High-Fidelity Audio Generation and Representation Learning with Guided Adversarial Autoencoder

Kazi Nazmul Haque\textsuperscript{1}, Rajib Rana\textsuperscript{1}, and Bjørn W Schuller\textsuperscript{2,3}

\textsuperscript{1}University of Southern Queensland, Australia
\textsuperscript{2}Imperial College London, UK
\textsuperscript{3}University of Augsburg, Germany

Abstract—Unsupervised disentangled representation learning from the unlabelled audio data, and high fidelity audio generation have become two linchpins in the machine learning research fields. However, the representation learned from an unsupervised setting does not guarantee its’ usability for any downstream task at hand, which can be a wastage of the resources, if the training was conducted for that particular posterior job. Also, during the representation learning, if the model is highly biased towards the downstream task, it loses its generalisation capability which directly benefits the downstream job but the ability to scale it to other related task is lost. Therefore, to fill this gap, we propose a new autoencoder based model named “Guided Adversarial Autoencoder (GAAE)”, which can learn both post-task-specific representations and the general representation capturing the factors of variation in the training data leveraging a small percentage of labelled samples; thus, makes it suitable for future related tasks. Furthermore, our proposed model can generate audio with superior quality, which is indistinguishable from the real audio samples. Hence, with the extensive experimental results, we have demonstrated that by harnessing the power of the high-fidelity audio generation, the proposed GAAE model can learn powerful representation from unlabelled dataset leveraging a fewer percentage of labelled data as supervision/guidance.

Index Terms—Unsupervised Representation Learning, Guided Adversarial Autoencoder, Audio Generation, GAN.

I. INTRODUCTION

Representation learning is a very requisite research field where the common belief is that any higher-dimensional data can be mapped into a lower-dimensional representation space where the variational factors of the data are disentangled. Thus implies that the distinct and informative characteristics/attributes of the data are easily separable in the representation space \cite{1}. Therefore, learning disentangled representation from unlabelled dataset opens a window of opportunity for the researchers to utilise the vastly available unlabelled dataset for any downstream tasks \cite{2} such as learned disentangled representation from freely available YouTube audios can be used to improve the emotion recognition task from audio where the large labelled dataset is unavailable.

Recently, the Generative Adversarial Neural Network (GAN) \cite{3} has shown prodigious success for generating real-like samples by capturing the training data distribution \cite{4}, \cite{5}, \cite{6}, \cite{7}. Here, the GAN is comprised of a Generator network and a Discriminator network where these networks are trained to beat each other based on a minimax game. During the training session, the Generator tries to fool the Discriminator by generating real-like samples from a random noise/latent distribution, and the Discriminator tries to defeat the Generator by differentiating the generated sample from the real samples \cite{8}. During this game-play, the Generator disentangles some underlying attributes of the data in the given random latent distribution \cite{9}. Therefore, researchers have achieved great success in terms of learning powerful representation \cite{9}, \cite{10}, \cite{8}, \cite{7}, \cite{11}, \cite{12}, \cite{5} with GAN based models in a completely unsupervised manner. Hence, GAN based models can be used successfully in the field of audio research where limited or no labelled data is available.

Here, the representation learning capability of the GANs is dependent on its’ sample generation quality \cite{7}. Though the GAN based models are successful at generating high fidelity images, it fails to perform likewise for the complex audio waveform generation \cite{13}. Thus, to successfully generate audio with GANs, researchers have focused on working with the spectrogram (image-like 2D representation) of the audio which can be converted back to the audio with minimal loss \cite{14}, \cite{13}, \cite{15}. However, still the recently proposed high performing GAN architectures such as BigGAN \cite{6} or StyleGAN \cite{5} are not well explored in this audio field thus leaving a room for the researchers to explore the compatibility of these models for audio data.

Now, the representation learned with GANs in a completely unsupervised manner does not guarantee the usability of the learned representation for any particular downstream task because it can ignore the important characteristics of the data during the training which is important for succeeding in the downstream job \cite{16}. So, some shorts of bias towards the downstream task is necessary during the unsupervised training to succeed in the posterior task \cite{2}.

Hence, learning meaningful representation from the unlabelled dataset using GAN models, requires good generation as well as some guidance towards the downstream task. Therefore, we proposed a BigGAN based architecture called “Guided Generative Adversarial Neural Network (GGAN)”, which is capable of learning powerful representation from an unlabelled dataset with the guidance based on some labelled data samples by harnessing the power of its’ high fidelity spectrogram generation. But the focus of the GGAN was to learn representation for any particular downstream task which makes the learned representation useless for any other unrelated task \cite{16}.
Nonetheless, in many cases, it is desirable to learn representation in a manner so that it can be used for any particular downstream task as well as can be used for any future tasks independent of the downstream job at hand [17]. So to address the shortcoming of the GGAN model and the gap in the superior audio generation research, in this paper, we propose a novel autoencoder based model named “Guided Adversarial Autoencoder (GAAE)”. Here, our GAAE model can generate diverse and high fidelity audio samples and using this superior generation quality; it can learn two types of useful representations from an unlabelled audio dataset with a minimal amount of labelled data as guidance. Here, among these two types of representations, one is the guided/post-task-specific representation for capturing the attributes/characteristics for the downstream task, and another one is the general/style representation for capturing other general attributes of the data which is independent of the task at hand. In this paper, our primary contributions can be summarised as follows.

- We have proposed a novel autoencoder based model named GAAE, which can learn to generate high fidelity audio samples capturing the diverse mode of the training data distribution leveraging the guidance from a fewer percentage of labelled data samples from that particular or related dataset. Hence, we evaluate the sample generation quality of the proposed model based on two audio datasets from different domains; the Speech Command dataset (S09) and the Musical Instrument Sound dataset (Nsynth). After comparing the models’ performance with the literature, we have demonstrated that the GAAE model has performed significantly better than the state-of-the-art (SOTA) models.

- We evince that our GAAE model can learn to disentangle general attributes/characteristics of the data in the representation space which can be beneficial for any other future potential tasks. Furthermore, we have also demonstrated that the GAAE model can learn post-task-specific representation from unlabelled dataset according to the given guidance, which directly benefits the downstream tasks at hand, where the guidance comes from a fewer percentage of labelled samples, either from the same dataset or from other related datasets. The representation learning of the GAAE model is evaluated on three different datasets; the Speech Command dataset (S09), the Audio Book Speech dataset (Librispeech) and the Musical Instrument Sound dataset (Nsynth). After the evaluation, we have demonstrated that the GAAE model performs better than SOTA models.

II. BACKGROUND AND RELATED WORK

A. Audio Representation Learning

1) Supervised Representation Learning: Neural Network can learn powerful representation from supervised training on the large dataset, and this learned representation can be used for similar tasks or scenarios, where limited labelled data is available [18]. This supervised representation learning is prevalent in the field of computer vision, and many researchers have shown successful implementations due to the availability of the enormous amount of labelled data [19], [20], [21], [22], [23], [24], [25], [26]. Likewise, in the Audio domain, there are some availabilities of the large labelled datasets, so the researchers have utilised this opportunity to train neural networks to learn representation in a supervised manner and then transfer that learning for further audio processing tasks where labelled data is limited. In this work [27], authors have conducted supervised training with Artificial Neural Network on multilingual speech database GlobalPhone [28] to get Language-Independent Bottleneck Features. In another work [29] related to speech, authors have learned supervised representation from Soundnet Dataset [30] and used it to improve the performance of the anger detection in the speech audio. Apart from speech audio, supervised representation learning is successful for other acoustic scene classification works [31], [32] viz; researchers have pretrained convolutional neural network on audioset [32] which is a dataset of weakly labelled sound events from YouTube videos, to learn representation and use it for audio classification scenario where the labelled dataset is limited [33]. For instance, Kumar et al. [34] have used this supervised representation learning from audioset to improve environmental sound classification tested on ESC-50 dataset [35]. In the musical domain, Million Song Dataset [36] is used to determine supervised representation to ameliorate other musical audio classification tasks [37], [38]. Apart from these, researchers have successfully transferred Inception-v4 [39] model, which is trained on image classification, to the acoustic domain for the classification of bird sounds [40]. So supervised representation learning is very rewarding when we have access to a large amount of labelled dataset, but there are many cases where we have unlabelled datasets and labelling those datasets are very expensive, which makes supervised representation learning method obsolete for this scenario. Thus, the unsupervised representation learning solves this problem by learning meaningful representation from the unlabelled dataset. Therefore, this learned representation can be used to improve any other tasks on related datasets where labels are limited [41], [42].

2) Unsupervised Representation Learning: In the context of the unsupervised representation learning, the self-supervised learning has become very popular recently due to its unprecedented success in the field of computer vision [43], [44], [45], [46], [47], [48], [49] and natural language processing [50], [51], [52], [53]. Here, the self-supervised learning methods use the information present in the unlabelled datasets to provide a supervision signal for the feature/representation learning [54]. Likewise, in the audio field, researchers have achieved noteworthy performances using self-supervised representation learning. Here, in a work of Deepmind [55], the authors have proposed a model to learn useful representation from unsupervised speech data through predicting future observation in the latent space. In another work from Google [56], the representation is learned by predicting instantaneous frequency based on the magnitude of the Fourier transform. Furthermore, Arsha and et al. (2020) [57] proposed a cross-modal self-supervised learning method to learn speech representation from the co-relationship between the face and the audio in the video. Other efforts have been made by the researchers
to learn general representation by predicting the contextual frames of any particular audio frames like wav2vec [58], speech2vec [59], and audio word2vec [60]. Likewise, there are other successful implementations [61], [62], [63], [64] of the self-supervised representation learning in the field of audio.

