss function and created a unique feature loss, which immediately fixed the mode collapse problem. The feature loss calculates the ratio of the difference between the two real samples and two generated samples. If the generator creates very similar or the same samples, it gets penalised and tries to find more modes. The Audio samples discussed in this paper can be found under the GitHub link.[1]

We used one-second audio following the current literature [8], [10]. One-second audio provides a good trade-off between complexity and information contained. Less than one-second audio might reduce the complexity but can cause information sparsity, impacting the generation and representation performance [77]. Usually, GAN-based frameworks show good performance with low-resolution data [1], [4]. Therefore, our GGAN model should work well for any audio dataset with lower temporal resolution (less than one second). However, it remains a challenge to make it useful for long audio with higher complexity. Nevertheless, any long audio can be divided into one-second audio and used with the GGAN model.

Our GGAN model is not likely to perform well at generating continuous speech audio where it is not conditioned on the speech-related features. To obtain good quality continuous speech generation with GGAN, different techniques from the recent literature can be used [6], [14], [27], [28], [78]. In the future, one should aim to explore the impact of using other high-performing GAN architectures such as progressive GAN [2] or the StyleGAN [3] within GGAN framework replacing the BigGAN architecture.

REFERENCES

[1] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[2] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proc. Int. Conf. Learn. Representations*, 2018.
[3] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4401–4410.
[4] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *Proc. Int. Conf. Learn. Representations*, 2019.
[5] J. Donahue and K. Simonyan, "Large scale adversarial representation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 10541–10551.
[6] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 6199–6203.
[7] M. Bińkowski *et al.*, "High fidelity speech synthesis with adversarial networks," *Proc. Int. Conf. Learn. Representations*, 2020.
[8] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *Proc. Int. Conf. Learn. Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=ByMVTsR5KQ
[9] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Dversarial neural audio synthesis," in *Proc. Int. Conf. Learn. Representations*, 2019.
[10] A. Marafioti, N. Perraudin, N. Holighaus, and P. Majdak, "Adversarial generation of time-frequency features with application in audio synthesis," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4352–4362.

[1] [Online]. Available: https://knhuq.github.io/GGAN.html

[11] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016, *arXiv:1511.06434*.

[12] M. Lucic, M. Tschannen, M. Ritter, X. Zhai, O. Bachem, and S. Gelly, "High-fidelity image generation with fewer labels," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4183–4192.

[13] F. Locatello *et al.*, "Challenging common assumptions in the unsupervised learning of disentangled representations," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4114–4124.

[14] K. Kumar *et al.*, "Melgan: Generative adversarial networks for conditional waveform synthesis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 14910–14921.

[15] G. Yang, S. Yang, K. Liu, P. Fang, W. Chen, and L. Xie, "Multi-band MelGAN: Faster waveform generation for high-quality text-to-speech," *Proc. IEEE Spoken Lang. Technol. Workshop*, 2021, pp. 492–498.

[16] A. van den Oord *et al.*, "WaveNet: A generative model for raw audio," in *Proc. Speech Synth. Workshop*, 2016.

[17] A. Van Den Oord *et al.*, "Parallel wavenet: Fast high-fidelity speech synthesis," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3918–3926.

[18] W. Ping, K. Peng, and J. Chen, "Clarinet: Parallel wave generation in end-to-end text-to-speech," in *Proc. Int. Conf. Learn. Representations*, 2019.

[19] Y. Wang *et al.*, "Tacotron: Towards end-to-end speech synthesis," in *Proc. Interspeech*, 2017.

[20] J. Shen *et al.*, "Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 4779–4783.

[21] J. Sotelo *et al.*, "Char2wav: End-to-End speech synthesis," in *Proc. Int. Conf. Learn. Representations*, 2017.

[22] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 2, pp. 236–243, April 1984.

[23] S. Mehri *et al.*, "Samplernn: An unconditional end-to-end neural audio generation model," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2017.