Though self-supervised learning is very efficacious at learning representation from the unlabelled dataset, it requires manual endeavour to design the supervised signal [42]. Hence, autoencoders are mostly used by the researchers to learn representation from unlabelled dataset [65], [66], [67] in a fully unsupervised manner. Therefore, in this paper [68], authors learned representation with autoencoder from a large unlabelled dataset, which improved the emotion recognition from speech audio. Similarly, in another work, the authors used denoising autoencoder to improve the affect recognition from speech data [69]. Several works [70], [71], [72] have utilised the Variational Autoencoders (VAEs) [73] to learn efficient speech representation from unlabelled dataset. Recently, given the popularity of the adversarial training, different works have been conducted by the researchers to learn robust representation with GANs [74], [75] and Adversarial Autoencoders [76], [77].

Though learning representation from prodigiously available unlabelled datasets is very intriguing, the recent work from Google AI has proved that the completely unsupervised representation learning is not possible without any form of supervision [2]. Also, representation learned from an unsupervised method does not guarantee the usability of this learned representation for any post-use case scenario. Thus, we proposed Guided Generative Adversarial Neural Network (GGAN) [16], which can learn powerful representation from unlabelled audio dataset according to the supervision given from a fewer amount of labelled dataset. Therefore, in the learned representation space, the GGAN disentangles attributes of the data according to the given categories from the labelled dataset, which benefits the related post-use case scenario. Still, the generalisation is lost thus can not be used for non-related tasks. For an example, if the GGAN is guided with small amount of dataset with emotion labels and trained on a large number of speech audios from different people, the GGAN will learn emotion-related representation ignoring the other attributes such as gender of the speaker, background noise, pitch, intensity etc. Therefore, this will help to improve the emotion recognition task but can not be used for other tasks such as speaker gender identification [16]. Hence, we overcome this shortcoming by proposing Guided Adversarial Autoencoder (GAAE) model, which can learn general attributes of the unlabelled dataset in the representation space as well as the characteristics according to the given guidance from the fewer labelled data samples.

B. Audio Generation

Most of the audios are periodic, and high fidelity audio generation requires modelling higher order magnitude of the temporal scales, which makes it a challenging problem for the researchers [13]. Most of the research works related to audio generation are based on the audio synthesis viz; Aaron and et al. (2016) have proposed a powerful autoregressive model named “Wavenet” where it works great on text to speech (TTS) synthesis for both English and Mandarin. Later the authors have improved this work by proposing “Parallel Wavenet”, which is 20 times faster than “Wavenet”. Other research works have utilised the seq2seq model for TTS such as Char2Wav [78] and TACOTRON [79]. However, these audio generation methods are conditioned on the text data and mainly focused on speech generation. Thus, these methods can not be generalised to all other audio domains, even for speech data where transcripts are not available.

Here, in the context of generating audio without any condition on the text data, the GANs are very promising due to its massive success in the field of computer vision [7], [80], [81], [82], [5]. However, porting these image GAN architectures directly to the audio domain does not offer similar performance as the audio waveform is very complex than the image [14], [13]. Therefore, researchers have focused on generating spectrogram (2D image-like representation of the audio) rather than generating direct waveform. Then the generated spectrogram is converted back to audio. Here, Chris et al. (2019) [14] has trained GAN based model to generate spectrograms and successfully converted back to the audio with Griffin-Lim algorithm [83]. Furthermore, in the TiFGAN paper [15], authors have proposed phase-gradient heap integration (PGHI) [84] algorithm for better reconstruction of the audio from the spectrogram with minimal loss. As PGHI algorithm is good at reconstructing audio from the spectrogram, now the challenge is to generate realistic spectrogram. As the spectrogram is an image-like representation of the audio, any GAN based framework from the image domain should be compatible. Hence, the BigGAN architecture [6] has shown promising performance at generating high fidelity image generation, but it was not explored for the audio generation. Therefore, to fill this gap, we proposed Guided GAN (GGAN) architecture [16], which can generate superior audio with a fewer labelled dataset as guidance. Here, the GGAN model suffers from severe mode collapse, which is solved to some extent by the feature loss. Though GGAN achieved a SOTA performance in the audio generation, it does not guarantee superiority in terms of diversity of the mode within the generated samples. So we improve this work by proposing Guided Adversarial Autoencoder (GAAE) which ensures the high fidelity image generation as well as the mode diversity.

C. Closely Related Architectures

The proposed GAAE model is a semi-supervised model as we leverage a small amount of labelled data during the training. Here, In this work [85], the authors proposed a semi-supervised version of the InfoGAN model [9] to capture specific representation and generation according to the supervision which comes from the small number of labelled data. But, the success of this model in terms of the complex data distribution is not evident. Other researchers have explored the scope of the semi supervision in the GAN architectures [86], [87], [88] to improve the conditional generation but most of these works are not explored in the audio domain which leaves
a major gap for the researchers to address. The GAAE model is based on Adversarial Autoencoder (AAE) \[12\], where we have extended the AAE model to learn task-specific and generalised representation from the unlabelled dataset in a semi-supervised fashion. Furthermore, in the GAAE model, we have implemented a unique way to leverage the small amount of labelled data for high-fidelity audio generation. Here, we have also proposed a way to utilise the generated samples for improving the representation learning. Moreover, the building blocks for our GAAE model is BigGAN architecture; thus, we further contribute by exploring the use of BigGAN in an autoencoder based model for audio data.

III. PROPOSED RESEARCH METHODS

A. Architecture of the GAAE

GAAE is consisted of five neural networks ; Encoder E, Decoder D, Classifier C, Latent Discriminator L and Sample Discriminator S. Let, the parameters for these networks be \( \theta_e, \theta_d, \theta_c, \theta_L, \theta_S \) respectively. The Figure [I] shows the whole architecture of the model and the description is as follows.

1) Encoder: The Encoder E takes any unlabelled data sample \( x_u \sim p_{data} \) and outputs two latent samples \( z_{x_u} \sim u_z \) and \( z_{x_u} \sim q_z \), where \( p_{data} \) is the true unlabelled data distribution and \( u_z,q_z \) are two different continuous distributions learned by the E. Here, we want the latent \( z_{x_u} \) to capture the post-task-specific attributes/characteristics of the data and the latent \( z'_{x_u} \) to capture the general/style attributes of the data.

2) Classifier: We have a classifier network C which is trained with limited labelled data \( x_l \sim \hat{p}_{data} \), where \( \hat{p}_{data} \) is the labelled data distribution and not necessarily \( p_{data} \subset \hat{p}_{data} \). Here, with this \( \hat{p}_{data} \) the whole model get the guidance thus we call this data as “guidance data”. Now, the C network takes any latent sample and predicts the category class for that latent sample. To train C, we pass \( x_l \) through the E network and get two latent vectors \( \{ z_{x_l},z'_{x_l} \} = E(x_l; \theta_e) \). Then we only forward \( z_{x_l} \) through C to get the predicted label \( \hat{y}_{x_l} = C(z_{x_l}; \theta_c) \) and train C against the true label \( y_l \sim Cat(y_l,k = n) \) of the sample \( x_l \), where \( Cat(y_l,k = n) \) is the categorical distribution with \( n \) numbers of categories/labels. These labels are used as one-hot vector. For now, lets consider that C can classify the label of any sample correctly.

3) Decoder: The Decoder D maps any latent and categorical class/label variable to the data sample. Now, to get the reconstructed sample of \( x_u \), we pass the latent \( z_{x_u} \) and the label of \( x_u \) through the D network. As \( x_u \) is an unlabelled data sample, we get the label \( \hat{y}_{x_u} = C(z_{x_u}; \theta_c) \) through the network C and get the reconstructed sample \( \hat{x}_u = D(z_{x_u}, \hat{y}_{x_u}; \theta_d) \) from the D network. Here, we also want to use the D network for generating samples according to the given condition along with the reconstruction. Therefore, the same latent \( z_{x_u} \) is used with a random categorical variable (one-hot vector) \( y_l \), sampled from categorical distribution \( Cat(y_l,K = n, p = \frac{1}{n}) \), where \( n \) is the number of categories/labels and sampling probability for each category is \( \frac{1}{n} \). Now, we get the generated sample \( \hat{x}_g \sim p_{data} \), where \( p_{data} \) is the generated data distribution by the D network and it is trained to match \( p_{data} \) with the true data distribution \( p_{data} \). Here, the size of \( n \) is the same as the guided data, and we want the D network to generate data according to the categories from the guided data. Therefore, we ensure this with the Discriminator where the Discriminator gets the labels of the data from the network C. As we use a small number of labelled data, it is hard to train C due to the problem of overfitting. So we use generated sample \( \hat{x}_g \) and train the C network considering \( y_r \) as the true label/category, where the predicted label is \( \hat{y}_{z_g} = C(E(\hat{x}_g, \theta_c), \theta_c) \).