[24] E. Song, F. K. Soong, and H. Kang, "Effective spectral and excitation modeling techniques for LSTM-RNN-based speech synthesis systems," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 11, pp. 2152–2161, Nov. 2017.

[25] Y. Ai and Z. Ling, "A neural vocoder with hierarchical generation of amplitude and phase spectra for statistical parametric speech synthesis," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 839–851, January 2020, doi: 10.1109/TASLP.2020.2970241.

[26] Z. Ling, Y. Ai, Y. Gu, and L. Dai, "Waveform modeling and generation using hierarchical recurrent neural networks for speech bandwidth extension," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 26, no. 5, pp. 883–894, May 2018.

[27] T. Kaneko and H. Kameoka, "Cyclegan-VC: Non-parallel voice conversion using cycle-consistent adversarial networks," in *Proc. 26th Eur. Signal Process. Conf.*, 2018, pp. 2100–2104.

[28] T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo, "StarGAN-VC2: Rethinking conditional methods for StarGAN-based voice conversion," in *Proc. Interspeech*, 2019.

[29] Z. Průša, P. Søndergaard, N. Holighaus, C. Wiesmeyr, and P. Balazs, "The large time-frequency analysis toolbox 2.0," in *Proc. Int. Symp. Comput. Music Multidisciplinary Res.*, Springer, 2013, pp. 419–442.

[30] H. Phan, L. Hertel, M. Maass, R. Mazur, and A. Mertins, "Learning representations for nonspeech audio events through their similarities to speech patterns," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 4, pp. 807–822, Apr. 2016.

[31] S. Parekh, S. Essid, A. Ozerov, N. Q. K. Duong, P. Pérez, and G. Richard, "Weakly supervised representation learning for audio-visual scene analysis," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 416–428, December 2020, doi: 10.1109/TASLP.2019.2957889.

[32] Y. Chen, S. Huang, H. Lee, Y. Wang, and C. Shen, "Audio word2vec: Sequence-to-sequence autoencoding for unsupervised learning of audio segmentation and representation," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 9, pp. 1481–1493, Sep. 2019.

[33] P. Agrawal and S. Ganapathy, "Interpretable representation learning for speech and audio signals based on relevance weighting," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 2823–2836, October 2020, doi: 10.1109/TASLP.2020.3030489.

[34] H. Phan, L. Hertel, M. Maass, P. Koch, R. Mazur, and A. Mertins, "Improved audio scene classification based on label-tree embeddings and convolutional neural networks," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 6, pp. 1278–1290, Jun. 2017.

[35] H. Xie and T. Virtanen, "Zero-shot audio classification via semantic embeddings," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 29, pp. 1233–1242, Mar. 2021, doi: 10.1109/TASLP.2021.3065234.

[36] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using wavenet autoencoders," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 12, pp. 2041–2053, Dec. 2019.

[37] R. Zhang, P. Isola, and A. Efros, "Colorful image colorization," in *Proc. Eur. Conf. Comput. Vis.*, 2016, vol. 9907, pp. 649–666.

[38] S. Liu, A. Davison, and E. Johns, "Self-supervised generalisation with meta auxiliary learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 1679–1689.

[39] X. Zhan, X. Pan, Z. Liu, D. Lin, and C. C. Loy, "Self-supervised learning via conditional motion propagation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1881–1889.

[40] Z. Feng, C. Xu, and D. Tao, "Self-supervised representation learning by rotation feature decoupling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10364–10374.

[41] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised Representation Learning by Predicting Image Rotations," *CoRR*, vol. abs/1803.07728, 2018. [Online]. Available: http://arxiv.org/abs/1803.07728

[42] A. Van Den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv:1807.03748*.

[43] F. de Chaumont Quitry, M. Tagliasacchi, and D. Roblek, "Learning audio representations via phase prediction," 2019, *arXiv:1910.11910*.

[44] A. Nagrani, J. S. Chung, S. Albanie, and A. Zisserman, "Disentangled speech embeddings using cross-modal self-supervision," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 6829–6833.

[45] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," in *Proc. Interspeech*, 2019.

[46] K. Kawakami, L. Wang, C. Dyer, P. Blunsom, and A. v. d. Oord, "Learning robust and multilingual speech representations," 2020, *arXiv:2001.11128*.

[47] M. Rivière, A. Joulin, P.-E. Mazaré, and E. Dupoux, "Unsupervised pretraining transfers well across languages," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 7414–7418.

[48] A. Baevski, M. Auli, and A. Mohamed, "Effectiveness of self-supervised pre-training for speech recognition," 2019, *arXiv:1911.03912*.

[49] S. Latif, R. Rana, S. Khalifa, R. Jurdak, J. Qadir, and B. W. Schuller, "Deep representation learning in speech processing: Challenges, recent advances, and future trends," 2020, *arXiv:2001.00378*.

[50] S. Amiriparian, M. Freitag, N. Cummins, and B. Schuller, "Sequence to sequence autoencoders for unsupervised representation learning from audio," in *Proc. DCASE 2017 Workshop*, 2017, pp. 17–21.

[51] Y. Xu *et al.*, "Unsupervised feature learning based on deep models for environmental audio tagging," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 6, pp. 1230–1241, Jun. 2017.

[52] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*.

[53] J. Chang and S. Scherer, "Learning representations of emotional speech with deep convolutional generative adversarial networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 2746–2750.

[54] H. Yu, Z. Tan, Z. Ma, and J. Guo, "Adversarial network bottleneck features for noise robust speaker verification," in *Proc. Interspeech*, 2017.

[55] A. Spurr, E. Aksan, and O. Hilliges, "Guiding infogan with semi-supervision," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*. Springer, 2017, pp. 119–134.

[56] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011.

[57] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 3730–3738.

[58] A. Krizhevsky, "Learning multiple layers of features from tiny images," Uni. Toronto, 05 2012.

[59] J. T. Springenberg, "Unsupervised and semi-supervised learning with categorical generative adversarial networks," 2015, *arXiv:1511.06390*.

[60] K. Sricharan, R. Bala, M. Shreve, H. Ding, K. Saketh, and J. Sun, "Semi-supervised conditional Gans," 2017, *arXiv:1708.05789*.

[61] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," 2016, *arXiv:1605.09782*.

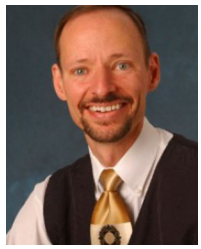
[62] Q. Mao, H.-Y. Lee, H.-Y. Tseng, S. Ma, and M.-H. Yang, "Mode seeking generative adversarial networks for diverse image synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1429–1437.

[63] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *Adv. Neural Inf. Process. Syst.*, vol. 29, pp. 2172–2180, 2016.
[64] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018, *arXiv:1804.03209*.
[65] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2015, pp. 5206–5210.
[66] J. Engel *et al.*, "Neural audio synthesis of musical notes with wavenet autoencoders," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1068–1077.
[67] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Adv. Neural Inf. Process. Syst.*, vol. 29, pp. 2234–2242, 2016.
[68] 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," *Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 6626–6637, 2017.
[69] S. Barratt and R. Sharma, "A note on the inception score," 2018, *arXiv:1801.01973*.
[70] K. Shmelkov, C. Schmid, and K. Alahari, "How good is my gan?" in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 213–229.
[71] F. Ribeiro, D. Florêncio, C. Zhang, and M. Seltzer, "Crowdmos: An approach for crowdsourcing mean opinion score studies," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2011, pp. 2416–2419.
[72] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.
[73] D. Dowson and B. Landau, "The fréchet distance between multivariate normal distributions," *J. Multivariate Anal.*, vol. 12, no. 3, pp. 450–455, 1982.
[74] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[75] P. Micikevicius *et al.*, "Mixed precision training," 2017, *arXiv:1710.03740*.
[76] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov.Nov., 2008.
[77] W.-C. Lin and C. Busso, "An efficient temporal modeling approach for speech emotion recognition by mapping varied duration sentences into fixed number of chunks," in *Proc. INTERSPEECH*, Shanghai, China, Oct. 2020, pp. 2322–2326.
[78] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang, "Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks," 2017, *arXiv:1704.00849*.