Here, \( C \) depends on the correct conditional generation from \( D \) and \( D \) depends on the classification from the C. During the training, the C network starts to predict the category of some samples from the given labelled data correctly. So the Discriminator learns to identify the correct category for those samples and force the D network to generate samples with the attributes related to these correctly classified samples. These, generated samples bring more characteristics with it, which is not present in the given labelled data but belongs to the data distribution. Now, as we feed these generated samples again to the C network with the associated conditional categories as correct labels, it learns to predict the correct category for more samples related to that generated samples. Then again, these new correctly classified samples improve the conditional generation of the D network. Hence, throughout the training, the C network and D network improve each other continuously. Meanwhile, during the training, representation learning (latent generation) capability of the E network is also ameliorated via the process of reconstructing sample \( x_u \), which also improves the performance of the C and D network eventually.

4) Discriminators: The GAAE model has two discriminators ; Sample Discriminator S and Latent Discriminator L. The \( S \) makes sure that the generated sample \( \hat{x}_g \) and reconstructed sample \( \hat{x}_u \), match the sample from the true data distribution \( p_{data} \). We train \( S \) with the sample and its label. Now, for the samples \( \hat{x}_g \) and \( \hat{x}_u \), we have labels \( y_l, \hat{y}_{z_g} \) respectively. So the pairs \( (\hat{x}_g, y_l) \) and \( (\hat{x}_u, \hat{y}_{z_g}) \) are considered fake labels for the \( S \). For the true data, both \( x_l \) and \( x_u \) is used together, where we get the label for the sample \( x_l \) from C and for the sample \( x_u \) we use the available true labels. Hence, in terms of distribution perspective, we get data distribution \( p_{data} \), mixing the distribution \( \hat{p}_{data} \) and \( p_{data} \). So \( S \) is trained with the true sample data \( x \sim p_{data} \) along with its’ associated label \( y \) if exists, otherwise the predicted label from \( C \).

Here, the E learns to map the general characteristics of the data in the latent distribution \( q_z \), excluding the categories from the guided data. Now, if we can draw the sample from \( Q_z \) distribution then, by using the categorical distribution as condition, we can generate diverse data for different categories (categories from the guided data) from the Decoder D. We can only sample from \( Q_z \) if the distribution is known to us. Therefore, we use another Discriminator \( L \) so that the E network is forced to match \( q_z \) to any known distribution \( p_z \), where \( p_z \) can be any known continuous random distribution (e.g. Continuous Normal Distribution, Continuous uniform distribution). The \( L \) network is trained through differentiating between the true latent \( z \sim p_z \) and the fake latent \( z'_{x_u} \).
B. Losses and Training

1) Encoder, Classifier and Decoder: For the E and D networks, we have sample generation loss \( G_{loss} \), sample reconstruction loss \( R_{loss} \) and latent generation loss \( L_{loss} \). To calculate generation and discrimination loss, we use hinge loss and for the reconstruction loss the Mean Squared Error (MSE) loss is used. For the \( G_{loss} \), we take the average of the generation loss for \( \hat{x}_u \) and \( \hat{x}_g \). Therefore,

\[
G_{loss} = -\frac{1}{2}(S(\hat{x}_u, y_{x_u}; \theta_s) + S(\hat{x}_g, y_{x_g}; \theta_s)).
\]

\[
L_{loss} = -(L(z'_{x_u}; \theta_l)).
\]

\[
R_{loss} = \frac{1}{N} \sum_{i=1}^{N} (\hat{x}_{ui} - x_{ui})^2.
\]

Now, for the C network, we calculate classification loss \( C_{loss} \) for the labelled data sample \( x_l \) and we detach the gradient of \( x_g \) when it is only used for the loss \( C_{loss} \). The model is implemented with pytorch and we detach the gradient of \( x_g \) when \( C_{loss} \) is calculated. Therefore,

\[
C_{loss} = -\sum y_l \log \hat{y}_{x_l}.
\]

We get the a combined loss \( EDC_{loss} \) for \( E, D \) and \( C \). The \( EDC_{loss} \) is calculated as,

\[
EDC_{loss} = \alpha \cdot (\omega_1 \cdot G_{loss} + \omega_2 \cdot (\lambda \cdot R_{loss} + \beta \cdot (\omega_3 \cdot C_{loss} + \omega_4 \cdot C_{loss} + \omega_5 \cdot L_{loss})).
\]

Here, the weights of the \( E, C \) and \( D \) networks are updated to minimize the loss \( EDC_{loss} \), where \( \omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \alpha, \beta, \lambda \) are the hyperparameters. The successful training of our GAAE model depends on these parameters. At the beginning of the training, we have noticed that the value of \( R_{loss} \) falls rapidly compared to other losses and results in very small gradient value. To mitigate this problem, we multiply \( R_{loss} \) with a hyperparameter \( \lambda \in R_{>0} \) and after hyperparameter tuning, we have found 20 as an optimal value for the \( \lambda \). The \( D \) network of the model is tuned for both the reconstruction loss \( R_{loss} \) and the generation loss \( G_{loss} \). Therefore, to balance between these two losses, the hyperparameter \( \omega_1 \) and \( \omega_2 \) is used where \( \omega_1, \omega_2 \in [0, 1] \) and \( \omega_1 + \omega_2 = 1 \). Here, we can force the model to focus more on either loss by increasing the hyperparameter for that particular loss. Likewise, for \( Cl_{loss}, C_{loss} \) and \( L_{loss} \) we use hyperparameters \( \omega_3, \omega_4, \omega_5 \) respectively where \( \omega_3, \omega_4, \omega_5 \in [0, 1] \) and \( \omega_3 + \omega_4 + \omega_5 = 1 \). In the \( EDC_{loss} \), \( G_{loss} \) and \( R_{loss} \) are responsible for sample generation quality, where \( Cl_{loss}, C_{loss} \) and \( L_{loss} \) are responsible for the latent generation quality. So to balance between sample generation quality.
Algorithm 1 Minibatch stochastic gradient descent training of the proposed GAAE model. The discriminator is updated $k$ times in one iteration. Here, for our experiment, we use $k = 2$ for better convergence.

1: for number of training iterations do
2: \hspace{1em} for $k$ steps do
3: \hspace{2em} Sample the latent/noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from $p_z$, the conditions (labels) $\{y_r^{(1)}, \ldots, y_r^{(m)}\}$ from $Cat(y_r)$, the unlabelled data samples $\{x_u^{(1)}, \ldots, x_u^{(m)}\}$ from $p_{data}$ and the labelled data samples $\{x_l^{(1)}, \ldots, x_l^{(m)}\}$ from $p_{data}$. Here, $m$ is the minibatch size.
4: \hspace{2em} Update the discriminator $S$ by ascending its stochastic gradient:
5: \hspace{3em} $\nabla_{\theta_s} \frac{1}{m} \sum_{i=1}^{m} \left[ S_{loss}^{(i)} \right]$. 
6: \hspace{2em} Update the discriminator $L$ by ascending its stochastic gradient:
7: \hspace{3em} $\nabla_{\theta_l} \frac{1}{m} \sum_{i=1}^{m} \left[ L_{loss}^{(i)} \right]$. 
8: \hspace{1em} Update the Encoder $E$, Decoder $D$ and Classifier $C$ by descending its stochastic gradient:
9: \hspace{2em} $\nabla_{\theta_e,\theta_d,\theta_c} \frac{1}{m} \sum_{i=1}^{m} \left[ EDC_{loss}^{(i)} \right]$. 
10: end for
11: end for

and latent generation, we use two hyperparameters $\alpha$ and $\beta$ where $\alpha, \beta \in [0, 1]$ and $\alpha + \beta = 1$.

2) Discriminators loss: For the Discriminator $S$ and $L$, we use hinge loss. The discrimination loss for the fake samples are averaged as we calculate the loss for both $\hat{x}_u$ and $\hat{x}_r$. Let the discrimination loss for $S$ and $L$ be $S_{loss}$, $L_{loss}$ respectively. Therefore,

\[
S_{loss} = -\min(0, -1 + S(x, C(E(x, \theta_c); \theta_s)); \theta_s)) \\
-\frac{1}{2} \left( \min(0, -1 - S(\hat{x}_u, \hat{y}_u); \theta_s)) \right) \\
L_{loss} = -\min(0, -1 + L(z, \theta_l)) \\
-\min(0, -1 - L(\hat{z}_u, \theta_l)),
\]

Here, we update the parameter $\theta_s$ and $\theta_l$ to maximise the loss $S_{loss}$ and $L_{loss}$ respectively. The algorithm 1 shows the training mechanism for the GAE model.

IV. DATA AND IMPLEMENTATION DETAIL

A. Datasets

For training the GAE model, we have used three audio datasets; S09 dataset [90], Librispeech dataset [91] and Nsynth dataset [92]. The S09 dataset consists of audios for different digits categories from zero to nine. This dataset is very noisy and comprised of 23,000 one-second audio samples uttered by 2618 speakers where the samples are labelled poorly. Furthermore, the S09 dataset only contains the labels for the audio digits [90]. Here, the Librispeech dataset is an English speech dataset with 1000 hours of audio recordings, and there are three subsets available in this Librispeech dataset containing approximately 100, 300 and 500 hours of recordings respectively. We used the subset with 100 hours of clean recordings as we do not need a large number of audios from this dataset. In this subset, the audios are uttered by 251 speakers where 125 are female, and 126 are male [91]. For our experiment, we only used the audios along with the gender labels of the speakers. Moreover, the Nsynth audio dataset contains 305,979 musical notes of size four seconds from ten different instruments where the sources are either acoustic, electronic or synthetic [92]. For this research work, we have used only three instruments with acoustic sources which are guitar, string and mallet.

B. Data Preprocessing

To evaluate the GAE model, we have used audio of length one second where the exact sample size was 16384, and the sampling rate was 16kHz. For the S09 dataset, we have zero-padded the one-second audios (16000 samples) to reach the sample size of 1634, where, for the Librispeech dataset, the one-second audio (16384 samples) was taken randomly from any particular audio clip. Furthermore, for the Nsynth dataset, the first one-second audio (16384 samples) was taken from any audio sample as it holds the majority of the instrument sound representation.

The audio data is converted to the log-magnitude spectrograms with the short-time Fourier Transform, and the generated log-magnitude spectrograms of the GAE model are converted to audio using PGGI algorithm [84]. From now on we refer the log-magnitude spectrogram as the spectrogram.

To obtain the spectrogram representation of the audio, the short-time Fourier Transform was calculated with the overlapping Hamming window of size 512 ms, and the hopping length was 128 ms. Therefore, the size of the spectrogram become $256 \times 128$ and then, we standardise the spectrogram with the equation $\frac{X - \mu}{\sigma}$ where, $X$ is the spectrogram, $\mu$ is the mean of the spectrogram and $\sigma$ is the standard deviation of the spectrogram. Now we clip the dynamic range of the spectrogram at $-r$, where, for the S09 and Librispeech dataset, the suitable value of $r$ was 10, and for the Nsynth dataset it was 15. After the clipping, we normalised the spectrogram values between -1 and 1. Now, this spectrogram representation of the audio is used as the input to the GAE model. Furthermore, the GAE model generates the spectrograms with values between -1 and 1. Then, we convert these spectrograms to audios with PGGI algorithm. For ease, we will refer these audios calculated from generated spectrogram as generated audios throughout the rest of the paper.

C. Measurement Metrics

We measured the performance of the GAE model, based on the generated samples and the learned representations.
Thus, the generated samples are evaluated with the Inception Score (IS) \[93\] and Frchet Inception Distance (FID) \[94, 95\] as these scores have become a de-facto standard for measuring the performance of any GAN based model \[96\]. To evaluate the representation/latent learning, we have considered classification accuracy, latent space visualisation and latent interpolation.

1) **Inception Score (IS):** The IS score is calculated based on the pretrained Inception Network \[97\] trained on the ImageNet dataset \[98\]. First, the logits are calculated for the images from the bottleneck layer of the Inception Network. Then the score is calculated with the equation given by,

\[
\exp(E_x KL(p(y|x)||p(y)))
\]

Here, \(x\) is the image sample, \(KL\) is the Kullback-Leibler Divergence (KL-divergence) \[99\], \(p(y|x)\) is the conditional class distribution for sample \(x\) predicted by the Inception Network and \(p(y)\) is the marginal class distribution. So the IS score computes the KL-divergence between the conditional label distribution and the marginal label distribution where the higher value indicates good quality generation.

2) **Frchet Inception Distance (FID):** The IS score is computed solely on the generated samples; thus no comparison is made between the generated and real samples and it is not a good measure for the samples diversity (mode) of the generated samples. So FID score solves this problem by comparing real samples with the generated samples \[90\] during the score calculation. Therefore, Frchet Inception Distance (FID) computes the Frchet Distance \[100\] between two multivariate Gaussian distributions for generated and real samples, parameterised by the mean and the covariance of the features extracted from the intermediate layer of the pretrained Inception Network. Therefore, the FID score is calculated based on,

\[
||μ_r - μ_g||^2 + Tr(Σ_r + Σ_g - 2(Σ_rΣ_g)^{1/2})
\]

Where, \(μ_r, μ_g\) are the means for the features of the real and generated samples respectively and similarly \(Σ_r, Σ_g\) are the covariances. Here, the lower value of the FID score indicates good generation quality.

The Inception Network is trained on the imagenet dataset thus offer reliable IS and FID score for related image dataset, but the spectrograms of the audios are entirely different from those imagenet samples. So, the Inception Network does not offer trustworthy scores for the audio spectrograms. Hence, instead of using Inception model, we train a classifier network based on the audio dataset and use this trained Classifier to calculate the IS and FID score for that particular dataset. In the case of S09 dataset, we used the pretrained Classifier released by the authors of the paper “Adversarial Audio Synthesis” \[14\] for a fair comparison and for the Nsynth dataset we train a simple Convolutional Neural Network (CNN) as the Classifier.

**D. Experimental Setup**

First, we have evaluated the overall sample generation quality of the GAAE model with the IS and FID scores, calculated based on the 50,000 generated samples for the random latent \(z\), and the random condition \(y_r\). Here, The spectrograms of the samples are generated from the \(D\) network and then they are converted to audios. After that, We used these generated audios to calculate the scores. Now, for all the dataset, we have used continuous normal distribution of size 128 for the latent \(z \sim N(μ = 0, σ^2 = 1)\) and ten digit categories (0-9) as the conditions \(y_r \sim Cat(y_r, K = 10, p = 0.1)\) for the S09 dataset. Furthermore, the three instrument categories (0-3) are used as the conditions \(y_r \sim Cat(y_r, K = 3, p = 0.33)\) for the Nsynth dataset.

The GAAE model is trained with different percentages (from 1% to 5%) of the data as guidance for both S09 and Nsynth dataset. For any particular percentage of data used as guidance, we have trained the GAAE model three times (in each run dataset was sampled randomly for the guidance), and the results are shown as the mean with the standard deviation. Here, due to having high wall time (approximately 21 hours on the two Nvidia p100 GPUs) for each run, we evaluate the model based on only three runs for any particular evaluation. Therefore, total wall time for the S09 and Nsynth dataset is approximately 21 × 3 × 5 (for different percentages of data) × 2 (for two datasets) = 630 hours or 26.25 days. Here, each run takes approximately 60,000 iterations with mixed-precision training \[101\] for the batch of size 128.

The results of the GAAE model are compared with the existing literature. Therefore, for comparing the GAAE model with Supervised BigGAN \[102\] and Unsupervised BigGAN \[102\], we have taken the results based on S09 dataset, from the GGAN paper \[16\]. Nevertheless, for the Nsynth dataset, we have trained these models with similar code and setting used in the GGAN paper. To calculate the IS and FID score for this Nsynth dataset, we have used our pretrained simple supervised CNN classifier, which is trained on the three classes (Guitar, String and Mallet) and achieved 92.01% ± 0.94 accuracy using the augmentation technique mentioned in the recent paper from google \[103\].

To evaluate the effectiveness of the guidance in the GAAE model in terms of generating correct samples from different categories/conditions, we have manually checked the audio samples generated for different categories based on both S09 and Nsynth dataset. However, it is not possible to check all the generated samples manually. So, we have trained a simple Convolutional Neural Network (CNN) Classifier with the samples generated for different random conditions/categories and used the random categories associate with the generated samples as the true labels. Then we evaluate the CNN Classifier on the test dataset based on the classification accuracy. If the GAAE model does not learn to generate correct samples for any given category and the generated samples do not match the training data distribution, the CNN model will never achieve good accuracy on the test dataset. Likewise, in this paper \[96\], the authors have suggested using this method to evaluate the performance of the overall generation quality of the model, along with the IS score and the FID score. For this evaluation, two CNN models with the same architecture are trained on the training data, and the generated data, respectively, where the size of the generated data is equal to the training data, and the class/category distribution is also kept the same. We
have conducted this experiment for both S09 and Nsynth dataset. For the sake comparison, we have also trained another two CNN models based on the generated samples from the supervised BigGAN and the GGAN model.

For both S09 and Nsynth Dataset, we have used the small amount of labelled data as guidance from the same dataset. So we wanted to investigate if the guidance from a completely different dataset works alike. In S09 dataset, we do not have the label available for the gender of the speakers, and we want to generate samples according to the conditions on the gender category. So we have collected random ten male and ten female speakers’ audio data from completely different Librispeech dataset to use as guidance during the training with S09 dataset. Here we used the S09 training data as the unlabelled dataset and Librispeech as the labelled dataset for the guidance on the gender labels. Similarly, like above experiments, we have used continuous normal distribution of size 128 for latent $z \sim \mathcal{N}(\mu = 0, \sigma^2 = 1)$ and two gender categories for the conditions $y_r \sim \text{Cat}(y_r, K = 2, p = 0.5)$.

Learning better Classifier with fewer labels is another prime goal of the GAAE model. The Classifier $C$ network of the GAAE model learns to classify the training data according to categories of the guidance data. For the S09 dataset, it learns to classify the digit categories, and for the Nsynth dataset, it learns the instrument classes. We designed the GAAE model to achieve the accuracy near to any supervised classifier. After training the GAAE model on any particular dataset, we did the evaluation based on the test data classification accuracy on that distinct dataset. For the sake of comparison, we trained a simple CNN classifier based on 1% to 5% training data where the data was heavily augmented with the techniques from googles’ paper [103] ( e.g. adding random noise, rotation of the spectrogram, multiplication with random zero patches etc.). Also, for further comparison, we trained BiGAN [81] model top of the unsupervised BigGAN and extracted the feature network after the training. Now, we train another feed-forward classifier network top of this feature network with the labelled data of size 1% to 5% where the weights for the feature network is fixed during the training. Then we evaluate this Classifier based on the test dataset. As the Classifier $C$ of the GAAE model is trained with fewer labelled data along with the generated samples from the decoder $D$, it will perform better only if the quality of the generated samples is near to the real sample and the generation is accurate according to the different categories. If the GAAE model does not learn the categorical distribution of the dataset according to the guidance, it will barely achieve a good result on the test dataset. Hence, we conducted this experiment for both S09 and Nsynth dataset.