Kazi Nazmul Haque received the master's degree in information technology from Jahangirnagar University, Dhaka, Bangladesh. He is currently working toward the Ph.D. degree with the University of Southern Queensland, Toowoomba, QLD, Australia. He is currently an Research Associate with the University of Southern Queensland. He has been working professionally in the field of machine learning for more than five years. His research interests include unsupervised representation learning for audio data.

Rajib Rana is an Experimental Computer Scientist, Advance Queensland Senior Research Fellow, and an Associate Professor with the University of Southern Queensland, Toowoomba, QLD, Australia. He is also the Director of the IoT Health research program with the University of Southern Queensland. His research work aims to capitalise on advancements in technology along with sophisticated information and data processing to better understand disease progression in chronic health conditions and develop predictive algorithms for chronic diseases, such as mental illness and cancer. His current research interests include unsupervised representation learning, reinforcement learning, adversarial machine learning, emotional speech generation, and domain adaptation.

Jiajun Liu received the B.Eng. degree from Nanjing University, Nanjing, China, in 2006 and the Ph.D. degree from the University of Queensland, Toowoomba, QLD, Australia, in 2012. He is currently the Science Leaderwith Distributed Sensing Systems Group, Data61, CSIRO, Australia. He was an Associate Professor with the Renmin University of China, Beijing, China. From 2006 to 2008, he was a Researcher with IBM China Research Labs.



John H. L. Hansen (Fellow, IEEE) received the B.S.E.E. degree from Rutgers University, New Brunswick, NJ, USA, and the M.S. and Ph.D. degrees in electrical engineering from the Georgia Institute of Technology, Atlanta, GA, USA. In 2005, he joined the Erik Jonsson School of Engineering and Computer Science, the University of Texas at Dallas, Richardson, TX, USA, where he is currently an Associate Dean of research and a Professor of electrical and computer engineering. He also holds the Distinguished University Chair of telecommunications engineering and a joint appointment as a Professor of speech and hearing with the School of Behavioral and Brain Sciences. From 2005 to 2012, he was the Head of the Department of Electrical Engineering, the University of Texas at Dallas. At UT Dallas, he established the Center for Robust Speech Systems. From 1998 to 2005, he was the Department Chair and a Professor of speech, language, and hearing sciences, and a Professor of electrical and computer engineering with the University of Colorado Boulder, Boulder, CO, USA, where he co-founded and was an Associate Director of the Center for Spoken Language Research. In 1988, he established the Robust Speech Processing Laboratory. He has authored or coauthored 765 journal and conference papers, including 13 textbooks. He is an ISCA Fellow. He was the recipient of the Acoustical Society of America's 25 Year Award in 2010, and is currently serving as ISCA President (2017-2022). He was also a Member and the past Vice-Chair on U.S. Office of Scientific Advisory Committees (OSAC) for OSAC-Speaker in the voice forensics domain from 2015 to 2021. He was the IEEE Technical Committee (TC) Chair and a Member of the IEEE Signal Processing Society: Speech-Language Processing Technical Committee (SLTC) from 2005 to 2008 and from 2010 to 2014, elected the IEEE SLTC Chairman from 2011 to 2013, and elected an ISCA Distinguished Lecturer from 2011 to 2012. In 2016, he was awarded the honorary degree Doctor Technices Honoris Causa from Aalborg University, Aalborg, Denmark in recognition of his contributions to the field of speech signal processing and speech or language or hearing sciences. He was the recipient of the 2020 Provost's Award for Excellence in Graduate Student Supervision from the University of Texas at Dallas and the 2005 University of Colorado Teacher Recognition Award. He organised and was the General Chair for ISCA Interspeech-2002, the Co-Organiser and Technical Program Chair for the IEEE ICASSP-2010, Dallas, TX, and the Co-Chair and Organiser for IEEE SLT-2014, Lake Tahoe, NV. He will be the Tech. Program Chair for the IEEE ICASSP-2024, and Co-Chair and Organiser for ISCA INTERSPEECH-2022.