In the GAAE model, the Classifier $C$ is built top of the latent $z_{x_{\text{ref}}} \sim u_z$, so $E$ network should learn this latent to disentangle the class categories according to the guided data. Like for the S09 dataset, we are using digit class as guidance, so in this latent space (representation space), the digit category should be disentangled. Furthermore, to explore this disentanglement, we have visualised the higher dimensional (128) latent space generated for the S09 test data in the 2D plane with the t-SNE (t-distributed stochastic neighbour embedding) [104] visualisation method.

| Model Name                   | IS Score | FID Score |
|------------------------------|----------|-----------|
| Real (Train Data) [14]       | 9.18 ± 0.04 | -         |
| Real (Test Data) [14]        | 8.01 ± 0.24 | -         |
| TiFGAN [106]                 | 5.97     | 26.7      |
| WaveGAN [14]                 | 4.67 ± 0.01 | -         |
| SpecGAN [14]                 | 6.03 ± 0.04 | -         |
| Supervised BigGAN            | 7.33 ± 0.01 | 24.40 ± 0.50 |
| Unsupervised BigGAN          | 6.17 ± 0.20 | 24.72 ± 0.05 |
| GGAN [16]                    | 7.24 ± 0.05 | 25.75 ± 0.10 |
| GAAE                         | 7.28 ± 0.01 | 22.60 ± 0.25 |

The $E$ network of the GAAE model is trained to match the $q_\alpha$ distribution with the known $p_z$ distribution. So we can sample $z'_{x_{\text{ref}}}$ from the $q_z$ distribution. To explore the learned representation $z'_{x_{\text{ref}}} \sim q_z$, we have generated audio samples for different categories/conditions keeping the $z'_{x_{\text{ref}}}$ same. Then, we manually hear the audios to investigate this scenario.

It is expected that the $D$ network of the GAAE model should learn to map the latent space $q_\alpha$ to the data distribution, so we can explore the latent space through generating sample for any particular latent. To investigate the latent space $q_\alpha$, further, we have conducted linear interpolation between two latent points like the DCGAN paper [8]. Therefore, a particular latent point $z_i$ within two latent points $z_0$ and $z_1$ is calculated with the equation $z_i = z_0 + \eta(z_0 - z_1)$, where $\eta$ is the step size from $z_0$ to $z_1$. Here, with this equation, we get the latent points in between the $z_0$ and $z_1$. Moreover, from the $D$ network, we get the generated samples for these latent points where the $y_r$ value is fixed.

For implementing our GGAN model, we have followed the network implementations, optimisation and hyperparameters from BigGAN paper [6]. For the optimisation we have used the Adam optimiser [105] with a learning rate of $5 \cdot 10^{-5}$ for network $E$, $D$ and $C$ where $2 \cdot 10^{-4}$ was the learning rate for the $S$ and $L$. The detailed architectures of the networks are given in the supplementary document.

V. Results

A. Sample Generation

Using only 5% labelled training data as guidance, the GAAE model has achieved $7.28 \pm 0.01$ IS score and FID score of $22.6 \pm 0.25$. The IS score is near to the supervised BigGAN and better than other research works mentioned in table I. In terms of the FID score, our GAAE model has performed superior to the other models mentioned in the table II which is the indication for more diversity/modes. The GAAE model has outperformed Supervised BigGAN model in terms of the diverse image generation, where GAAE has used only 5% labelled data and Supervised BigGAN is trained with 100% labelled training data. As our decoder is responsible for reconstructing all the training data as well
as for the generation, it is forced to learn more modes of the data distribution than the supervised BigGAN model. The figure 2 displays the spectrogram of the generated and the real samples where figure 4 show the samples for different conditions and latent samples. From these figures, we can observe that, due to the superior generation quality of the GAAE model, the generated samples are indistinguishable visually from the real samples. This is also true when we converted these spectrograms to audios.

To further validate the generation capability of the GAAE model, we used musical instrument dataset Nsynth with three acoustic class; Guitar, String and Mallet. The GAAE model has achieved the IS score of $2.58 \pm 0.01$ and FID score of $141.71 \pm 0.50$ with 5% labelled training data as guidance. In terms of the IS score, the performance is very near to supervised BigGAN ($2.64 \pm 0.08$) and better than unsupervised BigGAN ($2.21 \pm 0.11$). For the FID score, the performance is even greater than supervised BigGAN ($148.30 \pm 0.23$). The table II shows the comparison.

So, on both S09 and Nsynth dataset, the GAAE model has achieved superior generation quality like supervised BigGAN and in terms of sample diversity it has performed better than any other models mentioned in the table I and II. The audios can be found on the link: https://bit.ly/3coz5qO.

B. Guided Sample Generation

1) Guidance for learning categorical distribution: The generated samples for the S09 dataset based on different categories are shown in the fig 4. Furthermore, the generated samples for the Nsynth dataset is shown in the fig 5. However, it is not visually evident from these spectrograms that the model was able to generate correct samples according to the
Fig. 3. The figure demonstrates the difference between the generated spectrograms of the GAAE model and the real spectrograms of the data for the Nsynth dataset. The top row shows the generated samples, and the bottom row shows the real samples. The first block shows the spectrogram of the guitar, and the other two illustrate the spectrograms for the string and mallet.

| 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 |
| GAAE                 | 2.58 ± 0.01 | 141.71 ± 0.50 |

2) Classification accuracy based on the generated samples: For the S09 dataset, the test data classification accuracy for the CNN model trained with all the available labelled data is 95.52% ± 0.50, and 91.14% ± 0.17 for the CNN model, which is trained based on the generated samples from the GAAE model. The table shows the comparison between different models. For the generated samples from the GAAE model, the CNN model has achieved greater classification accuracy than the supervised BigGAN (86.58% ± 0.56) and the GGAN model (86.72% ± 0.47). This result demonstrates the superiority of the GAAE model in terms of the sample generation for different categories. So the small amount of labelled data used as the guidance during the training phase has assisted the GAAE model for learning the better conditional distribution of the training data thus demonstrates that the GAAE model has performed better than other models in terms of the sample diversity by capturing different modes of the data distribution.

When we trained the CNN model mixing the train data, and the generated samples from the GAAE model, the accuracy of the CNN model increased from 95.52% ± 0.50 to 97.33% ± 0.19. Here, along with the accuracy, the stability of the CNN model is also improved significantly in terms of the standard deviation in the results. We have also conducted the same evaluation on the Nsynth dataset and received similar results which can be found on the table IV. So, we further propose our GGAN model as a data augmentation model thus the generated samples from the GAAE model can be used to augment any related dataset.

3) Effect for the size of the guidance data: Here, The percentage of labelled training data used as guidance has a significant impact on the IS and FID score, which can be found from the table V. It is evident from the results that the more we feed the labelled data during the training, the more we boost the performance of the GAAE model in terms of the sample generation and the diversity. Furthermore, it is also noticeable from here that only with 1% labelled guided data the GAAE model has achieved acceptable performance.

4) Guidance from a different dataset: After calculating the scores for gender based training, we have noticed severe collapse in the performance as the GAAE model has achieved the IS score of 5.31 ± 1.8 and FID score of 35.87 ± 3.2. Because of mixing two different datasets during the training, the generated samples belong to both data distribution resulting in bad IS and FID scores as the scores are calculated with the CNN model trained on the digit classification tasks for S09 dataset, not for the gender classification task. Therefore, to eradicate this problem, we have trained a simple CNN model for the gender classification to calculate the IS and the FID score. So we have randomly selected 15 males and 15 females speaker from Librispeech dataset and used ten males and ten females for training (split into train and validation of 80%: 20%) and others for testing. We achieved an accuracy of 98.3 ± 0.50 and used this model to calculate the IS and FID Score for the generated samples from different models. The scores for different models are given on the table VI. Here in the table, there are two GAAE models; one is trained with the guidance from the S09 dataset with the digit labels and other given conditions/categories. Nevertheless, when we converted these spectrograms to audios, it was clear that the model was able to generate audios correctly according to the categories demonstrating the effectiveness of the guidance to learn the specific categorical distribution of the training dataset. The audios can be found on the link: https://bit.ly/3coz5qO.
C. Sample Classification

With 5% labelled data as guidance the GAAE model has achieved the digit classification accuracy of 94.6 ± 0.03 on the S09 test dataset, where the classification accuracy for the fully supervised CNN classifier is 95.52 ± 0.50. For the Nsyth dataset, the GAAE model has achieved the accuracy of 94.89% ± 0.01, which is better than the accuracy of the supervised CNN (92.01% ± 0.94). Furthermore, The relationship between the percentage of the data used as guidance and the test data classification accuracy is shown in table VII and VIII for S09 and Nsyth dataset, respectively. The results from both tables demonstrate that the classification accuracy on the test data increases along with the stability (standard deviation in the accuracy) as we increase the percentage of the data used as guidance. Also, the GAAE model has outperformed other models in terms of achieving better classification accuracy leveraging the minimal amount of label data. From the tables, we can observe that the GAAE model has performed better than the supervised classifier when it is trained with 100% labelled data because the classifier C takes the advantage of the generated samples as well as the labelled data. So for any classification task, our GAAE model can be used instead of any supervised classifier. Here, the GAAE model achieved great classification accuracy due to the generation of the samples with superior quality according to different categories.

D. Representation Learning

The GAAE model learns two types of representations/latent spaces: \( z_{x_u} \sim u_z \) to learn guidance specific characteristics of the data (Guided representation/post-task-specific representation) and \( z_{x_u} \sim q_z \) to learn general characteristics of the data (General representation/Style representation).

1) Guided Representation Learning: To investigate the impact of the guidance on the representation, we have visualised the latent \( z_{x_u} \) in the 2D plain. The figure 5 shows the representation space for S09 test dataset and figure 6 is the visualisation for the Nsynth dataset. From both figures, it is noticeable that the guided categories are clustered together and well separated in the representation space. So, \( E \) has successfully learned to map the data sample to the representation (latent) space \( u_z \) in a way so that the data categories which are used as guidance, are easily separable in the representation space.

2) General Representation/Style Representation: After investigating the generated audios, we have noticed that the voice of the speaker, audio pitch, background noise are the same for any latent sample \( z_{x_u} \). Therefore, the generated audio samples for the S09 dataset for a certain \( z_{x_u} \) and different digit categories have similar speaker voice and background noise. Furthermore, for the Nsynth dataset, we have noticed a similar pattern.

The digit categories of the generated audios are changed according to the given condition \( y_r \) and the general characteristics of the audio is changed with the change of \( z_{x_u} \), which infers that \( E \) has learned \( q_z \) in a way so that it captures the general attributes of the data. If this is true then pretrained \( E \) should be able to extract general attribute
TABLE V
The relationship between the percentage of the data used as guidance during the training and the sample generation quality of the GAAE model, measured with the IS and the FID score. The scores are calculated for the S09 and the Nsynth dataset.

| Labelled Data | IS Score (S09)  | FID Score (S09) | IS Score (Nsynth) | FID Score (Nsynth) |
|---------------|-----------------|-----------------|-------------------|-------------------|
| 1%            | 6.94 ± 0.04     | 24.21 ± 0.16    | 2.48 ± 0.08       | 145.89 ± 1.32     |
| 2%            | 7.06 ± 0.03     | 23.89 ± 0.11    | 2.53 ± 0.07       | 144.21 ± 0.65     |
| 3%            | 7.12 ± 0.04     | 23.15 ± 0.10    | 2.56 ± 0.05       | 143.01 ± 0.43     |
| 4%            | 7.19 ± 0.02     | 22.91 ± 0.08    | 2.57 ± 0.04       | 142.46 ± 0.38     |
| 5%            | 7.28 ± 0.01     | 22.60 ± 0.07    | 2.58 ± 0.03       | 141.71 ± 0.32     |
| 100%          | 7.45 ± 0.03     | 19.31 ± 0.01    | 2.67 ± 0.02       | 137.65 ± 0.02     |

TABLE VI
Comparison between the performance of the GGAN model trained with gender guidance and the other models on the S09 dataset, in terms of the quality of the generated samples based on the gender attributes of the speaker, 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|
| GAAE                | 1.61 ± 0.17| 29.84 ± 0.43|
| GAAE (Gender Guided)| 1.78 ± 0.03| 20.21 ± 0.01|

Fig. 6. t-SNE visualisation of the learnt representation of the test data of the Nsynth dataset. Here, different colours of points represent different instrument categories. In the representation space, the different instrument categories are clustered together and easily separable.

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. In the representation space, the different digit categories are clustered together and easily separable.

in latent $z'_u$ from any related dataset, which was not used during the training. So to explore this scenario, we have passed the test data from S09 dataset through $E$ to get the general representation $z'_{x,u}$. Then for a fixed $z'_{x,u}$ and different conditions (digit categories), we have generated samples from the pretrained $DE$ network. After converting the generated samples to audios, we have noticed that the generated audios preserved some similar characteristics like speaker gender, voice, pitch, tone, background noise etc. from the input data sample (S09 test data). We also noticed similar scenarios for the Nsynth dataset. The audios can be found on the link: https://bit.ly/36Oz9z9 Here the first second of any audio is the input audio data and rests are the generated audios.

Here, the GAAE model learns general/style attributes of the S09 dataset in the $z'_{x,u}$ latent, so we can expect that it has also disentangled the gender of the speaker in the latent space. To evaluate this, we have used the trained $E$ from the GAAE model to extract latent for an entirely different Librispeech dataset where gender labels are available. For 5000 data randomly sampled from Librispeech dataset, we have extracted the feature/latent $z_{x,u}$ from $E$ and visualized in 2D plain using t-SNE visualisation for exploration. The figure 7 shows the visualisation and here, we can observe that the latent for the same gender of the speakers are clustered together and easily separable from the latent space. This exploration exhibits that the GAAE model was able to learn gender attributes of the speaker from S09 dataset successfully though gender information of the speaker was never used during the training.

Now, The figure 8 shows the generated samples for both S09 and Nsynth dataset based on these interpolated points. Hence, from the figure, we can observe that the transition between two spectrograms generated based on two fixed latent $z_0$ and $z_1$ is very smooth. Moreover, when we converted the spectrograms to audio, we observed the same smooth transition, which indicates the disentanglement of the general attributes in the latent space $q_z$. The audios can be found on the link: https://bit.ly/36Oz9z9

Therefore, it is evident from these explorations that the
VI. IMPACT OF THE HYPERPARAMETERS

We tuned the hyperparameters based on the S09 dataset only because the tuning costs extensive amount of resource and time. Then, we used those hyperparameters for other datasets. From equation 6, $\omega_1$ and $\omega_2$ are two important hyperparameters for training the GAAE model, where $\omega_2 = 1 - \omega_1$. When we increase the $\omega_1$, the model focuses more on the generation and less on the reconstruction. Now if we reduce the $\omega_1$, the model increases the focus for reconstruction and reduces the focus for the generation. The relationship between $\omega_1$ and the IS score, FID score, Classification accuracy can be found in figure 9. The optimal value for the $\omega_1$ is 0.6 and for the $\omega_2$ it is 0.4. The hyperparameters $\alpha$ and $\beta$ from the equation 6 are two main hyperparameters. The value of $\alpha$ parameter determines how much the model will focus on generation and reconstruction loss where $\beta$ parameter determines the focus for the classification and latent loss. From figure 9, we can observe that 0.5 is the optimal value for both of the hyperparameters. In the equation 6 there are three more hyperparameters; $\omega_1, \omega_2$ and $\omega_3$. Hence, $\omega_1$ and $\omega_2$ determines the focus for classification loss for labelled data, generated data respectively, where the $\omega_3$ determines the focus for the latent loss. Here, equal balance is optimal between the classification and latent loss. So we have used 0.25 for $\omega_1, \omega_2$ and 0.50 for the $\omega_3$.

VII. CONCLUSION AND LESSON LEARNT

We have proposed Guided Adversarial Autoencoder (GAAE), where the model learns the conditional audio generation according to the labelled data samples which are used as guidance/supervision during the training. After evaluating the GAAE model based on one-second audio data, we have shown that the GAAE model can outperform the existing literature in terms of quality and mode diversity using only 5% labelled data samples as guidance. Furthermore, we have also proposed the GAAE model as a data augmenting model due to its superior sample generation aptness.

Along with high-fidelity audio generation, our GAAE model was able to disentangle the post-task-specific characteristics/attributes of the data in the learned latent space with fewer labelled data samples as guidance. Therefore, we have demonstrated that the guidance strategy during the training helps the model to focus on specific attributes of the dataset during the representation learning. Moreover, we have also shown that our GAAE model can outperform any supervised classification if it is trained with all the available labelled data. Along with the post-task-specific representation learning, the GAAE model is capable of learning the other variational factors of the training data in a different latent space. Hence, the GAAE model learns guided representation for the specific posterior task at hand and generalised representation for future unknown related jobs.

The GAAE model was evaluated based on the audio of the size of one second; thus, it remains a challenge to make this model work for longer audio samples generation. In the context of representation learning, the GAAE model can be used efficiently for any long audio samples by dividing it into one-second chunks. As we have achieved successful generation and representation with a minimum of 1% labelled data as guidance, we believe that our work will encourage other researchers to explore the GAAE model further for few-shot learning, where the GAAE model can perform similarly with very few number of labelled examples. We built the GAAE model based on the BigGAN architecture thus leaves a great opportunity for the researchers to study the progressive GAN or the Style GAN architecture in the GAAE model.
Fig. 9. The figure shows the relationship between the hyperparameters and the measurement metrics of the GAAE model. The top left plot explains the relationship between the $\omega_1$ and IS score, FID scores. Similarly, the top right explicates the relationship between the $\alpha$ and IS scores, FID scores. Here, The bottom left box illustrates the relationship between the $\omega_1$ and the Classification accuracy. Furthermore, the bottom right plot demonstrates the impact of $\alpha$ on the classification accuracy.