Nicholas Cummins is currently a Lecturer of AI for speech analysis for health with the Department of Biostatistics and Health Informatics, King's College London, London, U.K. His current research interests include speech processing, affective computing, and multisensory signal analysis. He is fascinated by the application of machine learning techniques to improve our understanding of different health conditions and mental health disorders in particular. He is actively involved in the RADAR-CNS project in which he assists in the management. After completing his Ph.D., he was a Postdoctoral Researcher with the Chair of Complex and Intelligent Systems with the University of Passau, Germany. Most recently, he was a habilitation candidate with the Chair of Embedded Intelligence for Health Care and Wellbeing with the University of Augsburg, Germany. During his time in Germany, he was involved in the DE-ENIGMA, RADAR-CNS, TAPAS, and sustAGE Horizon 2020 projects. He also wrote and delivered courses in speech pathology, deep learning and intelligent signal analysis in medicine.

Carlos Busso (Senior Member, IEEE) received the B.S. and M.S. degrees (with high honours) in electrical engineering from the University of Chile, Santiago, Chile, in 2000 and 2003, respectively, and the Ph.D. degree (2008) in electrical engineering from the University of Southern California (USC), Los Angeles, CA, USA, in 2008. He is an Associate Professor with Electrical Engineering Department, The University of Texas at Dallas (UTD), Richardson, TX, USA. He was selected by the School of Engineering of Chile, as the Best Electrical Engineer graduated in 2003 across Chilean universities. At USC, he received a Provost Doctoral Fellowship from 2003 to 2005 and a Fellowship in Digital Scholarship from 2007 to 2008. At UTD, he leads the Multimodal Signal Processing (MSP) laboratory. His research interests include human-centred multimodal machine intelligence and applications. His current research interests include affective computing, multimodal human-machine interfaces, nonverbal behaviours for conversational agents, in-vehicle active safety system, and machine learning methods for multimodal processing. He was the recipient of the NSF CAREER Award. In 2014, he was the recipient of the ICMI Ten-Year Technical Impact Award. In 2015, his student was the recipient of the third prize IEEE ITSS Best Dissertation Award (N. Li). He was also recipient of the Hewlett Packard Best Paper Award at the IEEE ICME 2011 (with J. Jain), and the Best Paper Award at the AAAC ACII 2017 (with Yannakakis and Cowie). He is the coauthor of the winner paper of the Classifier Sub-Challenge event at the Interspeech 2009 emotion challenge. His work has direct implication in many practical domains, including national security, health care, entertainment, transportation systems, and education. He was the General Chair of ACII 2017. He is a Member of ISCA, and AAAC, and a Senior Member of the ACM.

Björn W. Schuller (Fellow, IEEE) received the Diploma in 1999, the Doctoral degree for his study on automatic speech and emotion recognition in 2006, and the Habilitation and Adjunct Teaching Professorship in the subject area of signal processing and machine intelligence in 2012, all in electrical engineering and information technology from the Technical University of Munich, Munich, Germany. He is currently a Professor of artificial intelligence with the Department of Computing, Imperial College London, London, U.K., where he heads GLAM – the Group on Language, Audio, & Music, a Full Professor and the Head of the Chair of Embedded Intelligence for Health Care and Wellbeing with the University of Augsburg, Augsburg, Germany, and CEO of audEERING. He was previously a Full Professor and the Head of the Chair of Complex and Intelligent Systems with the University of Passau, Passau, Germany.