| Training Data Size | CNN Network | BiGAN | GGAN | GAAE |
|--------------------|-------------|-------|------|------|
| 1%                 | 82.21 ± 1.2 | 73.01 ± 1.02 | 84.21 ± 2.24 | 90.21 ± 0.16 |
| 2%                 | 83.04 ± 0.34 | 75.56 ± 0.41 | 85.39 ± 1.24 | 91.45 ± 0.12 |
| 3%                 | 83.78 ± 0.23 | 78.33 ± 0.07 | 88.25 ± 0.10 | 92.67 ± 0.06 |
| 4%                 | 84.11 ± 0.34 | 80.03 ± 0.01 | 91.02 ± 0.50 | 93.70 ± 0.05 |
| 5%                 | 84.50 ± 1.02 | 80.84 ± 1.72 | 92.00 ± 0.87 | 94.59 ± 0.03 |
| 100%               | 95.52 ± 0.50 | 86.77 ± 2.61 | 96.51 ± 0.07 | 97.68 ± 0.01 |

| Training Data Size | CNN Network | BiGAN | GGAN | GAAE |
|--------------------|-------------|-------|------|------|
| 1%                 | 85.76 ± 1.10 | 82.21 ± 0.84 | 88.52 ± 0.32 | 90.26 ± 0.09 |
| 2%                 | 89.79 ± 0.51 | 86.65 ± 0.57 | 91.69 ± 0.24 | 92.96 ± 0.07 |
| 3%                 | 89.83 ± 0.49 | 87.21 ± 0.46 | 91.95 ± 0.20 | 93.12 ± 0.05 |
| 4%                 | 90.52 ± 0.25 | 87.59 ± 0.41 | 92.16 ± 0.19 | 93.73 ± 0.02 |
| 5%                 | 91.07 ± 0.31 | 87.95 ± 0.39 | 92.45 ± 0.14 | 94.23 ± 0.02 |
| 100%               | 92.01 ± 0.94 | 88.09 ± 0.24 | 93.56 ± 0.09 | 94.89 ± 0.01 |

REFERENCES

[1] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 8, pp. 1798–1828, 2013.
[2] F. Locatello, S. Bauer, M. Lui, G. Rtsch, S. Gelly, B. Schlkopf, and O. F. Bachem, “Challenging common assumptions in the unsupervised learning of disentangled representations,” in International Conference on Machine Learning, 2019, best Paper Award. [Online]. Available: [http://proceedings.mlr.press/v97/locatello19a.html](http://proceedings.mlr.press/v97/locatello19a.html)
[3] 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 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: [http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf](http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf)
[4] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” 10 2017.
[5] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture
International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 2746–2750.
[75] H. Yu, Z.-H. Tan, Z. Ma, and J. Guo, "Adversarial network bottle-neck features for noise robust speaker verification," arXiv preprint arXiv:1706.03397, 2017.
[76] S. Sahni, R. Gupta, G. Sivaraman, W. AbdAlmageed, and C. Espy-Wilson, "Adversarial example encoders for speech-based emotion recognition," arXiv preprint arXiv:1806.02146, 2018.
[77] E. Principi, F. Vesperini, S. Squartini, and F. Piazza, "Acoustic novelty detection with adversarial autoencoders," in 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 3324–3330.
[78] J. Sotelo, S. Merti, K. Kamar, J. T. Santos, K. Kastner, A. Courville, and Y. Bengio, "Character2vec: End-to-end speech synthesis," 2017.
[79] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaity, Z. Yang, Y. Xiao, Z. Chen, S. Bengio et al., "Tacotron: Towards end-to-end speech synthesis," arXiv preprint arXiv:1703.10135, 2017.
[80] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Jaisovsky, and A. Courville, "Adversarially learned inference," arXiv preprint arXiv:1606.04704, 2016.
[81] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," CoRR, vol. abs/1605.09782, 2016. [Online]. Available: http://arxiv.org/abs/1605.09782
[82] K. Sricharan, R. Bala, M. Shreve, H. Ding, K. Saketh, and J. Sun, “Semi-supervised conditional gans,” 2017.
[83] M. Lucic, M. Tschannen, M. Ritter, X. Zhai, O. Bachem, and S. Gelly, “High-fidelity image generation with fewer labels,” 2019.
[84] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al., “Pytorch: An imperative style, high-performance deep learning library,” in Advances in Neural Information Processing Systems, 2019, pp. 8024–8035.
[85] P. Warden, “Speech commands: A dataset for limited-vocabulary speech recognition,” CoRR, vol. abs/1804.03320, 2018. [Online]. Available: http://arxiv.org/abs/1804.03320
[86] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: an asr corpus based on public domain audio books,” in 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2015, pp. 5206–5210.
[87] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi, “Neural audio synthesis of musical notes with wavenet autoencoders,” 2017.
[88] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, “Improved techniques for training gans,” in Advances in Neural Information Processing Systems 29, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 2234–2242. [Online]. Available: http://papers.nips.cc/paper/6032-unpaired-techniques-for-training-gans.pdf
[89] 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,” 2017.
[90] K. Shmelkov, C. Schmid, and K. Alahari, “How good is my gan?” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 2234–2242. [Online]. Available: http://papers.nips.cc/paper/6032-unpaired-techniques-for-training-gans.pdf
[91] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in CVPR09, 2009.
APPENDIX A
SUPPLEMENT MATERIAL

This section presents the details of the neural networks used in this paper. We have followed the abbreviations and description style from paper of Mario et al.[88]

**TABLE IX**
ABBREVIATIONS FOR DEFINING THE ARCHITECTURES.

| Full Name                        | Abbreviation |
|----------------------------------|--------------|
| Resample                         | RS           |
| Batch normalisation              | BN           |
| Conditional batch normalisation  | cBN          |
| Downscale                        | D            |
| Upscale                          | U            |
| Spectral normalisation           | SN           |
| Input height                     | h            |
| Input width                      | w            |
| True label                       | y            |
| Input channels                   | ci           |
| Output channels                  | co           |
| Number of channels               | ch           |

**A. Supervised BigGAN**

We have taken the exact implementation of the Supervised BigGAN from our GGAN paper[16]. Therefore, for the implementation of both Generator and Discriminator, we used Resnet architecture from the BigGAN paper[6]. The layers are shown in the X and XI. Generator and Discriminator architectures are shown in TableXII and XIII respectively. We use a learning rate of 0.00005 and 0.0002 for the Generator and the Discriminator, respectively. We set the number of channels (ch) to 16 to minimise the computational expenses, as the higher number of channels such as 64 and 32 only offer negligible improvements.

**TABLE X**
ARCHITECTURE OF THE RESBLOCK GENERATOR WITH UPSAMPLING FOR THE SUPERVISED BIGGAN.

| Layer Name       | Kernal Size | RS | Output Size |
|------------------|-------------|----|-------------|
| Shortcut         | [1,1,1]     | U  | 2h × 2w × c_-{o} |
| cBN, ReLU        | -           | -  | h × w × c_-{i} |
| Convolution      | [3,3,1]     | U  | 2h × 2w × c_-{o} |
| cBN, ReLU        | -           | -  | 2h × 2w × c_-{o} |
| Convolution      | [3,3,1]     | U  | 2h × 2w × c_-{o} |
| Addition         | -           | -  | 2h × 2w × c_-{o} |

**B. Unsupervised BigGAN**

Similarly, for the unsupervised BigGAN, we have followed the same implementation from GGAN paper[16]. The table XIV and XV shows the upsampling and downsampling layers respectively. The architectures of Generator and Discriminator are shown in the TableXVI and XVII respectively. The Learning rate and channels are the same as supervised BigGAN.
| Layer Name | Kernal Size | RS | Output Size |
|------------|-------------|----|-------------|
| Shortcut   | [1,1,1]     | D  | h/2 × w/2 × c_{o} |
| ReLU       | -           | -  | h × w × c_{i}   |
| Convolution| [3,3,1]     | U  | h × w × c_{o}   |
| ReLU       | -           | -  | h × w × c_{o}   |
| Convolution| [3,3,1]     | D  | h/2 × w/2 × c_{o} |
| Addition   | -           | -  | h/2 × w/2 × c_{o} |

| Layer Name | RS | SN | Output Size |
|------------|----|----|-------------|
| Input z    | -  | -  | 128         |
| Dense      | -  | -  | 4 × 2 × 16. ch |
| ResBlock   | U  | SN | 8 × 4 × 16. ch |
| ResBlock   | U  | SN | 16 × 8 × 16. ch |
| ResBlock   | U  | SN | 32 × 16 × 16. ch |
| ResBlock   | U  | SN | 64 × 32 × 16. ch |
| ResBlock   | U  | SN | 128 × 64 × 16. ch |
| Non-local block | - | - | 128 × 64 × 16. ch |
| ResBlock   | U  | SN | 256 × 128 × 1. ch |
| BN, ReLU   | -  | -  | 256 × 128 × 1  |
| Conv [3, 3, 1] | - | - | 256 × 128 × 1  |
| Tanh       | -  | -  | 256 × 128 × 1  |

| Layer Name | RS | Output Size |
|------------|----|-------------|
| Input z    | -  | 128         |
| Dense      | -  | 4 × 2 × 16. ch |
| ResBlock   | U  | 8 × 4 × 16. ch |
| ResBlock   | U  | 16 × 8 × 16. ch |
| ResBlock   | U  | 32 × 16 × 16. ch |
| ResBlock   | U  | 64 × 32 × 16. ch |
| ResBlock   | U  | 128 × 64 × 16. ch |
| Non-local block | - | - | 128 × 64 × 16. ch |
| ResBlock   | U  | 256 × 128 × 1. ch |
| BN, ReLU   | -  | 256 × 128 × 1  |
| Conv [3, 3, 1] | - | - | 256 × 128 × 1  |
| Tanh       | -  | 256 × 128 × 1  |

C. BiGAN

For the BiGAN model, we have trained Feature Extractor and Discriminator network top of the unsupervised BigGAN. The Feature Extractor network creates the features for real samples, and Discriminator tries to differentiate between the generated features and the random noise. The detail is exactly followed from the BiGAN paper [81]. The downsampling layer is same as the unsupervised BigGAN and can be found on table XVI. The architecture of the Feature Extractor network is shown in table XVIII. Furthermore, the architecture of the Discriminator is given on table XIX.

D. GAAE

In the GAAE model, the downsampling and upsampling layers are the same as those shown in table X and XI respectively.

The Encoder architecture is given in table XX where we used two Dense layers to get the z_{x,a} and z_{x,b} from Global sum pooling layer. For the Decoder, the conditional vector y_r or y_{z,a} is given through the conditional Batch Normaliser
For many classification tasks, we have mentioned about Simple Classifier throughout the paper. The architecture of

(cBN) from the upsampling layer. The Classifier network is built upon some Dense layer, and the architecture is given in table [XXII]. For the Sample Discriminator, we have exactly followed the implementation from table [XIII]. Here, in the table [XIII], the y is the conditional vector and h is the output from Global sum pooling layer. For the Latent Discriminator, we have used multi dense layers, and the architecture is given in the table [XXIII].

The learning rates for both Discriminators are 0.0002, and for other networks, the learning rate was 0.00005. We set the number of channels to 16 for all the experiment carried out with GAAE.

**E. Simple Classifier**

For many classification tasks, we have mentioned about Simple Classifier throughout the paper. The architecture of

| Layer Name            | RS | Output Size   |
|-----------------------|----|---------------|
| Input                 | -  | 256 × 128 × 1 |
| Spectrogram           | -  | 256 × 128 × 1 |
| ResBlock              | D  | 128 × 64 × 1. ch |
| Non-local block       | -  | 128 × 64 × 1. ch |
| ResBlock              | -  | 64 × 32 × 1. ch |
| ResBlock              | D  | 32 × 16 × 2. ch |
| ResBlock              | D  | 16 × 8 × 4. ch |
| ResBlock              | D  | 8 × 4 × 8. ch  |
| ResBlock              | D  | 4 × 2 × 16. ch |
| ResBlock (No Shortcut)| -  | 4 × 2 × 16. ch |
| ReLU                  | -  | 4 × 2 × 16. ch |
| Global sum pooling    | -  | 1 × 1 × 16. ch |
| Dense                 | -  | 128           |
| Table XVIII ARCHITECTURE OF THE FEATURE EXTRACTOR NETWORK FOR THE BIGGAN. | | | |

| Layer Name            | RS | Output Size   |
|-----------------------|----|---------------|
| Input                 | -  | 256 × 128 × 1 |
| Spectrogram           | -  | 256 × 128 × 1 |
| ResBlock              | D  | 128 × 64 × 1. ch |
| Non-local block       | -  | 128 × 64 × 1. ch |
| ResBlock              | -  | 64 × 32 × 1. ch |
| ResBlock              | D  | 32 × 16 × 2. ch |
| ResBlock              | D  | 16 × 8 × 4. ch |
| ResBlock              | D  | 8 × 4 × 8. ch  |
| ResBlock              | D  | 4 × 2 × 16. ch |
| ResBlock              | D  | 4 × 2 × 16. ch |
| ReLU                  | -  | 4 × 2 × 16. ch |
| Global sum pooling    | -  | 1 × 1 × 16. ch |
| Concat with input feature | - | 256+128=384 |
| Dense                 | -  | 128           |
| ReLU                  | -  | 128           |
| Dense                 | -  | 1             |
| Table XIX ARCHITECTURE OF THE DISCRIMINATOR FOR THE BIGGAN. | | | |

| Layer Name            | RS | Output Size   |
|-----------------------|----|---------------|
| Input                 | -  | 256 × 128 × 1 |
| Spectrogram           | -  | 256 × 128 × 1 |
| ResBlock              | D  | 128 × 64 × 1. ch |
| Non-local block       | -  | 128 × 64 × 1. ch |
| ResBlock              | -  | 64 × 32 × 1. ch |
| ResBlock              | D  | 32 × 16 × 2. ch |
| ResBlock              | D  | 16 × 8 × 4. ch |
| ResBlock              | D  | 8 × 4 × 8. ch  |
| ResBlock              | D  | 4 × 2 × 16. ch |
| ResBlock              | D  | 4 × 2 × 16. ch |
| ReLU                  | -  | 4 × 2 × 16. ch |
| Global sum pooling    | -  | 1 × 1 × 16. ch |
| Concat with input feature | - | 256+128=384 |
| Dense                 | -  | 128           |
| ReLU                  | -  | 128           |
| Dense                 | -  | 1             |
| Table XX ARCHITECTURE OF THE ENCODER FOR THE GAAE. | | | |

these classifiers are followed from the table [XXIV]. Here, c is the number of output according to the classification categories. The learning rates is used as 0.0001 for this Classifier network.
TABLE XXXI
ARCHITECTURE OF THE DECODER FOR THE GAAE.

| Layer Name          | RS | SN | Output Size |
|---------------------|----|----|-------------|
| Input latent vector | -  | -  | 128         |
| Dense               | -  | -  | 4 × 2 × 16. ch |
| ResBlock            | U  | SN | 8 × 4 × 16. ch |
| ResBlock            | U  | SN | 16 × 8 × 16. ch |
| ResBlock            | U  | SN | 32 × 16 × 16. ch |
| ResBlock            | U  | SN | 64 × 32 × 16. ch |
| ResBlock            | U  | SN | 128 × 64 × 16. ch |
| Non-local block     | -  | -  | 128 × 64 × 16. ch |
| ResBlock            | U  | SN | 256 × 128 × 1. ch |
| BN, ReLU            | -  | -  | 256 × 128 × 1 |
| Conv [3, 3, 1]      | -  | -  | 256 × 128 × 1 |
| Tanh                | -  | -  | 256 × 128 × 1 |

TABLE XXXII
ARCHITECTURE OF THE CLASSIFIER FOR THE GGGAN.

| Layer Name | Output Size |
|------------|-------------|
| Input latent vector | 128         |
| Dense       | 128         |
| ReLU        | 128         |
| Dense       | 10          |

TABLE XXXIII
ARCHITECTURE OF THE LATENT DISCRIMINATOR FOR THE GGGAN.

| Layer Name   | Output Size |
|--------------|-------------|
| Input latent vector | 128         |
| Dense        | 128         |
| ReLU         | 128         |
| Dense        | 1           |

TABLE XXXIV
ARCHITECTURE OF THE SIMPLE SPECTROGRAM CLASSIFIER.

| Layer Name | Output Size |
|------------|-------------|
| Input Spectrogram | 256 × 128 × 1 |
| Convolution [3, 3, 32] | 256 × 128 × 32 |
| Maxpool [2, 2] | 128 × 64 × 32 |
| Convolution [3, 3, 64] | 128 × 64 × 64 |
| Maxpool [2, 2] | 64 × 32 × 64 |
| Convolution [3, 3, 128] | 64 × 32 × 128 |
| Maxpool [2, 2] | 32 × 16 × 128 |
| Convolution [3, 3, 256] | 32 × 16 × 256 |
| Maxpool [2, 2] | 16 × 8 × 256 |
| Dense        | c           |

Kazi Nazmul Haque is a PhD student at University of Southern Queensland, Australia. He has been working professionally in the field of machine learning for more than five years. Kazi’s research work focuses on building machine learning models to solve diverse real-world problems. The current focus of his research work is unsupervised representation learning for the audio data. He has completed his Master in Information Technology from Jahangirnagar University, Bangladesh.

Rajib Rana is an experimental computer scientist, Advance Queensland Research Fellow and a Senior Lecturer in the University of Southern Queensland. He is also the Director of IoT Health research program at the University of Southern Queensland. He is the recipient of the prestigious Young Tall Poppy QLD Award 2018 as one of Queenslands most outstanding scientists for achievements in the area of scientific research and communication. Rana’s 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 focus is on Unsupervised Representation Learning. He received his B.Sc. degree in Computer Science and Engineering from Khulna University, Bangladesh with Prime Minister and Presidents Gold medal for outstanding achievements and Ph.D. in Computer Science and Engineering from the University of New South Wales, Sydney, Australia in 2011. He received his postdoctoral training at Autonomous Systems Laboratory, CSIRO before joining the University of Southern Queensland as Faculty in 2015.

Björn W. Schuller (M’05–SM’15–F’18) received his diploma in 1999, his doctoral degree for his study on Automatic Speech and Emotion Recognition in 2006, and his 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 TUM in Munich/Germany. He is Professor of Artificial Intelligence in the Department of Computing at the Imperial College London/UK, where he heads GLAM - the Group on Language, Audio & Music, Full Professor and head of the Z.D.B Chair of Embedded Intelligence for Health Care and Wellbeing at the University of Augsburg/Germany, and CEO of audEERING. He was previously full professor and head of the Chair of Complex and Intelligent Systems at the University of Passau/Germany. Professor Schuller is Fellow of the IEEE, Golden Core Member of the IEEE Computer Society, Senior Member of the ACM, President-emeritus of the Association for the Advancement of Affective Computing (AAAC), and was elected member of the IEEE Speech and Language Processing Technical Committee. He (co-)authored 5 books and more than 800 publications in peer-reviewed books, journals, and conference proceedings leading to more than overall 25 000 citations (h-index = 73). Schuller is general chair of ACII 2019, co-Program Chair of Interspeech 2019 and ICMI 2019, repeated Area Chair of ICASSP, and former Editor in Chief of the IEEE Transactions on Affective Computing next to a multitude of further Associate and Guest Editor roles and functions in Technical and Organisational Committees.